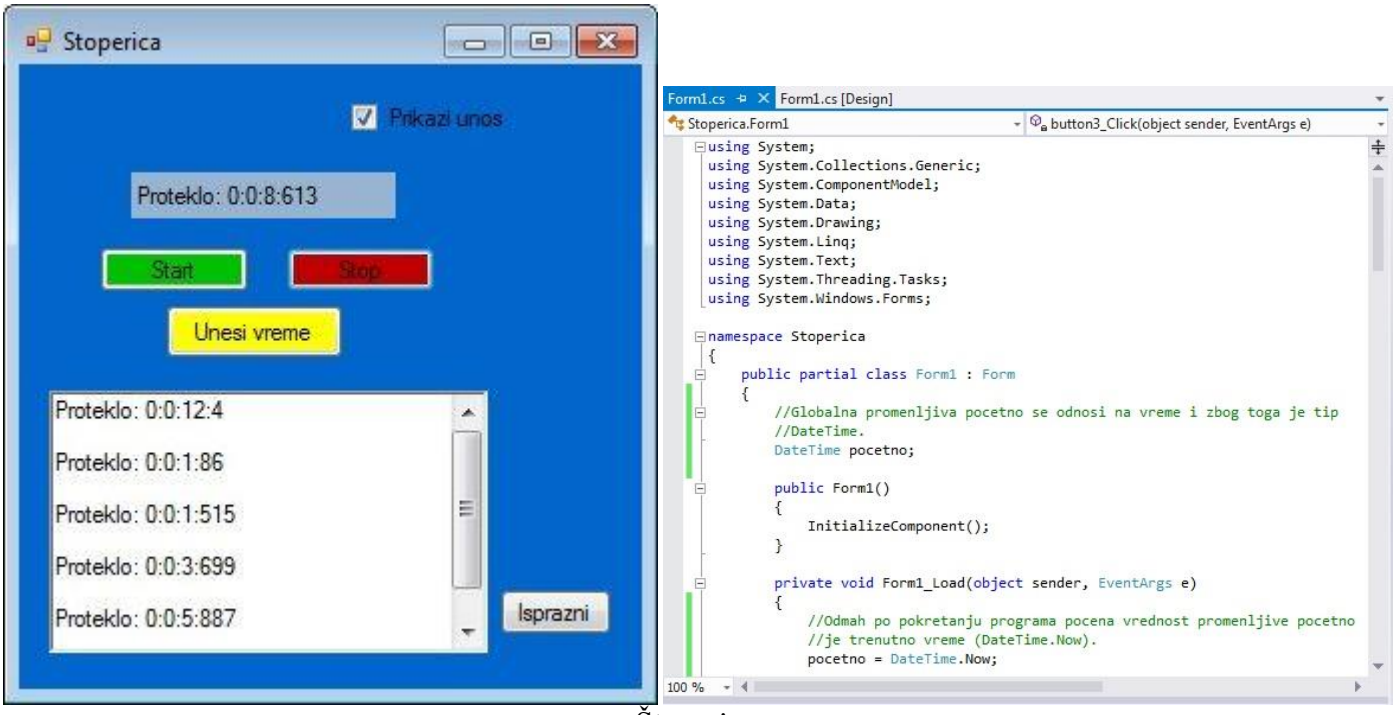
Увод у развојно окружење

Пре почетка рада потребно је да се упознамо са самим окружењем у коме ћемо радити. Бавићемо се креирањем разноразних апликација па је због тога потребно да знамо које су нам све опције на располагању како бисмо направили квалитетну апликацију.

Приликом креирања апликација користићемо разне компоненте као што су дугмићи, поља за унос текста, мениje, лабеле, поља за чекирања, оквири за слике и многе друге. Дакле, нама ће бити понуђени разни објекти са којима морамо да знамо да рукујемо на правилан начин. На врху хијерархије објеката налази се сама апликација, испод апликације се налази бар једна, а обично и више форми, а свака форма садржи компоненте. Само додавање компненти форми и креирање изгледа форме је јако једноставно у шта ћемо се и сами уверити чим почнемо са радом. На почетку програмирање Windows апликација је било веома тешко, али појавом визуелних алата креирање оваквих апликација је доста поједностављено. Сви елементи који се користе у програмирању апликација стоје нам на располагању са својим својствима тако да нама само преостаје да их организујемо на нама потребан начин, поставимо им жељена својства и напишемо потребан код. У колекцији 3.1 су приказани примери различитих апликација које се могу испробати.



Štoperica

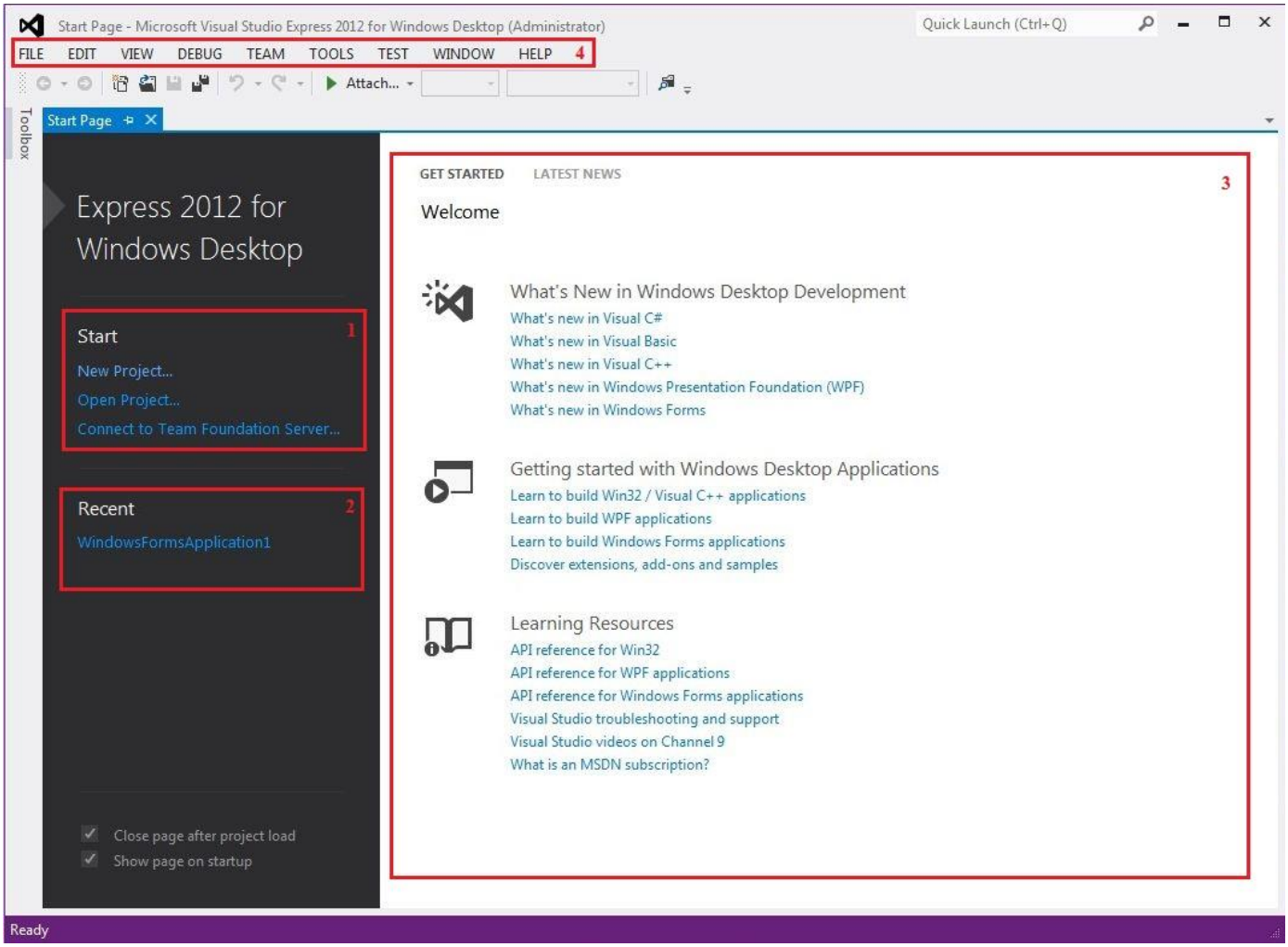
Следећи корак је писање кода, али и писање кода је такође олакшано јер се приликом куцања појављују листе разних функција које су нам на располагању и додатна објашњења како се оне правилно међусобно комбинују. Дакле, нема потребе напамет да учимо све те функције.

У овом курсу ће бити приказани и многи примери помоћу којих ће се приказати шта је то све могуће направити и који ће нам дати идеју за самостални рад. У Колекцији 3.2 су приказани делови програмских кодова апликација из Колекције 3.1.

Елементи развојног окружења

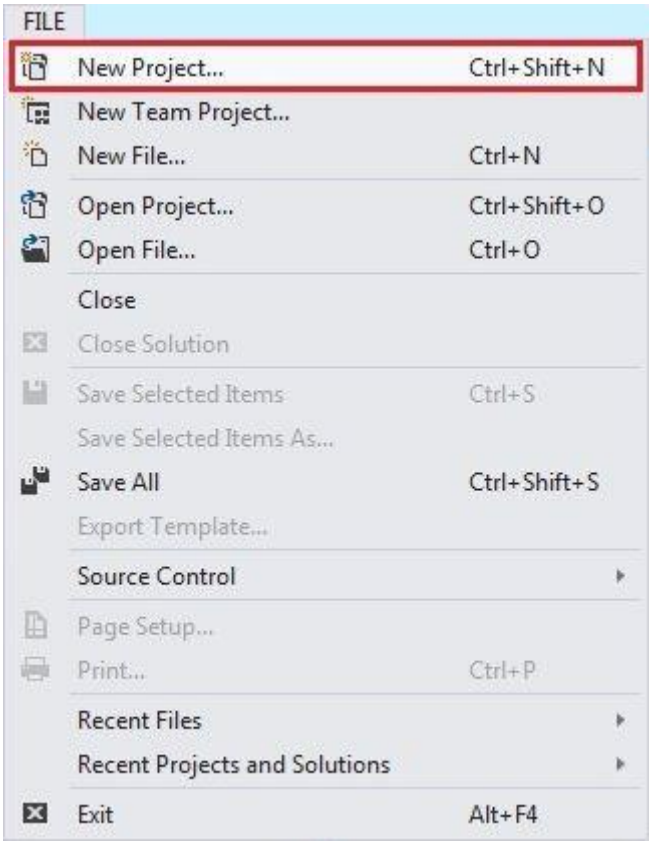
Отварање новог пројекта

У овом делу ћемо се упознати са елементима развојног окружења у коме ћемо радити, али пре тога је потребно да знамо да отворимо нови пројекат. После покретања програма на екрану ће се појавити почетна страна. Изглед почетне стране је приказан на Слици 3.1.



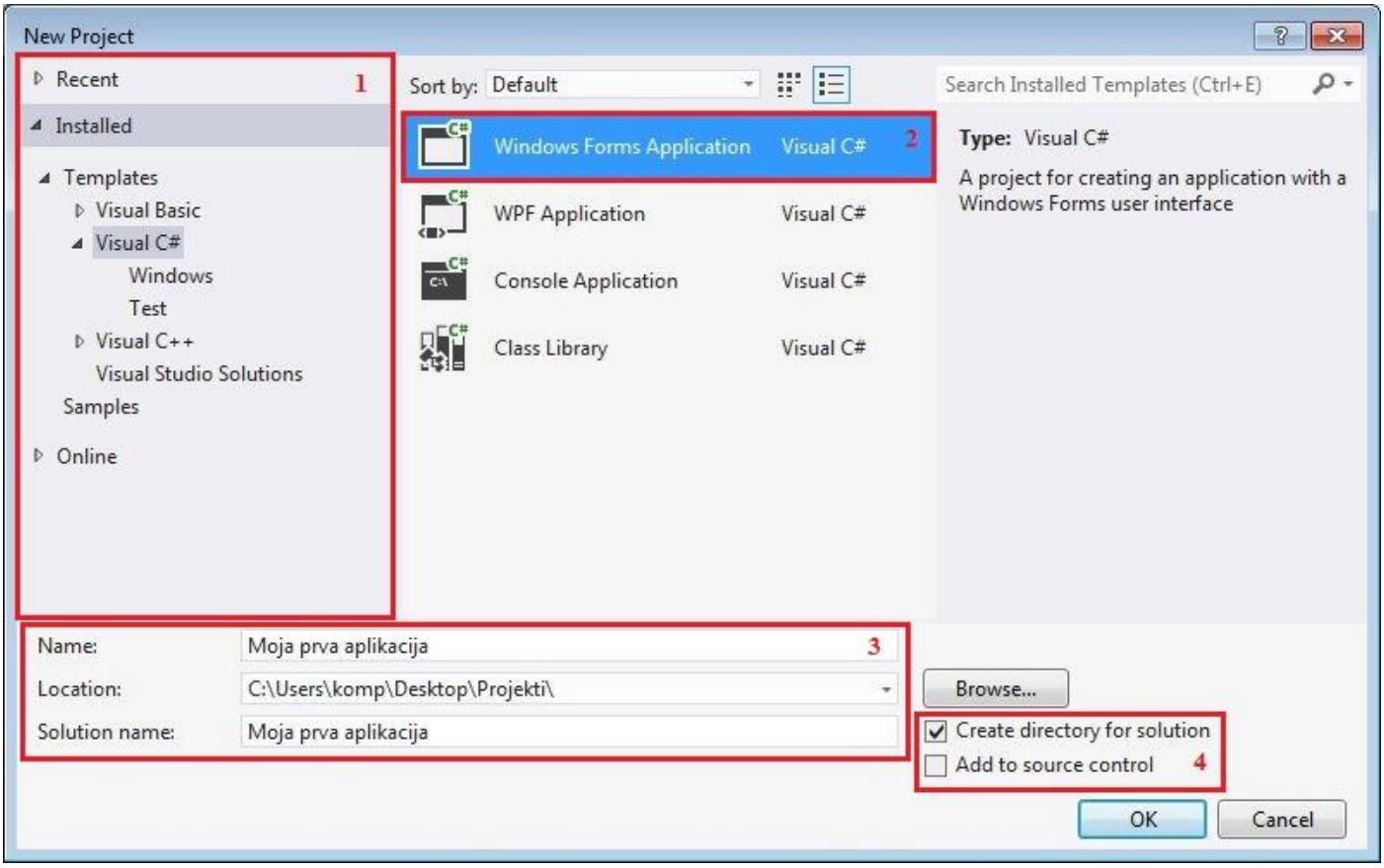
Слика 3.1. Почетна страна

На почетној страни се налази: трака главног менија (4), листа претходно отворених пројеката (2), део у коме се приказују најновије вести везане за Visual Studio које се аутоматски ажурирају уколико постоји интернет конекција и део са корисним линковима за почетнике (3) и опције за креирање новог пројекта и за отварање већ постојећег пројекта (1). Кликом на опцију ***Open Project...*** пружа нам се могућност отварања већ направљеног и сачуваног пројекта, док се кликом на опцију ***New Project...*** пружа могућност да креирамо нови пројекат, што је баш оно што нама треба. Када кликнемо на опцију ***New Project...*** отвориће се нови прозор у коме се нуде разне опције за које ми треба да се одлучимо (*Слика 3.3*). Ово исто смо могли да постигнемо и ако бисмо из траке главног менија (4) изабрали ***File*** мени, а затим из ***File*** менија ***New Project...*** (*Слика 3.2*).(На овај начин смо отварали нове документе и када смо радили у Word-у, Excel-у и PowerPoint-у.)



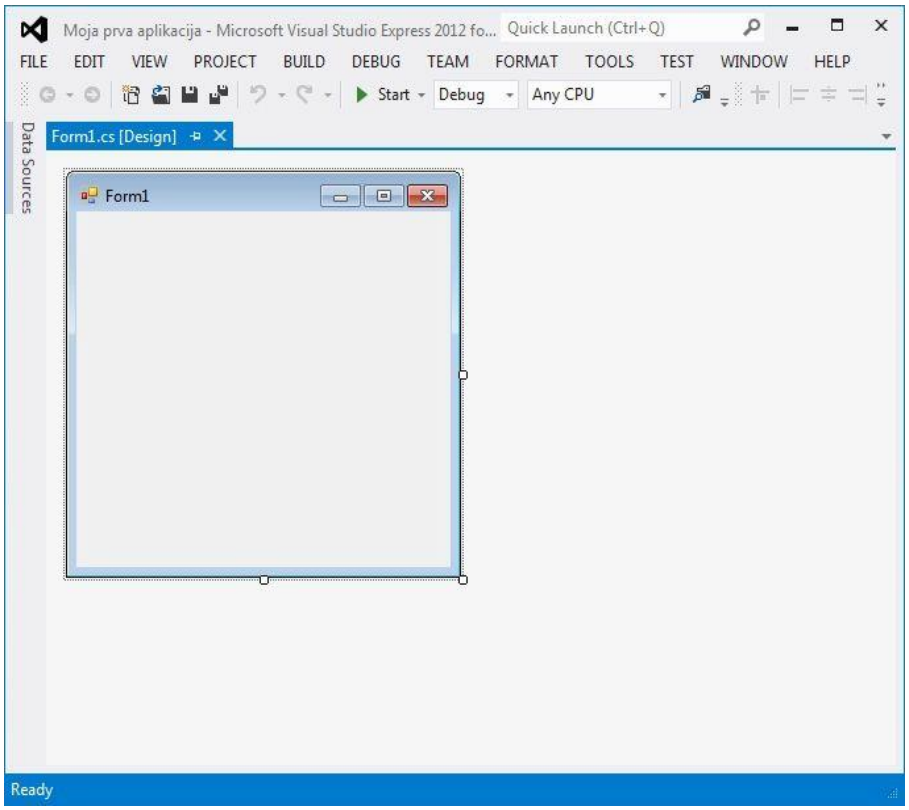
Слика 3.2. File мени

Вратимо се на опције које се нуде у прозору који је приказан на Слици 3.3. Битно је да у левом делу прозора под ставком Visual C# буде селектована опција Windows (1). Пошто ћемо ми да правимо апликацију потребно је да кликнемо на ставку под именом ***Windows Forms Application*** (2). У доњем делу прозора се налази поље за задавање имена апликацији (***Name***) и поље у које ћемо уписати адресу фолдера у коме ћемо чувати наше апликације (***Location***) (3). Локацију бирамо тако што кликнемо на дугме *Browse* и изаберемо фолдер у коме желимо да сачувамо наш пројекат. Значи ми овим поступком уједно и чувамо нови пројекат. Јако је битно да запамтимо ту локацију тј. тај фолдер, јер ако касније желимо да поново отворимо наш пројекат, наћи ћемо га баш у том фолдеру. На Слици 3.3, апликација је названа *Моја прва апликација*, а за локацију је одабран фолдер под именом *Пројекти* који се налази на десктопу. Такође је битно да ставка Create directory for solution буде штиклирана (4).

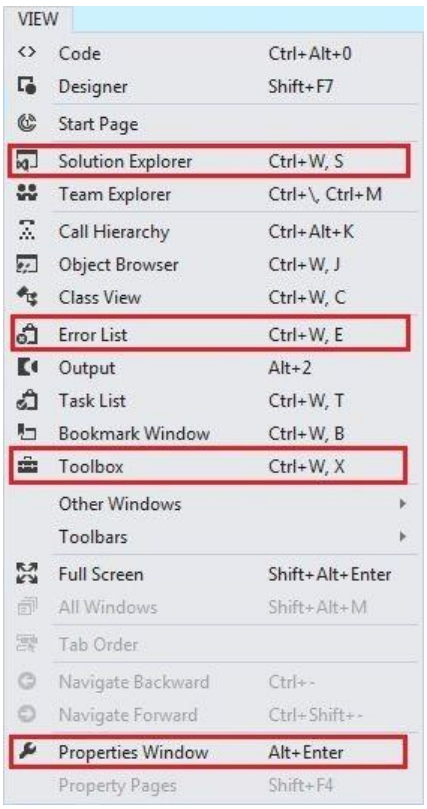


Слика 3.3. Прозор за отварање новог пројекта

Након што извршимо све наведене ставке за потврду кликнемо на дугме *ОК*. Неколико секунди након што кликнемо на дугме *ОК* отвориће се прозор за рад. Обично када први пут отварамо нови пројекат, прозор за рад ће изгледати као што је приказано на Слици 3.4.

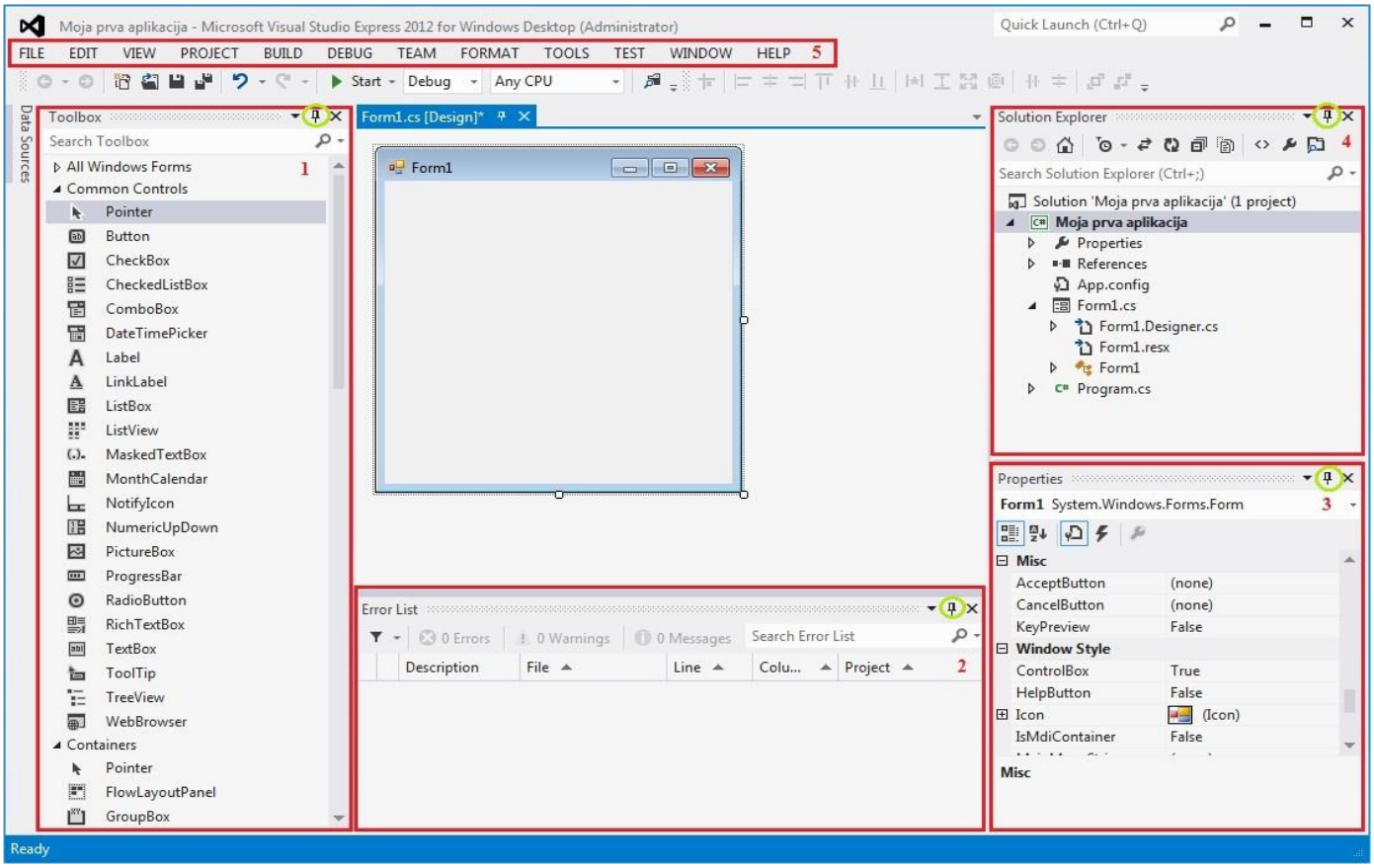


Слика 3.4. Радна површина



Слика 3.5. View мени

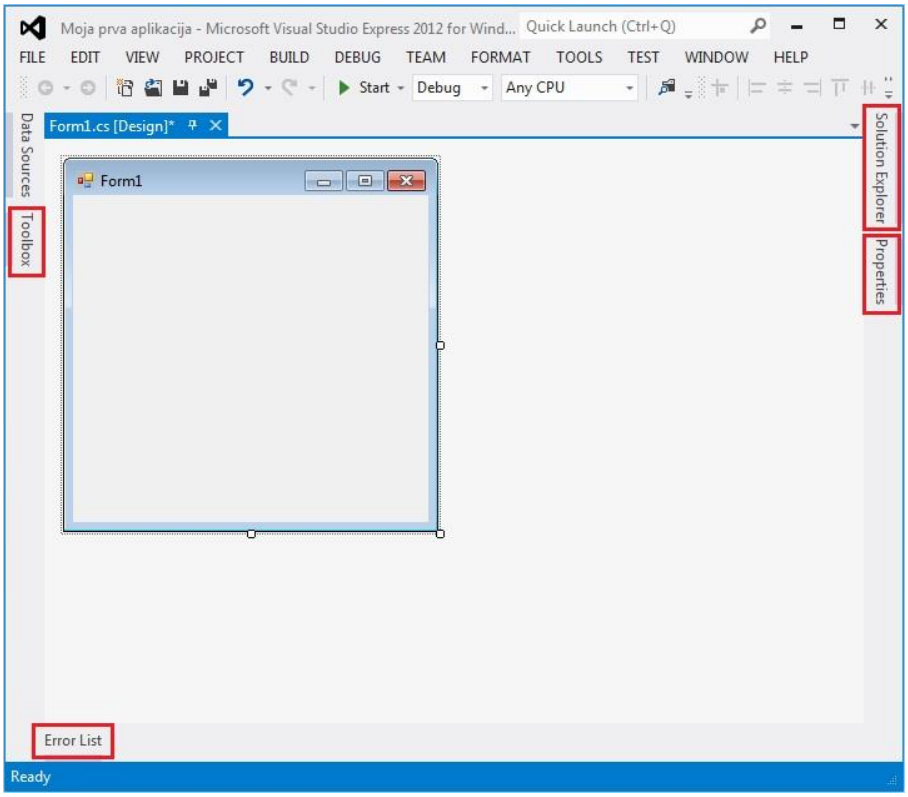
На прозору ће бити приказана само форма која је на почетку празна јер још увек ништа нисмо урадили. Међутим, нама ће бити потребни још неки елементи који нису приказани па их зато морамо сами додати прозору за рад. То ћемо урадити на следећи начин. Из траке главног менија изабраћемо *View* мени, а из *View* менија ***Solution Explorer, Error List, Toolbox*** и ***Properties Window***. На Слици 3.5 је приказано где се у *View* менију налазе нама потребни елементи. Након додавања наш прозор за рад ће изгледати као што је приказано на Слици 3.6.



Слика 3.6. Радна површина

На Слици 3.6 видимо да се *Toolbox* (палета са компонентама) налази у левом делу прозора за рад (1), *Error List* (листа грешака) при дну прозора (2), *Properties* (својства) у десном делу прозора (3), а *Solution Explorer* (претраживач решења) такође у десном делу прозора (4). Сада наш прозор за рад има све елементе који су нам потребни за креирање апликација. Такође, уколико случајно затворимо неки од ових елемената знаћемо како да их поново укључимо, увек се укључују на исти начин тј, баш као што смо то сада урадили.

Напомена: Постоји могућност да ће се сви ови елементи које смо додали из *View* менија сакрити када поново кликнемо на форму или на простор у коме се налази форма. Ако се то деси, радна површина ће изгледати као на Слици 3.7.



Слика 3.7. Радна површина

Ови елементи се још увек налазе на радној површини, али су сакривени и можемо их поново приказати ако кликнемо на „картице" у којима су исписана њихова имена (*Слика 3.7*). Да се ово не би стално дешавало, потребно је фиксирати све ове елементе за радну површину. Да бисмо их фиксирали потребно је да на сваком елементу кликнемо знак чиоде који се налази у десном горњем углу сваког од ових елемената (*Слика 3.6*).

Сви ови елементи имају своју улогу приликом рада на пројекту и користићемо их јако често када будемо правили апликације. Сада ћемо рећи нешто више о сваком од њих.

Елементи прозора за рад

1. *Палета са компонентама (Toolbox)*
2. *Листа грешака (Error List)*
3. *Листа својстава (Properties)*
4. *Претраживач решења (Solution Explorer)*
5. *Трака главног менија*

1. *Палета са компонентама (Toolbox)* је листа компонената које се могу додавати форми. Ту се налазе већ поменуте компоненте као што су дугмићи, лабеле, поља за унос текста и многе друге које ћемо користити. Приликом креирања апликација бирају се жељене компоненте и распоређују по форми.

2. *Листа грешака (Error List)* је, као што и само име говори, листа у којој се исписују грешке које се направе приликом куцања кода или приликом покретања програма. Уколико се направи нека грешка онда ће се у листи појавити порука да је грешка направљена, биће написано тачно у којој линији кода је грешка направљена и биће дато додатно објашњење које ће помоћи у исправљању грешака. Ово додатно олакшава програмирање. Раније је било доста теже исправити грешку јер јако често не можемо сами да нађемо грешку. Овим је процес проналажења грешака знатно убрзан, јер уместо да прегледамо цео код испочетка, тачно знамо линију кода у којој треба да је тражимо.

3. *Листа својстава (Properties)* приказује одређена својства изабране компоненте или форме. Та својства се могу мењати. То су на пример назив форме или неке компоненте, боја, ширина, висина, фонт слова и тако даље. Значи, променом ових својстава можемо постићи да изглед апликације буде онакав какав ми желимо.

4. *Претраживач решења (Solution Explorer)* пружа организовани приказ свих фајлова и фолдера који се налазе у нашем пројекту, јер приликом креирања апликације аутоматски се прави више различитих фолдера и фајлова који се могу наћи у фолдеру у коме смо сачували наш пројекат.

5. *Трака главног менија* такође је јако битна. Из траке главног менија ћемо најчешће користити ***FILE, EDIT, VIEW*** и ***DEBUG*** опције. Са неким од ових опција смо се већ сусрели приликом рада у *Wordu*-у на пример, а има доста и нових од којих нећемо све користити.

Из менија се најчешће користе следеће опције:

New Project	- омогућава отварање новог пројекта као што смо већ објаснили
Open Project	- омогућава отварање већ постојећег пројекта
Close	- омогућава затварање пројекта
Save	- омогућава чување било којих измена које смо направили приликом креирања пројекта
Save As	- омогућава чување активног пројекта под неким новим именом, користи се када задатак који смо урадили желимо да искористимо за неки други задатак који је сличан па уместо да све правимо испочетка само сачувамо под новим именом, као неки други пројекат и направимо потребне измене
Save All	- не разликује се умногоме од опције *Save*, дакле, такође омогућава чување било каквих измена до којих је дошло приликом рада

Из менија се најчешће користе следеће опције:

Undo	- поништава последњу акцију или брисање
Redo	- омогућава враћање последње акције која је поништена помоћу *Undo* опције
Cut	- селектовани део брише и премешта на *clipboard*
Copy	- селектовани део копира на *clipboard*, али за разлику од *Cut* опције не брише селектовани део
Paste	- омогућава копирање садржаја са *clipboard-a* у документ, дакле опције *Copy* (односно *Cut*) и *Paste* су повезане и обично када користимо једну одмах затим користимо и другу
Delete	- ова опција служи за брисање селектованог дела документа
Select All	- овом опцијом се селектује комплетан садржај документа

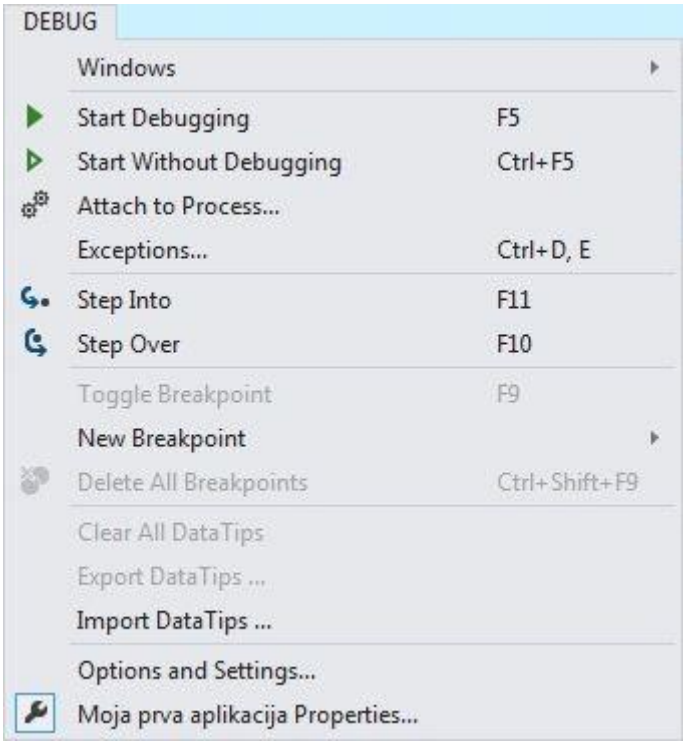
Напомена: Са свим овим опцијама, из *EDIT* менија, смо се већ сусрели приликом обраде текста у *Word-у*. Знамо да се оне користе приликом сређивања било каквог текста, према томе на исти начин се користе и приликом писања кода нашег пројекта који креирамо. Међутим, ове опције такође могу да се користе и приликом дизајнирања наше форме, само што сада није у питању неки текст већ компоненте које се додају форми (објекти). Значи од *EDIT* менија имамо велику корист и приликом писања кода и прилоком рада са објектима форме.

Из менија се најчешће користе следеће опције:

Већ смо се сусрели са неким опцијама из *View* менија на почетку и објаснили чему те опције служе. То су *Solution Explorer*, *Error List*, *Properties Window* и *Toolbox* (*Слика 3.5*). Поред ових опција користићемо још и следеће опције.

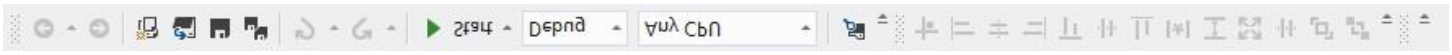
Code	- омогућава приказ кода нашег програма
Designer	- омогућава приказ форме са свим својим компонентама, односно омогућава приказ дизајна наше апликације
Start Page	- омогућава приказ почетне стране
Full Screen	- омогућава приказивање нашег прозора за рад преко целог екрана

Из менија се најчешће користе следеће опције:



Start Debugging	- ова опција служи за покретање програма, али програм се такође може покренути ако на тастатури притиснемо дугме *F5*
Stop Debugging	- ова опција служи за прекидање извршавања програма и у DEBUG менију ће се појавити тек када покренемо програм

Приметимо да већина ових опција из менија има и одговарајућу пречицу (комбинацију једног или више тастера). Ове пречице су написане поред имена опције у менију, то су на пример *Ctrl+S*, *F5*, *Ctrl+W,S* и тако даље. Неке од ових опција се могу наћи и у траци са алатима која се налази одмах испод траке главног менија (*Слика 3.8*). Трака са алатима служи да би се приступ опцијама које се јако често користе поједноставио, тако да уместо да их тражимо у поменутим менијима можемо се послужити траком са алатима.

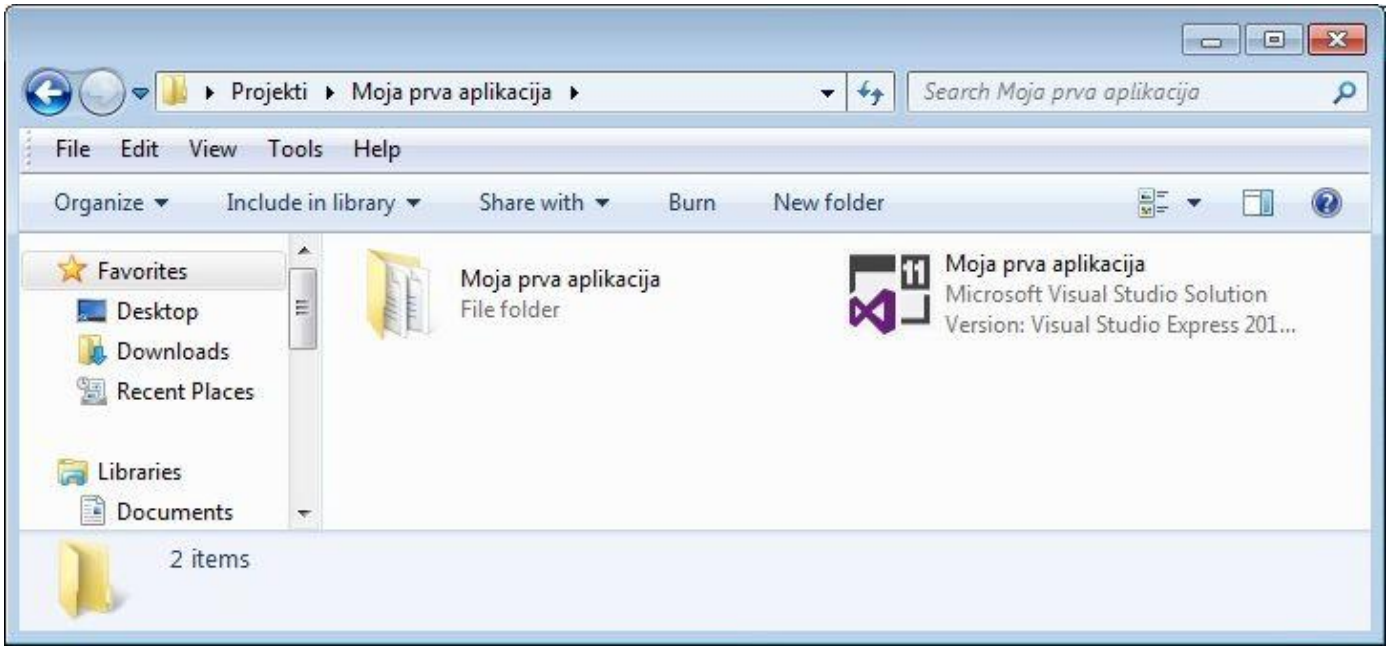


Слика 3.8. Трака са алатима

Још ћемо додатно напоменути по нешто о чувању пројекта, прокретању програма и о отварању сачувних пројеката.

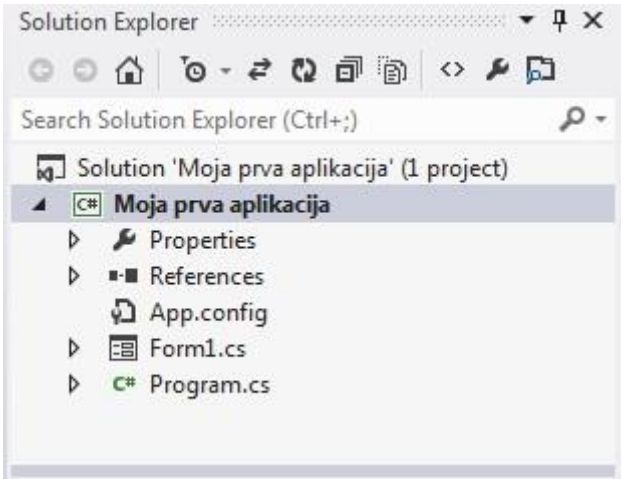
Чување пројекта

Као што смо већ напоменули на почетку, приликом отварања новог пројекта уједно се обавља и чување пројеката. Подсетимо се да смо наш први пројекат назвали *Моја прва апликација* и да смо га сачували у фолдеру под именом *Пројекти* који се налази на десктопу. Када бисмо отворили фолдер *Пројекти* унутар њега бисмо нашли још један фолдер под именом *Моја прва апликација*, отворимо ли и тај фолдер у њему бисмо видели следеће (*Слика 3.9*).



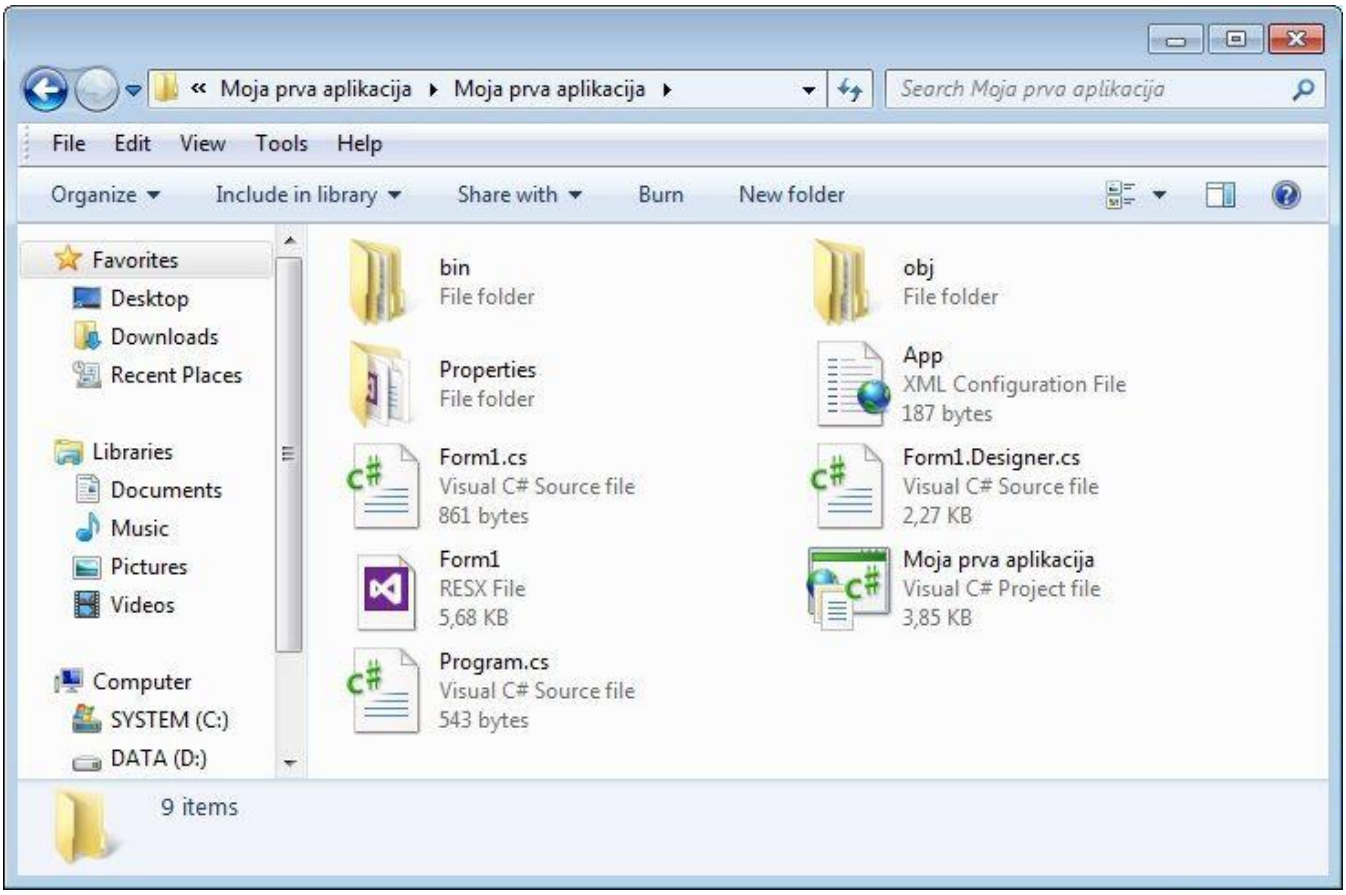
Слика 3.9. Садржај фолдера Пројекти

Дакле, унутра се налази још један фолдер под именом *Моја прва апликација* и један фајл под истим именом. Тај фајл заправо представља цео садржај Solution Explorer-a (*Слика 3.10*).



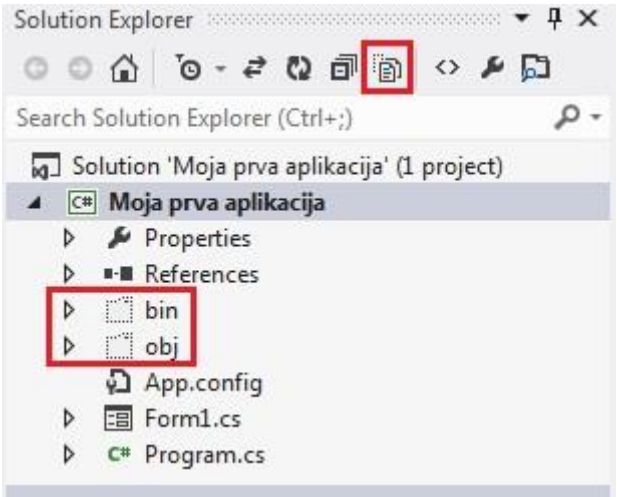
Слика 3.10. Садржај Solution Explorer-a

Што се тиче фолдера *Моја прва апликација*, у њему се налазе још три фолдера и још шест фајлова (*Слика 3.11*).



Слика 3.11. Садржај фолдера Моја прва апликација

Фолдери **bin** и **obj** се могу видети у Solution Explorer-у и то кад се кликне на иконицу **Show all Files** која је уоквирена црвеним на Слици 3.12. Уколико желимо да видимо шта се налази у тим фолдерима кликнемо на знак ▷.



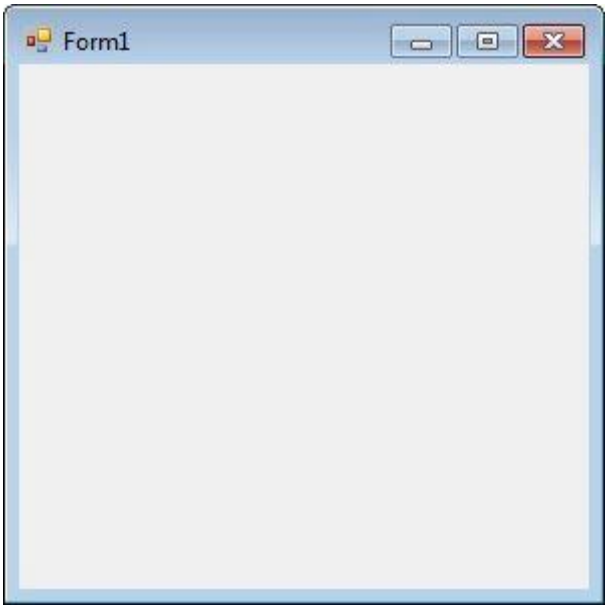
Слика 3.12. Solution Explorer

Дакле, иако још увек ништа нисмо испрограмирали, велики број фолдера и фајлова је направљен и сачуван. То је зато што су многе ствари унапред урађене за нас да би нам олакшале програмирање. Када будемо почели да радимо на пројекту битно је да све измене сачувамо тако што из *File* менија

изаберемо опцију *Save* (или изаберемо исту опцију из траке са алатима или коришћењем пречице *Ctrl+S*).

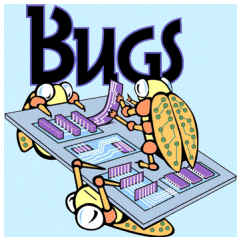
Покретање програма

Пројекат у коме још увек ништа није испрограмирано називамо празан пројекат. Али и у том случају можемо покренути програм. Програм се покреће тако што из *Debug* менија изаберемо опцију *Start Debugging* (или изаберемо исту опцију из траке са алатима, или притиснемо тастер F5 на тастатури). Уколико нема грешака, програм ће се покренути. После покретања програма на екрану ће се појавити празна форма (*Слика 3.13*).



Слика 3.13. Празна форма

На први поглед ова апликација не изгледа ништа посебно, али ако мало боље погледамо она има све функције као и било који прозор. Можемо је померати по екрану, има наслов Form1, иконицу, могућност за повећавање преко целог екрана или за минимизацију и за затварање, што уједно и зауставља извршавање програма. Форма има све ове функције зато што је много ствари за нас већ унапред урађено.

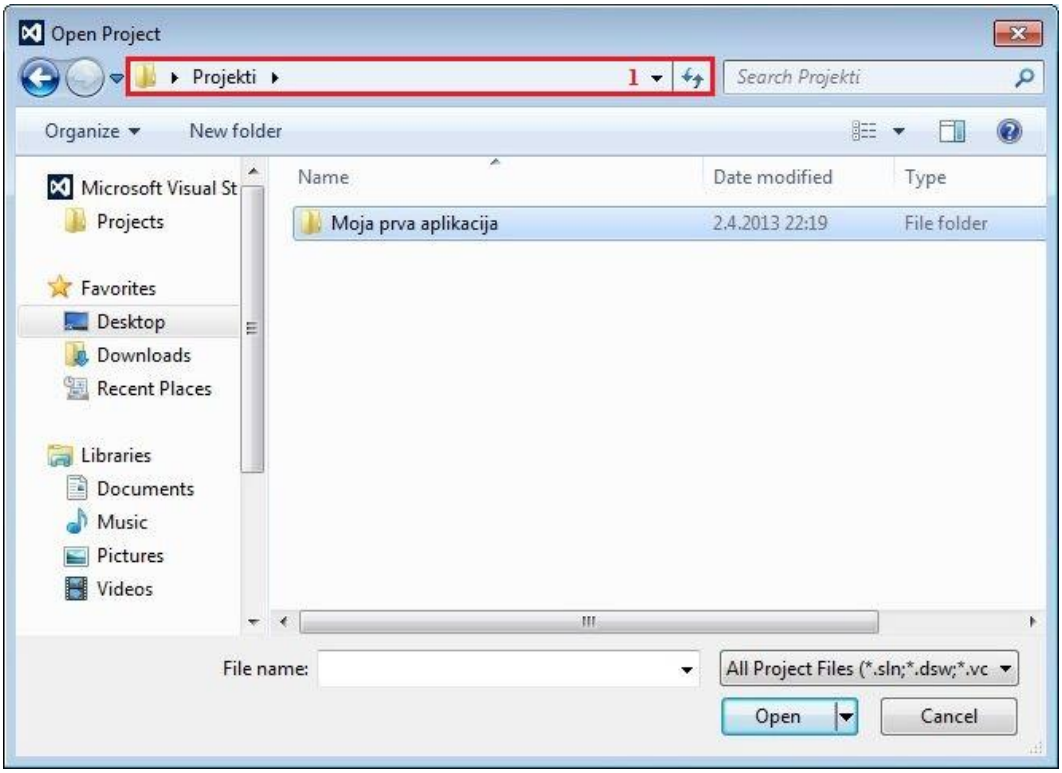


Занимљивост

Опште прихваћен израз за грешку у програмирању је ***bug***, што на енглеском значи буба. Одатле потиче и израз ***Debugging***, што би у слободном преводу значило отклањање буба. Први компјутери су се састојали из вакуумских цеви. Те цеви су се у току рада грејале и при томе емитовале спектар светлости који је привлачио разне инсекте. Једна од дужности особа које су тада одржавали рачунаре је било редовно чишћење вакуумских цеви од инсеката који су се на њих залепили и тиме ометали хлађење рачунара. Баш због овога је оваква терминологија заживела и дан данас се користи.

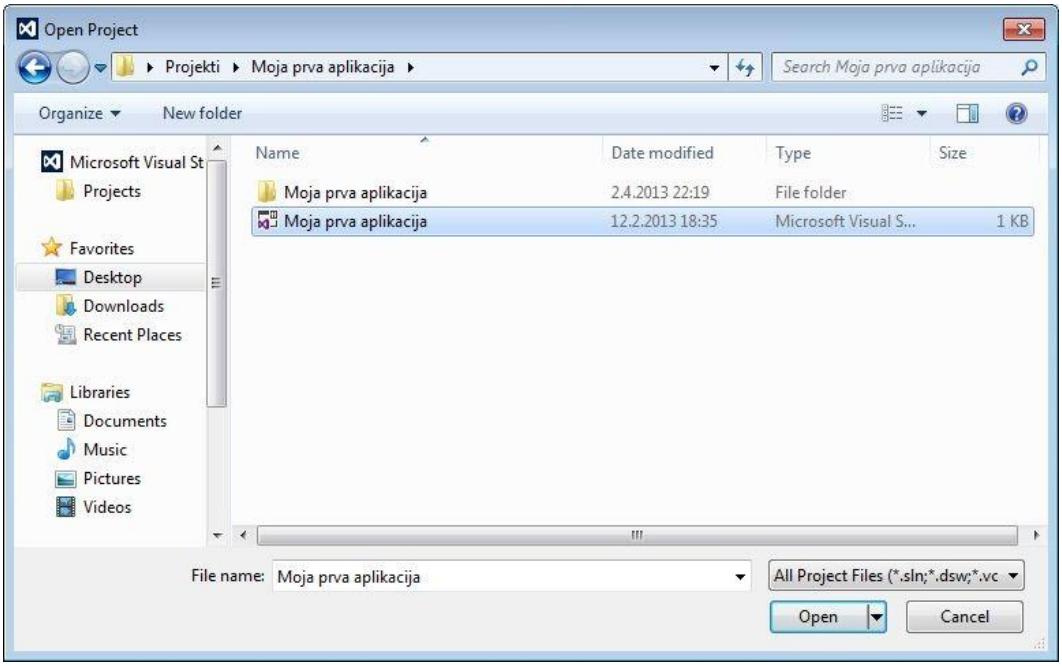
Отварање сачуваних пројеката

Пошто покренемо Visual Studio, из *File* менија изаберемо опцију *Open Project* ( ). Након овога ће се отворити прозор приказан на Слици 3.14.



Слика 3.14. Садржај фолдера Пројекти

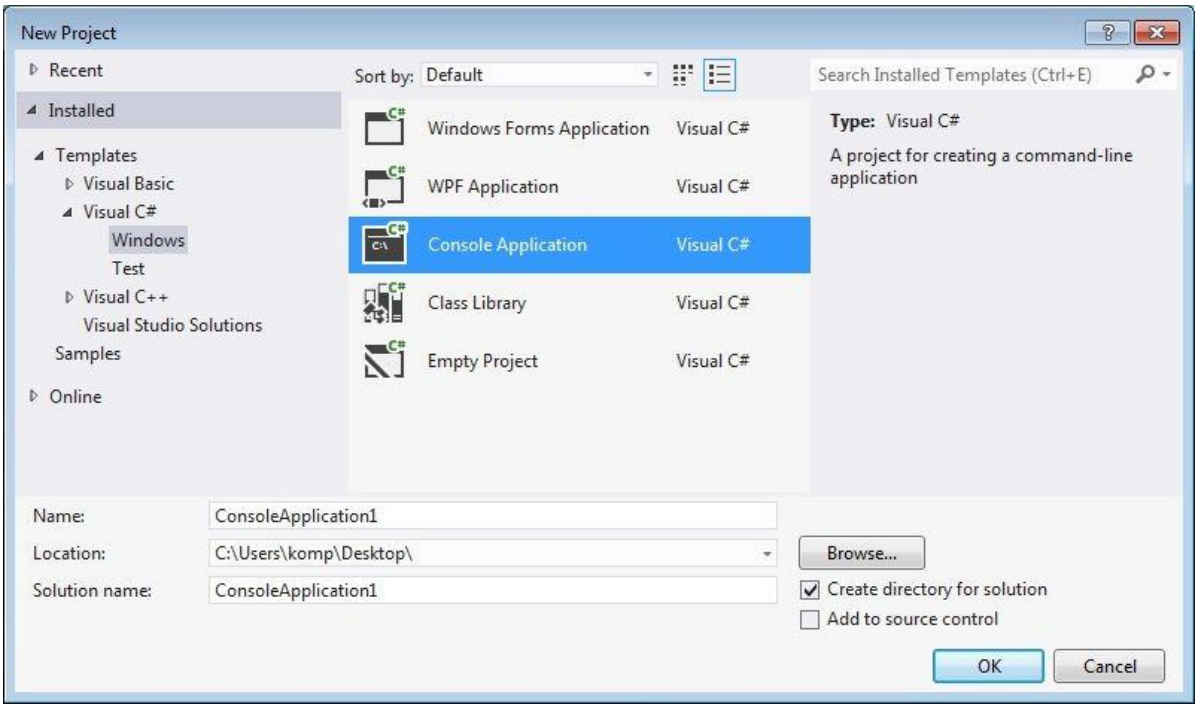
Потребно је да пронађемо фолдер у коме смо сачували апликацију (1) како би се приказао његов садржај, па онда отворимо фолдер *Moja prva aplikacija* и на крају отворимо фајл истог имена и на овај начин ћемо отворити наш пројекат (*Слика 3.15*).



Слика 3.15. Садржај фолдера Moja prva апликација

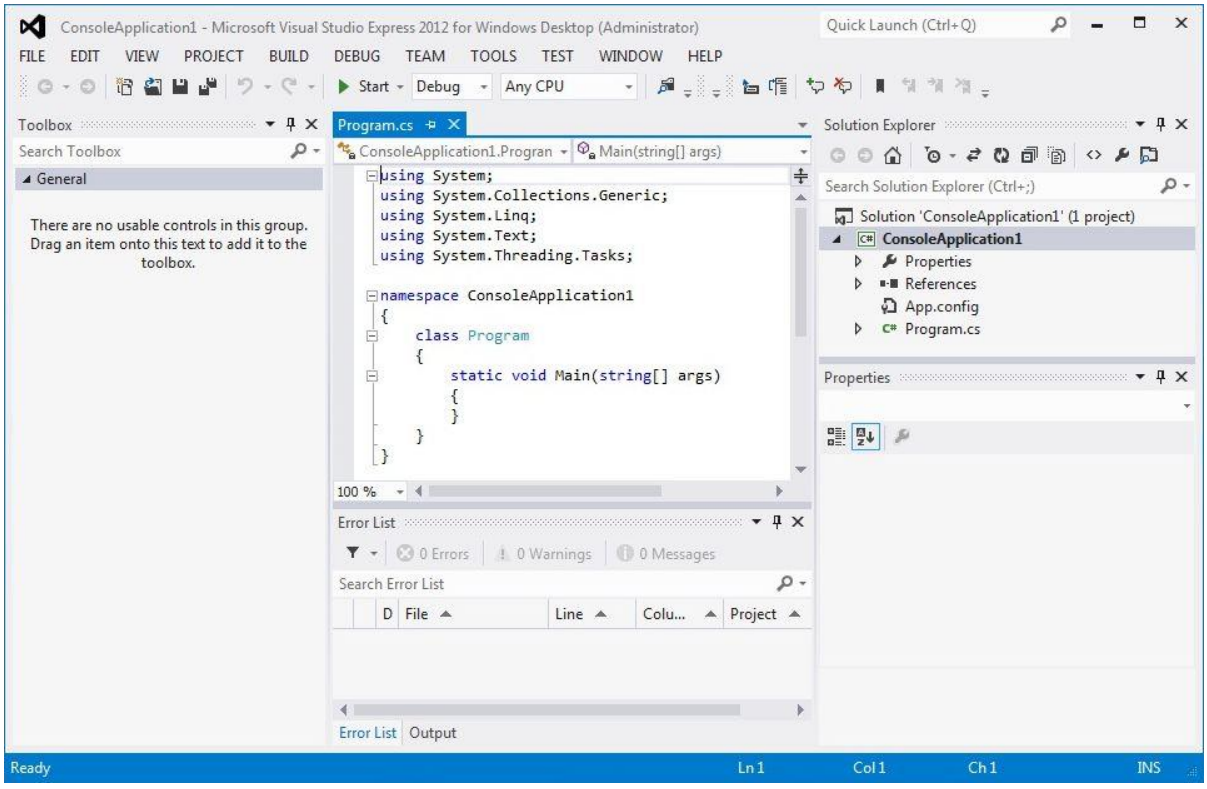
Конзола

Поред апликација заснованих на прозорима, у *C Sharp-y* се могу правити и конзолне апликације тј. *Console Application.* После покретања програма и пошто из *File* менија изаберемо опцију *New Project* на екрану ће се појавити прозор са којим смо се већ срели у претходном делу (*Слика 3.1*).



Слика 3.1. Прозор за отварање новог пројекта

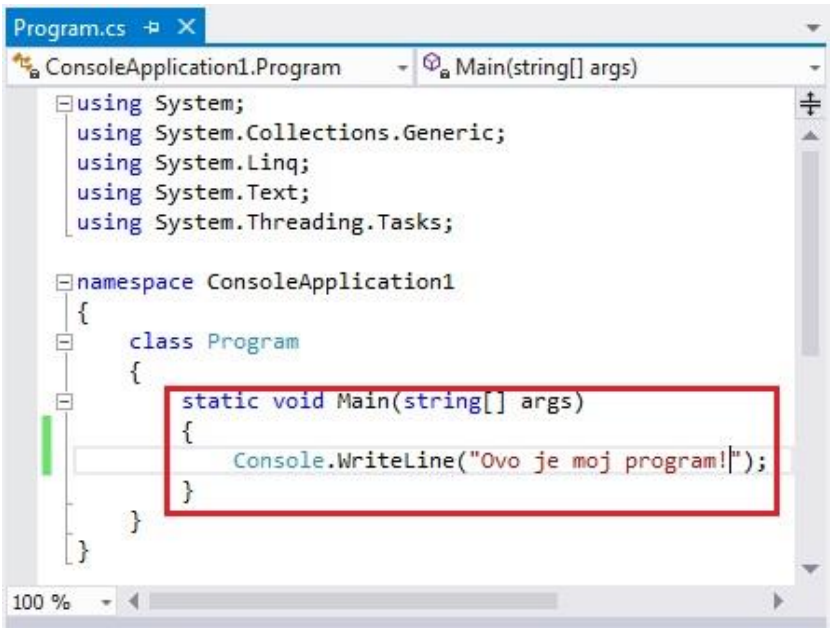
Поступак отварања новог пројекта је исти као и када правимо апликације засноване на прозорима, али уместо *Windows Application*, потребно је селектовати *Console Application* (Слика 3.1). Када именујемо пројекат и одредимо му фолдер у којем ће бити сачуван, за потврду кликнемо на дугме OK. После овога на екрану ће се појавити прозор за рад (*Слика 3.2*).



Слика 3.2. Радна површина

За разлику од *Windows* апликације у којој се у прозору за рад појавила форма, овде се одмах појављује код. То је зато што нећемо имати шта да дизајнирамо већ одмах пишемо код.

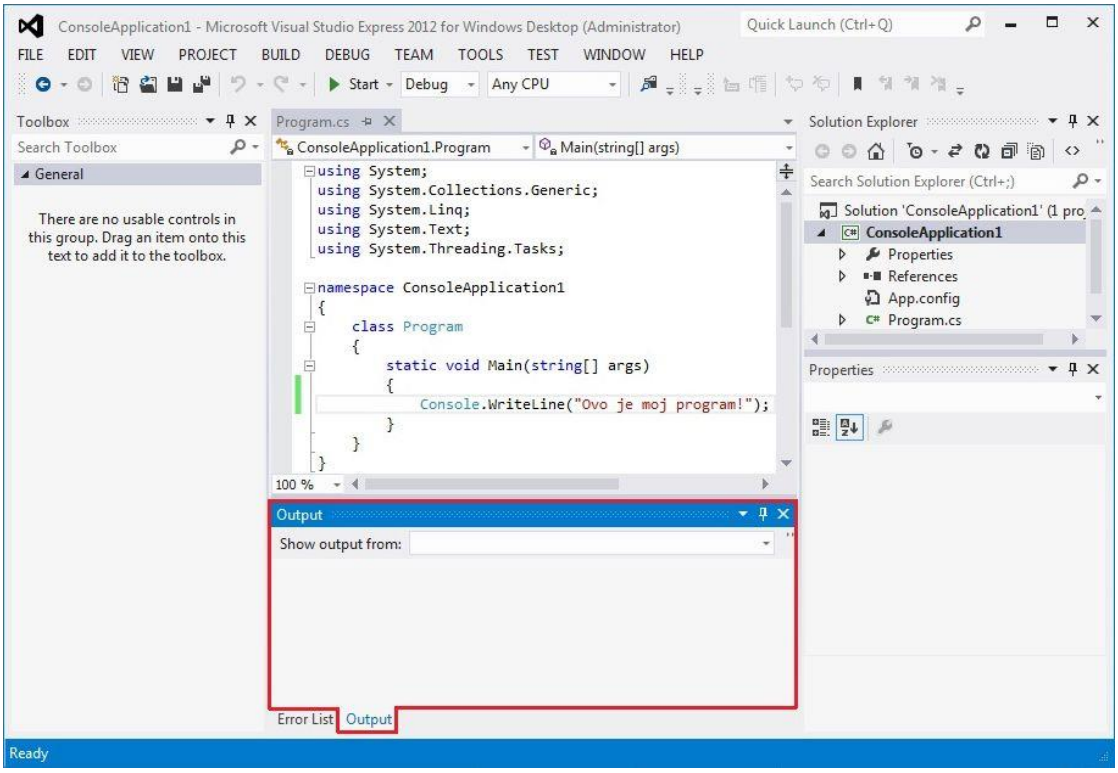
Наш циљ је да испишемо поруку *Ово је мој програм!* која ће се појавити у конзоли. Потребно је написати одговарајући код који ће омогућити исписивање ове поруке. Тај код се пише између заграда уоквиреног дела кода (*Слика 3.3*).



Слика 3.3. Програмски код

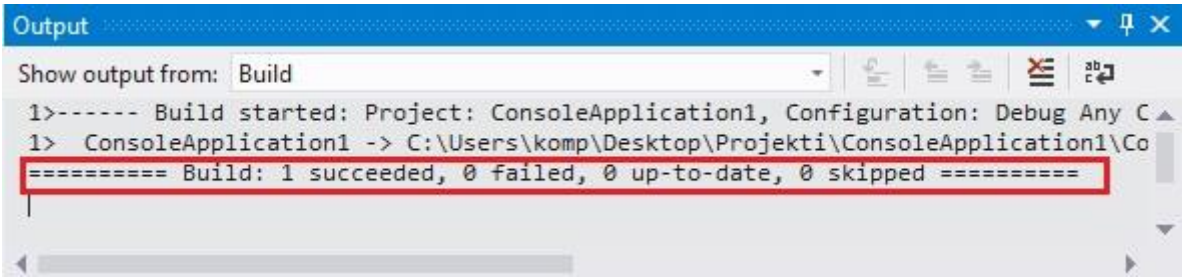
Прво напишемо *Console.* јер се програм односи на конзолу, а затим напишемо наредбу *WriteLine* која служи за исписивање текста у конзоли и на крају у заградама напишемо текст који желимо да се појави у конзоли (*Ово је мој програм*). Најзад наш програм изгледа као на Слици 3.3. Битно је обратити пажњу на то како се која наредба пише јер *C Sharp* разликује мала и велика слова.

Када сачувамо ове измене (из *File* менија изаберемо опцију *Save All*), из *View* менија треба да изаберемо опцију *Output* и на радној површини ће се појавити овај елемент који ће служити за проверу да ли је дошло до неке грешке приликом писања програма (*Слика 3.4*).

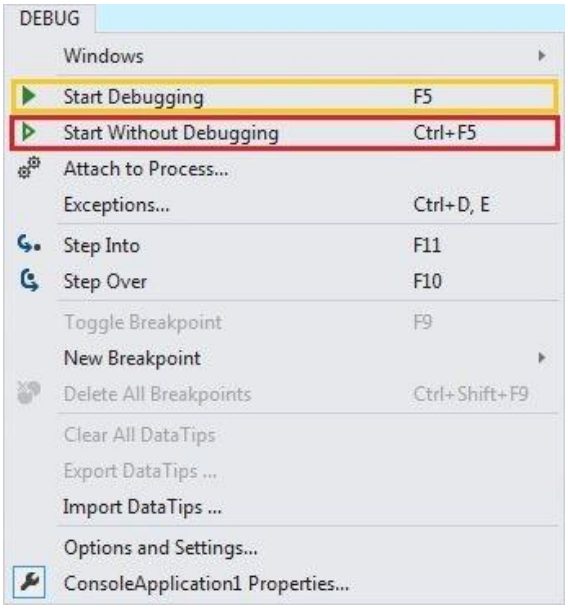


Слика 3.4. Позиција Output-а на радној површини

Из *Build* менија је потребно одабрати опцију *Build Solution* како би се у *Output-y* појавила порука о томе да ли постоје неке грешке. Уколико грешака нема порука у *Output-y* ће изгледати као на што је приказано на Слици 3.5 (ред који је уоквирен црвеном бојом говори о томе да у програму нема никаквих грешака). Након овога можемо да покренемо програм.

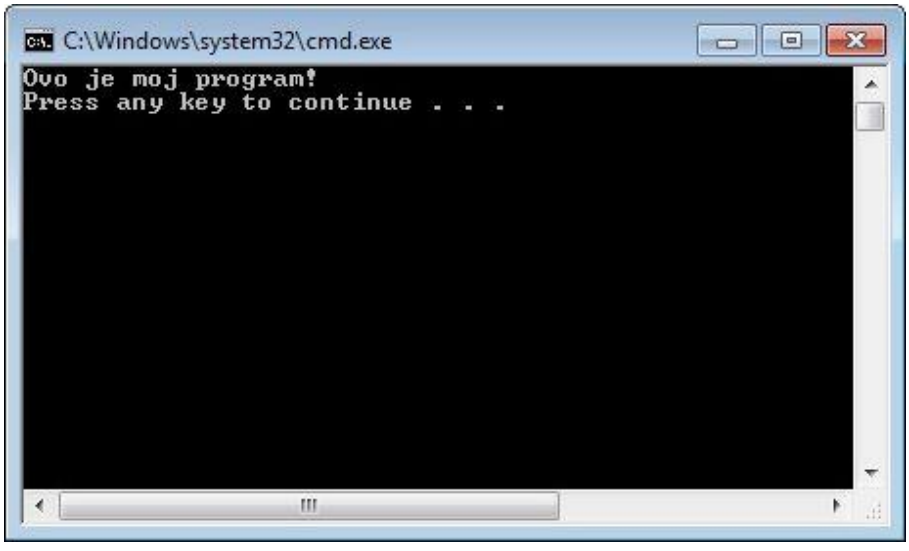


Слика 3.5. Output



Слика 3.6. Debug мени

Програм се покреће тако што из *Debug* менија изаберемо опцију *Start Debugging* (*Слика 3.6*). Међутим, ако програм покренемо на овај начин, конзола ће се појавити само на кратко на екрану и одмах ће се затворити, тако да нећемо моћи лепо да видимо оно што је исписано у нашој конзоли. Због овога ћемо из *Debug* менија да изаберемо опцију *Start Without Debugging* (*Слика 3.6*). На овај начин ће се на екрану појавити конзола и остаће приказана на њему док је корисник не затвори (*Слика 3.7*).



Слика 3.7. Конзола

Видимо да је у првом реду исписана наша порука, а у другом реду је исписана порука *Press any key to continue...* која нас обавештава да уколико хоћемо да наставимо са извршавањем програма треба да притиснемо било који тастер на тастатури. У овом случају ако то урадимо наш програм ће се затворити.

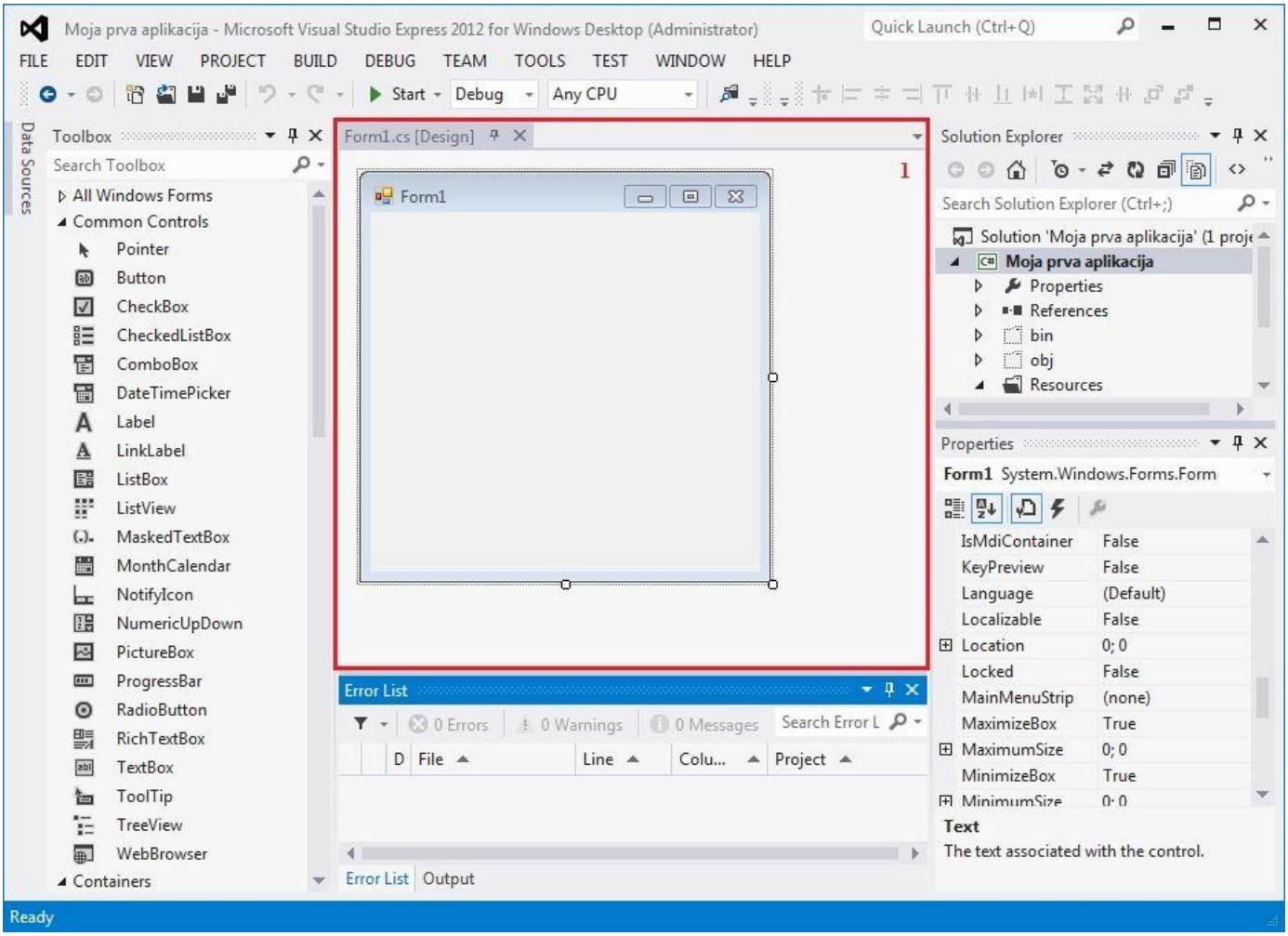
Пре су сви програми изгледали овако, односно изгледали су овако док се није развио графички кориснички интерфејс. Међутим, сада можемо да направимо веома занимљиве и атрактивне апликације. У овом курсу ћемо се са конзолним апликацијама срести још на само неколико места. У даљем раду ћемо се концентрисати на израду *Windows* апликација.

Форма и њена својства

Приликом креирања апликација разликујемо два битна дела:

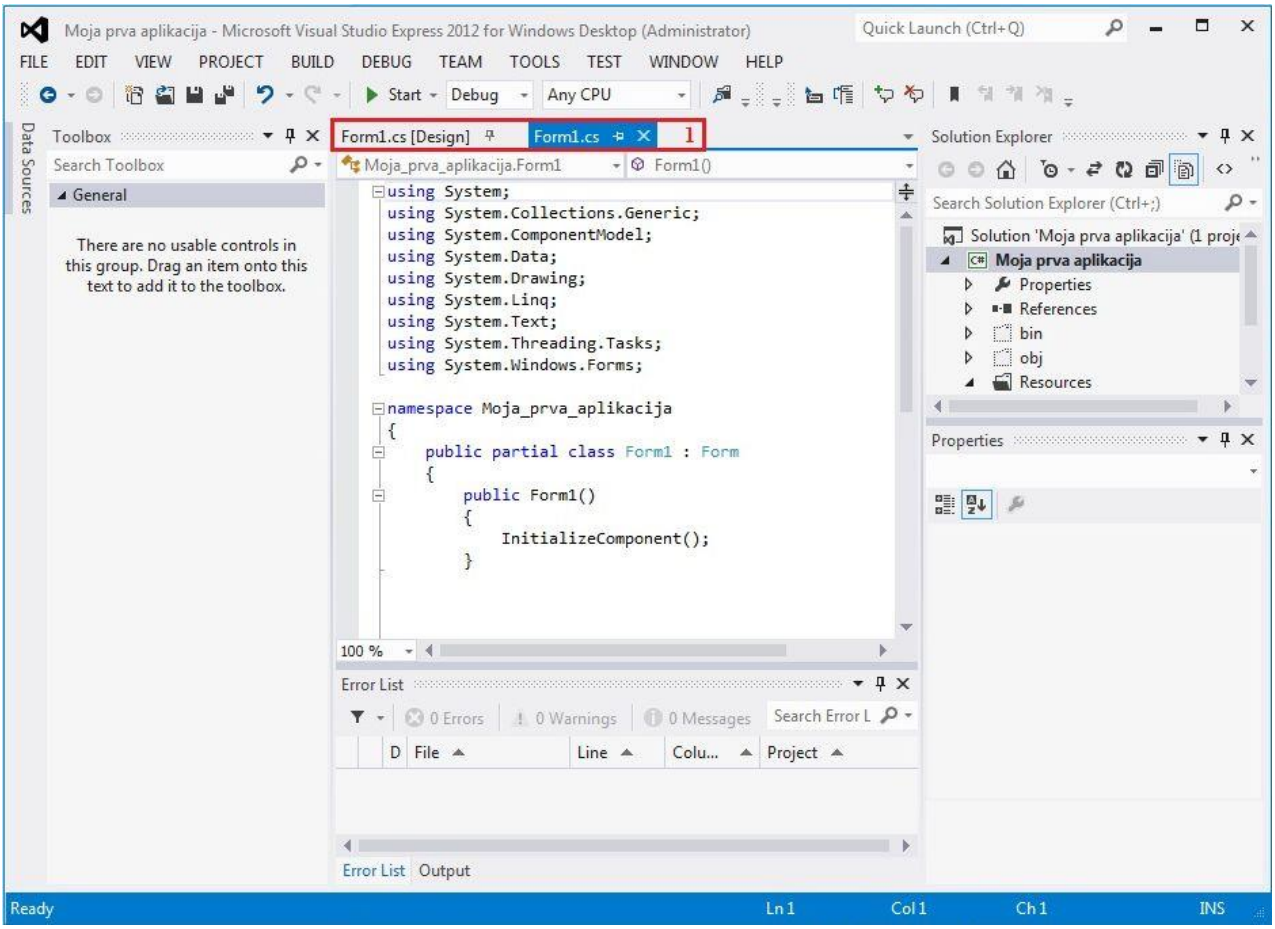
1. Дизајнирање форме
2. Писање кода.

Дизајнирање се врши на самој форми која се налази у централном делу прозора за рад (*Слика 3.1*). Форми ћемо додавати разне компоненте и подешаваћемо својства, како форме, тако и компоненти које ће се налазити на форми.



Слика 3.1. Позиција форме на радној површини

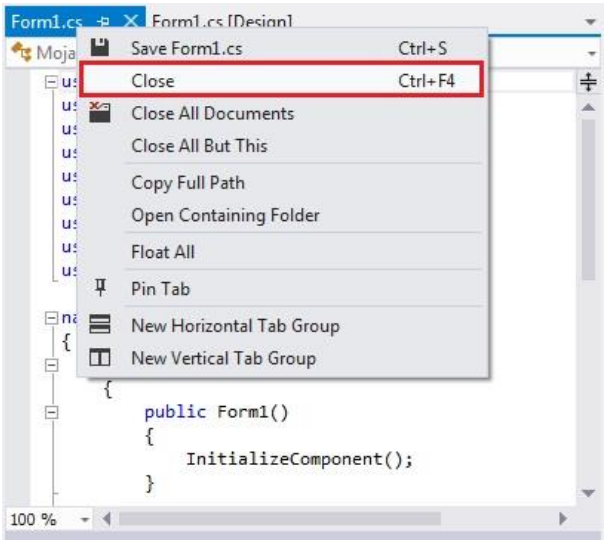
Што се тиче писања кода, прво морамо да из *View* менија изаберемо опцију *Code*. Приметимо да се сада у централном делу прозора за рад налази код (*Слика 3.2*). Постоје одређени делови кода који су већ унапред написани, као што је овај код који тренутно видимо, а за оно што нам додатно буде било потребно сами ћемо писати код.



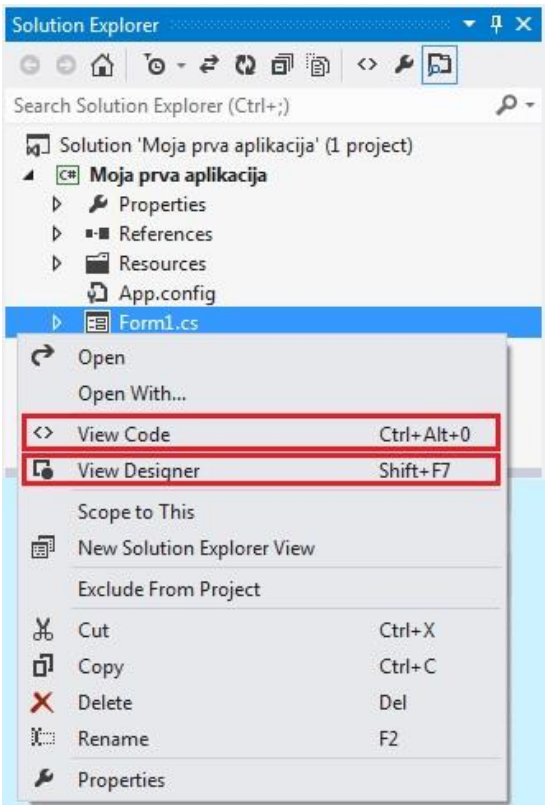
Слика 3.2. Позиција кода на радној површини

Обратимо пажњу на две "картице" које се налазе при врху централног дела прозора (1). То су: *Form1.cs [Design]* и *Form1.cs.* Тренутно је на Слици 3.2 приказана картица *Form1.cs* ,што значи да та картица приказује код нашег програма. Када бисмо кликнули на картицу *Form1.cs [Design]*, онда би се поново приказала форма као што је на Слици 3.1.

Затварање било које од ових картица можемо извршити одабиром опције *Close* из падајућег менија, који добијамо када кликнемо десним тастером миша на картицу коју желимо да затворимо (*Слика 3.3*). Уколико случајно затворимо приказ кода или форме на радној површини, можемо га поново отворити тако што ћемо из *View* менија изабрати опције за укључивање приказа кода (*Code*) или приказа форме на радној површини (*Designer*) као што је већ раније било објашњено, или десним кликом на *Form1.cs* који се налази у *Solution Explorer-y* и из падајућег менија изаберемо опције *View Code* или *View Designer* (*Слика 3.4*).



Слика 3.3. Падајући мени



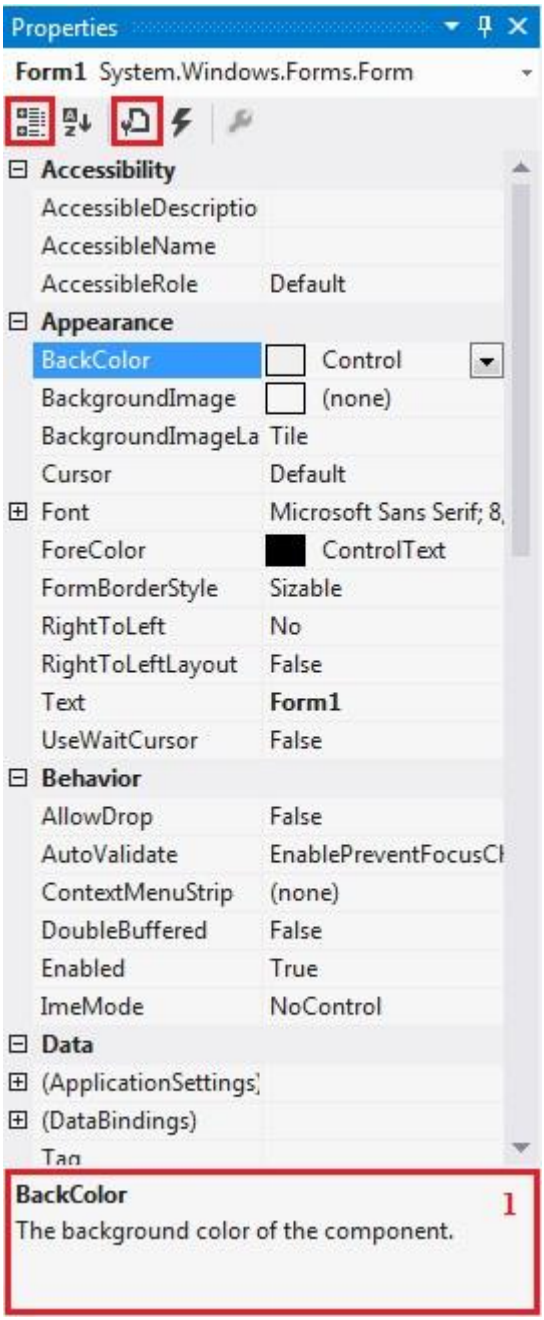
Слика 3.4. Опције Solution Explorer-a

Напомена: Приликом креирања апликације често ћемо прелазити са једног приказа на други што се постиже једноставним кликом на картицу која нам је у том тренутку потребна.

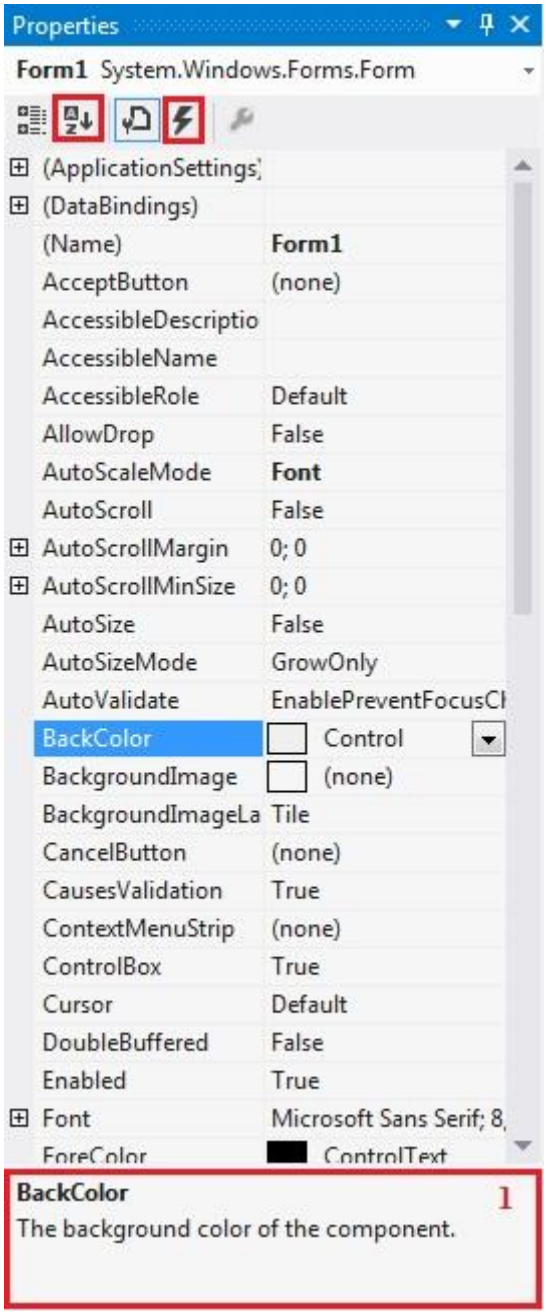
Рекли смо да се свака апликација састоји од најмање једне форме. Неке сложене апликације могу да се састоје од више форми, али ми ћемо за сада креирати апликације које у себи садрже једну форму. Изглед форме може да се мења. Форма има различита својства која можемо мењати па ћемо приказати нека од својстава која се често користе.

Својства форме

Да бисмо видели сва својства која су нам понуђена у *Properties Widows-y* прво морамо да кликнемо на форму, тј. форма мора да буде селектована. Уколико кликнемо на прву по реду иконицу , сва својства ће бити организована у разне категорије као што је приказано на Слици 3.5, а уколико желимо да сва својства буду организована по абецедном реду, како је приказано на Слици 3.6, онда кликнемо на другу по реду иконицу . Трећа по реду иконица служи за приказ свих својстава, а четврта за приказ листе догађаја о којима ће бити речи касније. У самом дну *Properties Widows-a* је део у коме се, када се кликне на неко својство, исписује додатно објашњење о том својству. На Слици 3.5 је селектовано својство *Back Color* (обележено плавом бојом), а објашњење које се ипписује (1) говори о томе да то својство служи за промену боје позадине компоненте, која је у нашем случају форма.



Слика 3.5. Својства организована по категоријама



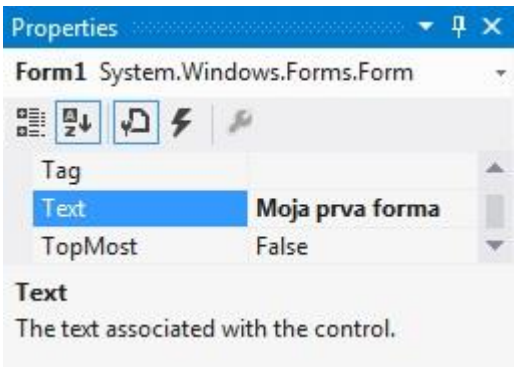
Слика 3.6. Својства организована по абецедном реду

Приметимо да се испред неких својстава налази симбол ⊞, који говори о томе да се нека одређена својства састоје од подсвојстава која се могу подешавати. Кликом на симбол ⊞приказују се сва подсвојства која припадају том својству, а симбол ⊞прелази у симбол ⊟. На пример својство ***Size*** се састоји од два подсвојства: ***Width*** (ширина) и ***Height*** (висина). Кликом на симбол ⊟затварају се сва подсвојства која припадају том својству и симбол ⊟прелази у симбол ⊞.

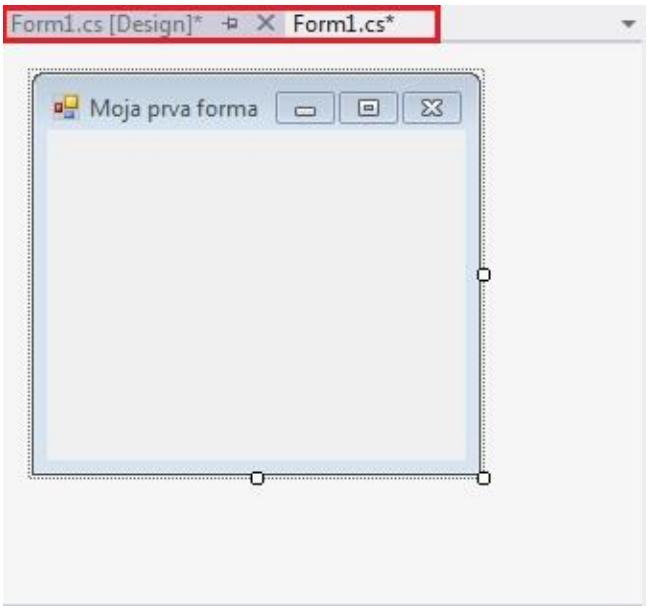
Својства која се најчешће користе:

Text је својство које се најчешће користи и служи за задавање имена компоненте или форме у складу са тим шта та компонента (или форма) треба да представља. Уколико правимо дигитрон, онда би наша форма добила име *Digitron* на пример. На почетку као текст стоји ***Form1***, а ми можемо да то променимо тако што једноставно избришемо то што је написано и напишемо оно што желимо и притиснемо тастер *Enter* на тастатури. Можемо на пример да напишемо ***Moja prva forma*** (*Слика 3.7*). Ако сада погледамо на форму видећемо да поред иконице више не пише *Form1*, већ *Moja prva forma* (*Слика 3.8*).

Напомена: Приметимо такође да се после ове измене у десном горњем углу картица *Form1.cs [Design]* и *Form1.cs* појавила по једна звездица (*Слика 3.8*). То значи да постоје несачуване измене у нашем пројекту и те звездице ће се пјављивати кад год будемо нешто мењали у пројекту, без обзира да ли се те измене врше на форми или у коду. Када будемо сачували измене, *Save* опција из *View* менија, звездице ће нестати.



Слика 3.7. Својство Text

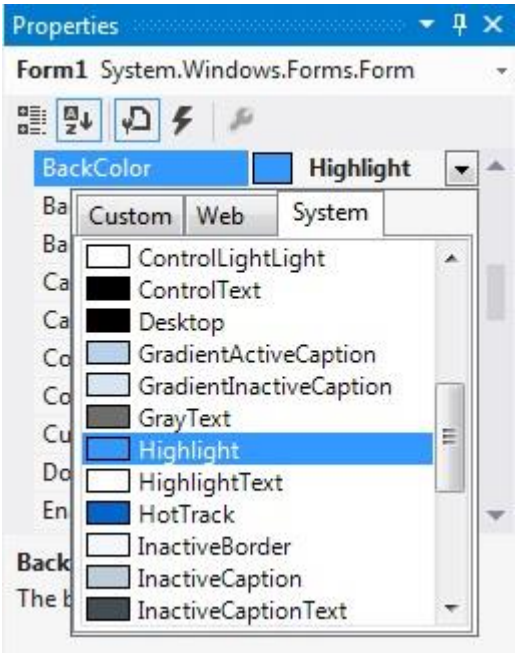


Слика 3.8. Приказ наслова форме

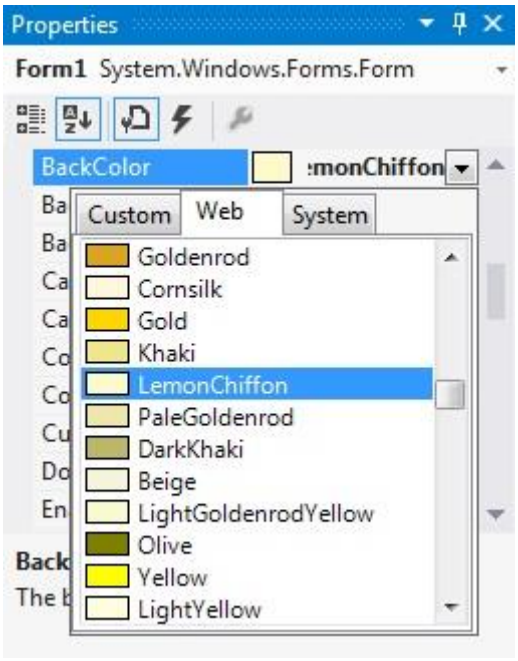
(Name) је својство које представља име форме које се користи у коду како би се идентификовала форма (тј. објекат). Форма аутоматски добија име Form1. Ово име се може променити помоћу овог својства, али ми ћемо радити са овим већ унапред задатим именом и нећемо га мењати. Ово својство треба разликовати од својства ***Text.***

Back Color је својство које служи за промену боје позадине форме. Када кликнемо на то својство, у десном углу ће се пјавити стрелица. Када кликнемо на ту стрелицу појавиће се падајући мени у коме су приказане три картице: *Custom*, *Web* и *System*. У *Web* и *System* картицама су приказане листе боја које можемо да користимо, (*Слика 3.9* и *3.10*), а у *Custom* картици је приказана палета боја које су нам на располагању (*Слика 3.11*). Можемо да изаберемо боју коју хоћемо и да видимо како она изгледа на

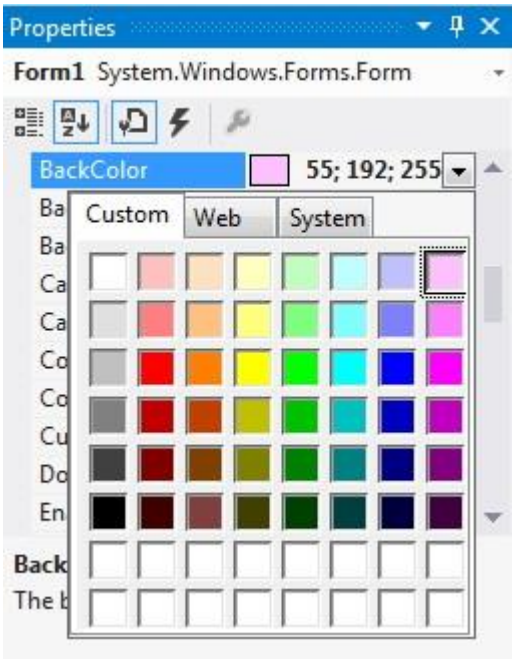
.



Слика 3.9. System картица

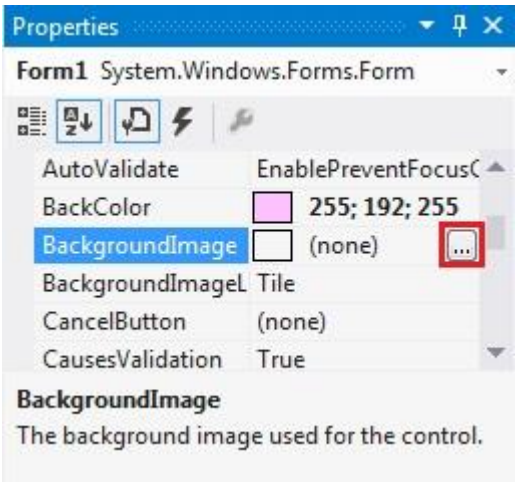


Слика 3.10. Web картица

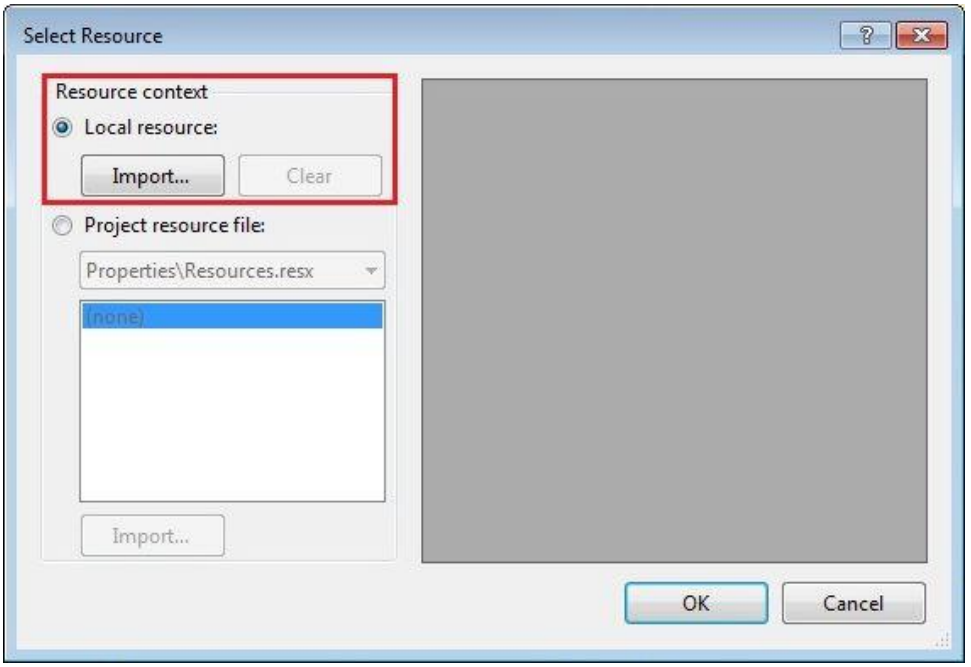


Слика 3.11. Custom картица

Background Image је својство које служи за постављање слике на позадину форме. Када кликнемо на то својство, у десном углу ће се појавити дугме са три тачке (*Слика 3.12*). Када кликнемо на то дугме појавиће се нови прозор који служи за импортовање слике (*Слика 3.13*).

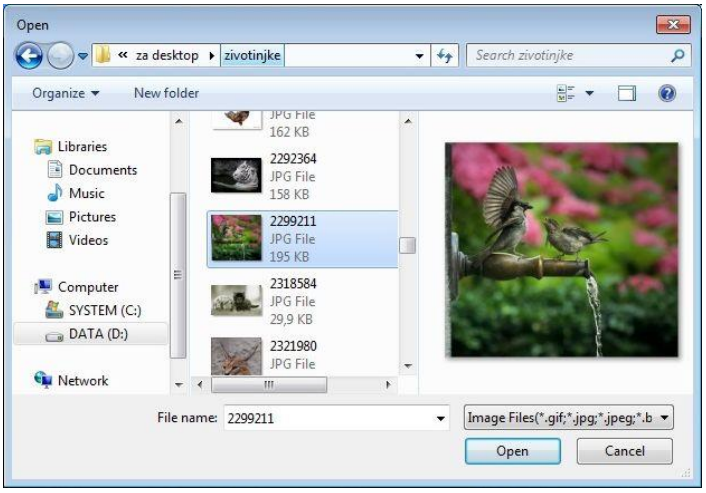


Слика 3.12. Background Image својство

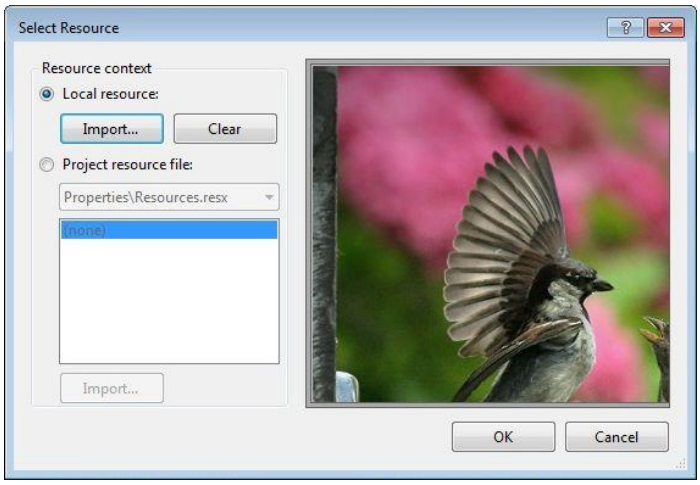


Слика 3.13. Изглед прозора за импортовање слика

Да бисмо могли да импортујемо слику, потребно је да селектујемо *Local resource*, а затим да кликнемо на дугме *Import* како је приказано на Слици 3.13. После овога ће се отворити *Windows Explorer* у коме треба да нађемо жељену слику (*Слика 3.14*). Када пронађемо жељену слику, селектујемо је и кликнемо на дугме *Open* и слика ће се појавити у прозору за импортовање (*Слика 3.15*).

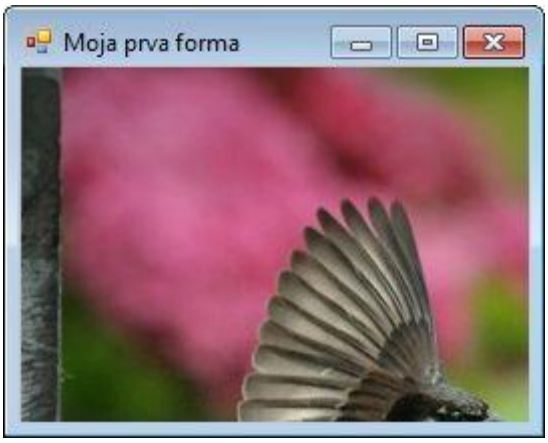


Слика 3.14. Фолдер са сликама



Слика 3.15. Импортовање слике

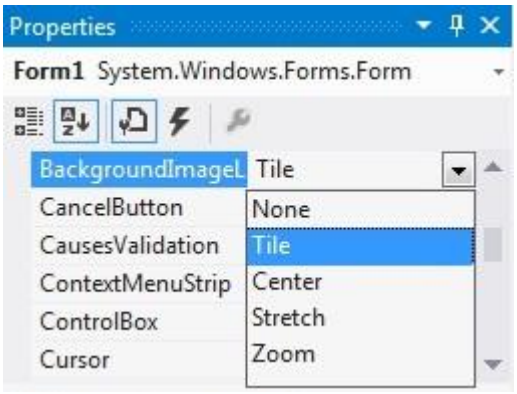
Међутим, на форми се слика неће видети у целости већ ће бити само један део као што је приказан један део на Слици 3.15. Следеће својство може да регулише овај проблем.



Background Image Layout је својство у коме су понуђени начини на које слика може да попуни позадину форме.

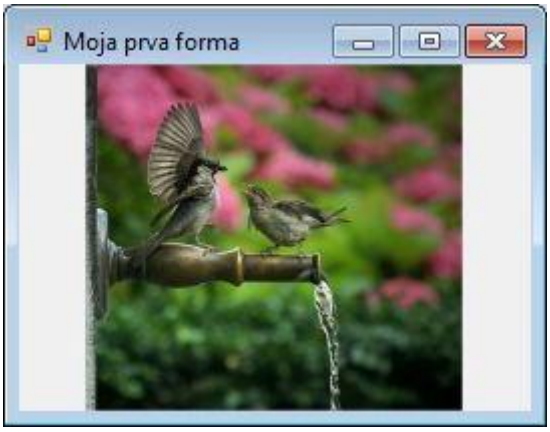
Ако кликнемо на ово својство са десне стране ће се појавити стрелица. Када кликнемо на ту стрелицу појавиће се листа понуђених опција (*Слика 3.16*). Понуђена су четири начина: *Tile*, *Center*, *Stretch* и *Zoom*. Најчешће се користи *Stretch* који развлачи слику преко целе форме и *Zoom* који прилагођава слику димензијама

.



Слика 3.16. Background Image Layout својство

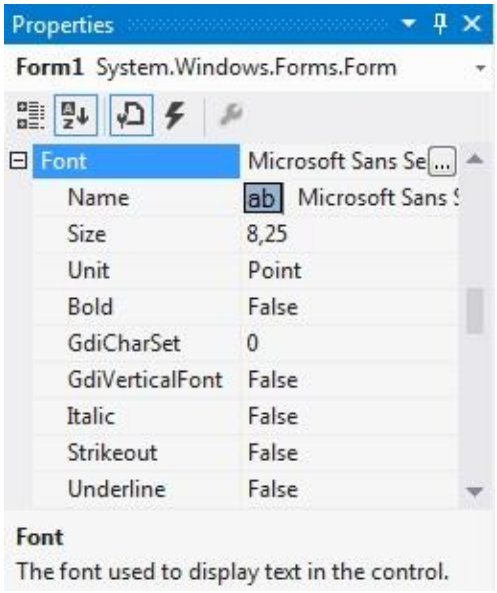
Stretch



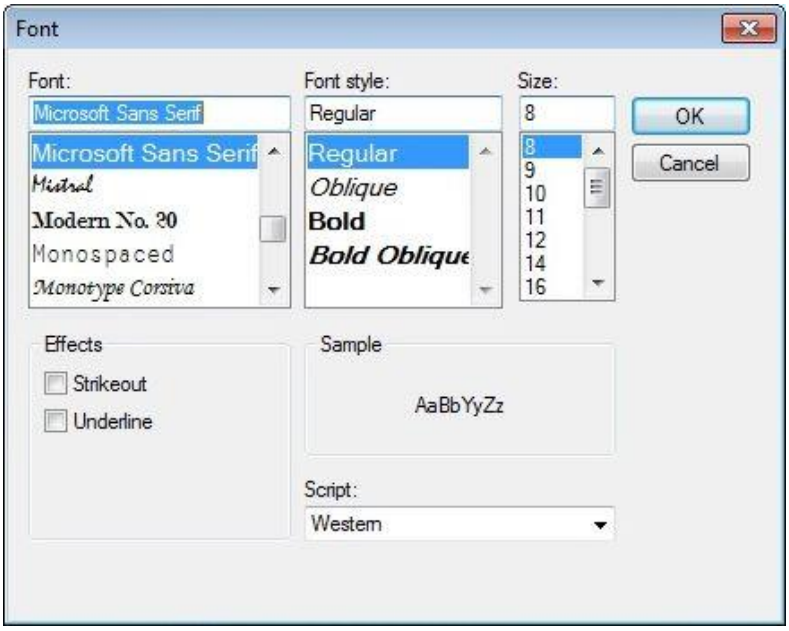
Zoom

Cursor је својство које нуди различите изгледе курсора. Када пређемо мишем преко форме, курсор ће попримити изабрани изглед.

Font је својство које служи за уређивање текста који се исписује на компонентама које се додају форми. Састоји се од подсвојстава која се могу видети када се кликне на симбол ⊞. Та подсвојства се директно могу мењати у оквиру *Properties Windows-a* (*Слика 3.17*). Ако кликнемо на својство ***Font*** у углу ће се појавити дугме са три тачке, када кликнемо на то дугме појавиће се нови прозор у коме су исто приказана подсвојства, тако да је ово још један начин на који се подсвојства могу подешавати (*Слика 3.18*).

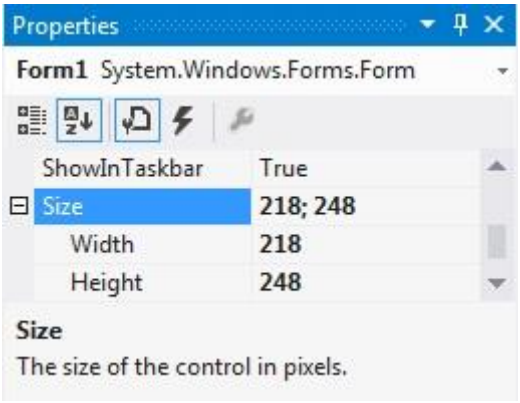


Слика 3.17. Font својство



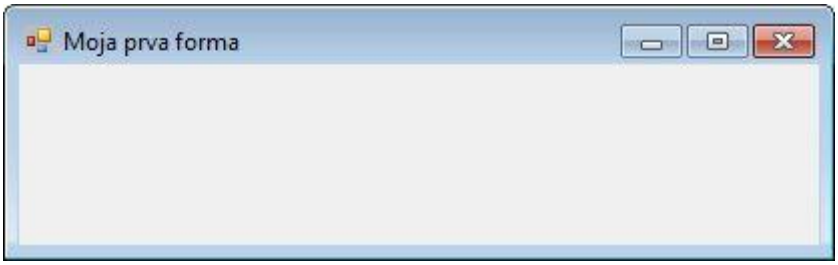
Слика 3.18. Опције Font својства

Size је својство помоћу кога се одређују димензије форме изражене у пикселима. Састоји се од два подсвојства: *Width*, које служи за подешавање ширине и *Height*, које служи за подешавање висине (*Слика 3.19*).



Слика 3.19. Size својство

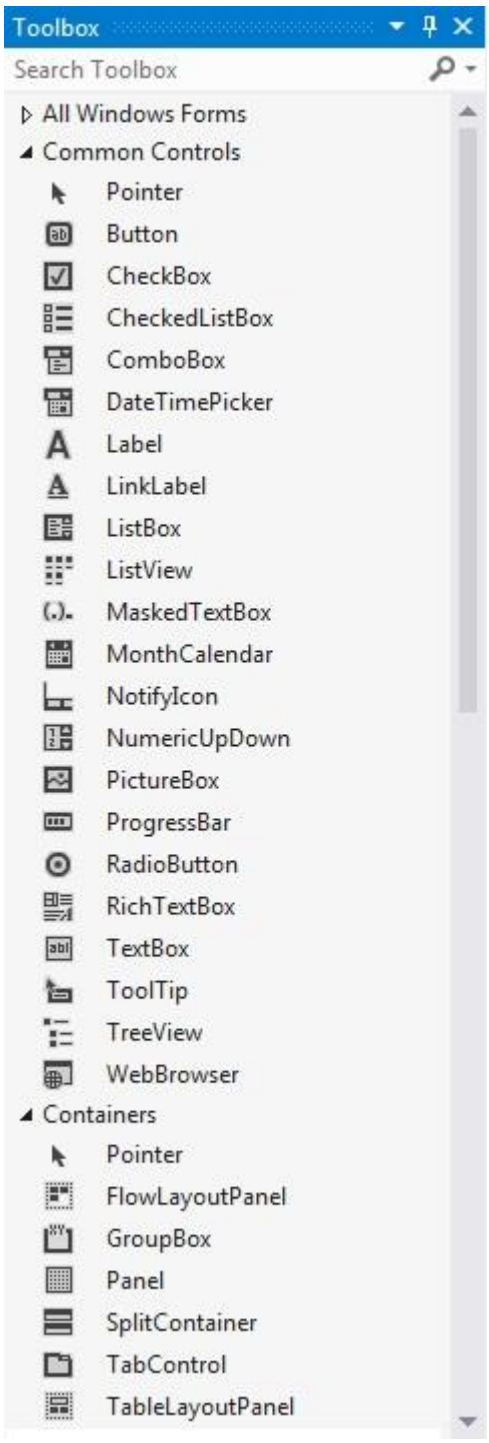
У овом случају смо променили и висину и ширину. Сада је ширина **415**, а висина је **128**.



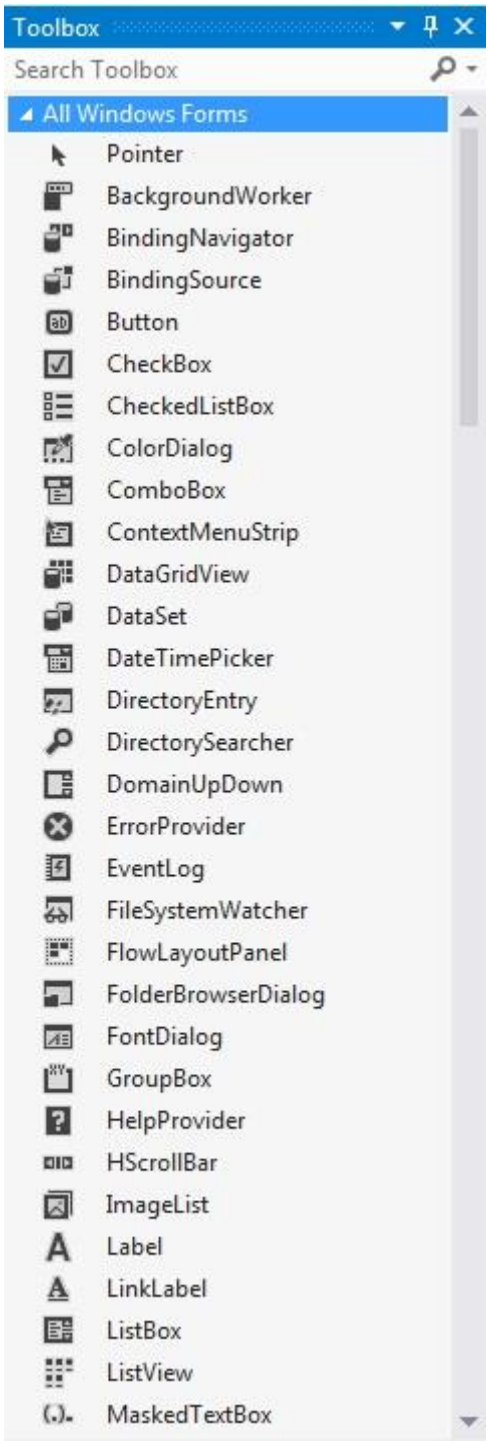
Компоненте и њихова својства

Додавање компоненти форми

Компоненте које се додају форми налазе се у *Toolbox-y* и оне су груписане по разним категоријама (*Слика 3.1*). Тих категорија има осам и то су: *Common Controls*, *Containers*, *Menus & Toolbars*, *Data*, *Components*, *Printing*, *Dialogs* и *Crystal Reports*. Такође, уколико кликнемо на симбол , који се налази испред категорије *All Windows Forms*, приказаће се све компоненте сортиране по абецедном реду (*Слика 3.2*).



Слика 3.1. Својства организована по категоријама



Слика 3.2. Својства организована по абецедном реду

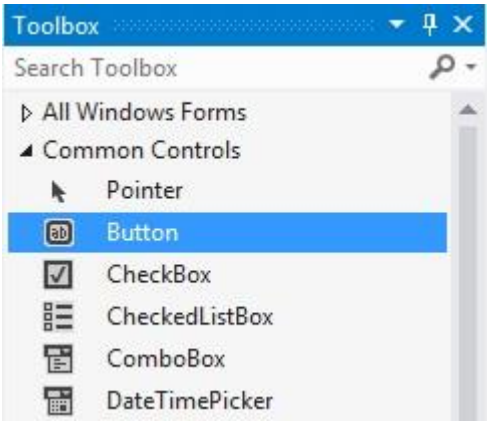
Ми нећемо користити све понуђене компоненте. За почетак ће бити приказане следеће компоненте:

1. *Button* (дугме)
2. *Label* (натпис)
3. *TextBox* (оквир за уношење и приказивање текста)
4. *RichTextBox* (оквир за уношење и приказивање текста)
5. *PictureBox* (оквир за приказивање слика)
6. *Timer* (тајмер).

Додавање компоненти се врши тако што се у *Toolbox-у* кликне на жељену компоненту, а затим се кликне било где на форму. Такође се додавање компоненте форми може извршити тако што два пута кликнемо жељену компоненту у *Toolbox-у* и компонента ће се одмах појавити на форми. Компоненту

можемо померати по форми где хоћемо тако што кликнемо на њу и померимо је (држимо стиснути леви тастер на мишу све време док померамо компоненту по форми). Свака компонента се додаје на исти начин. На форми ће се налазити више различитих компоненти, а које ће то компоненте бити зависи од тога какву апликацију правимо.

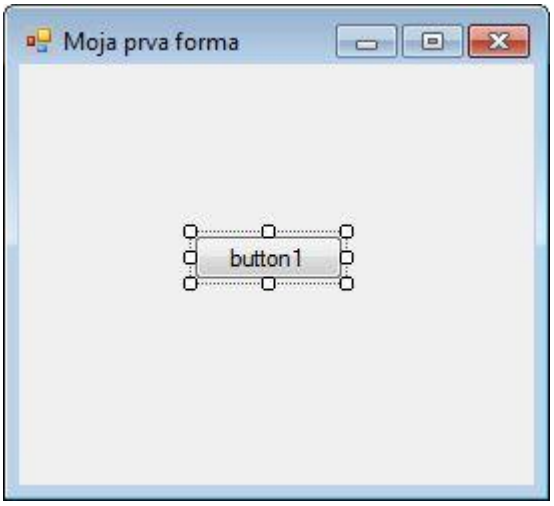
Приказ додавања компоненте *Button* форми



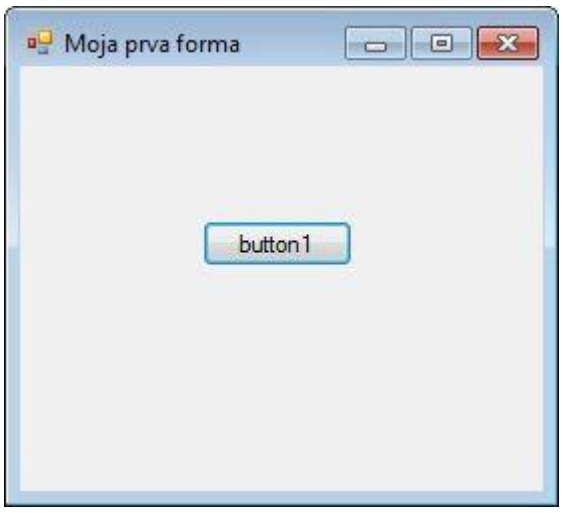
Слика 3.3. Позиција дугмета у Toolbox-у

Као што смо рекли, први корак у поступку додавања компоненте форми је да изаберемо компоненту из *Toolbox-а*, у нашем случају то је компонента *Button*, односно дугме (*Слика 3.3*).

Када се компонента појави на форми, позиционирамо је на место које хоћемо, на пример на сам центар форме, тако да у нашем примеру форма изгледа као што је приказано на Слици 3.4. Дугме се зове *button 1*, али његово име можемо променити, тј дугме можемо преименовати у зависности од тога каква ће акција бити извршена када се кликне на њега, и можемо му променити изглед (на пример величину, боју и тако даље).



Слика 3.4. Изглед дугмета на форми

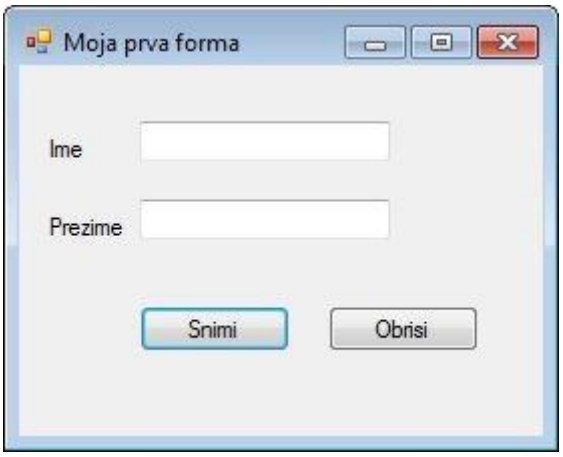


Слика 3.5. Изглед дугмета након покретања програма

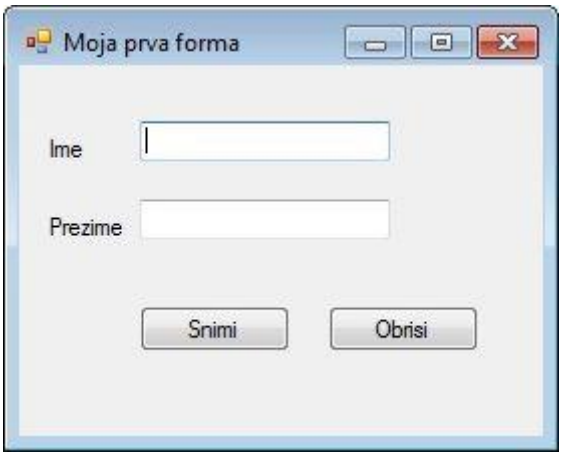
Када сачувамо измене, можемо и да покренемо програм. Можемо да кликнемо на дугме *button 1*, али ништа се неће десити јер нисмо написали никакав код који одговара некој акцији.

Компонента у жижи

У апликацијама се најчешће појављује више компоненти, али сваког тренутка само једна компонента може бити активна и за ту компоненту се каже да је у ***жижи*** (тј. у ***фокусу***). Компонента која је у жижи се разликује од других компоненти исте врсте. На Слици 3.6 дугме *Snimi* је у жижи и то се препознаје по томе што је уоквирено плавом бојом, а на Слици 3.7 у жижи је прво по реду поље за унос текст и разликује се од другог поља за унос текста по томе што је уоквирено плавом линијом и по курсору који је постављен на почетак поља, што значи да је све спремно за унос текста са тастатуре.



Слика 3.6. Компонента дугме у жижи



Слика 3.7. Поље за унос текста у жижи

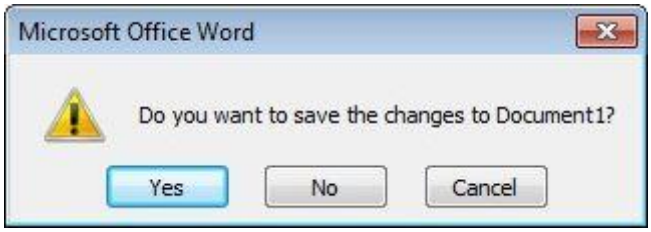
Притиском на неку компоненту жижа се помера на ту компонету. Жижа се може померати са једне компоненте на другу и тако што се притиска тастер ***Tab*** на тастатури, односно комбинација тастера ***Shift*** и ***Tab*** уколико желимо да жижу вратимо на претходну компоненту. Редослед померања жиже са компоненте на компоненту коришћењем ***Tab*** тастера, зависи од тога по ком редоследу смо додавали компоненте форми. Значи, ово је још једна ствар о којој треба да водимо рачуна приликом креирања апликације јер редослед померања жиже треба да следи логичан ток уношења података. Обично су компоненте распоређене тако да редослед померања жиже буде слева удесно и одозго надоле. На пример на Слици 3.7 редослед померања жиже теба да буде са првог поља за унос текст на друго поље за унос текста, па на дугме *Snimi* и на крају на дугме *Obrisi*, јер прво уносимо име па презиме и онда снимимо тај унос, а после тога бришемо тренутни унос како бисмо могли да унесемо нове податке (ово је логичи ток уношења података).

Свака компонента има различита својства која можемо подешавати. Компонента мора бити селектована да би се у *Properties Windows-y* појавила својства те компоненте. Својства компоненти се мењају на исти начин као што се мењају својства форме.

Компонента Button

Button је компонента која се у *Toolbox-y* налази у категорији *Common Controls*. Кликом на ову компоненту се извршава нека акција. Са овом компонентом смо се већ сусрели много пута приликом рада на компјутеру. Када завршимо са писањем неког документа (у *Word-y* на пример) и када хоћемо да га затворимо, а да га пре тога нисмо сачували, појавиће се порозор на екрану у коме је исписано питање *Да ли желите да сачувате измене у документу?* и три дугмета: Yes, No и Cancel (*Слика 3.1*). Кликом на свако дугме извршава се одговарајућа акција.

Такође је и дигитрон добар пример. Сви бројеви и све понуђене опције, као што су сабирање, одузимање, множење и друге, су дугмићи на које се кликом извршава одговарајућа акција (*Слика 3.2*). Ако кликнемо на дугме на коме је написан број 9, акција која ће се извршити је исписивање броја 9. Ако кликнемо на дугме на коме стоји знак плус, акција која ће се извршити је сабирање.



Слика 3.1. Примена компоненте Button

Ако боље погледамо, видећемо да смо се сусрели са компонентом ***Button*** и у овом електронском курсу и то баш у делу у коме смо причали о својствима форме. На пример у делу у коме се објашњава својство *Size* се појављује дугме *форма* и када се кликне на њега појави се текст и слика. То је акција која се изврши када се кликне на то дугме. Такође се појави и дугме *Затвори приказ* и акција која се извршава када се кликне на то дугме одговара самом називу дугмета.

Слика 3.2. Дигитрон

У апликацијама, које ћемо правити, често ћемо користити ову компоненту. Компонента ***Button*** такође има својства као форма и та својства се мењају на исти начин на који се мењају код форме.

<u>Својства која се најчешће користе:</u>

Text је својство које се најчешће користи и служи за задавање натписа који ће да стоји на компоненти у складу са акцијом која треба да се изврши када се кликне на њу. Текст који се иницијално на почетку појављује је button1, за прво дугме које се дода форми, button2, за друго дугме, button3 за треће и тако даље, али као што смо рекли ови натписи се помоћу овог својства могу променити.

Text Align је својство које нуди различите опције за поравнање натписа који стоји на дугмету. Натпис се може поравнати тако да стоји на пример у горњем левом углу, доњем десном углу, центру и тако даље.

(Name) је својство које представља име компоненте које се користи у коду како би се идентификовала та компонента (тј. објекат). Свако дугме аутоматски добија име и то према редоследу додавања форми *button1, button2, button3* и тако даље. Ова имена се могу променити помоћу овог својства, али ми ћемо радити са овим већ унапред задатим именима и нећемо их мењати. Ово својство треба разликовати од својства ***Text.***

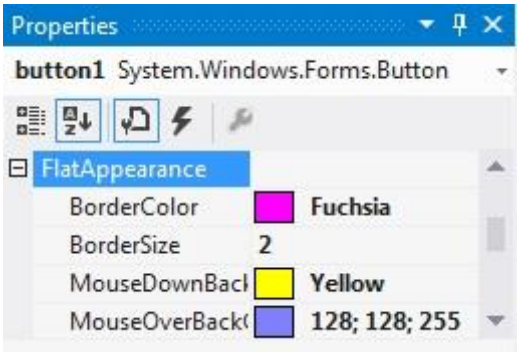
Back Color је својство које служи за промену боје дугмета. Начин на који се то ради је идентичан начину на који се мења боја позадине форме.

Background Image је својство које служи за постављање слике на позадину дугмета.

Background Image Layout је својство у коме су понуђени начини на које слика може да попуни позадину дугмета.

Cursor је својство које нуди различите изгледе курсора. Када пређемо мишем преко дугмета, курсор ће попримити изабрани изглед.

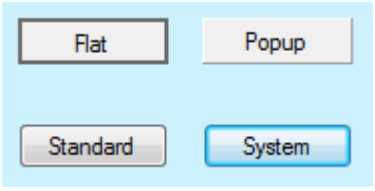
Flat Appearance је својство које се тиче изгледа дугмета и има четири подсвојства: *BorderColor*, служи за промену боје линије којом је оивичено дугме, *BorderSize*, служи за промену дебљине линије којом је оивичено дугме, *MouseDownBackColor*, служи за то да док држимо кликнуто дугме поприми изабрану боју и *MouseOverBackColor*, служи за то да када пређемо мишем преко дугмета, дугме поприми изабрану боју (*Слика 3.3*). Међутим, ова својства се могу мењати само ако је у својству ***Flat Style***, то је следеће својство, изабрана опција *Flat*, у супротном нема сврхе мењати ова подсвојства.

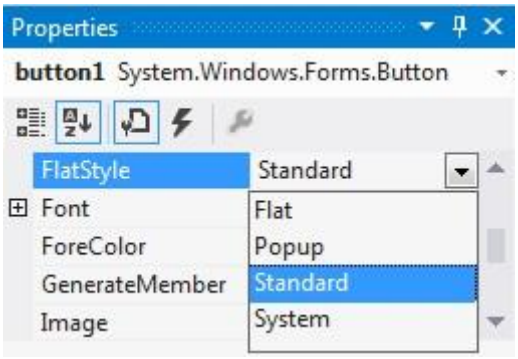


Слика 3.3. Flat Appearance својство

Flat Style је својство у коме се бира између четири понуђена изгледа дугмета и то су:

1. *Flat*
2. *Popup*
3. *Standard*
4. *System*.





Слика 3.4. Flat Style својство

Font је својство помоћу кога се може изабрати врста слова, начин исписа, величина слова натписа који се појављује на дугмету. Подсвојстава се могу видети када се кликне на симбол ⊞. Та подсвојства се директно могу мењати уоквиру *Properties Windows-a*.

Fore Color је својство помоћу кога се може изабрати боја натписа који се појављује на дугмету. Та боја се бира исто као што се бира и боја позадине дугмета.

Size је својство помоћу кога се одређују димензије дугмета изражене у пикселима. Састоји се од два подсвојства: *Width*, које служи за подешавање ширине и *Height*, које служи за подешавање висине.

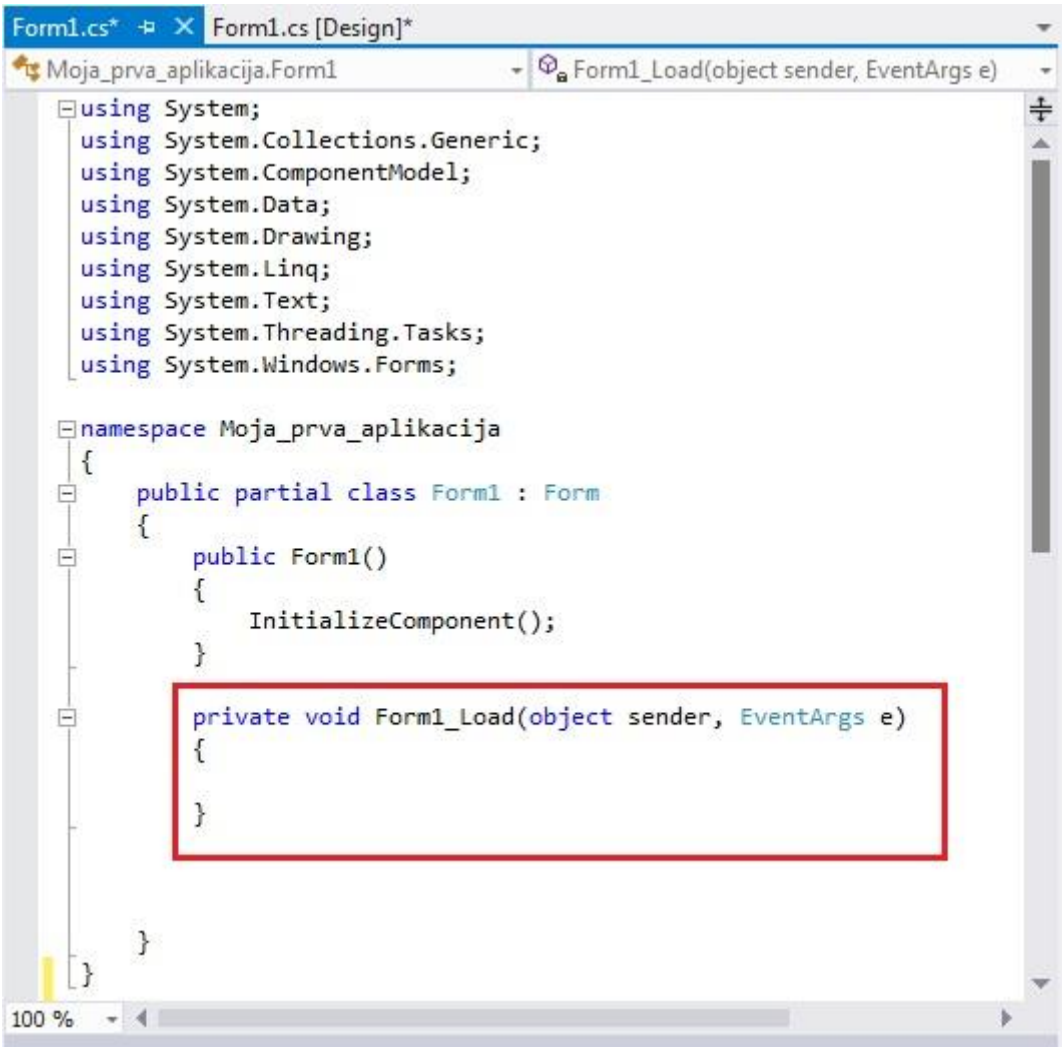
Tab Index је својство помоћу кога се одређује редослед по коме ће се, помоћу *Tab* тастера на тастатури, жижа померати са једне компоненте на другу компоненту. На овај начин можемо да решимо проблем логичног тока уношења података.

Tab Stop је својство помоћу кога се одређује да ли корисник може да помера жижу са једне компоненте на другу помоћу *Tab* тастера. Уколико је изабрана опција *True*, жижа се може померати са компоненте на компоненту на овај начин, а ако је изабрана опција *False*, жижа се не може померати са компоненте на компоненту на овај начин.

Visible је својство помоћу кога се утврђује да ли је компонента видљива или сакривена. Уколико је изабрана опција *True*, компонента ће бити видљива када се покрене програм, а уколико је изабрана опција *False*, компонента се неће видети када се покрене програм, односно биће сакривена.

Задавање својстава компоненти преко кода

Такође постоји још један начин на који можемо променити својства компоненте. Сва наведена својства се могу променити и исписивањем кода. Прикажимо и овај начин. Потребно је два пута да кликнемо на нашу форму како би се појавио део кода у коме ми додатно треба да пишемо наредбе за промену својстава компоненте (*Слика 3.5*). Овим ће дугме, чим покренемо програм, добити додељена својства. Ово је заправо једна акција, додељивање својстава компоненти преко кода, која се извршава приликом покретања програма.



Слика 3.5. Програмски код

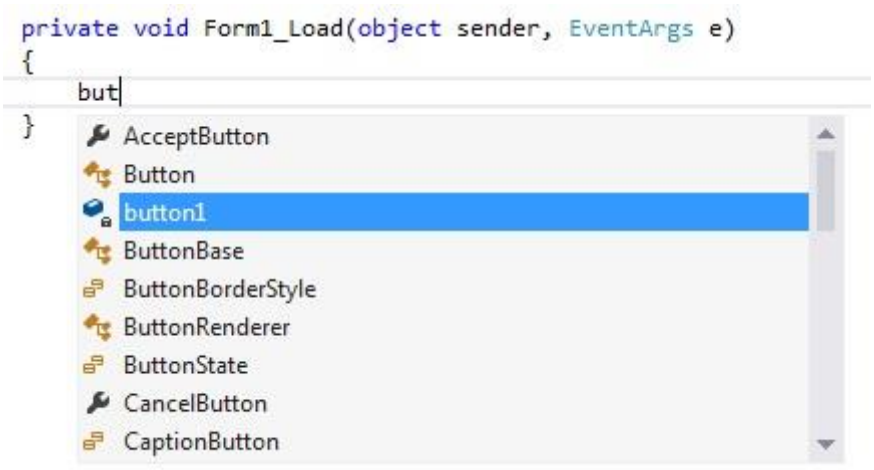
Код који се аутоматски исписује, када кликнемо два пута на форму, је заокружен црвеном линијом на Слици 3.5. Део кода који ми додајемо, куцамо између витичастих заграда.

Напомена: Приликом куцања ћемо додатно коментарисати код, односно писати коментаре. Коментари се пишу када додатно желимо да објаснимо неку наредбу или неку линију кода, како би особи која чита код био што јаснији. Битно је да се зна како се коментари пишу. Сви коментари у *C Sharp-y*, који се протежу у више редова, почињу са /*, а завршавају се са знаком */. Постоји још један начин писања коментара. На почетку коментара стоји знак //. Овај начин ћемо ми користити када будемо писали коментаре. Међутим овај начин се користи само за коментаре који су написани само у једном реду, тако да када желимо да пишемо коментаре у више редова, онда на почетку сваког реда мора да стоји знак //. Коментари се могу препознати и по томе што су зелене боје.

Код започињемо тако што пишемо име оног објекта коме желимо да променимо својство. У нашем случају тај објекат је компонента *Button* и њено име је *button1*. Чим напишемо првих неколико слова имена овог објекта, појавиће се листа са именима објеката који се тренутно налазе на форми и са још неким командама, која су сортирана по абецеди (*Слика 3.6*). Оно што је нама потребно обележено је плавом бојом на Слици 3.6.

Пошто селектујемо оно што нам треба, притиснемо *Enter* на тастатури и на тај начин ће се аутоматски исписати име објекта у коду. Ова листа је корисна када нисмо сигурни како се одређене речи пишу и она ће се стално појављивати и биће нам од велике помоћи приликом рада.

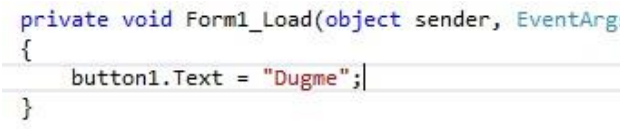
Након што се напише име објекта ставља се тачка, а затим се пише име својства које желимо да променимо. Поново је довољно написати почетна слова и наћи то својство у листи која ће се појавити (*Слика 3.7*).



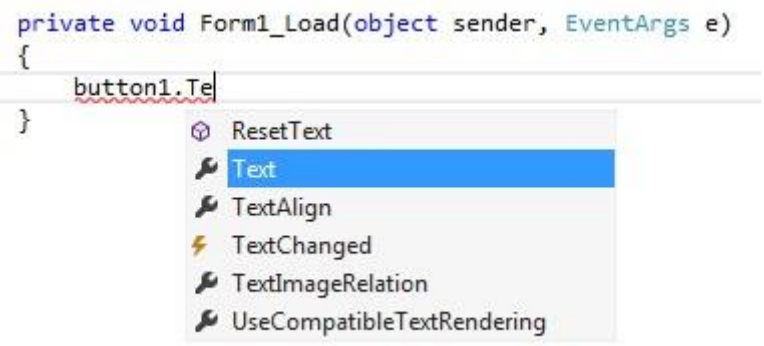
Слика 3.6. Падајућа листа

После овога се пише знак једнакости и између наводника се пише текст који ће се појавити као натпис на дугмету (*Слика 3.8*).

Још један јако важан детаљ је то да се свака линија кода оваквог типа завршава са знаком тачка-зарез (;).



Слика 3.8. Део програмског кода



Слика 3.7. Падајућа листа

Напоменимо још и то да *C-Sharp* разликује велика и мала слова, тако да о овоме додатно морамо водити рачуна. Сва својства и касније догађаји, који почињу великим словима, у коду се исто тако морају писати.

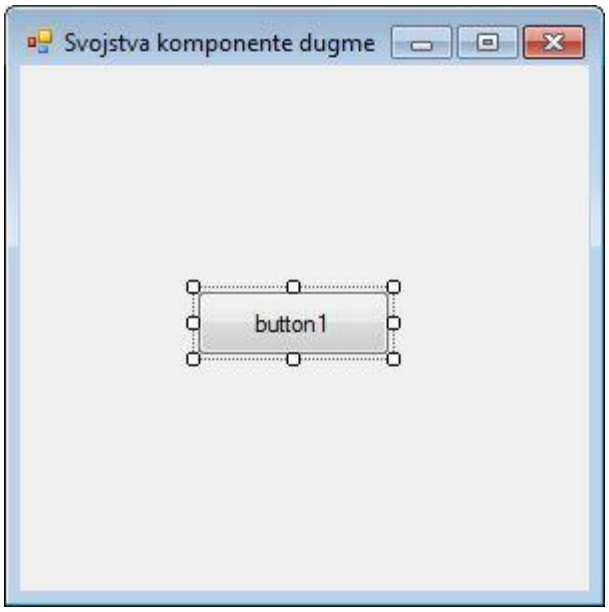
Остала својства се мењају на сличан начин. Следи код са коментарима.

```
private void Form1_Load(object sender, EventArgs e)
{
    button1.Text = "Dugme";

    //Pomocu sledece linije koda se poravnava natpis na dugmetu (svojstvo TextAlign).
Naredba
    //ContentAligment sluzi za odabir nacina na koji ce se sav tekst, koji se
pojavljuje na
    //dugmetu poravnati. U ovom slucaju ce se natpis poravnati u gornji levi ugao
(TopLeft).
    button1.TextAlign = ContentAlignment.TopLeft;

    //Pomocu sledece linije koda se menja boja dugmeta (svojstvo BackColor). Naredba
Color
    //sluzi za odabir boje, a u ovom slucaju smo odabrali zutu boju (Yellow).
    button1.BackColor = Color.Yellow;

    //Pomocu sledece linije koda se menja izgled kursora (svojstvo Cursor). Naredba
Cursor
```

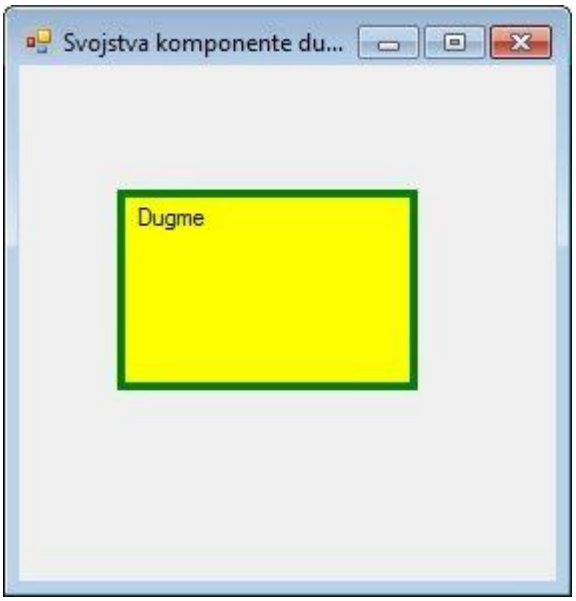
```
    //sluzi za biranje izgleda kursora, a u ovom slucaju smo izabrali krstasti izgled
(Cross).
    button1.Cursor = Cursors.Cross;

    //Pomocu sledece linije koda se menja izgled dugmeta (svojstvo FlatStyle). Naredba
FlatStyle
    //sluzi za izbor stila, a u ovom slucaju smo izabrali Flat stil.
    button1.FlatStyle = FlatStyle.Flat;

    //Pomocu sledece linije koda se menja boja ivice dugmeta (svojstvo FlatAppearance,
podsvojstvo
    //BorderColor). Naredba Color sluzi za odabir boje, a u ovom slucaju smo odabrali
zelenu boju
    //(Green).
    button1.FlatAppearance.BorderColor = Color.Green;

    //Pomocu sledece linije koda se menja velicina ivice dugmeta (svojstvo
FlatAppearance,
    //podsvojstvo BorderSize). U ovom slucaju smo odabrali da velicina bude 3.
    button1.FlatAppearance.BorderSize = 3;

    //Pomocu sledece linije koda se menja boja dugmeta kada kursor misa stoji na
dugmetu, a
    //levi taster na misu stisnut (svojstvo FlatAppearance, podsvojstvo
MouseDownBackColor).
    //Naredba Color sluzi za odabir boje, a u ovom slucaju smo odabrali lila boju
(Violet).
    button1.FlatAppearance.MouseDownBackColor = Color.Violet;

    //Pomocu sledece linije koda se menja boja dugmeta kada se predje misem preko njega
(svojstvo
    //FlatAppearance, podsvojstvo MouseOverBackColor). Naredba Color //sluzi za odabir
boje, a u
    //ovom slucaju smo odabrali ljubicastu boju (Purple).
    button1.FlatAppearance.MouseOverBackColor = Color.Purple;

    //Pomocu sledece linije koda se menja visina dugmeta (podsvojstvo Height). U ovom
slucaju smo
    //postavili da visina bude 100.
    button1.Height = 100;

    //Pomocu sledece linije koda se menja sirina dugmeta (podsvojstvo Width). U ovom
slucaju smo
    //postavili da sirina bude 150.
    button1.Width = 150;

    //Pomocu sledece linije koda se menja redosled po kome ce dugme da dobije fokus
(svojstvo
    //TabIndex). U ovom slucaju je postavjeno na 1.
    button1.TabIndex = 1;

    //Pomocu sledece linije koda se utvrdjuje da li se pomocu Tab tastera moze dugme
postaviti u
    //zizu (svojstvo TabStop). U ovom slucaju moze jer je postavljeno na true, da je
postavljeno
    //na false ne bi moglo.
    button1.TabStop = true;

    //Pomocu sledece linije koda se odredjuje da li ce dugme da bude vidljivo ili
sakriveno kada
    //se pokrene program (svojstvo Visible).
    button1.Visible = true;
}
```

На почетку је наше дугме изгледала као што је приказано на Слици 3.9. После писања кода и покретања програма, наше дугме изгледа као што је приказано на Слици 3.10, с'тим што када се

пређе мишем преко дугмета, дугме поприми лила боју, а када се миш држи притиснут на дугмету, дугме поприми љубичасту боју. Такође се види да када се прелази мишем преко дугмета курсор поприма крстаст изглед.



Слика 3.9. Изглед компоненте Button пре покретања програма

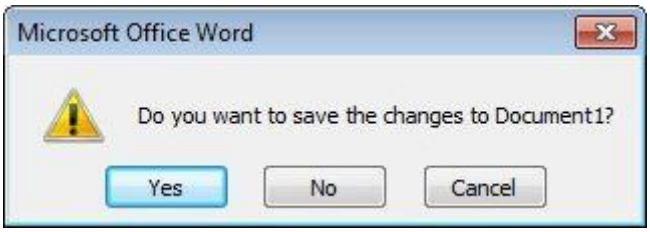


Слика 3.10. Изглед компоненте Button након покретања програма

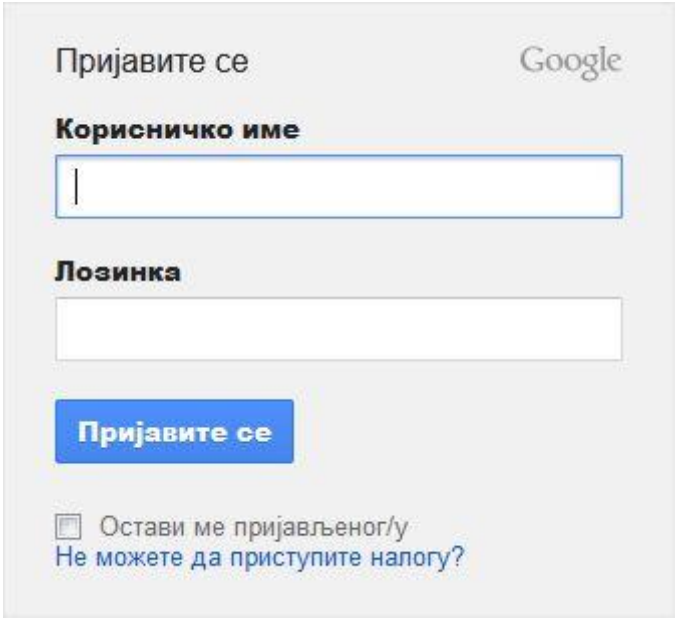
Компонента Label

Label је компонента која се у *Toolbox-y* налази у категорији *Common Controls*. Корисник не може да мења садржај ове компоненте. Када се програм покрене, ова компонента не може бити у жижи. Питање које се појављује на slici

, са претходне стране, је заправо једна лабела у којој је то питање исписано.



Са овом компонентом се такође можемо срести и када хоћемо да се улогујемо на неки налог, на пример *Google* налог (*Слика 3.1*). Приметимо да се овде налазе три лабеле. У првој је исписана порука *Пријавите се*, у другој је исписано *Корисничко име*, а у трећој *Лозинка*. Значи, лабела служи да би се преко ње описала нека друга компонента. У овом случају лабела *Корисничко име* служи за то да се нагласи да поље за унос текста, које се налази испод ње, служи за то да се у њега упише корисничко име, а лабела *Лозинка* служи за то да се нагласи да се у поље за унос текста, које се налази испод ње, уписује лозинка корисника.



Слика 3.1. Приказ компоненте Label

Својства која се најчешће користе:

Text је својство које се најчешће користи и служи за задавање натписа који ће да стоји у лабели. Најчешће се задаје када треба да опише неку другу компоненту. Текст који се иницијално на почетку појављује је *label1*, за прву лабелу које се дода форми, *label2*, за другу лабелу, *label3* за трећу и тако даље, али као што смо рекли овај текст се помоћу овог својства може променити.

Text Align је својство које нуди различите опције за поравнање натписа који стоји у лабели. Натпис се може поравнати тако да стоји на пример у горњем левом углу, доњем десном углу, центру и тако даље.

(Name) је својство које представља име компоненте које се користи у коду како би се идентификовала та компонента (тј. објекат). Свака лабела аутоматски добија име и то према редоследу додавања форми *label1, label2, label3* и тако даље. Ова имена се могу променити помоћу овог својства, али ми ћемо радити са овим већ унапред задатим именима и нећемо их мењати. Ово својство треба разликовати од својства ***Text.***

Auto Size је својство које служи да се димензије лабеле аутоматски прилагоде тексту који је исписан у лабели. Ако је ово својство постављено на *True*, онда се димензије аутоматски прилагођавају, а ако је постављено на *False*, онда ми можемо сами да задамо висину и ширину помоћу својства *Size*.

Back Color је својство које служи за промену боје позадине лабеле. Начин на који се то ради је идентичан начину на који се мења боја позадине форме.

Border Style својство има три понуђене опције које пружају различите изгледе ивица лабеле.

Cursor је својство које нуди различите изгледе курсора. Када пређемо мишем преко лабеле, курсор ће попримити изабрани изглед.

Font је својство помоћу кога се може изабрати врста слова, начин исписа, величина слова натписа који се појављује у лабели. Подсвојства се могу видети када се кликне на симбол ⊞. Та подсвојства се директно могу мењати у оквиру *Properties Windows-a*.

Fore Color је својство помоћу кога се може изабрати боја текста који се појављује у лабели. Та боја се бира исто као што се бира и боја позадине лабеле.

Visible је својство помоћу кога се утврђује да ли је компонента видљива или сакривена. Уколико је изабрана опција *True*, компонента ће бити видљива када се покрене програм, а уколико је изабрана опција *False*, компонента се неће видети када се покрене програм, односно биће сакривена.

Задавање својстава преко кода

Као што смо мењали својства компоненте *Button* преко кода, тако можемо да мењамо својства и ове компоненте. Поступак је исти, само се уместо *button1* пише *label1*, јер сада мењамо својства компоненте *Label*.

```
private void Form1_Load(object sender, EventArgs e)
{
    //Postavljanje teksta u labelu.
    label1.Text = "Ovo je labela!";

    //Na ovaj nacin ce se dimenzije labele prilagoditi tekstu koji
    //je u njoj ispisan.
    label1.AutoSize = true;

    //Promena boje pozadine labele.
    label1.BackColor = Color.Beige;

    //Na ovaj nacin ce kursor poprimiti drugi izgled kada se misem
    //predje preko labele.
    label1.Cursor = Cursors.Hand;

    //Promena izgleda ivica labele.
    label1.BorderStyle = BorderStyle.Fixed3D;

    //Labela ce biti vidljiva kada se pokrene program.
    label1.Visible = true;
}
```

На почетку је наша лабела изгледала као што је приказано на Слици 3.2. После писања кода и покретања програма, наша лабела изгледа као што је приказано на Слици 3.3. Такође се види да када се прелази мишем преко лабеле курсор поприма изглед руке.

Слика 3.2. Изглед компоненте Label пре покретања програма

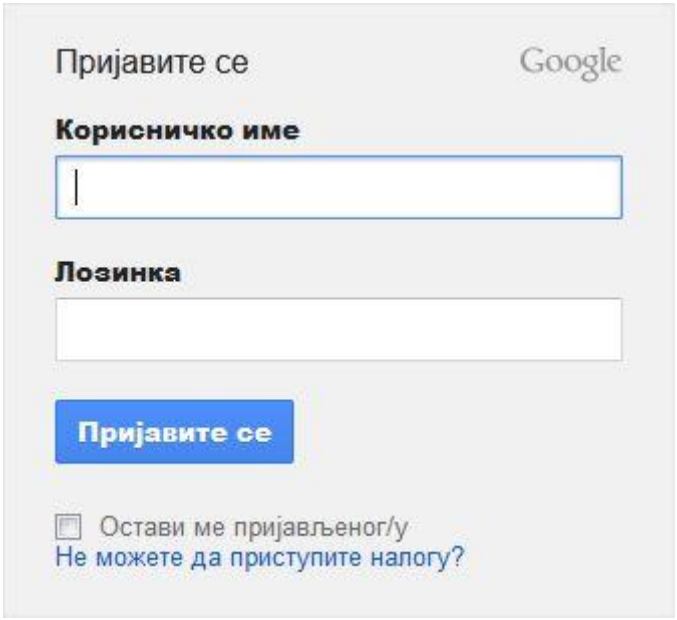
Слика 3.3. Изглед компоненте Label након покретања програма

Компонента TextBox

TextBox је компонента која се у *Toolbox-у* налази у категорији *Common Controls*. *TextBox* тј. оквир за уношење и приказивање текста је компонента која, како јој и само име каже, може да служи и за унос текста са тастатуре и за исписивање неког текста који се одређује помоћу програма. Када је ова компонента у жижи у њеном оквиу трепери
курсор.

Већ смо напоменули, када смо причали о лабели, да се са овом компонентом можемо срести када хоћемо да се улогујемо на неки налог (*Слика 3.1*). Оквири испод лабела *Корисничко име* и *Лозинка* су заправо оквири за унос и приказивање текста. У овом случају се у прво поље за унос текста уноси корисничко име, а у друго поље за унос текста се уноси лозинка корисника. На

, са стране на којој смо причали о компоненти *Button*, такође постоји *TextBox* и то је оквир у коме је приказан број 9. Приметимо да у овом примеру *TextBox* не служи за унос текста, већ за исписивање текст (у овом случају тај текст је бој девет).



Слика 3.1. Приказ компоненте Label

Својства која се најчешће користе:

Text је својство које служи за исписивање текста који ће да стоји у *TextBox-у*. Међутим, у већини случајева је потребно да ова компонента буде празна, тако да у пољу поред својства *Text* не треба да стоји ништа.

(Name) је својство које представља име компоненте које се користи у коду како би се идентификовала та компонента (тј. објекат). Сваки *TextBox* аутоматски добија име и то према редоследу додавања форми *textBox1*, *textBox2*, *textBox3* и тако даље. Ова имена се могу променити помоћу овог својства, али ми ћемо радити са овим већ унапред задатим именима и нећемо их мењати.

Back Color је својство које служи за промену боје позадине *TextBox-а*. Начин на који се то ради је идентичан начину на који се мења боја позадине форме.

Border Style својство има три понуђене опције које пружају различите изгледе ивица *TextBox-а*.

Cursor је својство које нуди различите изгледе курсора. Када пређемо мишем преко *TextBox-а*, курсор ће попримити изабрани изглед.

Font је својство помоћу кога се може изабрати врста слова, начин исписа, величина слова натписа који се појављује у *TextBox-у*. Подсвојства се могу видети када се кликне на симбол ⊞. Та подсвојства се директно могу мењати у оквиру *Properties Windows-а*.

Fore Color је својство помоћу кога се може изабрати боја текста који се појављује у пољу за унос текста. Та боја се бира исто као што се бира и боја позадине поља за унос текста.

Multiline је својство које омогућава да се текст исписује у више редова уместо само у једном.

Scroll Bars има смисла користити само ако је у својству ***Multiline*** изабрана опција *True* и у овом својству се могу изабрати опције за скроловање текста горе-доле или лево-десно или и једно и друго. Користи се када се текст који је исписан у *TextBox-y* не може видети у целости због малих димензија *TextBox-a*

Size је својство помоћу кога се одређују димензије *TextBox-a* изражене у пикселима. Састоји се од два подсвојства: *Width*, које служи за подешавање ширине и *Height*, које служи за подешавање висине.

Tab Index је својство помоћу кога се одређује редослед по коме ће се, помоћу *Tab* тастера на тастатури, жижа померати са једне компоненте на другу компоненту. На овај начин можемо да решимо проблем логичног тока уношења података.

Tab Stop је својство помоћу кога се одређује да ли корисник може да помера жижу са једне компоненте на другу помоћу *Tab* тастера. Уколико је изабрана опција *True*, жижа се може померати са компоненте на компоненту на овај начин, а ако је изабрана опција *False*, жижа се не може померати са компоненте на компоненту на овај начин.

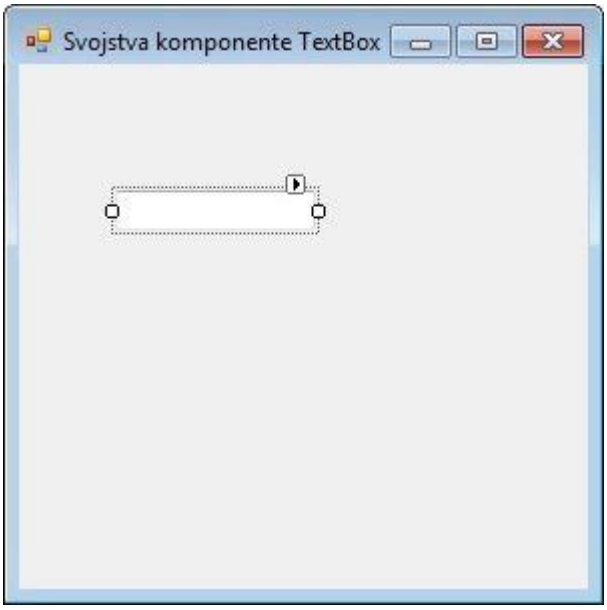
Visible је својство помоћу кога се утврђује да ли је компонента видљива или сакривена. Уколико је изабрана опција *True*, компонента ће бити видљива када се покрене програм, а уколико је изабрана опција *False*, компонента се неће видети када се покрене програм, односно биће сакривена.

Задавање својстава преко кода

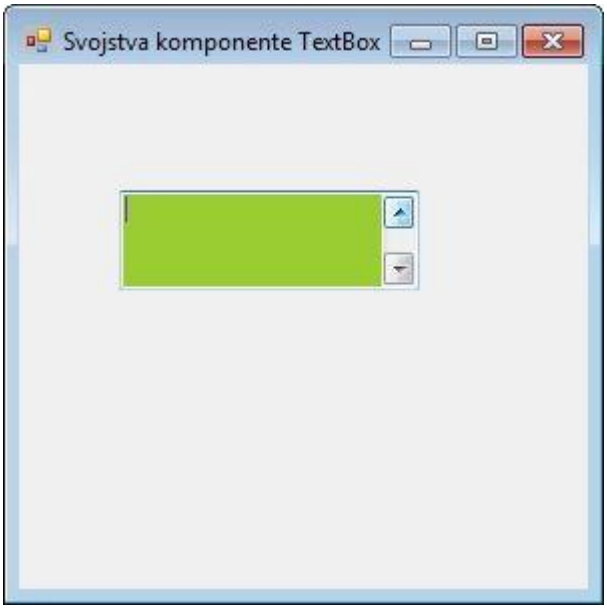
```
private void Form1_Load(object sender, EventArgs e)
{
    //Posto ce najcesce biti potrebno da TextBox bude prazan
    //onda cemo pomocu ove linije koda to i postici. Izmedju
    //znakova navodnika nista ne pisemo.
    textBox1.Text = "";

    //Promena boje pozadine TextBox-a
    textBox1.BackColor = Color.YellowGreen;

    //Promena izgleda ivica
    textBox1.BorderStyle = BorderStyle.Fixed3D;

    //Promena izgleda kusora
    textBox1.Cursor = Cursors.AppStarting;

    //Omogucava unos teksta u vise redova
    textBox1.Multiline = true;

    //Postavlja strelice za pomeranje teksta gore dole, ovo
    //ima smisla stavljati samo kada je svojstvo Multiline
    //postavljeno na true.
    textBox1.ScrollBars = ScrollBars.Vertical;

    //Zadavanje visine TextBox-a i ovo ima smisla stavljati
    //samo kada je svojstvo Multiline postavljeno na true.
    textBox1.Height = 50;

    //Zadavanje sirine TextBox-a
    textBox1.Width = 150;

    //Zadavanje redosleda po kome ce TextBox da bude u zizi
    textBox1.TabIndex = 1;

    //Omogucava da se pomocu Tab tastera ziza premesti na ovu
    //komponentu
    textBox1.TabStop = true;
```

```
    //TextBox ce biti vidljiv kada se pokrene program.
    textBox1.Visible = true;
}
```

На почетку је *TextBox* изгледао као што је приказано на Слици 3.2. После писања кода и покретања програма, *TextBox* изгледа као што је приказано на Слици 3.3. На почетку *TextBox-а* стоји курсор. У њему се може куцати неки текст и то у више линија и помоћу стрелица са десне стране, текст може да се помера горе доле.



Слика 3.2. Изглед компоненте TextBox пре покретања програма



Слика 3.3. Изглед компоненте TextBox након покретања програма

Компонента RichTextBox

RichTextBox је компонента која се у *Toolbox-y* налази у категорији *Common Controls*. Компонента *RichTextBox* је јако слична компоненти *TextBox*. Значи, ова компонента такође служи и за унос текста са тастатуре и за исписивање неког текста који се одређује помоћу програма. Разлика између ове компоненте и компоненте *TextBox* је у томе што је компонента *RichTextBox* предвиђена за уношење и иписивање текста већег обима, док је *TextBox* компонента предвиђена за унос углавном једнолинијског текста и ређе текста у неколико линија. Када је ова компонента у жижи у њеном оквиу трепери курсор.

Ове две компоненте имају јако велики број заједничких својстава, баш зато што постоји велика сличност између њих.

Својства која се најчешће користе:

Text је својство које служи за исписивање текста који ће да стоји у *RichTextBox-y*. Међутим, у већини случајева је потребно да ова компонента буде празна, тако да у пољу поред својства *Text* не треба да стоји ништа.

(Name) је својство које представља име компоненте које се користи у коду како би се идентификовала та компонента (тј. објекат). Сваки *RichTextBox* аутоматски добија име и то према редоследу додавања форми *richTextBox1, richTextBox2, richTextBox3* и тако даље. Ова имена се могу променити помоћу овог својства, али ми ћемо радити са овим већ унапред задатим именима и нећемо их мењати.

Back Color је својство које служи за промену боје позадине *RichTextBox-a*. Начин на који се то ради је идентичан начину на који се мења боја позадине *TextBox-a*.

Border Style својство има три понуђене опције које пружају различите изгледе ивица *RichTextBox-a*.

Cursor је својство које нуди различите изгледе курсора. Када пређемо мишем преко *RichTextBox-a*, курсор ће попримити изабрани изглед.

Detect Urls је својство које служи за детекцију URL адреса и аутоматски их формира као линкове.

Font је својство помоћу кога се може изабрати врста слова, начин исписа, величина слова текста који се појављује у *RichTextBox-y*. Подсвојства се могу видети када се кликне на симбол ⊞. Та подсвојства се директно могу мењати у оквиру *Properties Windows-a*.

Fore Color је својство које служи за промену боје текста који се појављује у *RichTextBox-y*.

Multiline је својство које омогућава да се текст исписује у више редова уместо само у једном.

Scroll Bars има смисла користити само ако је у својству ***Multiline*** изабрана опција *True* и у овом својству се могу изабрати опције за скроловање текста горе-доле или лево-десно или и једно и друго. Користи се када се текст који је исписан у *RichTextBox-y* не може видети у целости.

Size је својство помоћу кога се одређују димензије *RichTextBox-a* изражене у пикселима. Састоји се од два подсвојства: *Width*, које служи за подешавање ширине и *Height*, које служи за подешавање висине.

Tab Index је својство помоћу кога се одређује редослед по коме ће се, помоћу *Tab* тастера на тастатури, жижа померати са једне компоненте на другу компоненту.

Tab Stop је својство помоћу кога се одређује да ли корисник може да помера жижу са једне компоненте на другу помоћу *Tab* тастера. Уколико је изабрана опција *True*, жижа се може померати са компоненте на компоненту на овај начин, а ако је изабрана опција *False*, жижа се не може померати са компоненте на компоненту на овај начин.

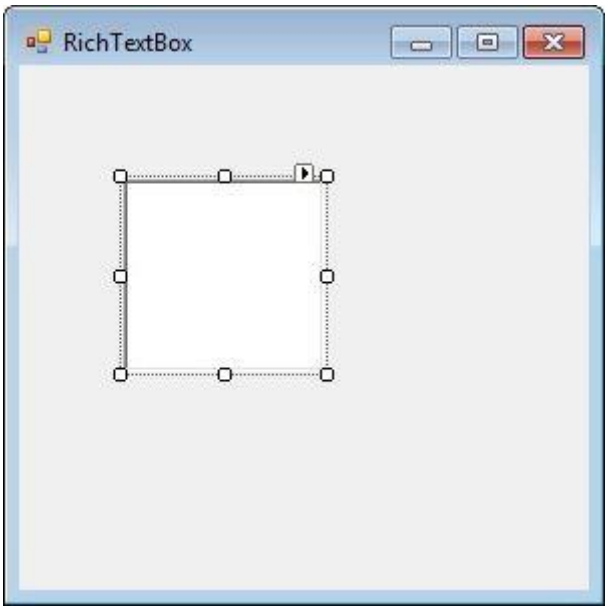
Visible је својство помоћу кога се утврђује да ли је компонента видљива или сакривена. Уколико је изабрана опција *True*, компонента ће бити видљива када се покрене програм, а уколико је изабрана опција *False*, компонента се неће видети када се покрене програм, односно биће сакривена.

Задавање својстава преко кода

```
private void Form1_Load(object sender, EventArgs e)
{
    //Posto ce najcesce biti potrebno da RichTextBox bude prazan
    //onda cemo pomocu ove linije koda to i postici. Izmedju
    //znakova navodnika nista ne pisemo.
    richTextBox1.Text = "";

    //Promena boje pozadine RichTextBox-a
    richTextBox1.BackColor = Color.Thistle;

    //Promena izgleda ivica
    richTextBox1.BorderStyle = BorderStyle.Fixed3D;

    //Promena izgleda kusora
    richTextBox1.Cursor = Cursors.IBeam;

    //Detekcija linkova
    richTextBox1.DetectUrls = true;
```

```
    //Promena boje teksta koji ce biti ispisan u RichTextBox-u
    richTextBox1.ForeColor = Color.Yellow;

    //Omogucava unos teksta u vise redova
    richTextBox1.Multiline = true;

    //Zadavanje visine
    richTextBox1.Height = 50;

    //Zadavanje sirine
    richTextBox1.Width = 150;

    //Zadavanje redosleda po kome ce RichTextBox da bude u zizi
    richTextBox1.TabIndex = 1;

    //Omogucava da se pomocu Tab tastera ziza premesti na ovu
    //komponentu
    richTextBox1.TabStop = true;

    //RichTextBox ce biti vidljiv kada se pokrene program.
    richTextBox1.Visible = true;
}
```

На почетку је компонента *RichTextBox* изгледала као што је приказано на Слици 3.1. После писања кода и покретања програма, *RichTextBox* изгледа као што је приказано на Слици 3.2. У њему смо откуцали текст и видимо да је текст жуте боје, а позадина розе.

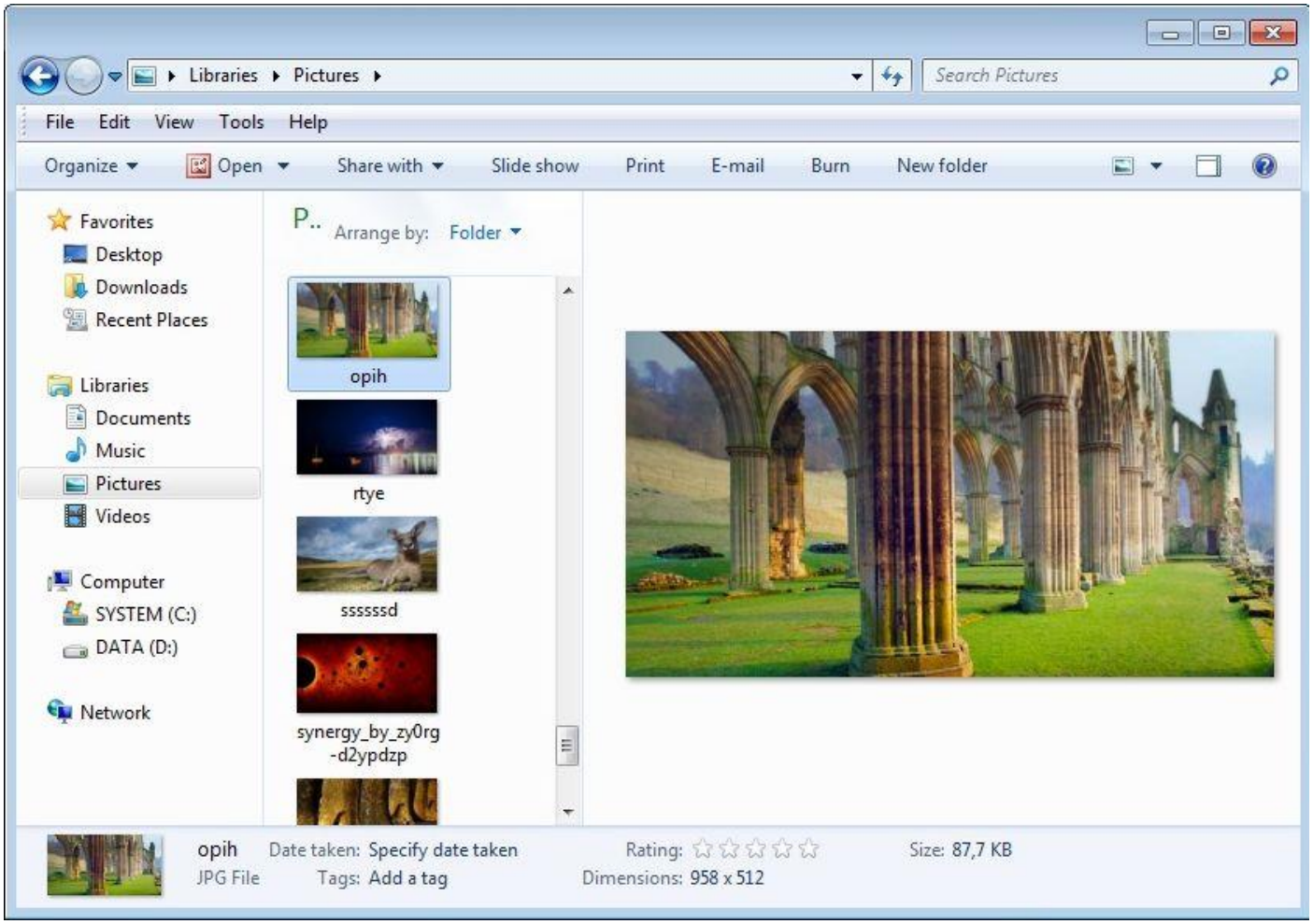


Слика 3.1. Изглед компоненте RichTextBox пре покретања програма

Слика 3.2. Изглед компоненте RichTextBox након покретања програма

Компонента PictureBox

PictureBox је компонента која се у *Toolbox-у* налази у категорији *Common Controls*. Компонента *PictureBox*, односно оквир за графички објекат, служи најчешће за приказивање слика. Када се програм покрене, ова компонента не може бити у жижи исто као што компонента *Label* не може бити у жижи. Са овом компонентом можемо да се сретнемо и када отворимо неки фолдер у коме су сачуване слике (*Слика 3.1*). Када кликнемо на неку слику из фолдера у десном делу прозора ће се приказати селектована слика. Оквир у коме се она приказује је заправо једна *PictureBox* компонента.



Слика 3.1. Приказ примене компоненте PictureBox

Својства која се најчешће користе:

(Name) је својство које представља име компоненте које се користи у коду како би се идентификовала та компонента (тј. објекат). Свака *PictureBox* компонента аутоматски добија име и то према редоследу додавања форми *pictureBox1*, *pictureBox2*, *pictureBox3* и тако даље. Ова имена се могу променити помоћу овог својства, али ми ћемо радити са овим већ унапред задатим именима и нећемо их мењати.

Back Color је својство које служи за промену боје позадине *PictureBox-a*.

Background Image је својство које служи за постављање слике у *PictureBox*.

Background Image Layout је својство у коме су понуђени начини на које слика може да попуни *PictureBox*.

Border Style својство има три понуђене опције које пружају различите изгледе ивица *PictureBox-a*.

Cursor је својство које нуди различите изгледе курсора. Када пређемо мишем преко *PictureBox-a*, курсор ће попримити изабрани изглед.

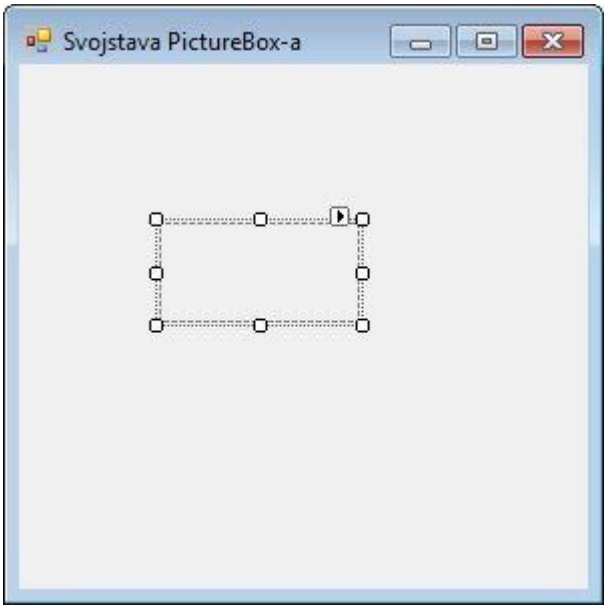
Size је својство помоћу кога се одређују димензије *PictureBox-a* изражене у пикселима. Састоји се од два подсвојства: *Width*, које служи за подешавање ширине и *Height*, које служи за подешавање висине.

Visible је својство помоћу кога се утврђује да ли је компонента видљива или сакривена. Уколико је изабрана опција *True*, компонента ће бити видљива када се покрене програм, а уколико је изабрана опција *False*, компонента се неће видети када се покрене програм, односно биће сакривена.

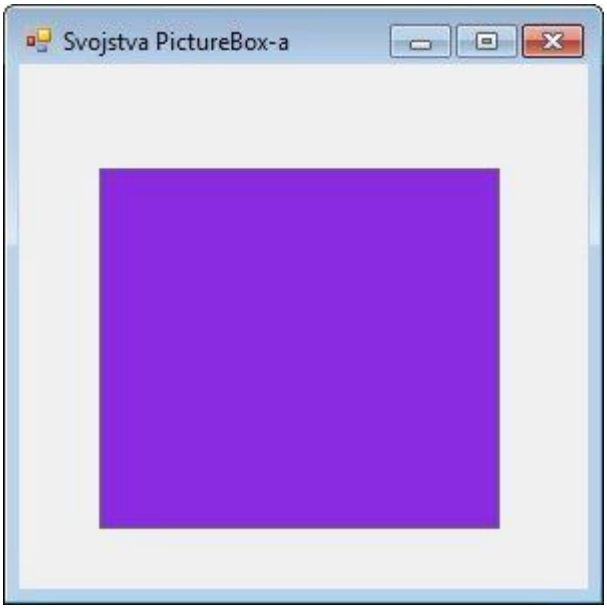
Задавање својстава преко кода

```
private void Form1_Load(object sender, EventArgs e)
{
    //Promena boje pozadine PictureBox-a
    pictureBox1.BackColor = Color.BlueViolet;

    //Promena izgleda ivica
    pictureBox1.BorderStyle = BorderStyle.FixedSingle;

    //Promena izgleda kursora
    pictureBox1.Cursor = Cursors.Arrow;

    //Zadavanje visine PictureBox-a
    pictureBox1.Height = 180;

    //Zadavanje sirine PictureBox-a
    pictureBox1.Width = 200;

    //PictureBox ce biti vidljiv kada se pokrene program
    pictureBox1.Visible = true;
}
```

На почетку је компонента *PictureBox* изгледала као што је приказано на Слици 3.2. После писања кода и покретања програма, *PictureBox* изгледа као што је приказано на Слици 3.3.



Слика 3.2. Изглед компоненте PictureBox пре покретања програма

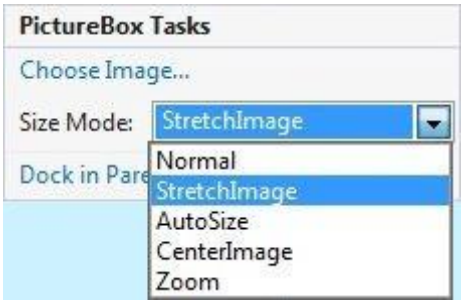


Слика 3.3. Изглед компоненте PictureBox након покретања програма

Приметимо такође да када је компонента *PictureBox* селектован, као на Слици 3.2, у горњем десном углу се појављује стрелица (▶). Када се кликне на ту стрелицу отвориће се мени (*Слика 3.4*).

Слика 3.4. Мени

У том менију се налазе три опције: *Choose Image*, које служи за импортовање слике у *PictureBox*, *Size Mode*, који служи да се изабере начин на који ће слика испунити *PictureBox* (*Слика 3.5*), и *Dock in parent container*, који служи за то да се *PictureBox* прошири преко целе форме, односно да се прилагоди димензијама.

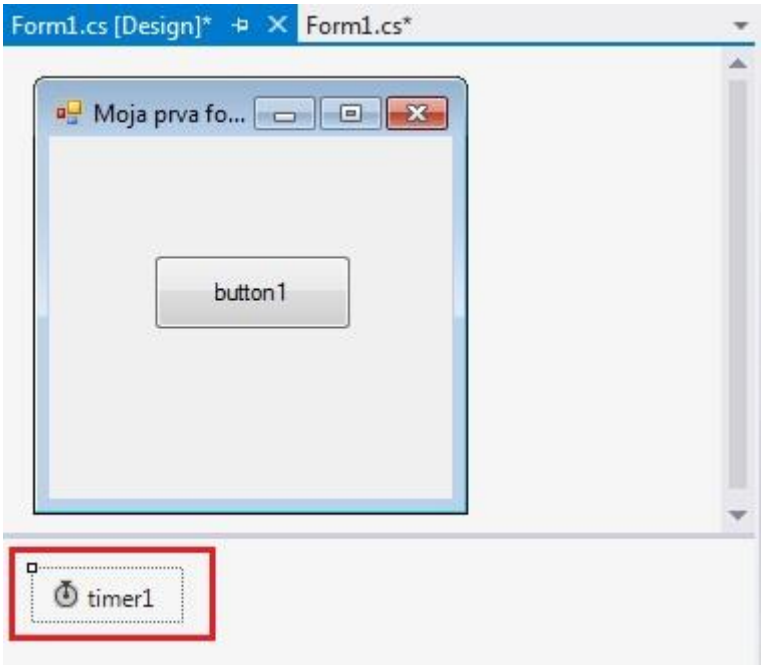


Слика 3.5. Size Mode

Компонента Timer

Timer компонента се у *Toolbox-у* налази у категорији *Components*. Ова компонента омогућава извршавање неких акција периодично, тј. у тачно одређеном тренутку у односу на њено активирање. Користи се код симулације кретања и код визуелних ефеката који треба да се понављају у једнаким временским интервалима. Ова компонента се не види на форми, већ се види на дну *DesignView-a* (*Слика 3.1*).

Ова компонента има своју улогу у разним програмима. На пример, када гледамо слике на рачунару, уместо сами да их листамо, на располагању имамо опцију *Slide show* која омогућава аутоматско листање слика у одређеном временском интервалу. Увек протекне исти временски интервал између смењивања две узастопне слике. Тај временски интервал се обезбеђује помоћу компоненте *Timer*.



Слика 3.1. Изглед компоненте Timer

Својства која се најчешће користе:

(Name) је својство које представља име компоненте које се користи у коду како би се идентификовала та компонента (тј. објекат). Свака *Timer* компонента аутоматски добија име и то према редоследу додавања форми *timer1*, *timer2*, *timer3* и тако даље. Ова имена се могу променити помоћу овог својства, али ми ћемо радити са овим већ унапред задатим именима и нећемо их мењати.

Enabled је својство које служи за активацију компоненте *Timer*.

Interval је својство којим се подешава временски интервал у коме ће се одређена акција понављати. Овај временски интервал је изражен у милисекундама.

Задавање својстава преко кода

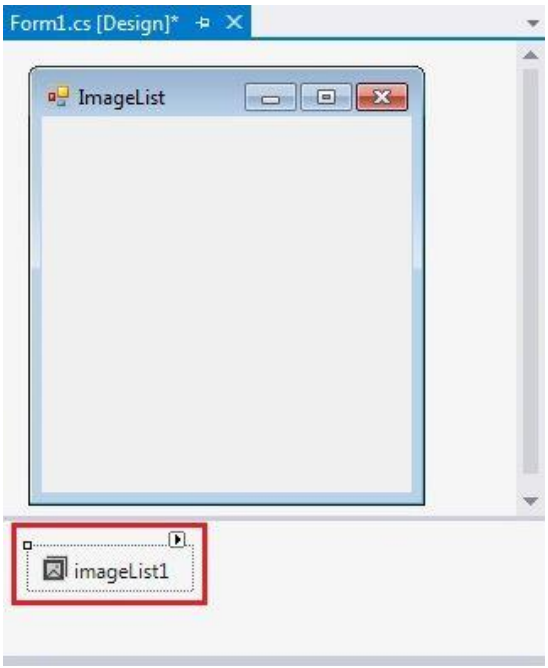
```
private void Form1_Load(object sender, EventArgs e)
{
    //Podesavanje intervala tajmera
    timer1.Interval = 1000;

    //Ovim kodom se aktivira Timer. Ukoliko zelimo da zautavimo Timer, onda pisemo istu
    //liniju koda, samo umesto true pisemo false (timer.Enabled = false;).
    timer1.Enabled = true;
}
```

За разлику од осталих компонената и њихових својстава, својства ове компоненте ћемо најчешће задавати преко кода.

Компонента ImageList

ImageList компонента се у *Toolbox-у* налази у категорији *Components* и представља колекцију слика исте величине. Све слике у листи су нумерисане односно свака слика има свој редни број који се још назива *индекс*. Нумерација слика у листи почине од нуле. Приступ свакој слици у листи се врши преко њеног индекса. Ова компонента се не види на форми, већ се види на дну *DesignView-a* (*Слика 3.1*).

Ова компонента се често користи за управљање великим сетом иконица или системских слика. Такође се може користити и за управљање слика из галерије. Овакве листе не садрже само слике већ су у оквиру ових листа понуђене и опције за додавање нових слика већ постојећој листи и уклањање слика из листе. Приликом прављења апликација ова компонента се користи у комбинацији са другим компонентама, на пример користићемо је ако правимо апликацију која треба да има опције које су понуђене у фото галеријама, али ће уз њу бити коришћене и многе друге компоненте као што су дугмићи, лабеле, поља за приказивање слика и друге.



Слика 3.1. Изглед компоненте ImageList

Проналажење броја слика, индекса последње слике у листи

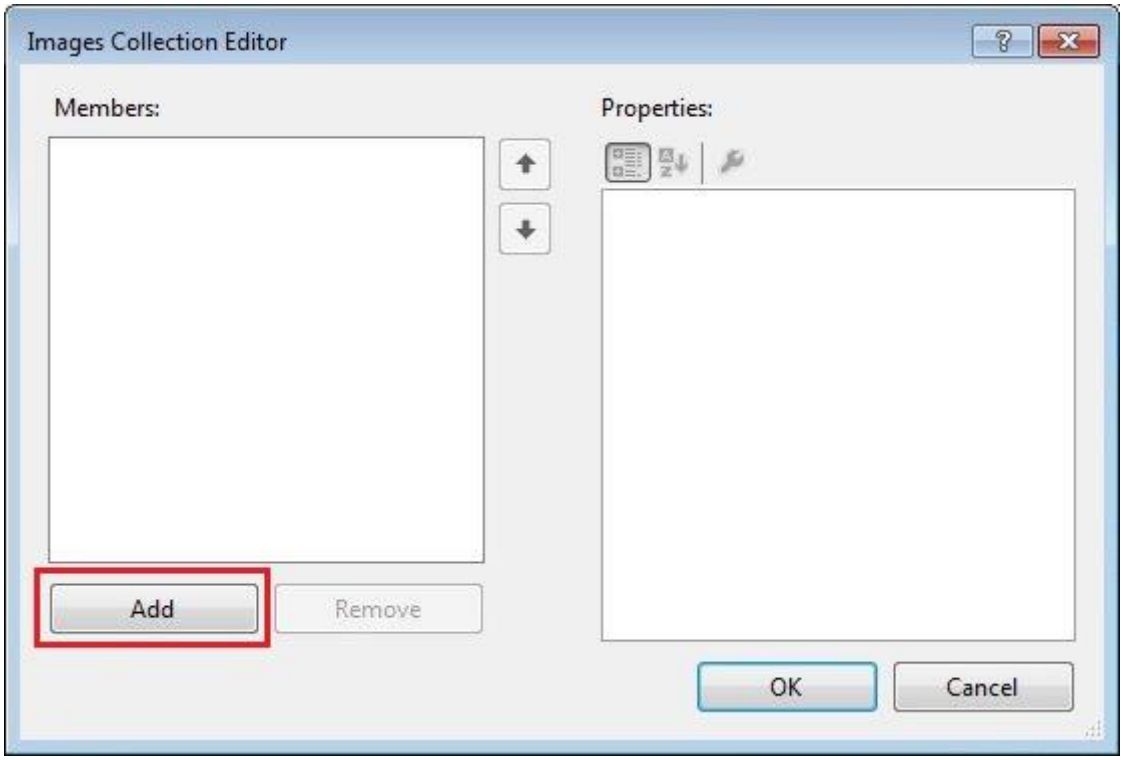
и приступ сликама преко њихових индекса

Својства која се најчешће користе:

(Name) је својство које представља име компоненте које се користи у коду како би се идентификовала та компонента (тј. објекат). Свака *ImageList* компонента аутоматски добија име и то према редоследу додавања форми *imageList1*, *imageList2*, *imageList3* и тако даље. Ова имена се могу променити помоћу овог својства, али ми ћемо радити са овим већ унапред задатим именима и нећемо их мењати.

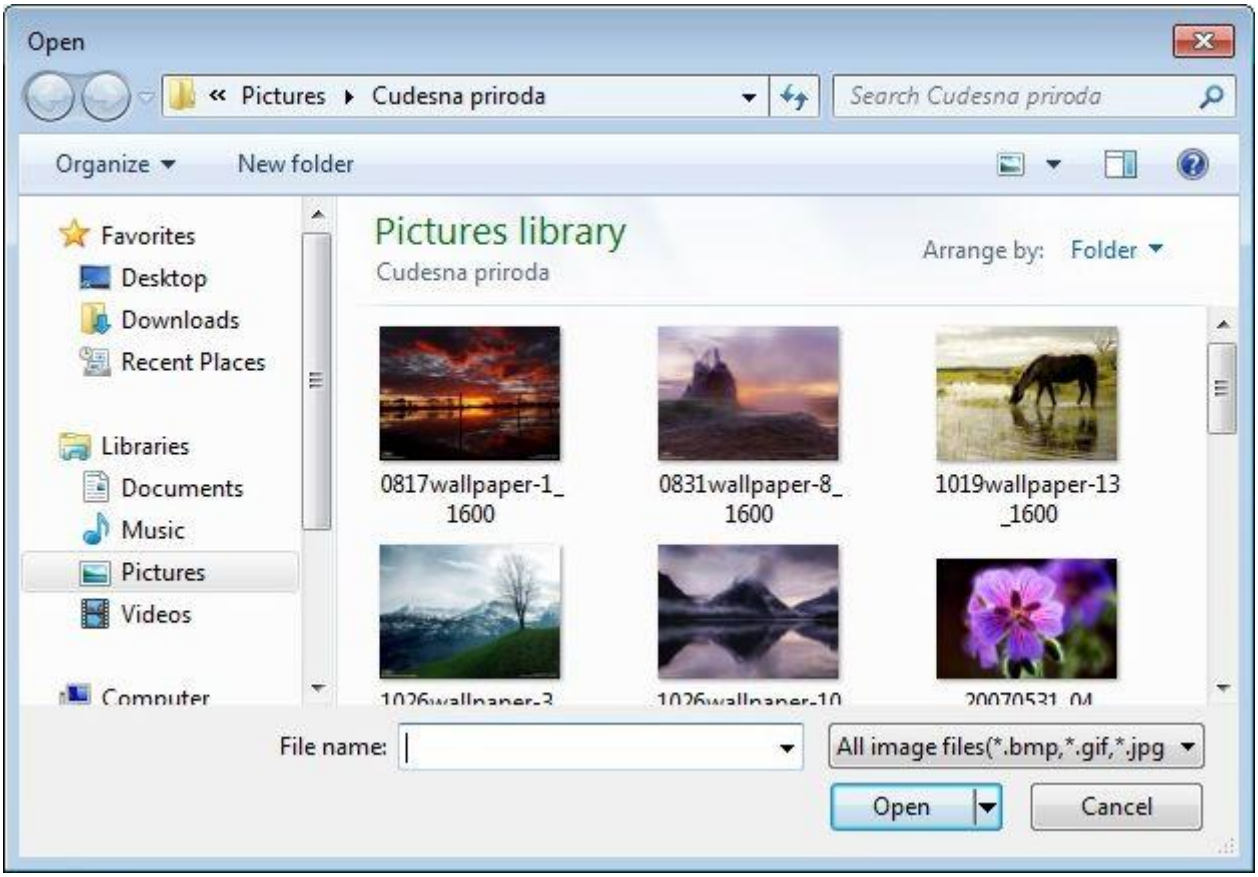
ColorDepth је својство које служи за подешавање квалитета слика у листи.

Images је својство које служи за додавање слика у листу. Кликом на иконицу ... појављује се нови прозор који слжи за импортовање слика (*Слика 3.2*)



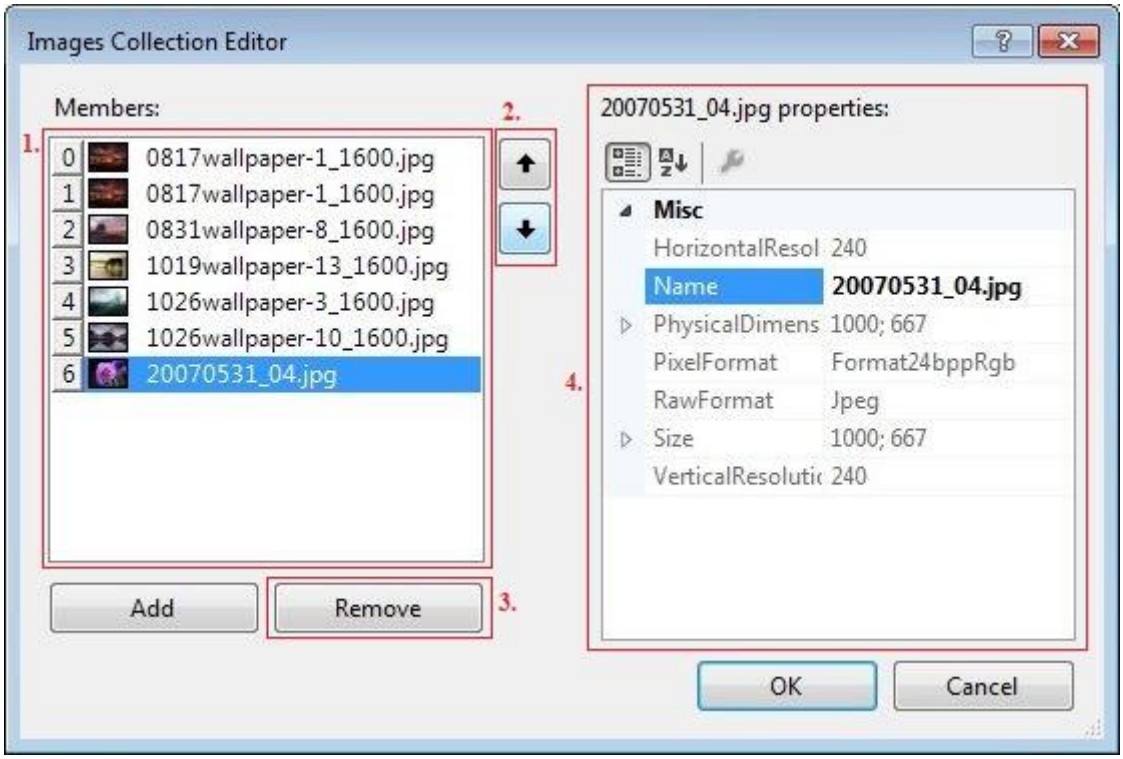
Слика 3.2. Убацивање слика у листу

Да бисмо могли да убацимо слику у листу потребно је да кликнемо на дугме *Add* након чега ће се појавити *Windows Explorer* у коме треба да нађемо фолдер у коме су смештене нама потребне слике и да изаберемо слике из тог фолдера које желимо да се нађу у нашој листи (*Слика 3.3*).



Слика 3.3. Windows Explorer

Када завршимо са селектовањем слика кликнемо на дугме *Open* након чега ће се слике појавити у листи (*Слика 3.4*).



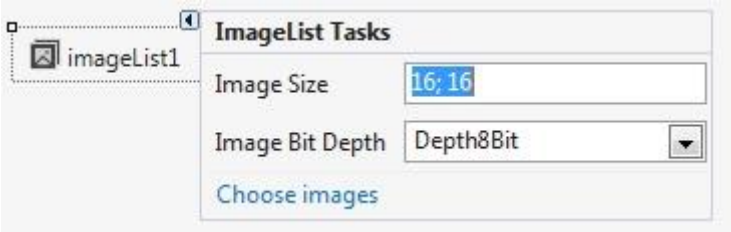
Слика 3.4. Убацивање слика у листу

Поменимо још и опције које су приказане на Слици 3.4:

1. приказ нумерисаног списка убачених слика
2. помоћу ових стелица могуће је мењати редослед селектоване слике
3. помоћу дугмета *Remove* могуће је уклонити слектовану слку из листе
4. приказ особина селектоване слике (име слике, димензије, резолуција, тип слике).

ImageSize је својство помоћу кога се одређују димензије слика из листе изражене у пикселима. Састоји се од два подсвојства: *Width*, које служи за подешавање ширине и *Height*, које служи за подешавање висине слика.

Ова свосјтва се, осим у *Properties Windows-y*, могу подесити и коришћењем менија који се појављује када се кликне на стрелицу ▶ која се налази у горњем десном углу саме компоненте (*Слика 3.5*).



Слика 3.5. Мени

Задавање својстава преко кода

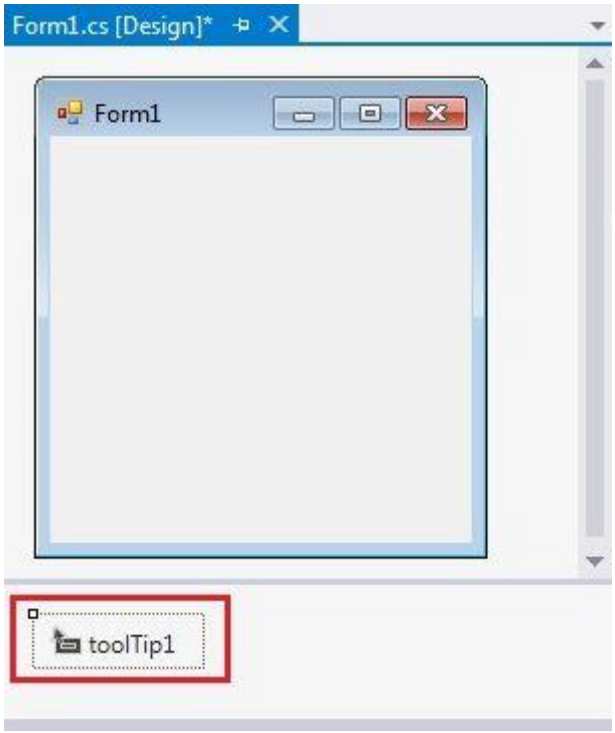
```
private void Form1_Load(object sender, EventArgs e)
{
    //Podesavanje svojstva ColorDepth.
    imageList1.ColorDepth = ColorDepth.Depth32Bit;

    //Ubacujemo sliku iz nekog odredjenog foldera tak sto kopiramo adresu
    //tog foldera i ime slike. U ovom primeru smo ubacili u listu sliku koja
    //se zove slika1 (tip slike je jpg) i koja se nalazi u folderu Pictures.
    imageList1.Images.Add(Image.FromFile("c:\\Pictures\\slika1.jpg"));

    //Uklanjanje slike iz liste se vrsi tako sto se kao argument upisuje
    //indeks one slike koju zelimo da uklonimo. U ovom slucaju uklanja se
    //slika ciji je indeks 0.
    imageList1.Images.RemoveAt(0);
}
```

Компонента ToolTip

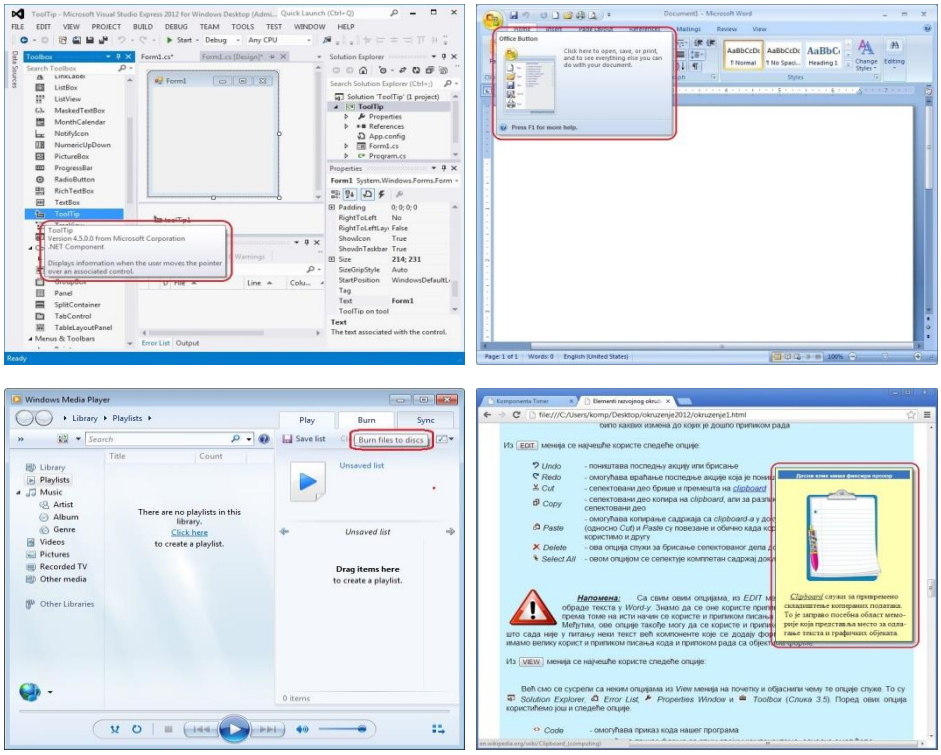
ToolTip компонента се у *Toolbox-y* налази у категорији *Common Controls* и служи за приказивање поруке када се мишем пређе преко неког одређеног објекта. Ова компонента се не види на форми, већ се види на дну *DesignView-a* (*Слика 3.1*).

Ова компонента се јако често користи када желимо да додатно објаснимо неку акцију или уколико желимо да прикажемо поруку која садржи неку врсту упоузорења. Осим текста, може се поставити и нека од понуђених системских иконица. Пример ове компоненте се може наћи и на самој радној површини, ако мишем пређемо преко ToolTip компоненте у Toolbox-у појавиће се порука са додатним информацијама, или на пример у Word-у ако мишем пређемо преко Office дугмета, или у Windows Media Player-у ако пређемо мишем преко Burn картице такође ће се појавити порука са додатним објашњењима или битним информацијама (*Слика 3.2* - кликни на слику да би се увећала). Са овом компонентом смо се већ срели и у овом електронском курсу. У делу *Елементи развојног окружења* када мишем пређемо преко речи *clipboard* појави се порука у којој је додатно објашњен поменути појам (*Слика 3.2*). Ова компонента се користи у комбинацији са другим компонентама, на

пример уколико желимо да се појави нека порука када пређемо мишем преко дугмета користићемо ToolTip компоненту.



Слика 3.1. Изглед компоненте ToolTip



Слика 3.2. Примери ToolTip-а

Придруживање ToolTip компоненте другим компонентама

Постоје два начина на која можемо да придружимо ToolTip компоненту некој другој компоненти. Први начин придруживања је преко кода.

```
private void Form1_Load(object sender, EventArgs e)
{
    toolTip1.SetToolTip(argument1, argument2);
}
```

Приликом писања кода користи се функција SetToolTip која има два аргумента. Први аргумент је име објекта, односно компоненте којој желимо да прикружимо ToolTip компоненту, на пример дугме, лабела, оквир за приказивање слика и друге. Други агумент је текст поруке који желимо да се појави.

Други начин придурживања је коришћењем *Properties Widnows-a.* Неопходно је прво додати ToolTip компоненту форми, селектовати компоненту којој желимо да придружимо ToolTip компоненту, а затим у простору поред својство *ToolTip on toolTip1*, које се налази у Properties Windows-у, написати текст поруке.

Својства која се најчешће користе:

(Name) је својство које представља име компоненте које се користи у коду како би се идентификовала та компонента (тј. објекат). Свака *ToolTip* компонента аутоматски добија име и то према редоследу додавања форми *toolTip1*, *toolTip2*, *toolTip3* и тако даље. Ова имена се могу променити помоћу овог својства, али ми ћемо радити са овим већ унапред задатим именима и нећемо их мењати.

Active је својство помоћу кога се ToolTip компонента активира. Порука која треба да се појави, појавиће се само ако је у овом својству одабрана опција *True*.

Automatic Delay је својство које служи за подешавање вредности за појављивање поруке.

Auto Pop Delay је својство помоћу кога се одређују колико ће дуго порука остати видљива док је курсор миша постављен на региону предвеђеном за приказивање поруке.

Back Color је својство помоћу кога се бира боја позадине области у којој је исписана порука (бира се на исти начин као и позадина форме).

Fore Color је својство помоћу кога се бира боја текста поруке која се појављује (бира се на исти начин као и позадина форме).

Initial Delay је својство помоћу кога се одређују колико дуго курсор миша треба да мирује на објекту како би се порука појавила.

Is Balloon је својство помоћу кога се одређују да ли ће прозор у коме се појављује порука попримити облик облака или не.

Reshow Delay је својство помоћу кога се одређује дужина времена која прође до поновног појављивања поруке ако се курсор миша помера са једног објекта на други.

Tool Tip Icon је својств које служи за одабир системске иконице уколико желимо да се поред текста поруке појави и једна од три понуђене системске иконице (*Info, Warning, Error*).

Tool Tip Title је својств које служи за постављање наслова поруке.

Use Fading је својств које даје ефекат постепеног појављивања или нестајања прозора у коме је исписана порука.

Задавање својстава преко кода

```
private void Form1_Load(object sender, EventArgs e)
{
    //ToolTip se namesta tako sto se za prvi argument funkcije SetToolTip
    //postavi ime objekta za koji zelimo da kada se predje misem preko
    //njega iskoci poruka (button1), a drugi argument je tekst same poruke.
    toolTip1.SetToolTip(button1, "Ovo je Tool Tip poruka!");

    //Aktivacija ToolTip komponente.
    toolTip1.Active = true;

    //Podesavanje vrednosti za pojavljivanje poruke.
    toolTip1.AutomaticDelay = 500;

    //Na ovaj nacin se podesava vremenski period za koji ce poruka ostati
    //vidljiva dok je kursor postavljen na objektu.
    toolTip1.AutoPopDelay = 5000;

    //Podesavanje boje pozadine prozora u kome je ispisana poruka.
    toolTip1.BackColor = Color.White;

    //Podesavanje boje teksta poruke.
    toolTip1.ForeColor = Color.Black;

    //Pomocu ovog koda se podesava koliko dugo kursor misa treba da
    //miruje na objektu kako bi se poruka pojavila.
    toolTip1.InitialDelay = 500;

    //Podesavanje oblkika prozora u kome se pojavljue poruka.
    toolTip1.IsBalloon = true;

    //Podesavanje vremenskog prerioda koji treba da prodje do
    //ponovnog pojavljivanja poruke posle pomeranja kursora.
    toolTip1.ReshowDelay = 100;

    //Postavljanje sistemske ikonice unutar ToolTip prozora.
    toolTip1.ToolTipIcon = ToolTipIcon.Info;

    //Postavljanje naslova poruke.
    toolTip1.ToolTipTitle = "Naslov";

    //Aktivacija efekta postepenog pojavljivanja ili nestajanja
    //prozora u kome je ispisana poruka.
    toolTip1.UseFading = true;
}
```

Након задавања својстава преко кода и након покретања програма прозора у коме се појављује порука изгледа као што је приказано на Слици 3.3.

Слика 3.3. Изглед ToolTip прозора након покретања програма

Догађаји

Догађаји компоненти

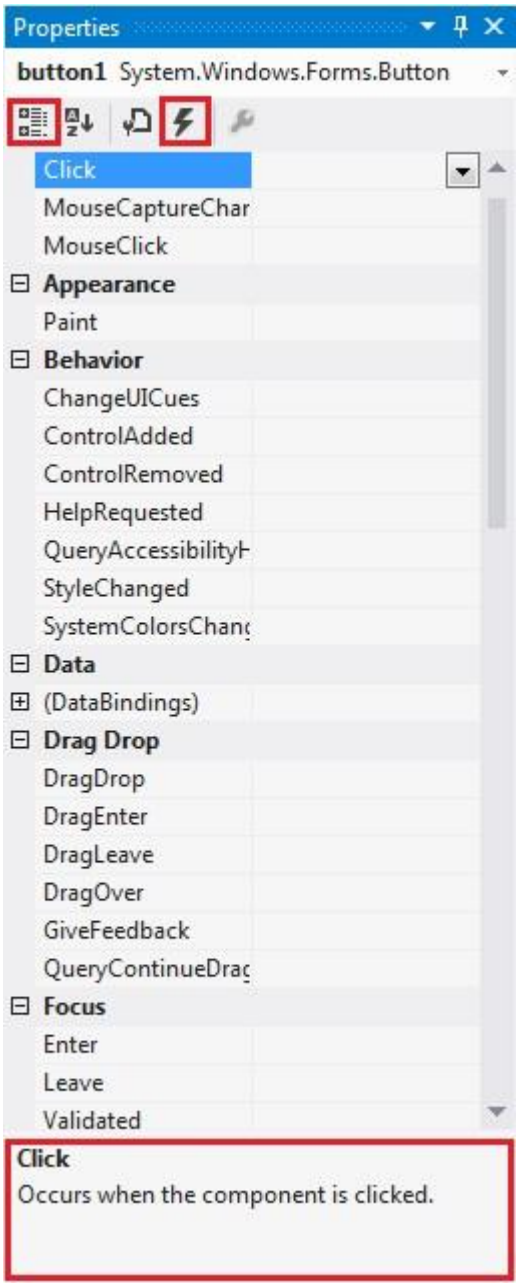
За све наведене компоненте постоје и разноразни догађаји. Већ смо спомињали неке догађаје, а овде ћемо детаљније објаснити шта све може да буде догађај.

**Све што урадимо директном акцијом помоћу миша или тастатуре, али и све што је последица извршавања програма или неког дела програма, може бити догађај.*

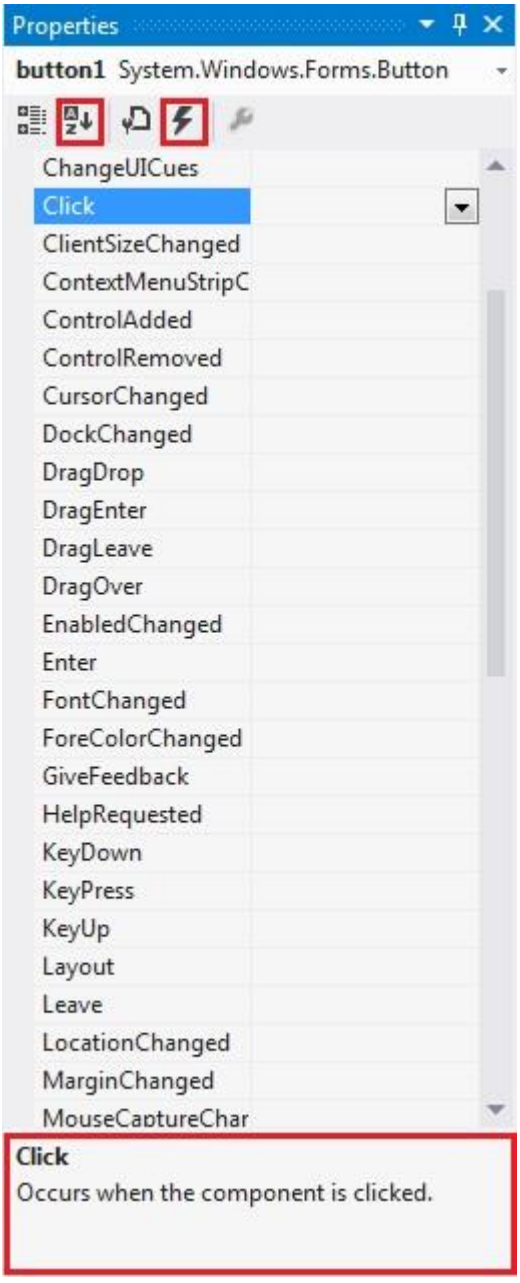
На пример, клик мишем, притисак на неко дугме на тастатури или отпуштање дугмета на тастстури, прелазак показивача миша преко компоненте (тј. објекта), представљају догађаје.

Задатак програма је да одлучи који догађаји ће одговарати којим компонентама и шта ће се дешавати после одређеног догађаја у тој апликацији. Ако ни за један догађај није написан код, онда се у апликацији неће дешавати ништа.

Као што се сва својства компоненти налазе у *Properties Widows-y*, тако се ту налазе и догађаји. Као што прво морамо са форме да селектујемо компоненту којој желимо да подесимо својства, тако морамо исто да селектујемо и компоненту којој желимо да доделимо неки догађај. Да бисмо видели све понуђене догађаје за селектовану компоненту, потребно је да кликнемо на четврту по реду иконицу  . Уколико кликнемо на прву по реду иконицу  , сви догађаји ће бити организовани у разне категорије као што је приказано на Слици 3.1, а уколико желимо да сви догађаји буду организована по абецедном реду, како је приказано на Слици 3.2, онда кликнемо на другу по реду иконицу  . У самом дну *Properties Widows-a* је део у коме се, када се кликне на неки догађај, исписује додатно објашњење о том догађају.



Слика 3.1. Догађаји организовани по категоријама



Слика 3.2. Догађаји организовани по абецедном реду

Напомена: Приликом рада са компонентама, често ћемо користити и својства и догађаје из *Properties Widows-a*. Због овога треба да обратимо пажњу на иконицу , која пружа приказ свих својстава компоненте, и , која пружа приказ свих догађаја компоненте. Кад год је потребно да се у *Properties Widows-y* прикажу својства, кликнемо на иконицу , а кад год је потребно да се у *Properties Widows-y* прикажу догађаји компоненти, кликнемо на иконицу .

Не реагују све компоненте на исти број догађаја. Неке компоненте могу да реагују на велики број догађаја, док неке реагују на врло мали број догађаја.

Догађаји који се најчешће користе

Приликом рада са компонентама *Button*, *Label*, *TextBox*, *RichTextBox*, *PictureBox* и приликом рада са формом, најчешће се користе следећи догађаји.

Click - клик на објекат (тај објекат може бити нека од ових пет наведених компоненти, или форма)

DoubleClick - двоструки клик на објекат

MouseClick - када се кликне мишем на објекат (нема суштинске разлике између овог догађаја и догађаја *Click*)

DoubleMouseClick - двоструки клик мишем на објекат

MouseDown - када је показивач миша на објекту и леви тастер на мишу је стистнут

MouseMove - када показивачем миша пређемо преко објекта

MouseUp - када је показивач миша на објекту и леви тастер миша се отпусти

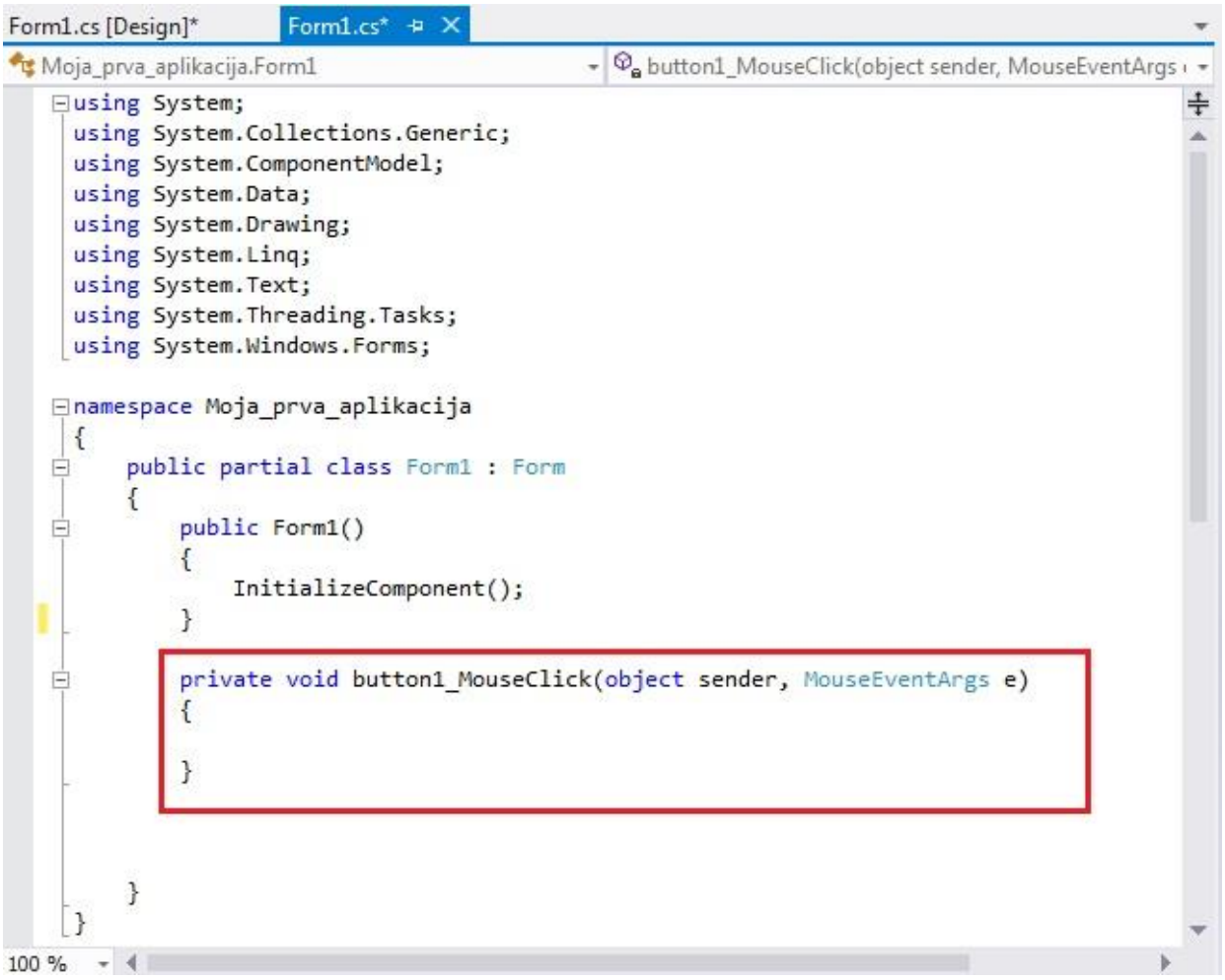
Load - када се покрене програм

Ово су само неки догађаји које ћемо за почетак да користимо.

За разлику од ових компоненти и од форме, компонента *Timer* има само један догађај.

Tick - деси се нека акција када прође одређени временски интервал и та акција се периодично понавља

Уколико желимо да придружимо неки догађај некој компоненти, прво селектујемо ту компоненту, а затим у *Properties Windows-y* кликнемо два пута на догађај који желимо да придружимо тој компоненти. Када кликнемо два пута на тај догађај појавиће се део кода, баш као када смо мењали својства компоненти преко кода. На пример, желимо компоненти *Button* да придружимо догађај *MouseClick* (клик мишем на дугме), па зато кликнемо два пута на догађај *MouseClick* у *Properties Windows-y*. Појавиће се део кода који је приказан на Слици 3.3.



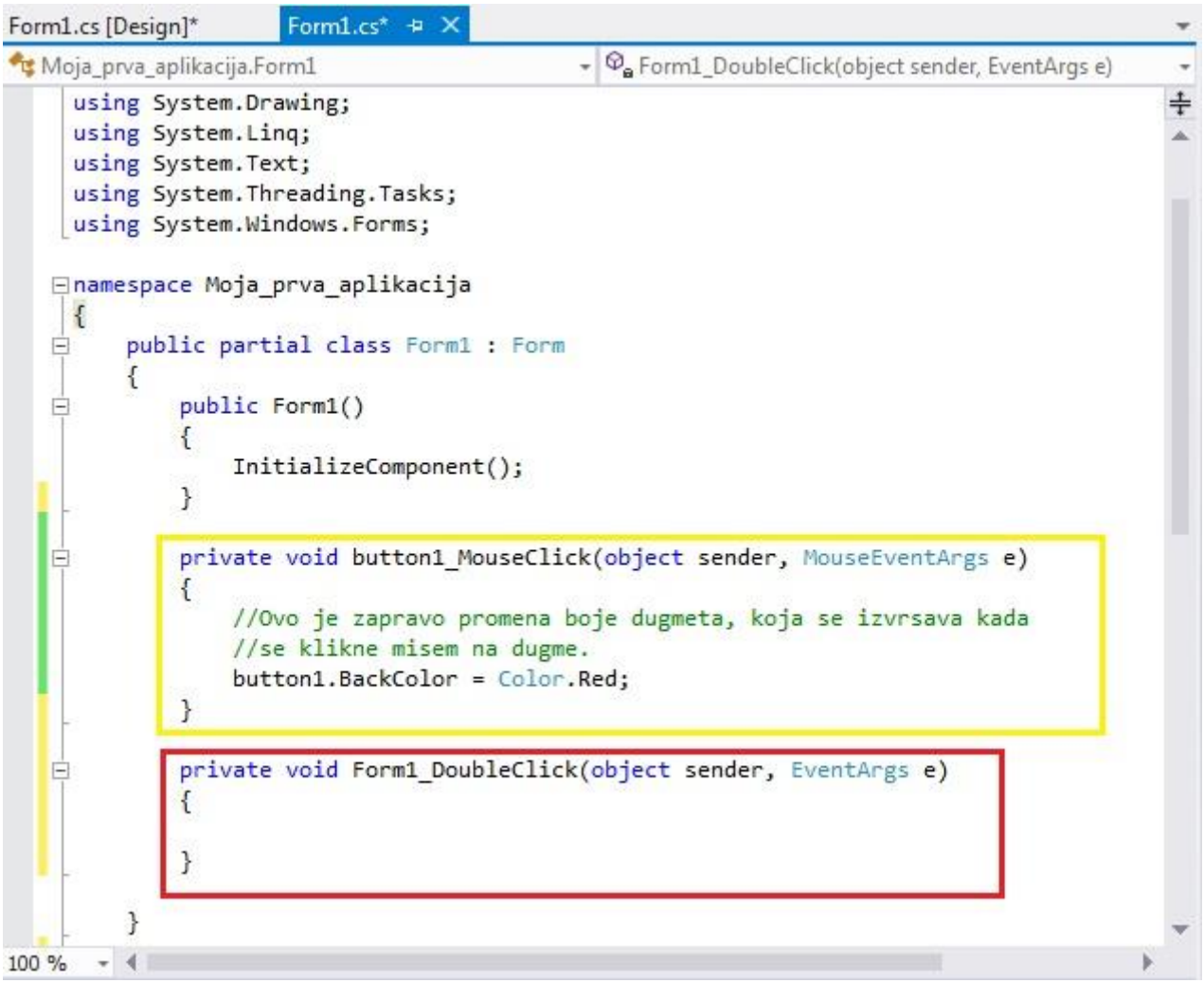
Слика 3.3. Унапред генерисан код

Између заграда уоквиреног кода, пишемо део програма који ће се извршити када мишем кликнемо на дугме. На пример, кликом миша на дугме, дугме мења боју у црвено. Део програма који одговара овој акцији и који се пише на поменутом месту изгледа овако:

```
private void button1_MouseClick(object sender, MouseEventArgs e)
{
    //Ovo je zapravo promena boje dugmeta, koja se izvrsava kada
    //se klikne misem na dugme
    button1.BackColor = Color.Red;
}
```

У једном пројекту ће се појављивати по неколико догађаја, па можемо да обрадимо још један догађај и у овом нашем примеру. На пример, када кликнемо два пута на форму, у лабели, коју смо додали форми, се испише порука *Кликнули сте два пута на форму!* и то црвеном бојом.

Пошто ова акција треба да се изврши када се кликне два пута на форму, онда селиктујемо форму и два пута кликнемо на догађај *DoubleClick* у *Properties Windows-y*. Одмах испод дела кода који одговара предходном догађају који смо обрадили ће се појавити део кода везан за догађај *DoubleClick* (*Слика 3.4*).



Слика 3.4. Програмски код

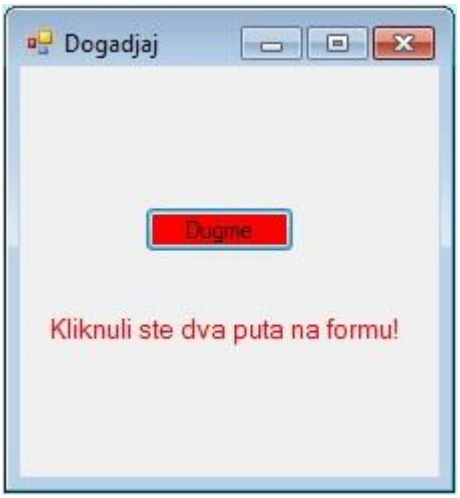
На Слици 3.4 жутом бојом је уоквирен код који одговара догађају који смо већ обрадили, а црвеном бојом је уоквирен код који одговара догађају *DoubleClick*. Део програма који омогућава извршавање акције треба да изгледа овако:

```
private void Form1_DoubleClick(object sender, EventArgs e)
{
    //Ovo je zapravo promena svojstva Text labele koja se izvrsava
    //kada se dva puta klikne na formu
    label1.Text = "Kliknuli ste dva puta na formu!";

    //Ovo je zapravo promena boje teksta, koji se ispisuje u labeli,
    //koja se izvrsava kada se dva puta klikne na formu
    label1.ForeColor = Color.Red;

}
```

Када сачувамо измене можемо да покренемо програм. После покретања програма форма изгледа као на Слици 3.5. Када се мишем кликне на дугме форма ће изгледати као на Слици 3.6, а када се два пута кликне на форму, форма ће изгледати као што је приказано на Слици 3.7.

Слика 3.5. Изглед апликације након покретања програма

Слика 3.6. Изглед апликације након клика на дугме



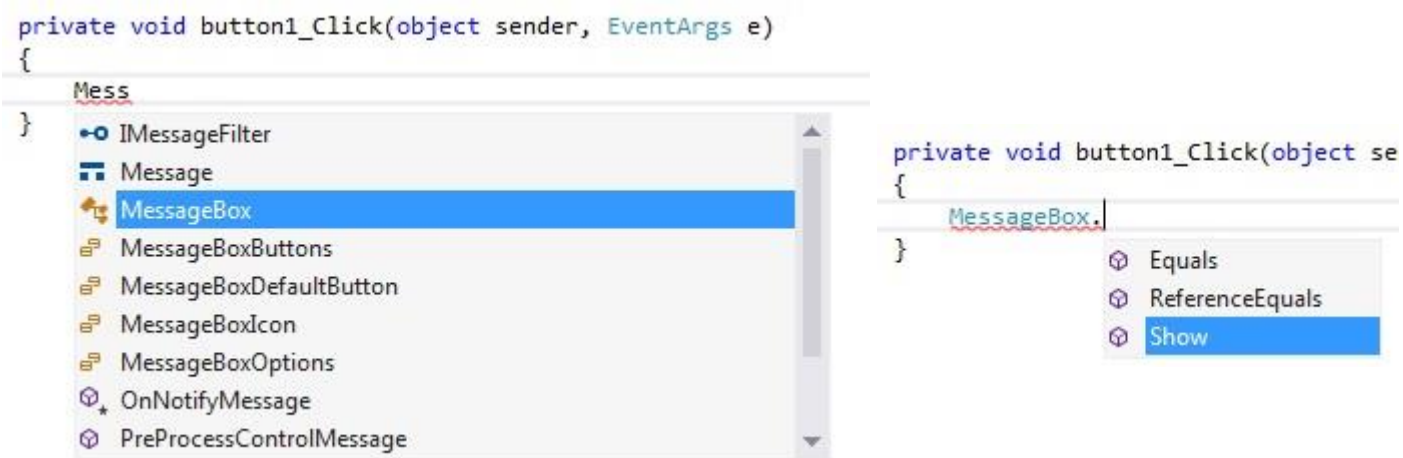
Слика 3.7. Изглед апликације након двоструког клика на форму

Напомена: Уколико желимо да догађај *Click* придружимо некој компоненти, не морамо да два пута кликнемо на догађај *Click* у *Properties Windows-y*, већ кликнемо два пута на компоненту на форми којој желимо да придружимо тај догађај. Међутим, уколико кликнемо два пута на форму, онда ћемо форми придружити догађај *Load*.

Примери

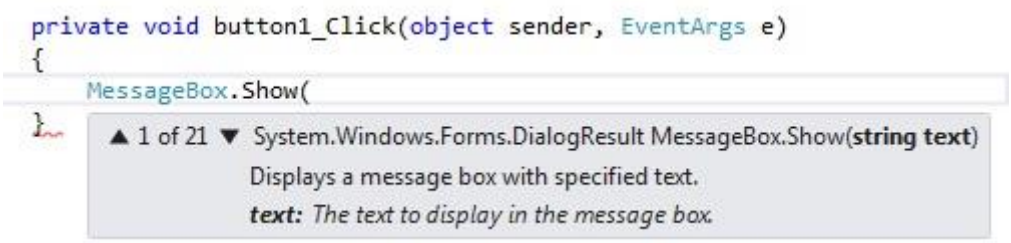
Пример 1. **Написати програм који притиском на дугме PORUKA исписује поруку *Ovo je moja prva aplikacija!* у *MessageBox-y.***

Овај први пример ћемо детаљније објаснити. Прво форми треба да додамо компоненту дугме. Након тога у *Properties Windows-y* треба да променимо својство *Text* компоненте *Button* тако да на њему буде исписано PORUKA. Пошто желимо да се *MessageBox* са исписаном поруком појави када кликнемо на дугме PORUKA, онда прво кликнемо два пута на дугме да би му придружили догађај *Click*. Одмах ће се отворити део кода и курсор ће бити постављен на место на коме треба да куцамо потребан код помоћу кога ће се извршити оно што се тражи у задатку. Напишемо ли неколико почетних слова појавиће нам се листа у којој селектујемо *MessageBox* (*Слика 3.1*).



Слика 3.1. Падајућа листа *Слика 3.2. Падајућа листа*

После овог се ставља тачка и опет ће се појавити листа са понуђеним методама (*Слика 3.2*). Ми ћемо изабрати *Show* методу. Ова метода може да има више аргумената у заградама, или само један, тако да сада отворимо заграду и појавиће се уоквирено објашњење које описује који то све аргументи могу бити и како се они пишу. На почетку се налазе мале стрелице на које када се кликне приказују се различити аргументи *MessageBox-a* (*Слика 3.3*).



Слика 3.3. Оквир са објашњењима

Ова објашњења могу да буду јако корисна и треба се навићи да их користимо. Наш аргумент ће бити текст који треба да се појави, а сваки текст у *C Sharp-y* се пише под наводницима, тако да ће на крају наш код изгледати као на Слици 3.4.

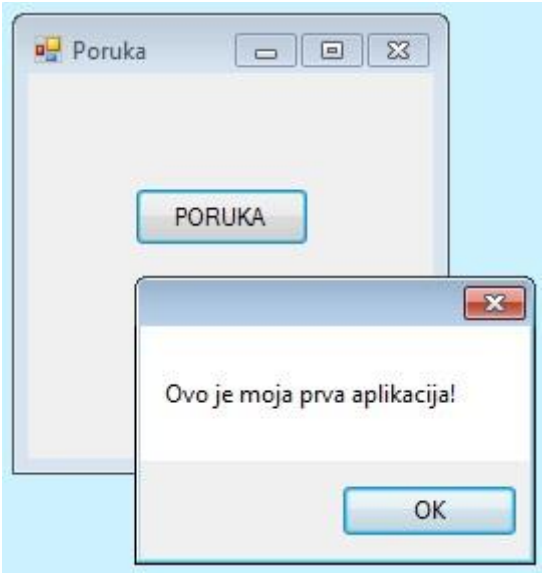
```
private void button1_Click(object sender, EventArgs e)
{
    //Ispisujemo poruku u MessageBox-u.
    MessageBox.Show("Ovo je moja prva aplikacija!");
}
```

Слика 3.4. Програмски код

После писања кода и након што сачувамо измене, можемо покренути програм.

Слика 3.5. Изглед апликације након покретања програма

Након покретања програма појавиће се апликација која има једно дугме (*Слика 3.5*), на које када се кликне појави се *MessageBox* са поруком (*Слика 3.6*). *MessageBox*, поред исписане поруке, садржи дугме *OK*. Кликом на дугме OK *MessageBox* се затвара.



Слика 3.6. Изглед апликације након клика на дугме

Пример 2. **Написати програм који притиском на дугме PORUKA исписује питање *Da li je ovo Vasa prva aplikacija?* у *MessageBox-y*. *MessageBox* треба да има два дугмета *Yes* и *No*, име и иконицу.**

Овај пример је јако сличан претходном примеру. Понављамо поступак као из претходног примера, али сада уместо само једног аргумента у *Show* методи ће бити више аргумената (после сваког аргумента се пише зарез).

```
private void button1_Click(object sender, EventArgs e)
{
    MessageBox.Show(
}
```

▲ 7 of 21 ▼ System.Windows.Forms.DialogResult MessageBox.Show(**string text**, string caption, MessageBoxButtons buttons, MessageBoxIcon icon)
Displays a message box with specified text, caption, buttons, and icon.
text: *The text to display in the message box.*

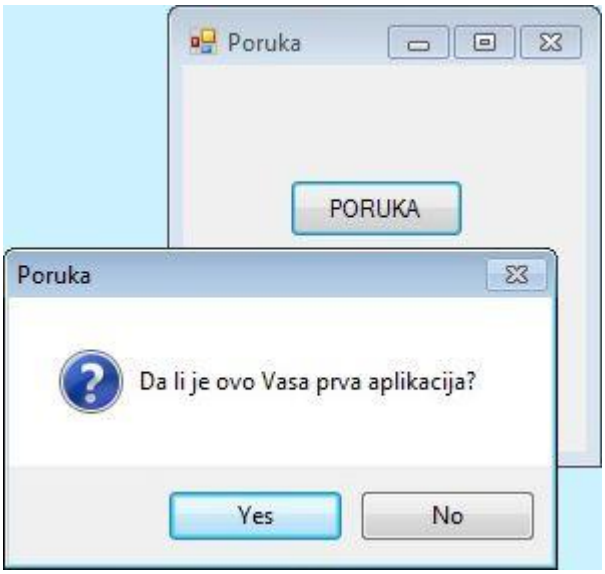
Слика 3.7. Аргументи MessageBox-а

Потребно је да кликнемо неколико пута на другу по реду стрелицу, односно све док се не појави седми облик исписивања аргумената (*Слика 3.7*). Први аргумент је порука, други је име *MessageBox-а*, трећи је наредба за приказивање дугмића у *MessageBox-у* и четврта је наредба за приказивање иконице у *MessageBox-у*. На крају, код ће изгледати као што је приказано на Слици 3.8.

```
private void button1_Click(object sender, EventArgs e)
{
    //Ispisujemo poruku u MessageBox-u, koji ima ime, dva dugmeta i
    //ikonicu.
    MessageBox.Show("Da li je ovo Vasa prva aplikacija?", "Poruka",
                    MessageBoxButtons.YesNo, MessageBoxIcon.Question);
}
```

Слика 3.8. Програмски код

Слика 3.10. Изглед апликације након клика на дугме



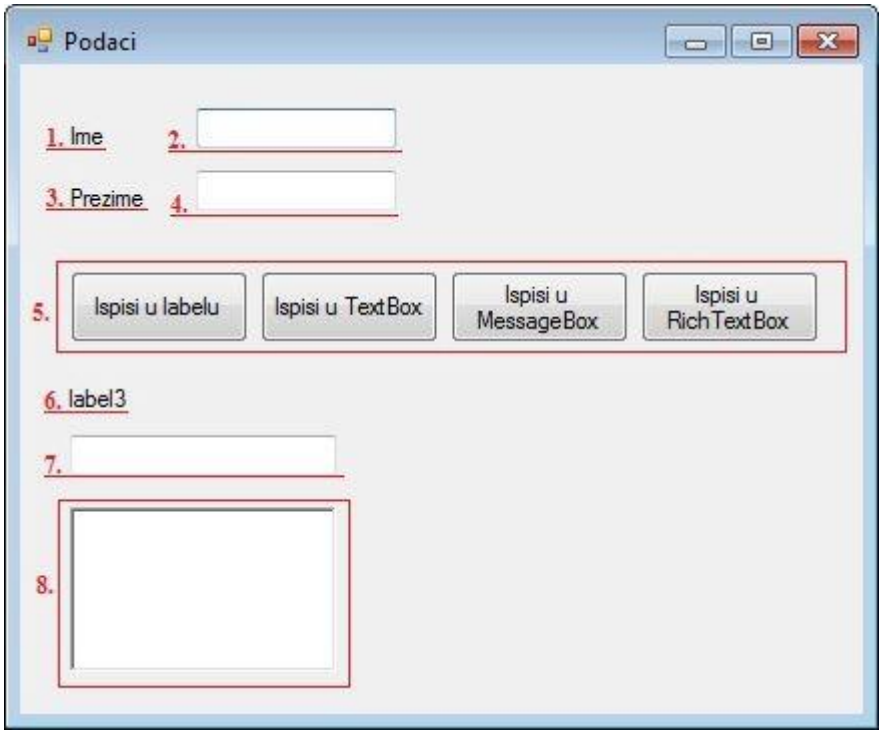
Слика 3.9. Изглед апликације након покретања програма

После покретања програма појавиће се апликација која има једно дугме (*Слика 3.9*), на које када се кликне појави се *MessageBox* у коме је исписана порука. *MessageBox* има два дугмета *Yes* и *No*, име *Poruka* и иконицу знак питања као што се може видети на Слици 3.10.

Пример 3. **Написати програм који има два *TextBox-a* и у први се уноси име, а у други презиме особе.**

1. **Кликом на дугме *Ispisi u labelu* се исписује унето име и презиме у лабелу.**
2. **Кликом на дугме *Ispisi u textBox* се исписује унето име и презиме у TextBox.**
3. **Кликом на дугме *Ispisi u MessageBox* се исписује унето име и презиме у MessageBox.**
4. **Кликом на дугме *Ispisi u RichTextBox* се исписује унето име и презиме у RichTextBox.**

У овом примеру ћемо видети различите начине исписивања података унетих са тастатуре.



Слика 3.11. Изглед форме у примеру 3

Потребно је да додамо следеће компоненте форми:

1. label1- у својству *Text* напишемо текст Ime
2. textBox1- овде ћемо са тастатуре унети име
3. label2-у својству *Text* напишемо текст Prezime
4. textBox2- овде ћемо са тастатуре унети презиме
5. Четири компоненте Button- свакој променимо својство *Text* тако да натписи буду као што је приказано
6. label3- у ову лабелу ће се исписати подаци када се кликне на прво дугме
7. textBox3- у ово поље ће се исписати подаци када се кликне на друго дугме
8. richTextBox1- у ово поље ће се исписати подаци када с кликне четврто дугме

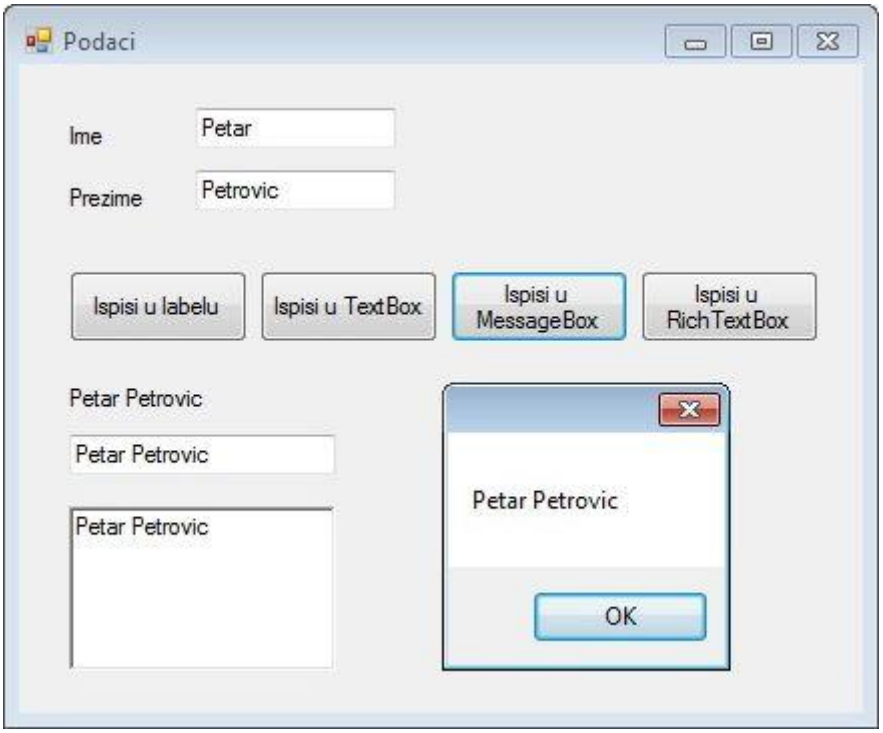
Компоненте можемо да распоредимо по форми као што је приказано на Слици 3.11.

Након овога следи писање кода. Два пута кликнемо на свако дугме да би им придружили догађај *Click*, а онда пишемо одговарајући код за свако дугме.

```
private void button1_Click(object sender, EventArgs e)
{
    //Na ovaj nacin se u label3 ispisuje prvo ono sto pise u prvom TextBox-u pa onda
pored toga
    //ono sto pise u drugom TextBox-u. Medjutim, izmedju imena i prezimena treba da
stoji razmak
    //zato dodajemo prazno msto izmedju znakova navodnika. Znak plus je neophodan jer
se pmocu
    //njega povezuju ove naredbe.
    label3.Text = textBox1.Text + " " + textBox2.Text;
}

private void button2_Click(object sender, EventArgs e)
{
    //Na ovaj nacin se u textBox3 ispisuje isti ovaj sadrzaj.
    textBox3.Text = textBox1.Text + " " + textBox2.Text;
```

```
}

private void button3_Click(object sender, EventArgs e)
{
    //Na ovaj nacin se u MessageBox ispisuje isti ovaj sadrzaj.
    MessageBox.Show(textBox1.Text + " " + textBox2.Text);
}

private void button4_Click(object sender, EventArgs e)
{
    //Na ovaj nacin se u richTextBox1 ispisuje isti ovaj sadrzaj.
    richTextBox1.Text = textBox1.Text + " " + textBox2.Text;
}
```

Након уношења података, и пошто кликнемо на свако дугме, наша апликација треба да изгледа као што је приказано на Слици 3.12.



Слика 3.12. Изглед апликације у примеру 3

Пример 4. **Написати програм који има шест *TextBox* компоненти у које ће са тастатуре да се уносе одређени подаци (име, презиме, град, улица и број, број телефона), кликом на дугме NARUCI се подаци исписују у richTextBox1, а кликом на дугме POTVRDI се појављује *MessageBox* са исписаном поруком: *Vasa porudzbina je uspesno poslata*.**

Слика 3.13. Изглед форме у примеру 4

Потребно је да додамо следеће компоненте форми:

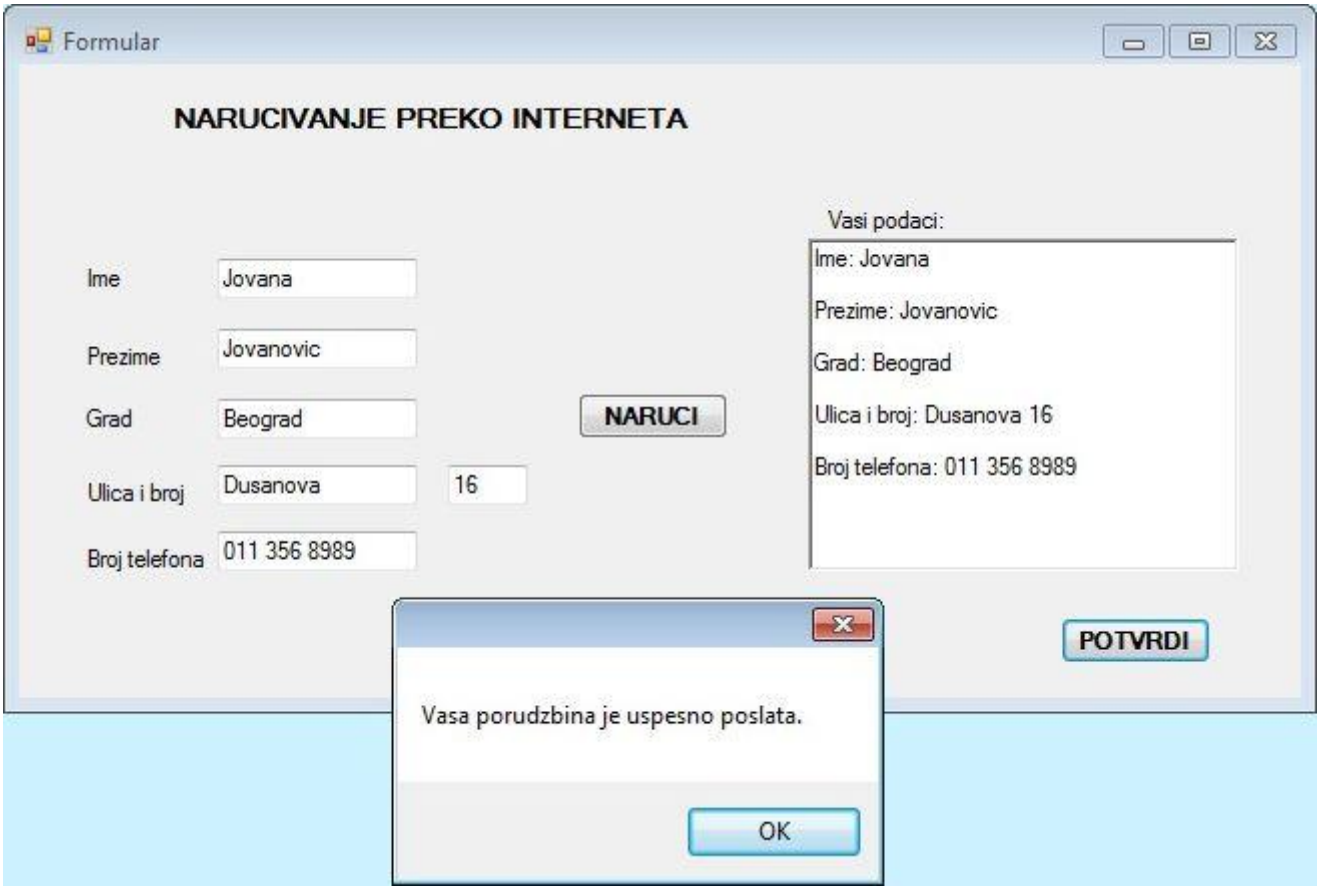
1. label1- служи као наслов и у својству *Text* је написан тај наслов, а у својству *Font* је подсвојство *Bold* постављено на *True*
2. Пет компоненти *Label* којима је у својству *Text* исписан одговарајући текст као што је приказано и служе да се наговести шта треба да се упише у текстуална поља која стоје поред сваке лабеле
3. Шест TextBox компоненти које служе за унос података са тастатуре
4. button1- у својству *Text* је промењено име дугмета
5. label7- у својству *Text* је постављен текст Vasi podaci
6. richTextBox1- у њему ћемо исписати податке
7. button2- у својству *Text* је промењено име дугмета

Након овога следи писање кода. Два пута кликнемо на дугме NARUCI и напишемо одговарајући код за то дугме, а затим два пута кликнемо на дугме POTVRDI и напишемо одговарајући код за то дугме.

```
private void button1_Click(object sender, EventArgs e)
{
    //Na ovaj nacin kada kliknemo na dugme NARUCI u richTextBox1 upisujemo sve podatke.
Pisali
    //smo u vise redova zato sto bi u jednom redu bilo previse dugacko. Ovde nam se
pojavljuje
    //na vise mesta znak \n pod znacima navodnika, on sluzi za to da se posle njega
sledeca
    //naredba ispisuje u novom redu u richTextBox1.
    richTextBox1.Text = "Ime: " + textBox1.Text + "\n" + "\n" +
                        "Prezime: " + textBox2.Text + "\n" + "\n" +
                        "Grad: " + textBox3.Text + "\n" + "\n" +
                        "Ulica i broj: " + textBox4.Text + " " + textBox5.Text + "\n" +
"\n" +
                        "Broj telefona: " + textBox6.Text;
}
```

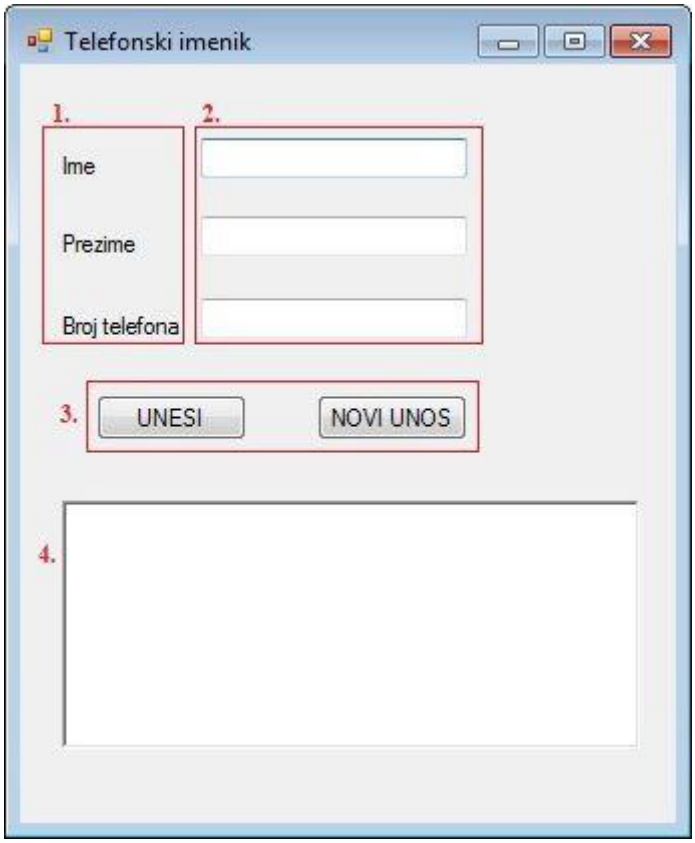
```
private void button2_Click(object sender, EventArgs e)
{
    //Kada kliknemo na dugme POTVRDI pojavljuje se poruka u MessageBox-u.
    MessageBox.Show("Vasa porudzbina je uspesno poslata.");

    //Brisemo tekst iz svake TextBox komponente kako bi bile spremne za novi unos.
    textBox1.Text = "";
    textBox2.Text = "";
    textBox3.Text = "";
    textBox4.Text = "";
    textBox5.Text = "";
    textBox6.Text = "";

    //Brisemo tekst iz RichTextBox komponente kako bi bio spreman za novi unos.
    richTextBox1.Text = "";
}
```

Након уношења података, и пошто кликнемо прво на дугме NARUCI, а онда на дугме POTVRDI, наша апликација треба да изгледа као што је приказано на Слици 3.14.



Слика 3.14. Изглед апликације у примеру 4

Пример 5. **Написати програм *Телефонски именик*. Омогућити унос имена, презимена и броја телефона (за свако посебно поље за унос текста). Кликом на дугме UNESI у RichTextBox се исписују подаци. Кликом на дугме NOVI UNOS поља за унос се празне, а у RichTextBox се исписује нови унос одмах после старог уноса (значи претходни унос остаје исписан).**



Слика 3.15. Изглед форме у примеру 4

Потребно је да додамо следеће компоненте форми:

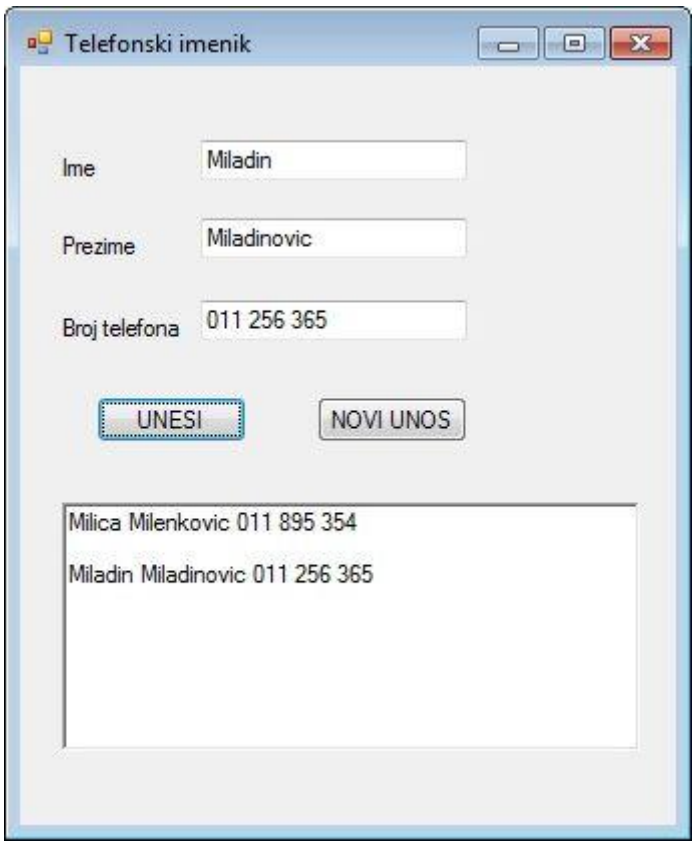
1. Три лабеле- label1, label2, label3 које ћемо преименовати у Ime, Prezime и Borj telefona
2. Три поља за унос текста- textBox1, textBox2, textBox3 која ће нам служити за унос података
3. Два дугмета- button1, button2 која ћемо преименовати у UNESI и NOVI UNOS, прво дугме ће служити за исписивање унетих података у richTextBox1, а када се кликне на друго дугме испразниће се све три TextBox компоненте
4. Један RichTextBox- richTextBox1 у коме ће се исписати унети подаци

Након овога следи писање кода. Два пута кликнемо на дугме UNESI и напишемо одговарајући код за то дугме, а затим два пута кликнемо на дугме NOVI UNOS и напишемо одговарајући код за то дугме.

```
private void button1_Click(object sender, EventArgs e)
{
    //Kada kliknemo na dugme UNESI, u richTextBox1 se ispisuje prvo ono sto je vec
pisalo
    //u njemu pa tek onda ono sto pise u ova tri TextBox-a. Na samom pocetku je
richTextBox1
    //prazan, ali u drugom unosu se prvo ispise ono sto smo uneli u prvom unosu pa onda
ono
    //iz drugog unosa. U trecem unosu se prvo ispise ono sto smo uneli u prva dva unosa
pa
    //onda ono iz treceg unosa i tako dalje.
    richTextBox1.Text = richTextBox1.Text + textBox1.Text + " " + textBox2.Text + " " +
                        textBox3.Text + "\n\n";
}

private void button2_Click(object sender, EventArgs e)
{
    //Kada kliknemo na dugme NOVI UNOS na ovaj nacin iz polja
    //textBox1 brisemo trenutni unos i posle ovoga polje ostaje
    //prazno kao sto je bilo na pocetku.
```

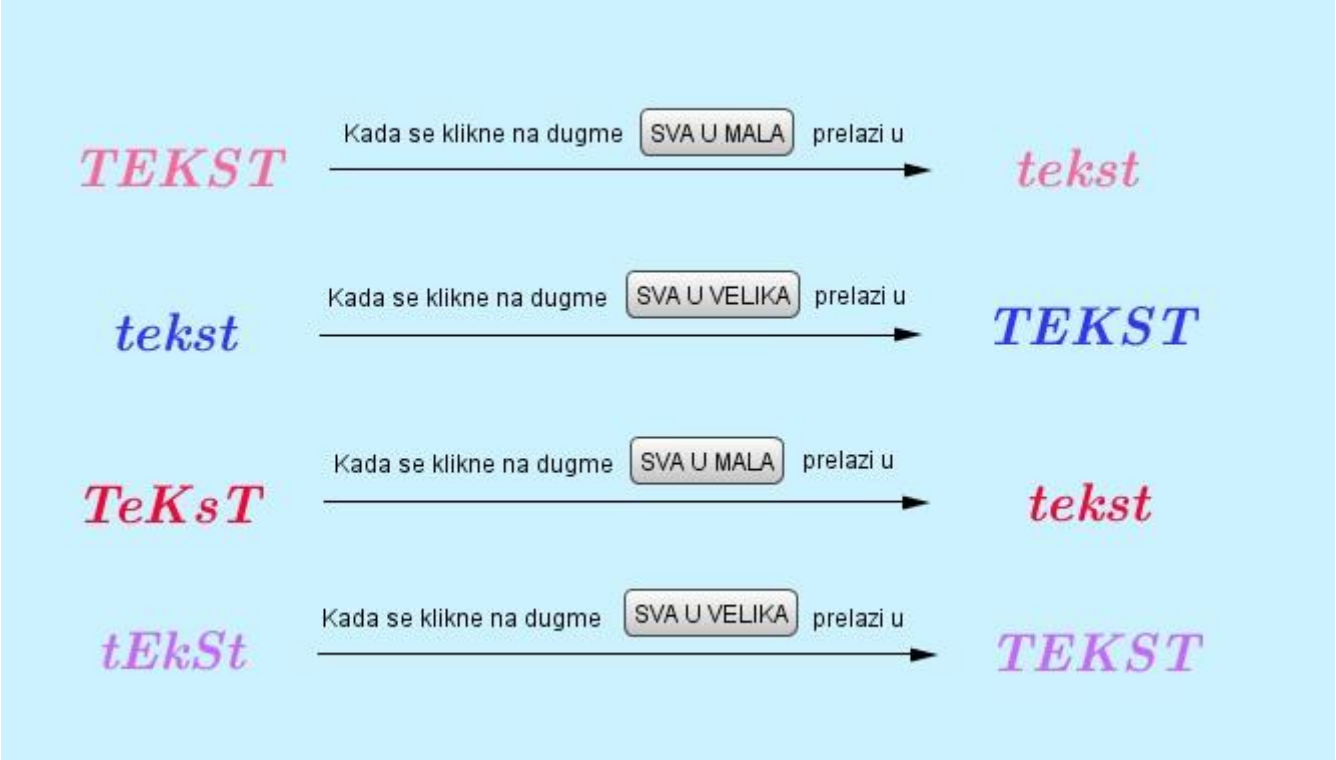
```
    textBox1.Text = "";

    //Kada kliknemo na dugme NOVI UNOS na ovaj nacin iz polja
    //textBox2 brisemo trenutni unos.
    textBox2.Text = "";

    //Kada kliknemo na dugme NOVI UNOS na ovaj nacin iz polja
    //textBox3 brisemo trenutni unos.
    textBox3.Text = "";
}
```

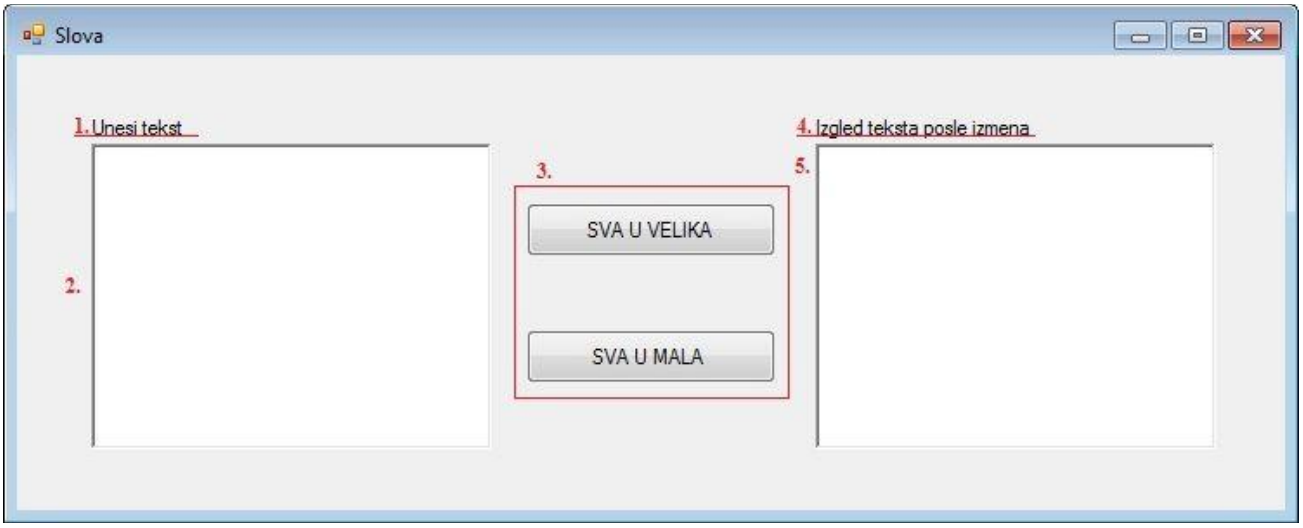
Након уношења података, и пошто кликнемо прво на дугме UNESI, па онда на дугме NOVI UNOS и унесемо нове податке, а онда пново кликнемо на дугме UNESI , наша апликација треба да изгледа као што је приказано на Слици 3.15.



Слика 3.16. Изглед апликације у примеру 5

Пример 6. **Написати програм који кликом на дугме SVA U VELIKA претвара сва слова текста исписаног у richTextBox1 у велика и исписује их у richTextBox2, а кликом на дугме SVA U MALA претвара сва слова у мала и исписује их у richTextBox2 (у првом RichTextBox-у остаје оригиналан текст, а у другом се исписује у измењеном облику).**





Слика 3.17. Изглед форме у примеру 6

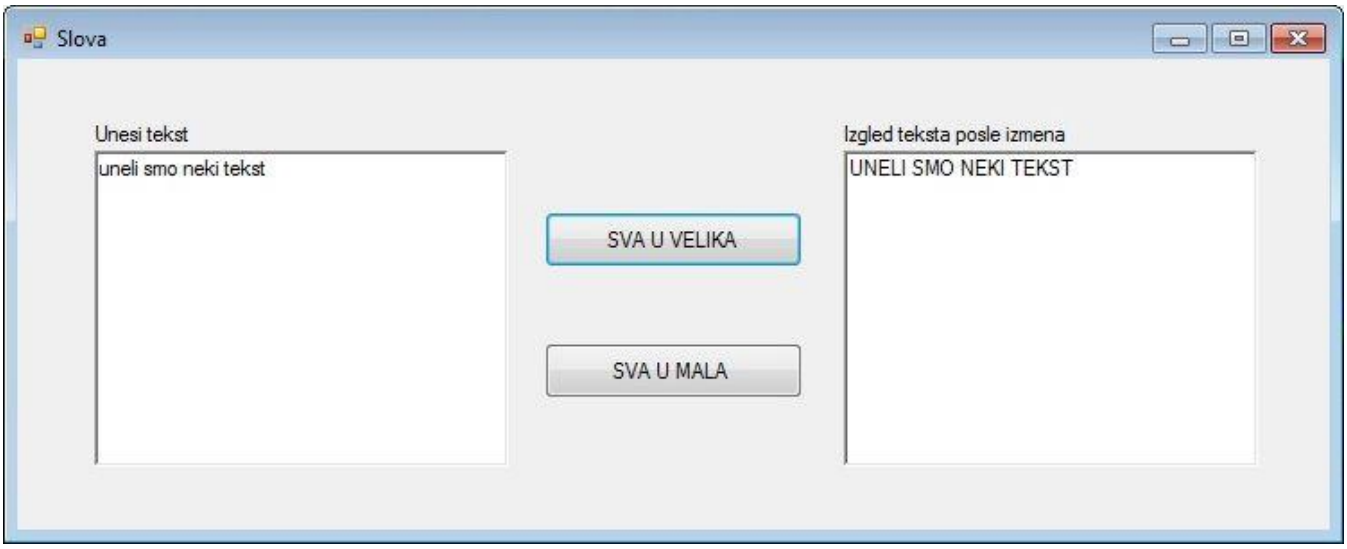
Потребно је да додамо следеће компоненте форми:

1. label1- служиће да објасни шта треба да се унесе у richTextBox1
2. richTextBox1- који ће служити за унос текста
3. button1, button2- прво дугме ће да претвара сва слова у велика, друго сва слова у мала
4. label2- служиће да објасни шта ће се десити у компоненти richTextBox2
5. richTextBox2- у њему ће се исписивати текст у новом облику

Након овога следи писање кода. Два пута кликнемо на дугме SVA U VELIKA и напишемо одговарајући код за то дугме, а затим два пута кликнемо на дугме SVA U MALA и напишемо одговарајући код за то дугме.

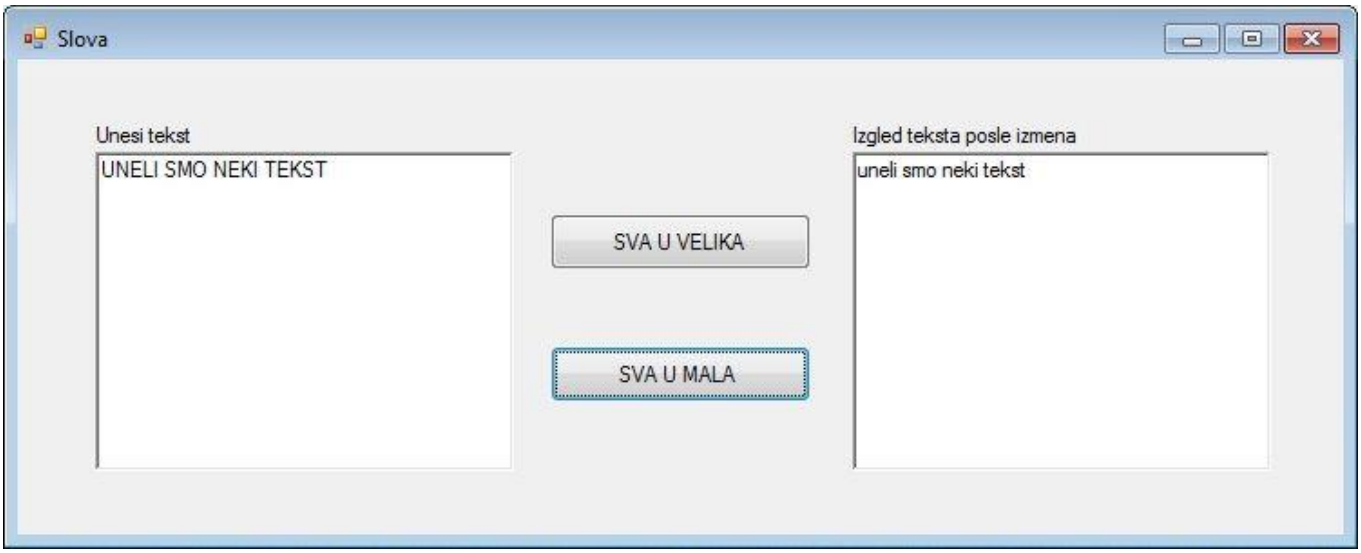
```
private void button1_Click(object sender, EventArgs e)
{
    //Klikom na dugme SVA U VELIKA u richTextBox2 se upisuje tekst iz okvira
richTextBox1
    //i pretvara sva slova u velika.
    richTextBox2.Text = richTextBox1.Text.ToUpper();
}

private void button2_Click(object sender, EventArgs e)
{
    //Klikom na dugme SVA U VMALA u richTextBox2 se upisuje tekst iz okvira
richTextBox1
    //i pretvara sva slova u mala.
     richTextBox2.Text = richTextBox1.Text.ToLower();
}
```

Након уношења података, и пошто кликнемо на дугме SVA U VELIKA, наша апликација треба да изгледа као што је приказано на Слици 3.18.



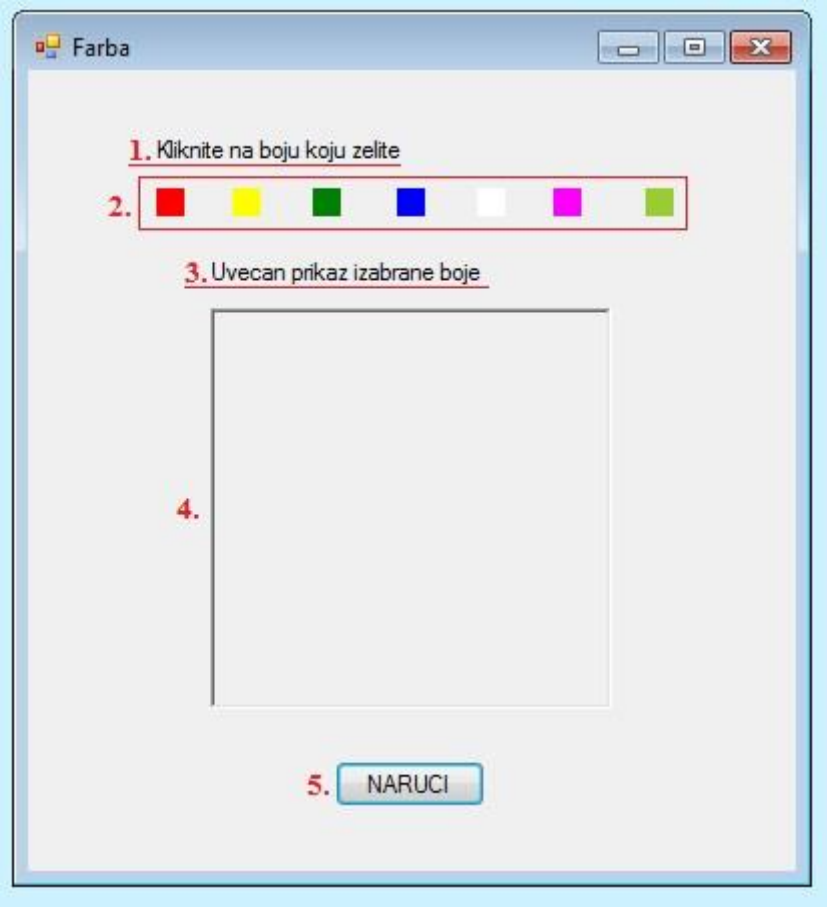
Слика 3.18. Изглед апликације у примеру 6

Уколико кликнемо на дугме SVA U MALA, слова унетог текста која су била велика ће се претворити у мала као што је приказано на Слици 3.19.



Слика 3.19. Изглед апликације у примеру 6

Пример 7. **Написати програм који има приказаних 7 боја на које кад се кликне у новом PictureBox-у се појављује увећан приказ те боје. Кликом на дугме NARUCI исписује се порука *За потврду кликните на ОК да бисте наручили фарбу!* у MessageBox-у (MessageBox треба да има два дугмета ОК и Cancel).**



Слика 3.20. Изглед форме у примеру 7

Потребно је да додамо следеће компоненте форми:

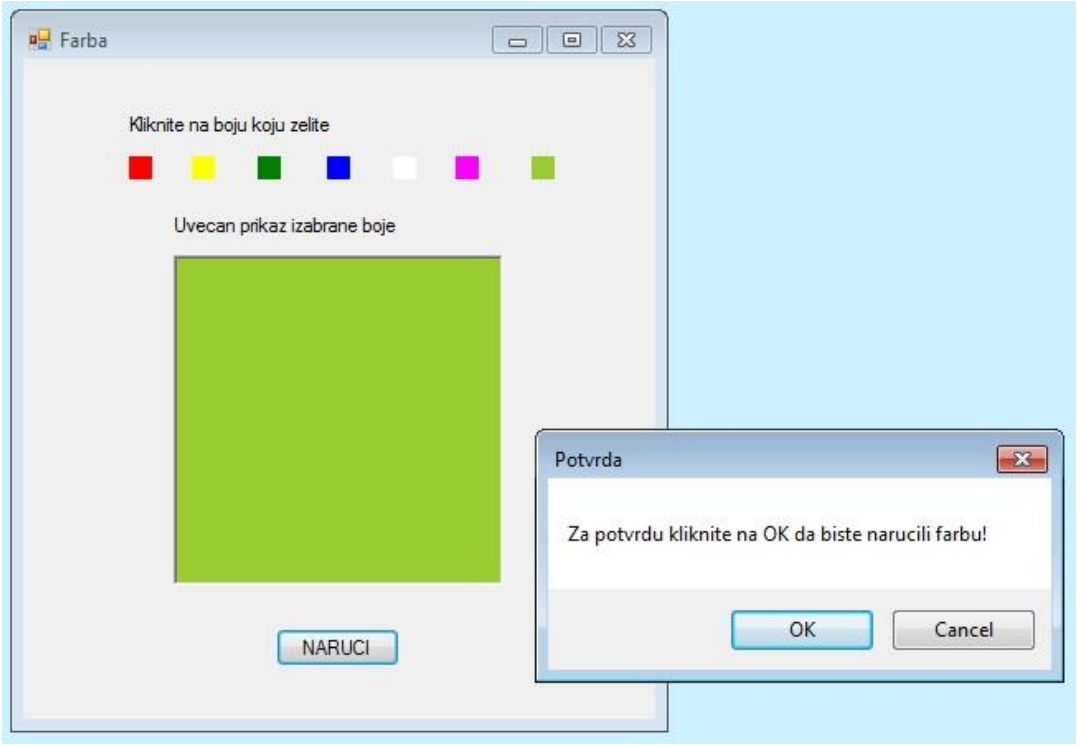
1. label1- служиће да објасни шта треба корисник да уради
2. седам PictureBox компоненти, свакој подесити својства *Size* у *PropetriesWindows-y* тако да буду малих димензија, свакој подесити својство *BackColor* тако да буду различитих боја и свакој подесити својство *Cursor* тако да када се пређе мишем преко сваке курсор поприми облик руке.
3. label2- служи да објасни где ће да се појави увећани приказ боје на коју смо кликнули
4. pictureBox8- служи за увећани приказ боје на коју смо кликнули, поставити да својство *BorderStyle* у *PropetriesWindows-y* буде Fixed3D
5. button1- када кликнемо на дугме појавиће се MessageBox

Након овога следи писање кода. Два пута кликнемо на сваки од седам малих PictureBox компоненти и напишемо одговарајући код за сваку, а затим два пута кликнемо на дугме NARUCI и напишемо одговарајући код за то дугме.

```
private void pictureBox1_Click(object sender, EventArgs e)
{
    //Menjamo boju pozadine komponente pictureBox8.
    pictureBox8.BackColor = Color.Red;
}
```

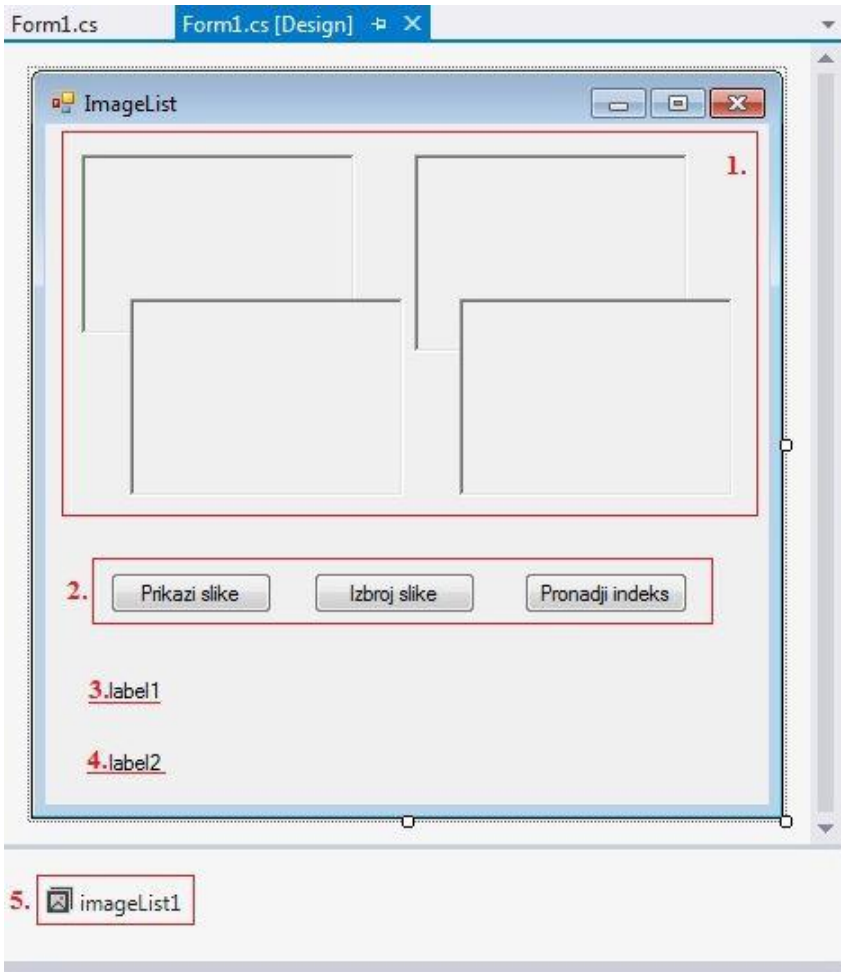
```
private void pictureBox2_Click(object sender, EventArgs e)
{
    //Menjamo boju pozadine komponente pictureBox8.
    pictureBox8.BackColor = Color.Yellow;
}

private void pictureBox3_Click(object sender, EventArgs e)
{
    //Menjamo boju pozadine komponente pictureBox8.
    pictureBox8.BackColor = Color.Green;
}

private void pictureBox4_Click(object sender, EventArgs e)
{
    //Menjamo boju pozadine komponente pictureBox8.
    pictureBox8.BackColor = Color.Blue;
}

private void pictureBox5_Click(object sender, EventArgs e)
{
    //Menjamo boju pozadine komponente pictureBox8.
    pictureBox8.BackColor = Color.White;
}

private void pictureBox7_Click(object sender, EventArgs e)
{
    //Menjamo boju pozadine komponente pictureBox8.
    pictureBox8.BackColor = Color.Fuchsia;
}

private void pictureBox6_Click(object sender, EventArgs e)
{
    //Menjamo boju pozadine komponente pictureBox8.
    pictureBox8.BackColor = Color.YellowGreen;
}

private void button1_Click(object sender, EventArgs e)
{
    //Ispisujemo poruku u MessageBox-u (prvi argument u zagradama), ime
    //MessageBox-a (drugi argument) i postavljamo dugmice OK i Cancel u
    //MessageBox (treci argument).
    MessageBox.Show("Za potvrdu kliknite na OK da biste narucili farbu!",
                            "Potvrda", MessageBoxButtons.OKCancel);
}
```

Када кликнемо на неку боју, а затим када кликнемо на дугме NARUCI, наша апликација треба да изгледа као што је приказано на Слици 3.21.



Слика 3.21. Изглед апликације у примеру 7

Пример 8. **Написати програм који кликом на дугме *Prikazi slike* убацује слике из *ImageList* компоненте у *PictureBox* компоненте, кликом на дугме *Izbroj slike* исписује у лабелу број слика из *ImageList* компоненте, а кликом на дугме *Pronadji indeks* исписује у другу лабелу индекс последње слике у листи.**



Слика 3.22. Изглед форме у примеру 8

Потребно је да додамо следеће компоненте форми:

1. Четири *PictureBox* компоненте које ће служити за приказивање слика из *ImageList* компоненте (својство *BorderStyle* поставити на **Fixed3D**)
2. Три дугмета (код првог у својсту *Text* исписати *Prikazi slike*, код другог *Izbroj slike*, а код трећег *Pronadji indeks*)
3. label1- служи за исписивање броја слика
4. label2- служи за исписивање индекса последње слике у листи
5. imageList1 (унети слике у листу, у својсту *ImageSize* величину слика у листи поставити на 160-Width и 100-Height, а својство *ColorDepth* поставити на **Depth32Bit**)

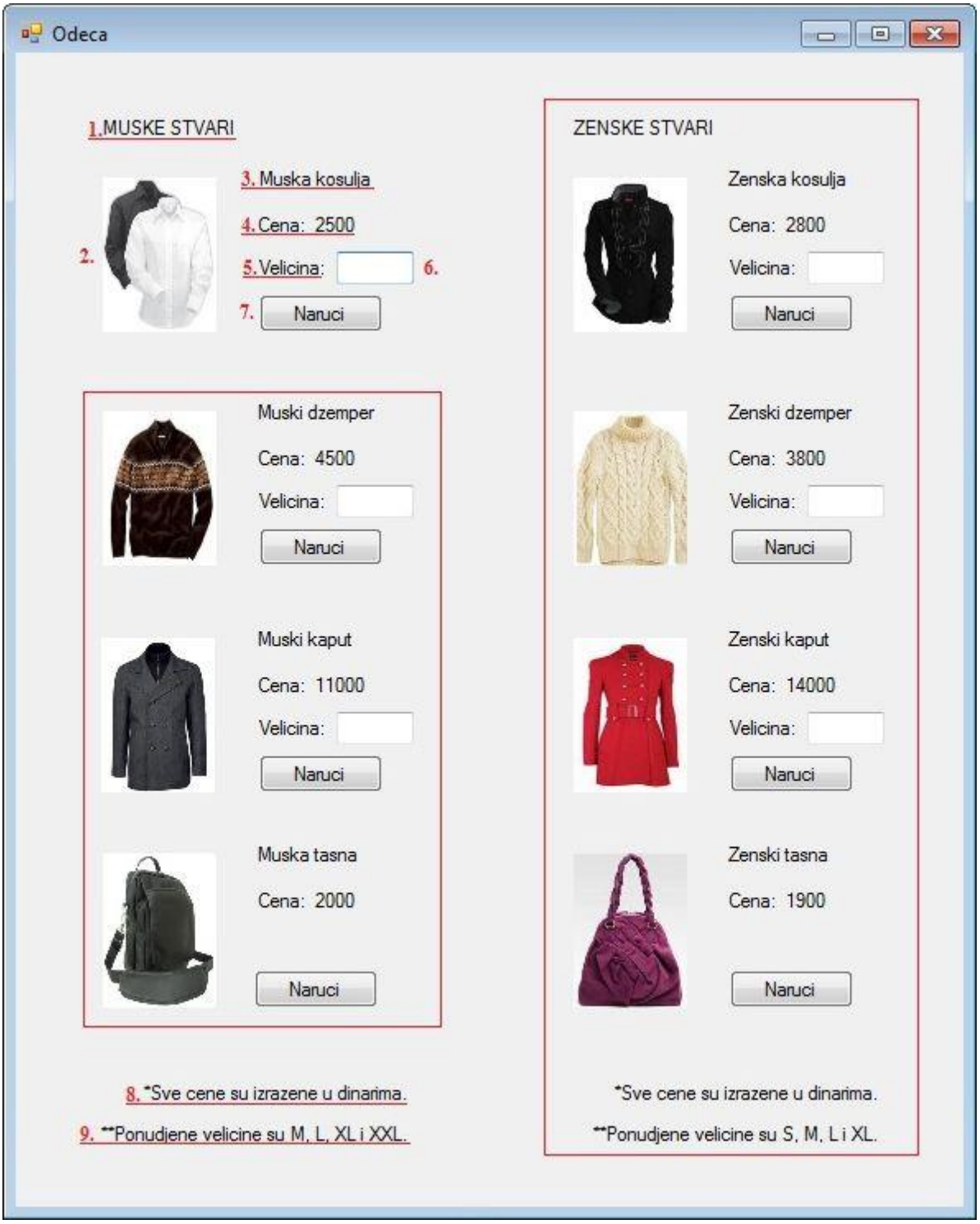
Након овога следи писање кода. Потребно је два пута кликнути на свако дугме понаособ и исписати одговарајући код за свако од њих.

```
private void Form1_Load(object sender, EventArgs e)
{
    //Podesavanje svojstva ColorDepth.
    imageList1.ColorDepth = ColorDepth.Depth32Bit;
}

private void button1_Click(object sender, EventArgs e)
{
    //U pictureBox komponente ubacujemo slike (pictureBox1.BackgroundImage)
    //iz liste tako sto pristupamo indeksima slika iz liste (imageList1.Images[0]).
    pictureBox1.BackgroundImage = imageList1.Images[0];
    pictureBox2.BackgroundImage = imageList1.Images[1];
    pictureBox3.BackgroundImage = imageList1.Images[2];
    pictureBox4.BackgroundImage = imageList1.Images[3];
}

private void button2_Click(object sender, EventArgs e)
{
    //U komponentu label1 upisujemo tekst koji hocemo da se pojavi, a zatim
    //na taj tekst nadovezujemo broj slika u listi koji se dobija tako sto
    //se upotrebi funkcija Count (imageList1.ImagesCount).
    label1.Text = "Broj slika u listi je: ";
    label1.Text = label1.Text + imageList1.Images.Count + ".";
}

private void button3_Click(object sender, EventArgs e)
{
    //U komponentu label2 upisujemo tekst koji hocemo da se pojavi, a zatim
    //na taj tekst nadovezujemo indeks poslednje slike u listi koji se dobija
    //tako sto se upotrebi funkcija Count (imageList1.ImagesCount) i od toga
    //oduzme 1 jer numeracija indeksa krece od nule.
    label2.Text = "Indeks poslednje slike u listi je: ";
    label2.Text = label2.Text + (imageList1.Images.Count - 1) + ".";
}
```

Након покретања програма и након извршавања акција, наша апликација треба да изгледа као што је приказано на Слици 3.23.

Слика 3.23. Изглед апликације у примеру 8

Пример 9. **Написати програм који приказује неколико слика мушке и женске гардеробе, а поред сваке слике се налазе подаци везани за ту слику (назив артикла и цена). Такође се поред сваке слике налази поље у које се уноси величина артикла и дугме Naruci. Кликом на дугме Naruci се у MessageBox-у исписује порука шта је наручено и која је величина наручена.**



Слика 3.24. Изглед форме у примеру 9

Потребно је да додамо следеће компоненте форми:

1. label1- служи као наслов
2. pictureBox1- служи да се прикаже слика у њему, у *PropetriesWindows-y* подесити својство *SizeMode* на StretchImage (слика ће бити развучена преко целог PictureBox-а)
3. label2- служи да се у њој напише име артикла
4. label3- служи да се у њој напише цена артикла
5. label4- служи да нагласи шта треба да се унесе у поље за унос текста које се налази поред ње
6. textBox1- служи за унос величине са тастатуре
7. button1- када кликнемо на дугме појавиће се MessageBox са поруком
8. label13- служи као обавештење о ценама за кориснике
9. label14- служи као обавештење о величинама

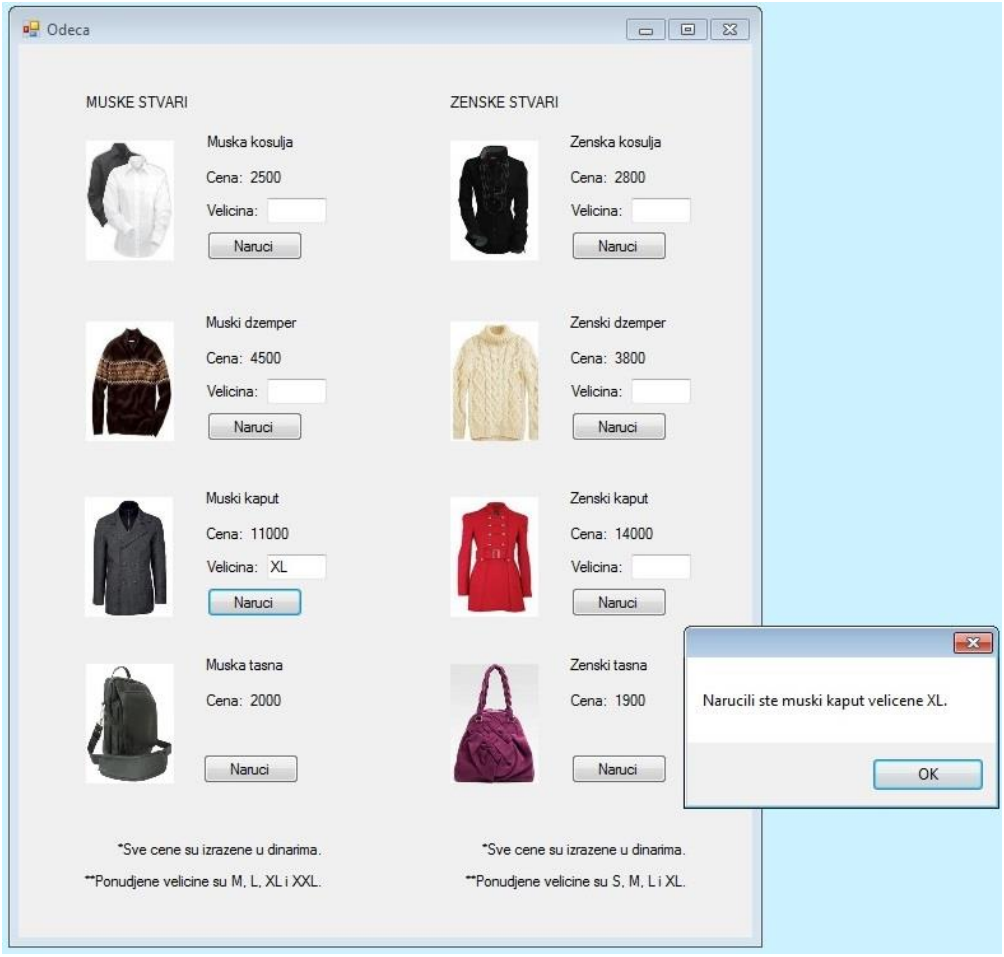
Следеће три уоквирене групице се праве на исти начин. Можемо чак, када направимо прву групу, да је селектујемо на форми и копирамо три пута, а онда само променимо слику, потребне податке и поставимо је на форму где желимо. Десна уоквирена страна се прави исто као и лева. Такође можемо, када направимо леву страну, да је селектујемо и целу копирамо, а онда изменимо слике и потребне податке.

Слике можемо наћи на интернету, ако их већ немамо у неком фолдеру на рачунару, и сачувати их у фолдеру у коме чувамо и пројекте. На овај начин ће, ако смо некоме послали овај пројекат заједно са сликама, програм моћи да учита и прикаже ове слике.
Треба два пута да кликнемо на свако дугме, а затим за свако да испишемо одговарајући код.

```
private void button1_Click(object sender, EventArgs e)
{
    //Ispisujemo u MessageBox tekst pod navodnicima i tekst koji je unet u kompo-
    //nenti textBox1.
    MessageBox.Show("Narucili ste musku kosulju velicene " + textBox1.Text + ".");

    //Nakon sto se MessageBox zatvori, komponenta textBox1 treba da se isprazni
    //kako bi bila spremna za novi unos. Dakle, sledecom linijom koda se ovo polje
    //prazni.
    textBox1.Text = "";
}

private void button2_Click(object sender, EventArgs e)
{
    //Ispisujemo u MessageBox tekst pod navodnicima i tekst koji je unet u kompo-
    //nenti textBox2.
    MessageBox.Show("Narucili ste muski dzemper velicene " + textBox2.Text + ".");

    //Nakon sto se MessageBox zatvori, komponenta textBox2 treba da se isprazni
    //kako bi bila spremna za novi unos. Dakle, sledecom linijom koda se ovo polje
    //prazni.
    textBox2.Text = "";
}

private void button3_Click(object sender, EventArgs e)
{
    //Ispisujemo u MessageBox tekst pod navodnicima i tekst koji je unet u kompo-
    //nenti textBox3.
    MessageBox.Show("Narucili ste muski kaput velicene " + textBox3.Text + ".");

    //Nakon sto se MessageBox zatvori, komponenta textBox3 treba da se isprazni
    //kako bi bila spremna za novi unos. Dakle, sledecom linijom koda se ovo polje
    //prazni.
    textBox3.Text = "";
}

private void button4_Click(object sender, EventArgs e)
{
    //Ispisujemo u MessageBox tekst pod navodnicima.
    MessageBox.Show("Narucili ste musku tasnu.");
}

private void button5_Click(object sender, EventArgs e)
{
    //Ispisujemo u MessageBox tekst pod navodnicima i tekst koji je unet u kompo-
    //nenti textBox4.
    MessageBox.Show("Narucili ste zensku kosulju velicene " + textBox4.Text + ".");

    //Nakon sto se MessageBox zatvori, komponenta textBox4 treba da se isprazni
    //kako bi bila spremna za novi unos. Dakle, sledecom linijom koda se ovo polje
    //prazni.
    textBox4.Text = "";
}
```

```
private void button6_Click(object sender, EventArgs e)
{
    //Ispisujemo u MessageBox tekst pod navodnicima i tekst koji je unet u kompo-
    //nenti textBox5.
    MessageBox.Show("Narucili ste zenski dzemper velicene " + textBox5.Text + ".");

    //Nakon sto se MessageBox zatvori, komponenta textBox5 treba da se isprazni
    //kako bi bila spremna za novi unos. Dakle, sledecom linijom koda se ovo polje
    //prazni.
    textBox5.Text = "";
}

private void button7_Click(object sender, EventArgs e)
{
    //Ispisujemo u MessageBox tekst pod navodnicima i tekst koji je unet u kompo-
    //nenti textBox6.
    MessageBox.Show("Narucili ste zenski kaput velicene " + textBox6.Text + ".");

    //Nakon sto se MessageBox zatvori, komponenta textBox6 treba da se isprazni
    //kako bi bila spremna za novi unos. Dakle, sledecom linijom koda se ovo polje
    //prazni.
    textBox6.Text = "";
}

private void button8_Click(object sender, EventArgs e)
{
    //Ispisujemo u MessageBox tekst pod navodnicima.
    MessageBox.Show("Narucili ste zensku tasnu.");
}
```

Када унесемо величину, а затим када кликнемо на дугме NARUCI, наша апликација треба да изгледа као што је приказано на Слици 3.25.



Слика 3.25. Изглед апликације у примеру 9