Универзитет у Крагујевцу Факултет инжењерских наука



Софтверски инжењеринг

Пројектни задатак:

Судоку

Студент: Предметни наставник:

Тијана Танасковић 656/2017 Проф. Др. Ненад Филиповић

Крагујевац, Фебруар 2021.

**Садржај:**

1. [Поставка задатка и опис коришћења апликације 3](#_bookmark0)
   1. [Дефинисање пројекта 3](#_bookmark1)
   2. [Опис коришћења апликације 3](#_bookmark2)
2. [Опис делова програма 7](#_bookmark3)
   1. [Класа SudokuGenerator 7](#_bookmark4)
   2. [Класа SudokuPolje 10](#_bookmark5)
   3. [Класа SudokuForm 15](#_bookmark6)
3. [UML дијаграми 17](#_bookmark7)
   1. [Дијаграм случајева коришћења 17](#_bookmark8)
   2. [Дијаграм секвенци 18](#_bookmark9)
   3. [Дијаграм активности 19](#_bookmark10)
   4. [Дијаграм стања 20](#_bookmark11)
   5. [Дијаграм класа 21](#_bookmark12)
   6. [Дијаграм објеката 22](#_bookmark13)
4. [Закључак 23](#_bookmark14)
5. [Литература 24](#_bookmark15)

# Поставка задатка и опис коришћења апликације

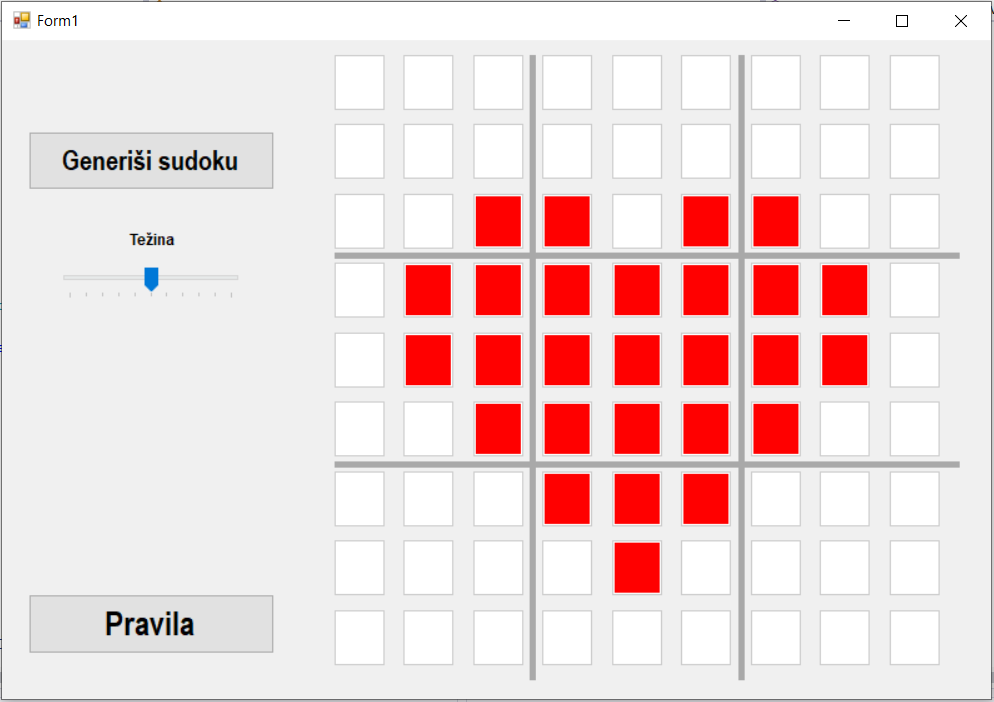
## Дефинисање пројекта

У овом пројектном задатку је било потребно направити реплику постојеће игре Судоку.

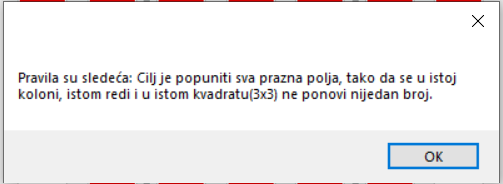
## Опис коришћења апликације

Када корисник покрене апликацију појављује се форма са слике 1. Уколико није упућен у правила игре, кликом на дугме “Pravila” појављује се позор са објашњењем.

Корисник може да изабере тежину игре, тако што помера TrackBar**.** Што је више постављен у десно, то је условно теже решити Судоку. Игра се започиње кликом на дугме “Generiši sudoku”, где програм сам генерише једну насумичну верзију игре за решавање. [2]

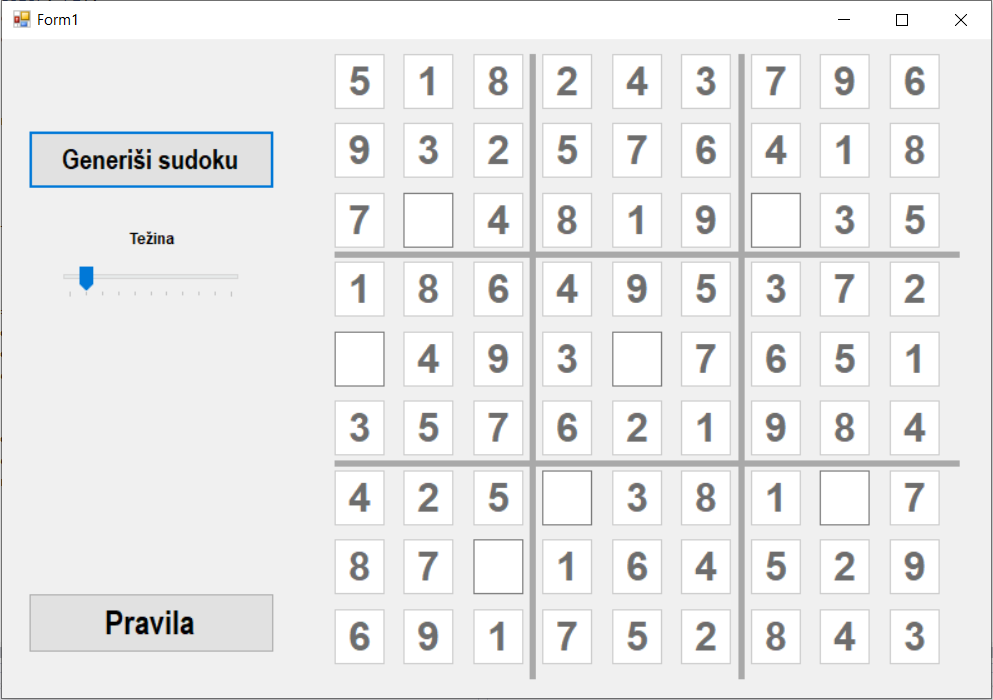


Слика 1 - Изглед форме при покретању програм

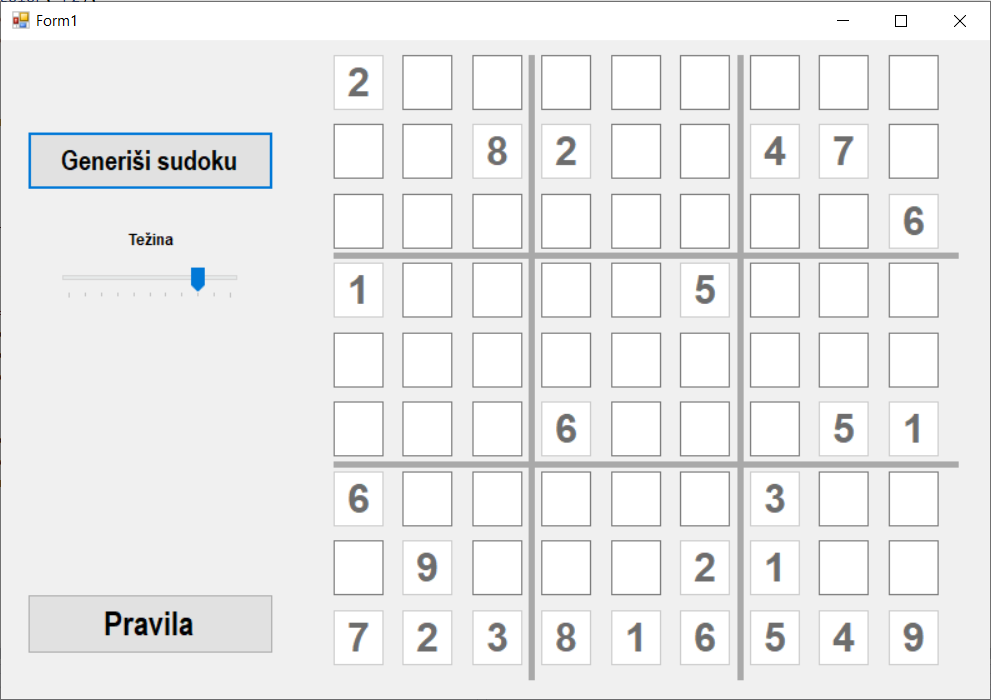


Слика 2 – Приказ објашњења правила игре

Када корисник одабере жељену тежину игре, генерише се табла са више или мање празних поља која треба да се попуне. На сликама испод је приказана једна лакша и једна тежа партија игрице.



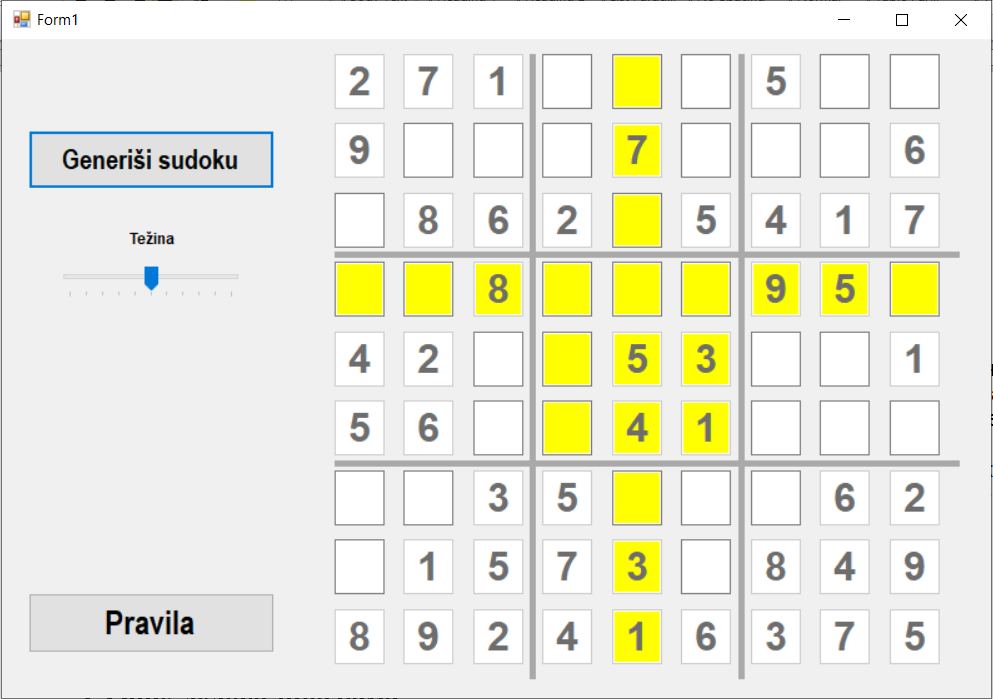
Слика 3 – Пример лакше партије



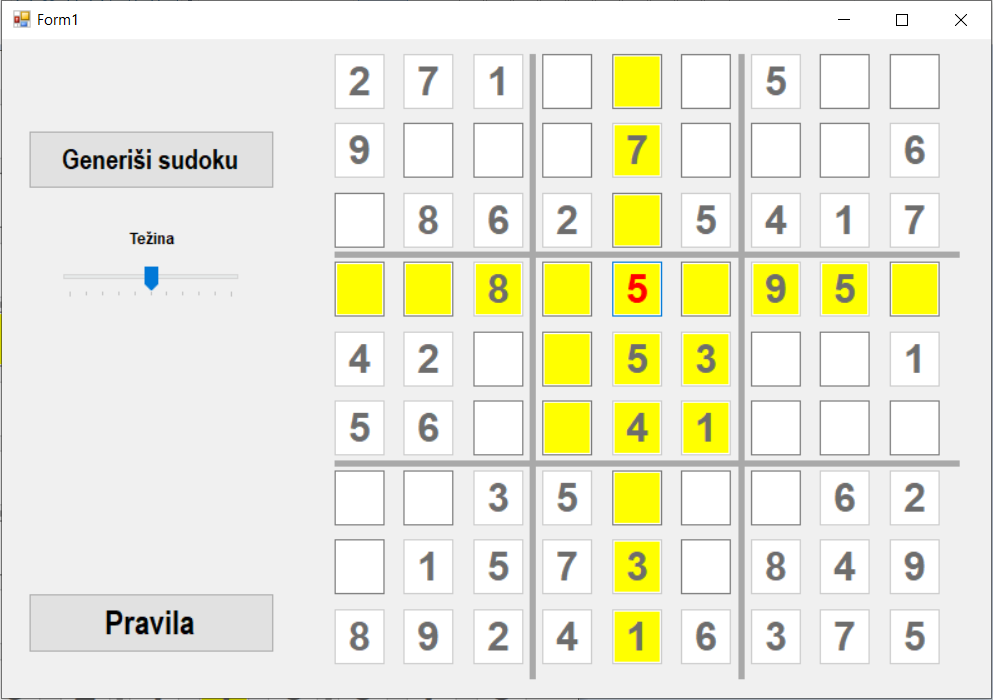
Слика 4 – Пример теже (захтевније) партије

Корисник започиње игру тако што уписује бројеве у празна поља по већ утврђеном правилу. Када стрелицом миша стане на било које празно поље хајлајтује се ред, колона и квадрат тог поља, што му помаже на шта треба да обрати пажњу и лакше уочи бројеве који ће утицати на то који број треба да унесе.

Још једна карактеристика је то што ако корисник унесе неки очигледно погрешан број (који се већ налази или у колони или реду или квадрату) тај број ће се обојити у црвено. Међутим не приказује му се где је тачно проблем већ он то мора сам да пронађе.

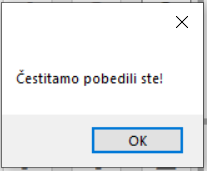


Слика 5 – Хајлајтован ред, колона и квадрат



Слика 5 – Пример очигледне грешке у решавању и обојена цифра као знак

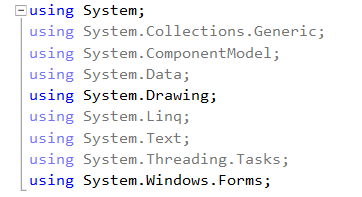
Када корисник успе да реши Судоку појављује му се прозорчић који га о томе обавештава.



Слика 7 – Пример краја успешне партије

# Опис делова програма

За израду пројектног задатка коришћен је програмски језик C#.



Слика 8 – Коришћене библиотеке

Библиотека System садржи фундаменталне класе и класе које дефинишу уобичајене типове података и њихове вредности (у нашем коду нпр. int, object).

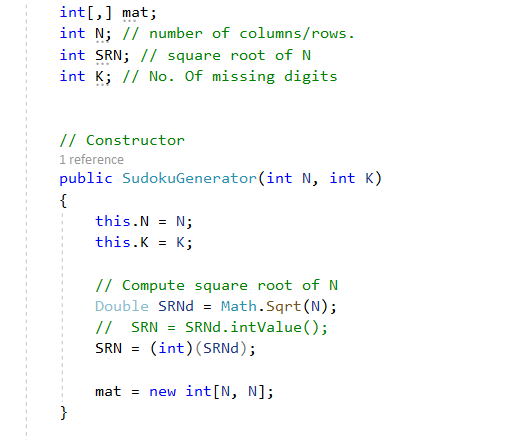
Библиотека System.Drawing нам омогућава цртање на гафичкој површини, коритећи разне врсте оловки, четки и слика. Користили смо је приликом бојења геометријских гифура као и боја генерално (System.Drawing.Color).

Последња библиотека нам служи за прављење форми помоћу којих приказујемо кориснички интерфејс (нпр. System.Windows.Forms.Form).

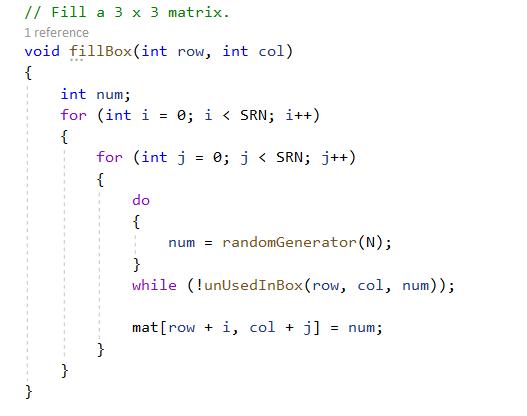
## Класа SudokuGenerator

Класа SudokuGenerator је заслужна за генерисање табле. Идеја је да се прво генеришу 3 дијагонална квадрата која нису повезана ни колоном, ни редом и независни су један од другог, што значи да је само битно да се бројеви унутар тих квадрата не понављају.

Остатак табеле се попуњава тако што се све време пази да ли су сви услови задовољени.

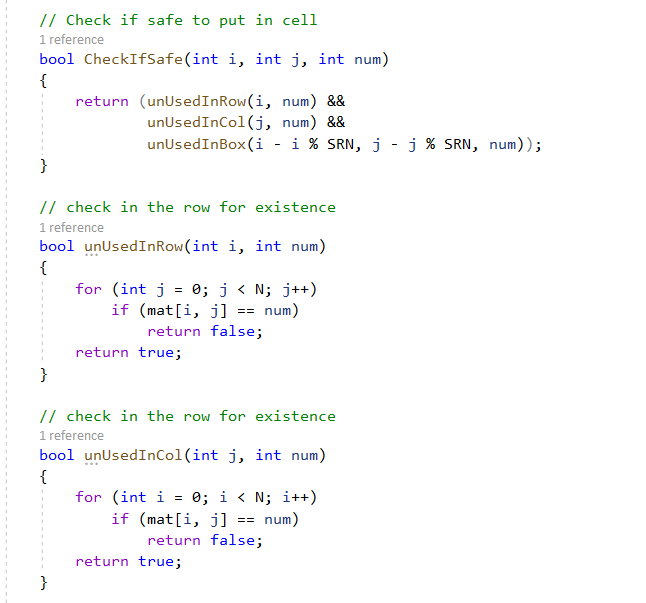


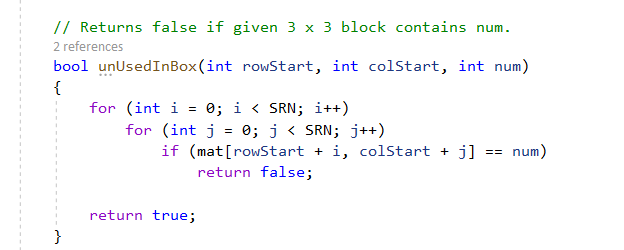
Слика 9 – Конструктор класе



Слика 10 – Метода за попуњавање дијагоналних квадрата квадрата

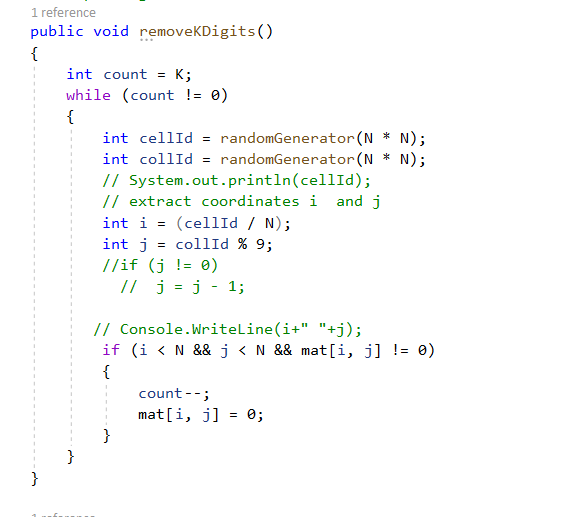
CheckIfSafe проверава три ф-је: да ли се неки број већ налази у колони, да ли се неки број већ налази у реду и да ли се тај број налази у квадрату.





Слика 11 – CheckIfSafe ф-ја са методама за провере

Када се попуни цела матрица, склањамо К рандом поља са табле.

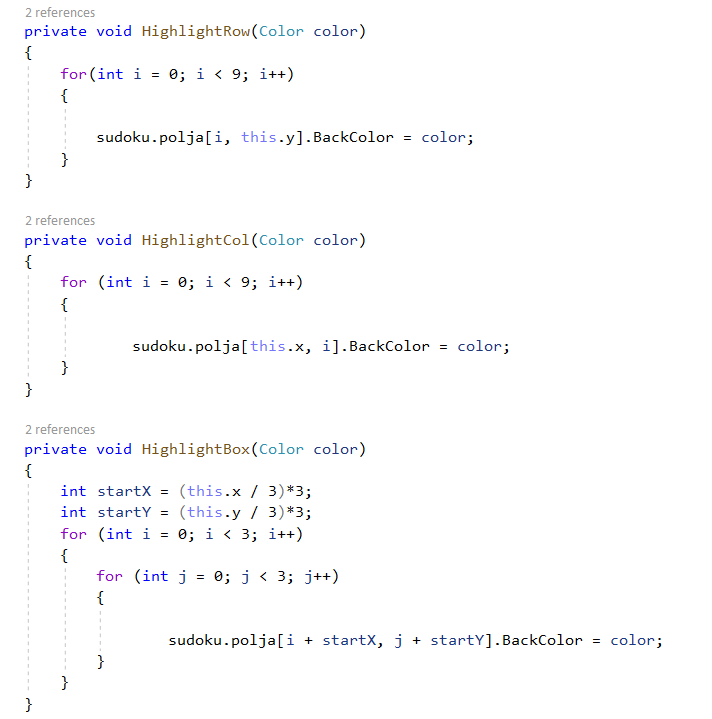
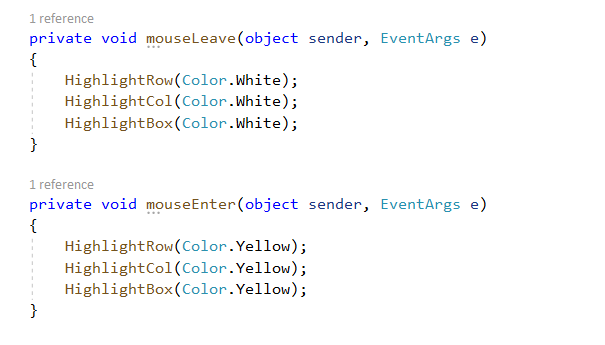


Слика 12 – Метода за уклањање бројева

SudokuGenerator прима две вредности број колона и редова и број поља која требају бити уклоњена.

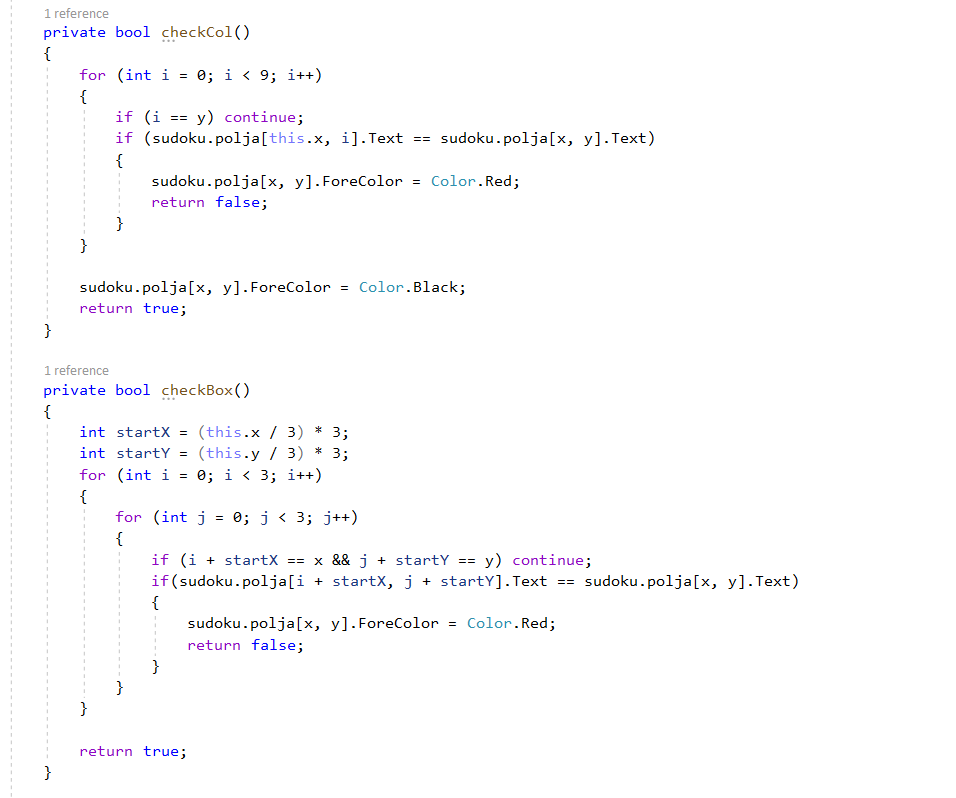
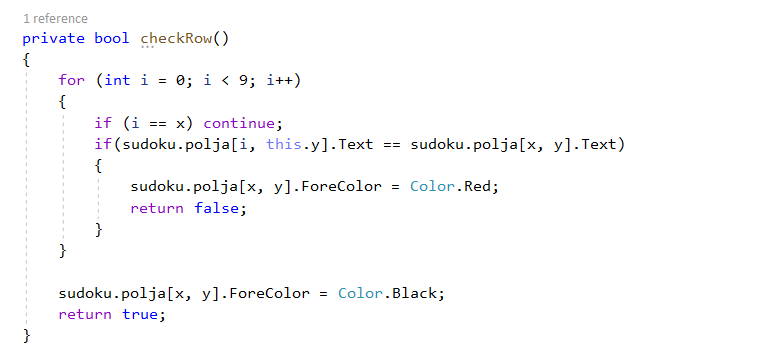
## Класа SudokuPolje

Ова класа је задужена за функционалности целе игре. У овој класи, између осталог, су написане методе које боје одређене делове табеле. Када мишем станемо на неко поље обележи се ред, колона и квадрат тог поља а када се склони враћа се на почетну белу боју.



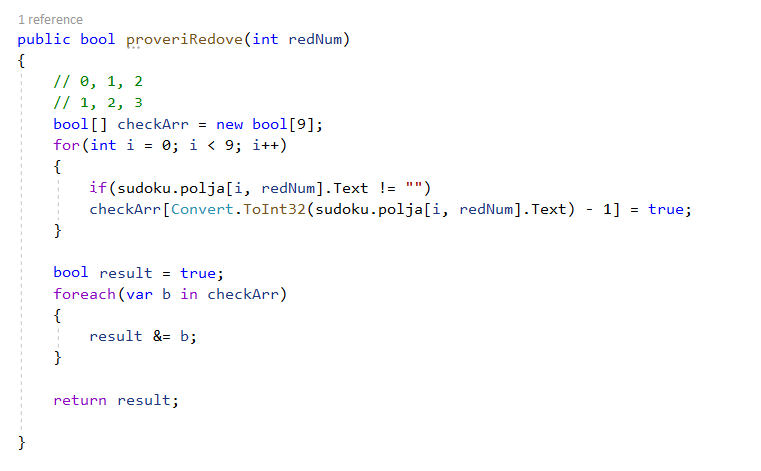
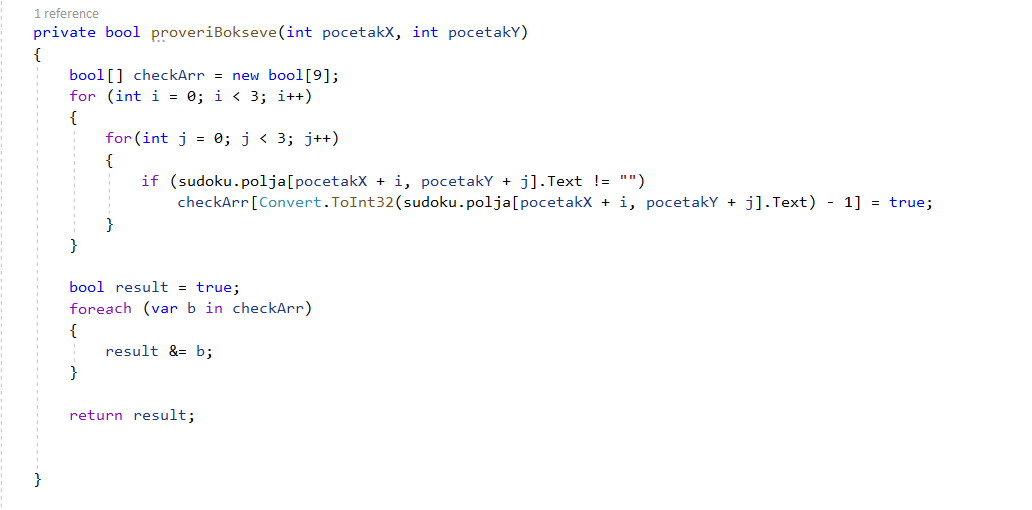
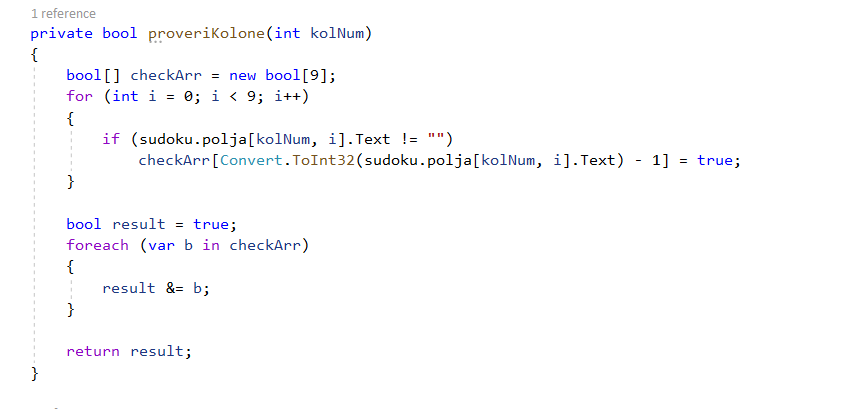
Слике 13 и 14 – Описане методе

Три методе (checkRow, checkCol, checkBox) проверавају да је тај број који ми унесемо током игре већ у реду, колони или квадрату, и ако јесте обоји га у црвено, како би кориснику приказао очигледну грешку.



Слике 15 и 16 – Методе које боје проблематичне бројеве

Три методе које проверавају да ли нам је сваки ред, колона и квадрат валидан одређују да ли је корисник победио или не.

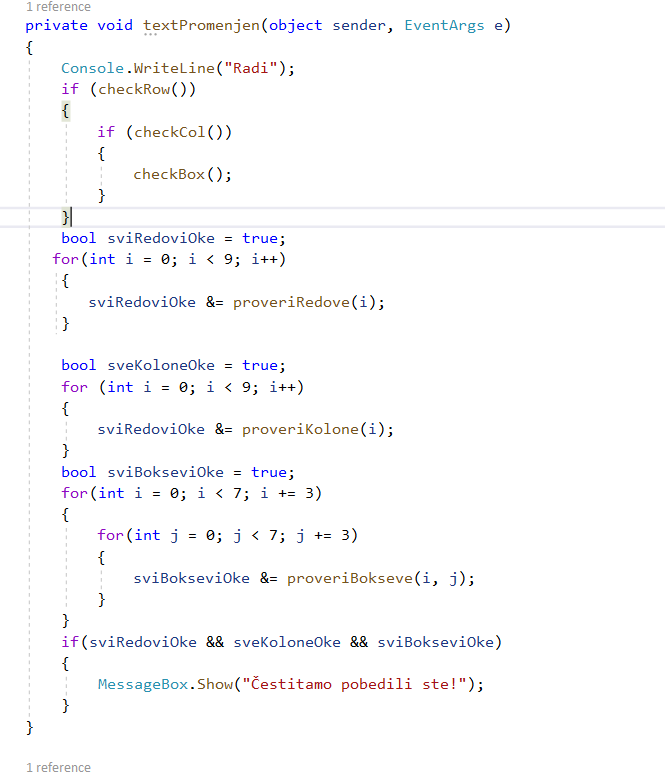


Слике 17, 18 и 19 – Методе о валидацији колона, квадрата и редова

Свих 6 метода се позивају у ф-ји textPromenjen која се активира сваки пут кад ми унесемо неки број у било које поље. Проверава све горе наведене ф-је.

Све фј-е имају bool вредности, како бисмо лакше могли да извришимо нешто у случају да је неки случај задовољен(true) или не(false).

И само ако је сваки ред, свака колона и сваки квадрат добар, корисник може да победи.



Слика 20 - Функција textPromenjen

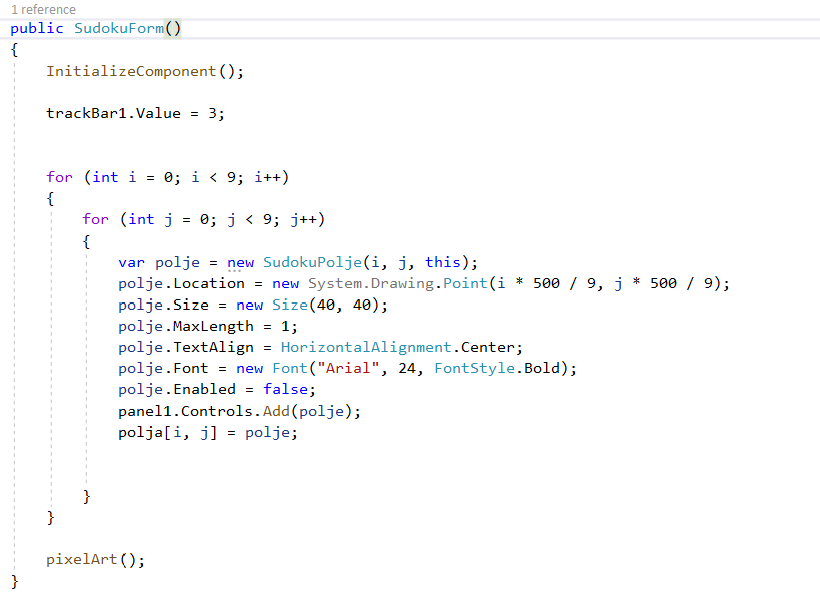
## Класа SudokuForm

Класа је заслужна за изглед форме, панела, постављање поља, постављање линија између квадрата, фонт, величину слова, боје...

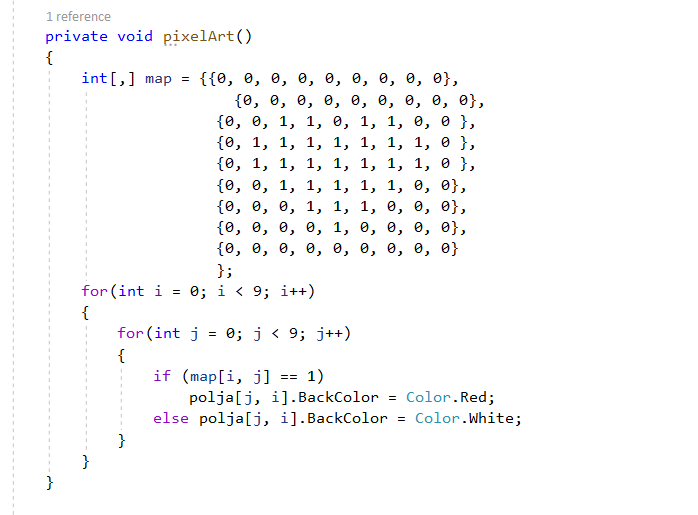
Функција SudokuForm одређује величину поља, позицију, фонт..

PixelArt је метода која на почетку боји одређена поља како би се постигао изглед који можемо видети када покренемо програм.

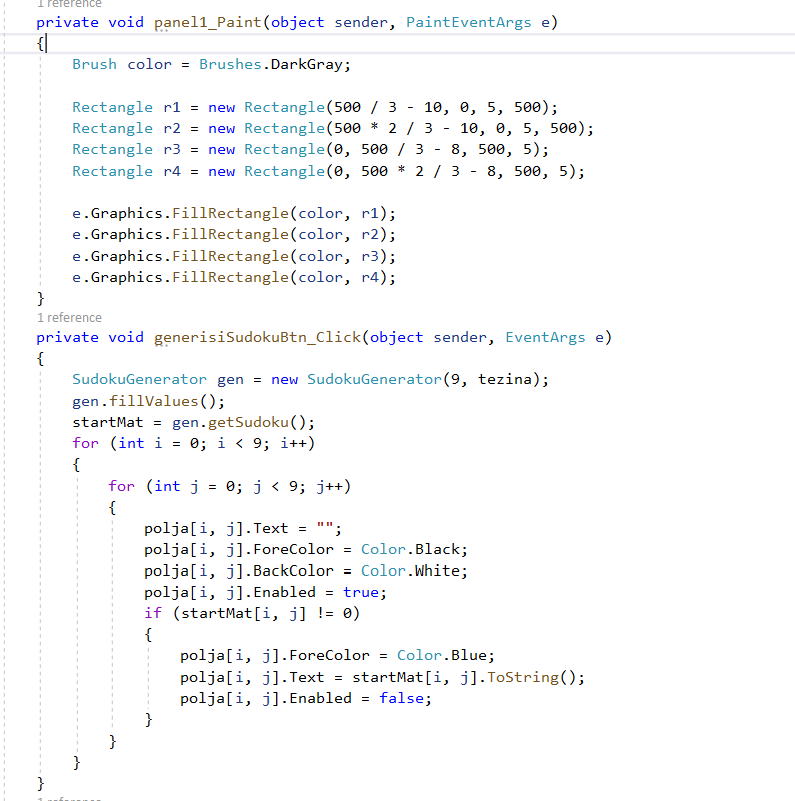
Ова класа додељује функцију дугмету за генерисање табле.



Слика 21 – Метода постављања поља



Слика 22 – Метода за улепшавање



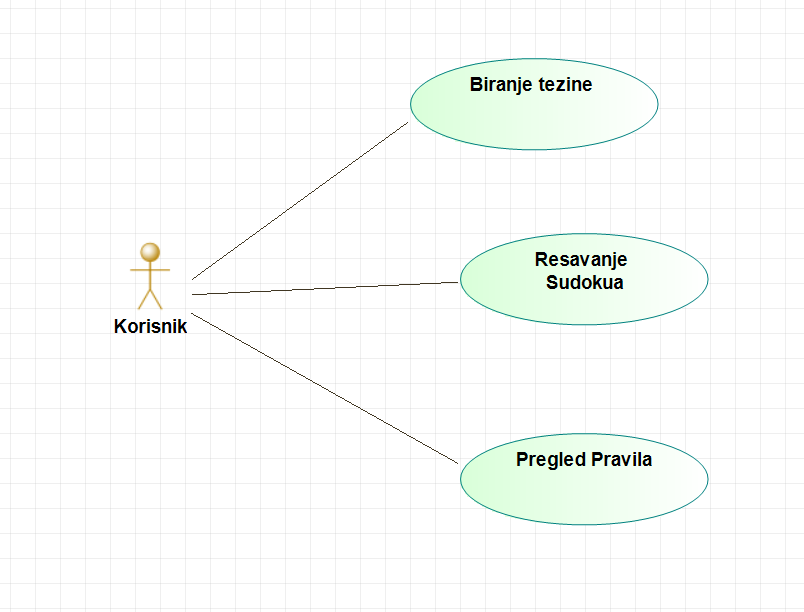
Слика 23 – Подешавање панела и дугмета

# UML дијаграми

UML(unified modeling language) је стандардни графички језик за моделовање објектно-орјентисаног софтвера. [1]

## Дијаграм случајева коришћења

Дијаграм случајева коришћења (use-case) приказује скуп случајева коришћења и актера. Случај коришћења специфира ШТА субјекат ради, а не КАКО ради. [1]

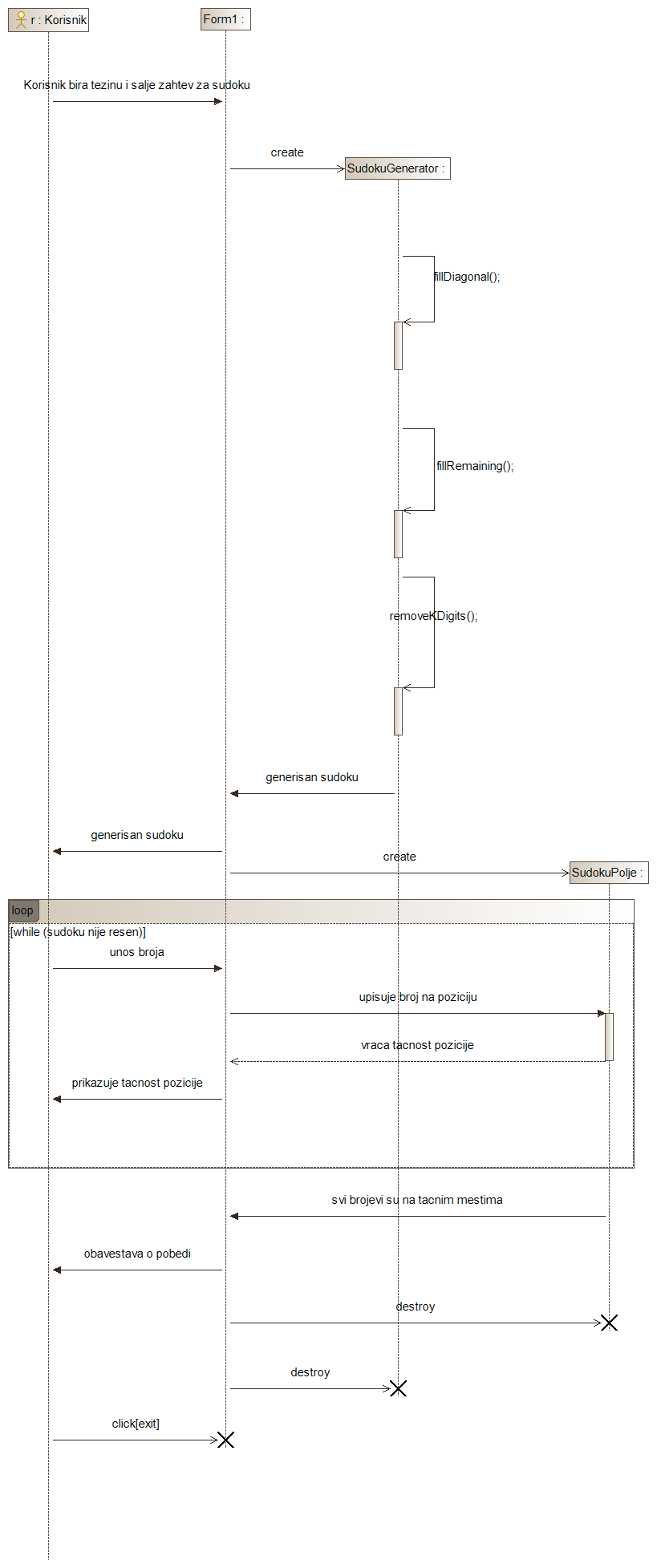


Слика 24 – Дијаграм случајева коришћења

## Дијаграм секвенци

Дијаграми секвенци користе се да прикажу комуникацију између скупа објеката током времена у облику порука.

Вертикална оса представља ток времена, и време тече одозго на доле.

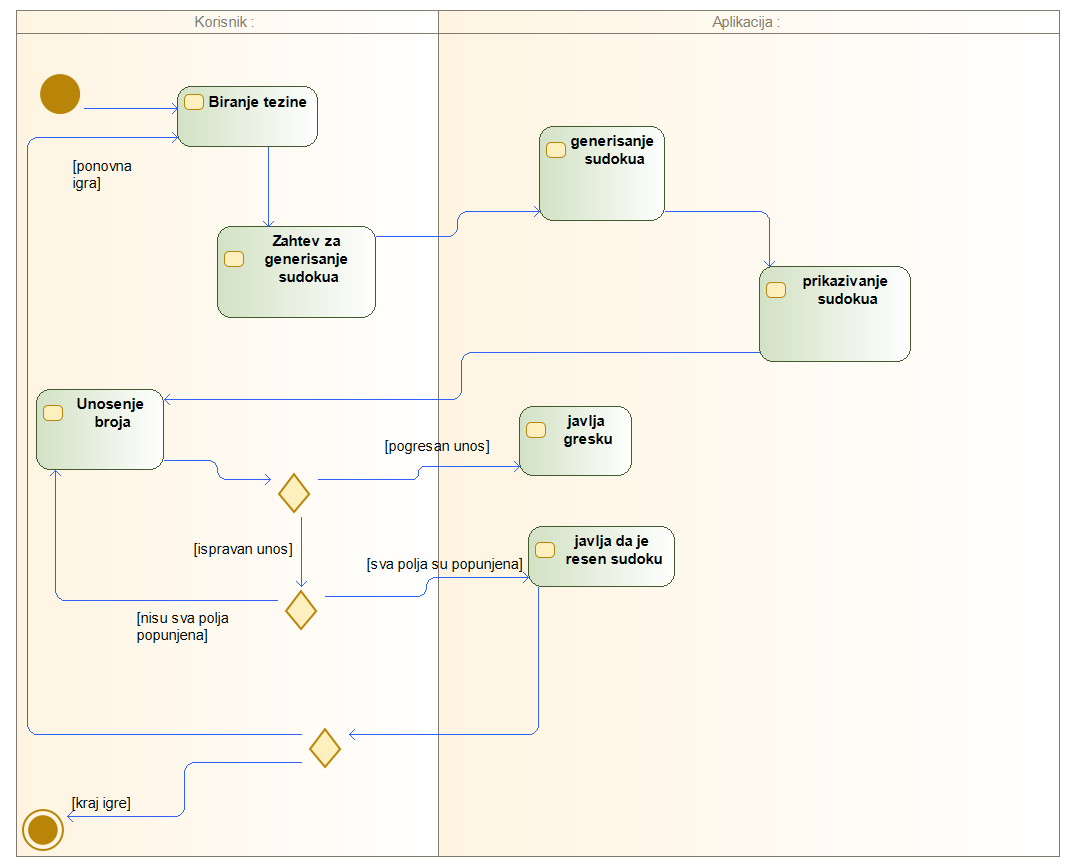


Слика 25 – Дијаграм секвенци

## Дијаграм активности

Дијагам акивности приказује редослед активности које извршавају објекти и ток објеката између корака активности. [1]

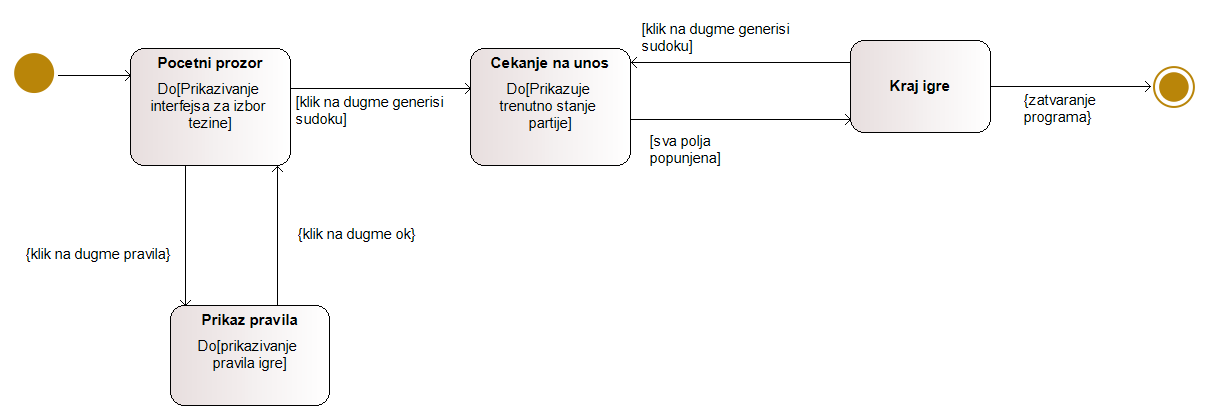
За активности са леве стране одговоран је корисник а са десна стране апликација.



Слика 26 – Дијаграм активности

## Дијаграм стања

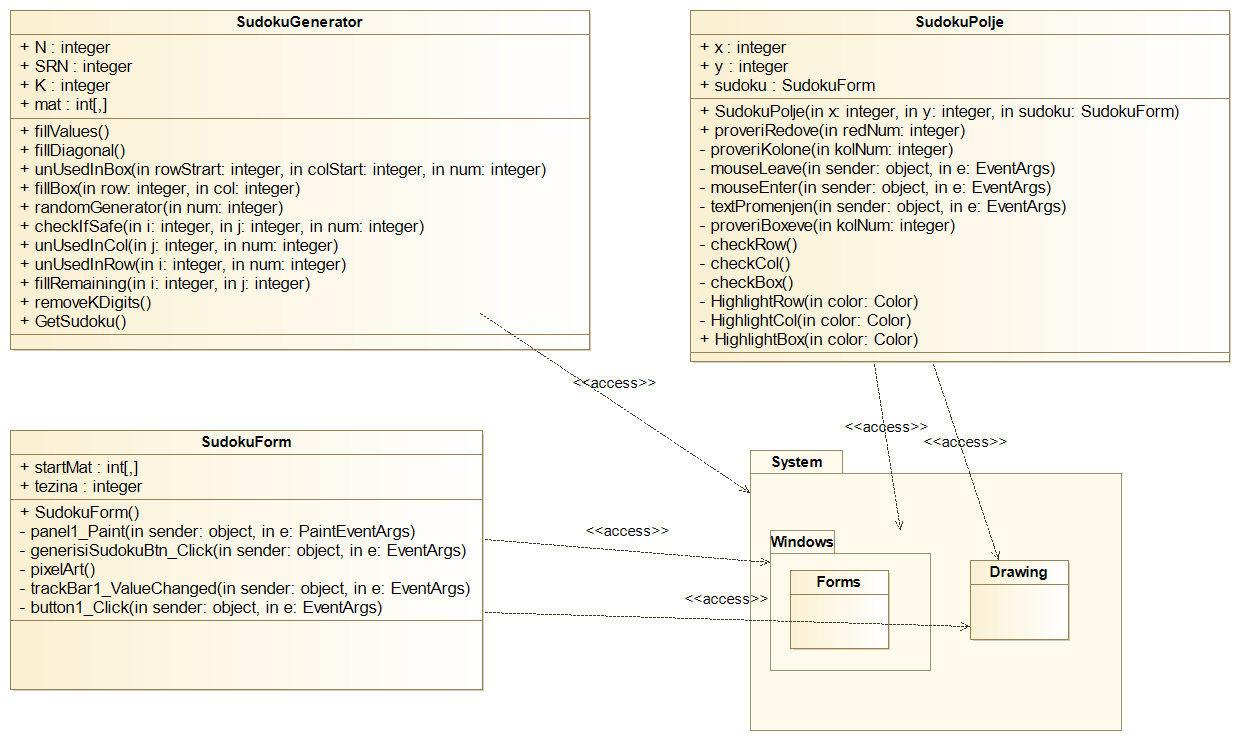
Дијаграм стања приказује стања и псеудостања, прелазе између стања, догађаје који проузрокују промену стања и акције које резултују из промене стања. У једном стању ентитет задовољава неки услов или обавља неку активност или чека неки догађај. [1]



Слика 27 – Дијаграм стања

## Дијаграм класа

Дијаграм класа приказује скуп класа, интерфејса, сарадњи и других ствари структуре, повезаним релацијама. [1]



Слика 28 – Дијаграм класа

## Дијаграм објеката

## Дијаграми објеката приказују примерке (објекте) апстракције (класа) и њихове везе преко којих објекти могу да комуницирају. [1]

## 

## Слика 29 – Дијаграм објеката

# Закључак

Програм је најлакше објаснити коришћењем UML дијаграма, пошто су дијаграми лакши за разумевање од самог кода.

Кориснику је само битно како апликација фукнционише, шта све може да им понуди а не шта се налази у позадини исте.

А уколико бисмо се ми, као програмери, нашли у неком новом окружењу (нпр новој фирми) и када бисмо били прикључени неком тиму који обавља одређени задатак, UML дијаграми би нам користили више од саме документације целог пројекта и олакшали би нам комуникацију са члановима тима. То је још један битан разлог и објашњење зашто су корисни за коришћење.

# Литература

1. Курс : Софтверски инжењеринг, 09.02.2021. <http://moodle.fink.rs/course/view.php?id=978>
2. GeeksforGeeks, 07.02.2021. <https://www.geeksforgeeks.org/program-sudoku-generator/>
3. Učim programiranje, 11.02.2021. <https://www.ucim-programiranje.com/2012/12/uml-osnove/>