Универзитет у Крагујевцу

Факултет инжењерских наука

Семинарски рад из предмета

Софтверски инжењеринг

Тема:

Исцртавање графика задате функције две праве и одређивање њиховог међусобног положаја

Студент:
Ђорђе Гачић 626/2018

Предметни професор:
проф. др. Ненад Филиповић

Асистент:
маст. инж. маш. Тијана Шуштершич

Крагујевац, јул 2021. године

Садржај

1. Поставка задатка

Кроз поставку задатка дефинисано је да је потребно направити апликацију која исцртава график задатих функција двије праве преко унесених координата или одговарајућег нагиба и одсјечка. Након исцртавања двије праве потребно је одредити координате тачке која се налази у пресјеку. Одрадити комплетан кориснички интерфејс за ову апликацију и понудити више опција кориснику. Предвидјети и неке изузетке као што је нпр. изостављен унос или унос неодговарајућих вриједности.

2. Опис намјене и коришћења апликације

Апликација која је у наставку имплементирана намјењена је исцртавању графика двије праве као и одређивању њихове тачке пресјека уколико су праве у положају да се сијеку. Такође је предвиђен случај да се праве поклапају или да су паралелне једна другој и о томе се корисник обавјештава на адекватан начин.

Након покретања апликације отвара се прозор преко кога је могућ одабир начина уноса података за праве. Унос вриједности за одређивање линеарних функција (тј. функција правих) могућ је на два начина:

1. Унос преко координата двије тачке за сваку праву
2. Унос преко нагиба и одсјечка за сваку праву

У оба случаја се провјерава садржај из поља за унос тако што се води рачуна да поља за унос нису празна као и да су у њима вриједности на основу којих се даљим прорачуном може доћи до параметара према којима ћемо касније одредити општи облик једначина правих и исцртати график. Такође, уз график се приказује и општи облик обје једначине правих као и координате тачке пресјека или информација о томе да ли се праве поклапају или су паралелне уколико се не сијеку.

Након одабира начина уноса података за праве, корисник је у могућности да се врати на поновни одабир уколико погријеши или се предомисли.

3. Опис дијелова програма са самим изворним кодом

За израду апликације коришћен је програмски језик Пајтон верзије 3.9.5. Додатни пакети уз Пајтон коришћени при изради апликације су:

- pyqt5 5.15.4
- sympy 1.21.0
- matplotlib 3.4.2

PyQt5 библиотека функционалности коришћена је за израду графичког корисничког интерфејса. Највећи дио кода написао сам ручно, а не преко графичког дизајнера (Qt Designer) из разлога јер сам хтио да имам већу контролу над кодом апликације обраћајући тако више пажње на функционалности саме апликације него на њен изглед за који сам пазио да буде што једноставнији и разумљивији кориснику.

SymPy библиотека је намјењена за бројне математичке прорачуне. У овој апликацији користио сам је за рјешавање система линеарних једначина као и за исцртавање графика двије праве.

matplotlib је базна библиотека за SymPy.

Код је раздвојен у двије датотеке:

1. lineDrawingMain.py
2. globalFunctions.py

Такође је присутан и директоријум „icons“ у коме се налазе слике које се користе као иконице за одређене прозоре графичког корисничког интерфејса. Неопходно је да се поменути директоријум налази у истом директоријуму као и фајлови са кодом како би слике биле исправно учитане.

Уређивач текста који сам користио при писању кода је „Kate“ на Linux оперативном систему односно „Notepad++“ на Windows оперативном систему, а апликацију сам покретао преко Linux конзоле односно Windows командног прозора.

3.1 Датотека „lineDrawingMain.py“

Ова датотека представља датотеку која служи за покретање апликације. Начин који сам користио за покретање је:

```
python lineDrawingMain.py
```

из директоријума у коме су датотеке апликације (важи и за Linux конзолу и за Windows командни прозор уколико су системске путање за Пајтон исправно подешене).

У наставку је садржај датотеке „lineDrawingMain.py“ који ће бити приказиван постепено уз објашњења након одређених дијелова кода:

```
1  import sys
2  from PyQt5.QtWidgets import QApplication, QWidget, QPushButton, QDialog, QLabel,
QLineEdit
3  from PyQt5.QtGui import QIcon, QFont
4  from PyQt5.QtCore import Qt
5  from math import *
6  from sympy import *
7  from sympy.plotting import plot, plot_parametric
8
9  import globalFunctions
10 from globalFunctions import *
```

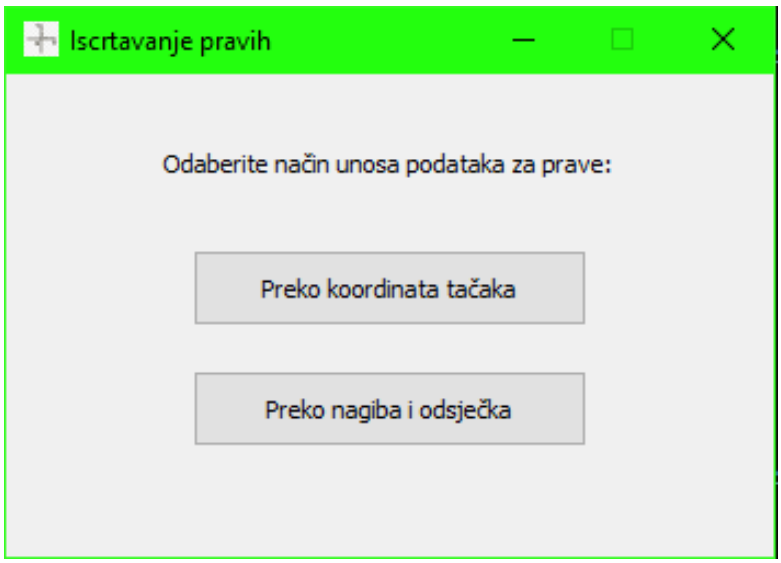
Претходни дио кода односи се на учитавање потребних Пајтон библиотека као што су „PyQt5“ за графичко окружење и „SymPy“ за математичке прорачуне и исцртавање графика. Такође је учитана и датотака „globalFunctions“ и читав њен садржај како бисмо имали на располагању и све глобалне функције направљене за потребе апликације.

Класа „WindowChoose(QWidget)“:

```
13 class WindowChoose(QWidget):
14     def __init__(self):
15         super().__init__()
16
17         self.setGeometry(500, 200, 350, 220)
18         self.setWindowTitle("Iscrtavanje pravih")
19         self.setWindowIcon(QIcon('icons/icon.jpg'))
20
21         self.setFixedHeight(220)
22         self.setFixedWidth(350)
23
24
25         self.labelTxt = QLabel("Odaberite način unosa podataka za prave:", self)
26         self.labelTxt.setGeometry(40, 30, 270, 20)
27         self.labelTxt.setWordWrap(True)
28         self.labelTxt.setAlignment(Qt.AlignCenter)
29
30         self.btnCoord = QPushButton("Preko koordinata tačaka", self)
31         self.btnCoord.setGeometry(85, 80, 180, 35)
32         self.btnCoord.clicked.connect(self.clicked_btnCoord)
33
34         self.btnSlopeIntercept = QPushButton("Preko nagiba i odsječka", self)
35         self.btnSlopeIntercept.setGeometry(85, 135, 180, 35)
36         self.btnSlopeIntercept.clicked.connect(self.clicked_btnSlopeIntercept)
37
38     def clicked_btnCoord(self):
```

```
39          windowCoord.show()
40          self.close()
41
42      def clicked_btnSlopeIntercept(self):
43          windowSlopeIntercept.show()
44          self.close()
```

Класа „WindowChoose(QWidget)“ је класа у којој су имплементиране почетне функционалности апликације које се односе на одабир начина уноса података за праве. Класа наслиједђује класу „QWidget“ која је садржана у библиотеци „PyQt5“. При иницијализацији се постављају параметри графичког окружења и иницијализује се надкласа, а изглед самог прозора је приказан на слици 1:



Слика 1: Почетни прозор – одабир начина уноса за праве

Кликом на горње дугме позива се функција „clicked_btnCoord()“ која затвара почетни прозор за одабир начина уноса, а приказује прозор за унос координата тачака за обје праве.

Кликом на доње дугме позива се функција „clicked_btnSlopeIntercept()“ која такође затвара тренутни прозор за одабир начина уноса, а приказује прозор за унос нагиба и одсјечка за обје праве.

Класа „WindowCoordinates(QWidget)“:

```
47 class WindowCoordinates(QWidget):
48     def __init__(self):
49         super().__init__()
50
51         self.setGeometry(300, 200, 675, 300)
52         self.setWindowTitle("Iscrtavanje pravih preko koordinata")
```

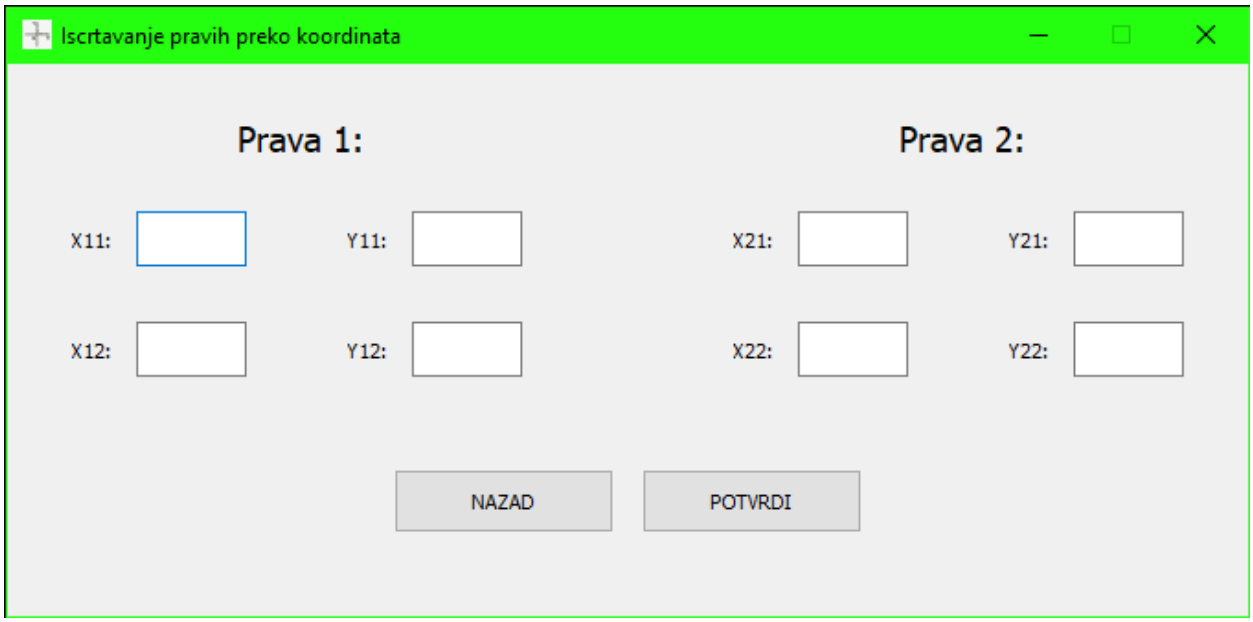
```
53          self.setWindowIcon(QIcon('icons/icon.jpg'))
54
55          self.setFixedHeight(300)
56          self.setFixedWidth(675)
57
58
59          self.labelOne = QLabel("Prava 1:", self)
60          self.labelOne.setGeometry(125, 30, 70, 20)
61          self.labelOne.setFont(QFont("Sanserif", 14))
62
63          self.labelX11 = QLabel("X11:", self)
64          self.labelX11.setGeometry(35, 80, 30, 30)
65          self.inputX11 = QLineEdit(self)
66          self.inputX11.setGeometry(70, 80, 60, 30)
67
68          self.labelY11 = QLabel("Y11:", self)
69          self.labelY11.setGeometry(185, 80, 30, 30)
70          self.inputY11 = QLineEdit(self)
71          self.inputY11.setGeometry(220, 80, 60, 30)
72
73          self.labelX12 = QLabel("X12:", self)
74          self.labelX12.setGeometry(35, 140, 30, 30)
75          self.inputX12 = QLineEdit(self)
76          self.inputX12.setGeometry(70, 140, 60, 30)
77
78          self.labelY12 = QLabel("Y12:", self)
79          self.labelY12.setGeometry(185, 140, 30, 30)
80          self.inputY12 = QLineEdit(self)
81          self.inputY12.setGeometry(220, 140, 60, 30)
82
83
84          self.labelTwo = QLabel("Prava 2:", self)
85          self.labelTwo.setGeometry(485, 30, 70, 20)
86          self.labelTwo.setFont(QFont("Sanserif", 14))
87
88          self.labelX21 = QLabel("X21:", self)
89          self.labelX21.setGeometry(395, 80, 30, 30)
90          self.inputX21 = QLineEdit(self)
91          self.inputX21.setGeometry(430, 80, 60, 30)
92
93          self.labelY21 = QLabel("Y21:", self)
94          self.labelY21.setGeometry(545, 80, 30, 30)
95          self.inputY21 = QLineEdit(self)
96          self.inputY21.setGeometry(580, 80, 60, 30)
97
98          self.labelX22 = QLabel("X22:", self)
99          self.labelX22.setGeometry(395, 140, 30, 30)
100         self.inputX22 = QLineEdit(self)
101         self.inputX22.setGeometry(430, 140, 60, 30)
102
103         self.labelY22 = QLabel("Y22:", self)
104         self.labelY22.setGeometry(545, 140, 30, 30)
105         self.inputY22 = QLineEdit(self)
106         self.inputY22.setGeometry(580, 140, 60, 30)
107
108
109         self.btnBackCoord = QPushButton("NAZAD", self)
```

```
110         self.btnBackCoord.setGeometry(210, 220, 120, 35)
111         self.btnBackCoord.clicked.connect(self.clicked_back_coord)
112 
113 
114         self.btnDrawCoord = QPushButton("POTVRDI", self)
115         self.btnDrawCoord.setGeometry(345, 220, 120, 35)
116         self.btnDrawCoord.clicked.connect(self.clicked_submit_coord)
117 
118     def clicked_back_coord(self):
119         windowChoose.show()
120         self.close()
121 
122     def clicked_submit_coord(self):
123 
124         dictInputs = {}
125         dictInputs['x11'] = self.inputX11.text()
126         dictInputs['x12'] = self.inputX12.text()
127         dictInputs['y11'] = self.inputY11.text()
128         dictInputs['y12'] = self.inputY12.text()
129 
130         dictInputs['x21'] = self.inputX21.text()
131         dictInputs['x22'] = self.inputX22.text()
132         dictInputs['y21'] = self.inputY21.text()
133         dictInputs['y22'] = self.inputY22.text()
134 
135         messageInvalidInput = invalidInputMessage(dictInputs, 'coordinates')
136 
137         if messageInvalidInput != '':
138             dialogInvalidInput.labelWarning.setText(messageInvalidInput)
139             dialogInvalidInput.show()
140             return 0
141 
142              line1 = setLinFuncFromPoints(dictInputs['x11'], dictInputs['y11'],
dictInputs['x12'], dictInputs['y12'])
143 
144              line2 = setLinFuncFromPoints(dictInputs['x21'], dictInputs['y21'],
dictInputs['x22'], dictInputs['y22'])
145 
146         drawPlot(line1, line2)
147 
148         return 1
```

Класа „WindowCoordinates(QWidget)“ такође наслијеђује класу „QWidget“ и односи се на прозор и функционалности које се учитавају након одабира уноса података преко координата тачака. Поред поља за унос, прозор садржи и два дугмета:

- „NAZAD“
- „POTVRDI“

Изглед прозора приказан је на слици 2:



Слика 2: Прозор за унос координата тачака за двије праве

Координате се уносе на следећи начин:

- Права1: тачка1(x11, y11) и тачка2(x12, y12)
- Права2: тачка1(x21, y21) и тачка2(x22, y22)

Кликом на дугме „NAZAD“ затвара се тренутни прозор за унос координата и поново се отвара прозор за одабир начина уноса података за праве. То се извршава позивом функције „clicked_back_coord()“.

Кликом на дугме „POTVRDI“ позива се функција „clicked_submit_coord()“ у којој се врши провјера валидности унесених података преко глобалне функције „invalidInputMessage(dictInputs, inputType)“ која враћа празан стринг уколико су унесени подаци исправни, у супротном враћа поруку одговарајуће грешке. Уколико је унос исправан прелази се на подешавање параметаара линеарних једначина (глобална функција „setLinFuncFromPoints(x1, y1, x2, y2)“) као и на само исцртавање графика (глобална функција drawPlot(line1, line2)) на коме су исцртане праве и исписани подаци о њиховом међусобном положају.

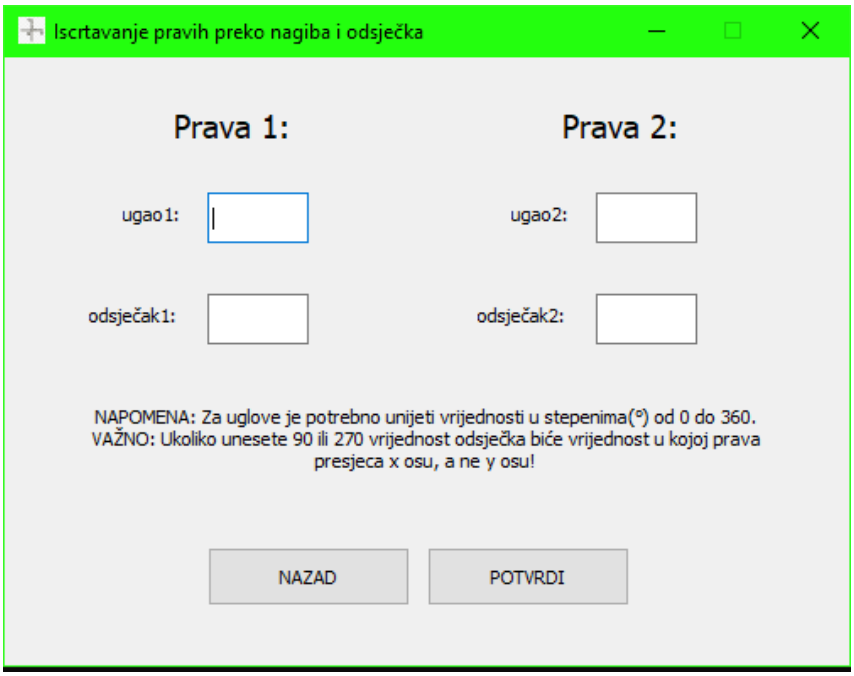
Класа „WindowSlopeIntercept(QWidget)“:

```
151 class WindowSlopeIntercept(QWidget):
152     def __init__(self):
153         super().__init__()
154 
155         self.setGeometry(350, 200, 500, 360)
156         self.setWindowTitle("Iscrtavanje pravih preko nagiba i odsječka")
```

```
157         self.setWindowIcon(QIcon('icons/icon.jpg'))
158 
159         self.setFixedHeight(360)
160         self.setFixedWidth(500)
161 
162 
163         self.labelOne = QLabel("Prava 1:", self)
164         self.labelOne.setGeometry(100, 30, 70, 20)
165         self.labelOne.setFont(QFont("Sanserif", 14))
166 
167         self.labelAngle1 = QLabel("ugao1:", self)
168         self.labelAngle1.setGeometry(70, 80, 50, 25)
169         self.inputAngle1 = QLineEdit(self)
170         self.inputAngle1.setGeometry(120, 80, 60, 30)
171 
172         self.labelIntercept1 = QLabel("odsječak1:", self)
173         self.labelIntercept1.setGeometry(50, 140, 70, 25)
174         self.inputIntercept1 = QLineEdit(self)
175         self.inputIntercept1.setGeometry(120, 140, 60, 30)
176 
177 
178         self.labelTwo = QLabel("Prava 2:", self)
179         self.labelTwo.setGeometry(330, 30, 70, 20)
180         self.labelTwo.setFont(QFont("Sanserif", 14))
181 
182         self.labelAngle2 = QLabel("ugao2:", self)
183         self.labelAngle2.setGeometry(300, 80, 50, 25)
184         self.inputAngle2 = QLineEdit(self)
185         self.inputAngle2.setGeometry(350, 80, 60, 30)
186 
187         self.labelIntercept2 = QLabel("odsječak2:", self)
188         self.labelIntercept2.setGeometry(280, 140, 70, 25)
189         self.inputIntercept2 = QLineEdit(self)
190         self.inputIntercept2.setGeometry(350, 140, 60, 30)
191 
192             self.labelNote = QLabel("NAPOMENA: Za uglove je potrebno unijeti
vrijednosti u stepenima(°) od 0 do 360.\nVAŽNO: Ukoliko unesete 90 ili 270
vrijednost odsječka biće vrijednost u kojoj prava presjeca x osu, a ne y osu!",
self)
193         self.labelNote.setGeometry(50, 190, 400, 70)
194         self.labelNote.setWordWrap(True)
195         self.labelNote.setAlignment(Qt.AlignCenter)
196 
197         self.btnBackSlopeIntercept = QPushButton("NAZAD", self)
198         self.btnBackSlopeIntercept.setGeometry(120, 290, 120, 35)
199 
self.btnBackSlopeIntercept.clicked.connect(self.clicked_back_slope_intercept)
200 
201         self.btnDrawSlopeIntercept = QPushButton("POTVRDI", self)
202         self.btnDrawSlopeIntercept.setGeometry(250, 290, 120, 35)
203 
self.btnDrawSlopeIntercept.clicked.connect(self.clicked_submit_slope_intercept)
204 
205     def clicked_back_slope_intercept(self):
206         windowChoose.show()
207         self.close()
208 
```

```
209     def clicked_submit_slope_intercept(self):
210 
211         dictInputs = {}
212         dictInputs['ugao1'] = self.inputAngle1.text()
213         dictInputs['odsjecak1'] = self.inputIntercept1.text()
214         dictInputs['ugao2'] = self.inputAngle2.text()
215         dictInputs['odsjecak2'] = self.inputIntercept2.text()
216 
217         messageInvalidInput = invalidInputMessage(dictInputs, 'slope_intercept')
218 
219         if messageInvalidInput != '':
220             dialogInvalidInput.labelWarning.setText(messageInvalidInput)
221             dialogInvalidInput.show()
222             return 0
223 
224                     line1   =   setLinFuncFromSlopeIntercept(dictInputs['ugao1'],
dictInputs['odsjecak1'])
225 
226                     line2   =   setLinFuncFromSlopeIntercept(dictInputs['ugao2'],
dictInputs['odsjecak2'])
227 
228         drawPlot(line1, line2)
229 
230         return 1
```

Класа „WindowSlopeIntercept(QWidget)“, као и претходне, наслиједјује „QWidget“ класу и представља прозор са опцијама уноса нагиба (преко угла у степенима) и одсјечака за праве. Прозор се отвара након одабира уноса података за праве преко нагиба и одсјечка на почетном прозору. Поред поља за унос, прозор, као и претходни, садржи и два дугмета: „NAZAD“ и „POTVRDI“. Изглед прозора приказан је на слици 3:



Слика 3: Прозор за унос нагиба и одсјечака за двије праве

$$y = k * x + n$$

k - нагиб

n – одсјечак на у оси

Права1: нагиб је тангенс угла „ugao1“, а одсјечак је „odsječak1“

Права2: нагиб је тангенс угла „ugao2“, а одсјечак је „odsječak2“

ВАЖНО: Потребно је обратити пажњу на напомену гдје пише да уколико унесемо 90 или 270 за вриједности угла у степенима тада ће за ту праву одсјечак представљати вриједност на x оси кроз коју пролази права, а не на у оси јер је таква права паралелна са у осом и не сијече је.

Кликом на дугме „NAZAD“ затвара се тренутни прозор за унос нагиба и одсјечака и поново се отвара прозор за одабир начина уноса података за праве. То се извршава позивом функције „clicked_back_slope_intercept()“.

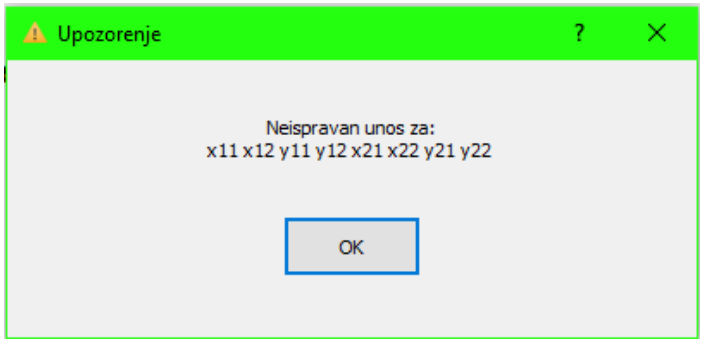
Кликом на дугме „POTVRDI“ позива се функција „clicked_submit_slope_intercept()“ у којој се врши провјера валидности унесених података преко глобалне функције „invalidInputMessage(dictInputs, inputType)“ која враћа празан стринг уколико су унесени подаци исправни, у супротном враћа поруку одговарајуће грешке. Уколико је унос исправан прелази се на подешавање параметаара линеарних једначина (глобална функција „setLinFuncFromSlopeIntercept(angle, intercept)“) као и на само исцртавање графика (глобална функција drawPlot(line1, line2)) на коме су исцртане праве и исписани подаци о њиховом међусобном положају.

Класа „DialogInvalidInput(QDialog)“:

```
233 class DialogInvalidInput(QDialog):
234     def __init__(self):
235         super().__init__()
236 
237         self.setGeometry(500, 220, 400, 160)
238         self.setWindowTitle("Upozorenje")
239         self.setWindowIcon(QIcon('icons/warning.png'))
240 
241         self.setFixedHeight(160)
242         self.setFixedWidth(400)
243 
244         self.labelWarning = QLabel("", self)
245         self.labelWarning.setGeometry(40, 20, 320, 50)
246         self.labelWarning.setWordWrap(True)
247         self.labelWarning.setAlignment(Qt.AlignCenter)
248 
249         self.btnOK = QPushButton("OK", self)
```

```
250         self.btnOK.setGeometry(160, 90, 80, 35)
251         self.btnOK.clicked.connect(self.clicked_ok)
252 
253     def clicked_ok(self):
254         self.close()
```

Класа „DialogInvalidInput(QDialog)“, наслиједђује „QDialog“ класу која даље наслиједђује класу „QWidget“. Служи за испис поруке о погрешном уносу података било преко координата било преко нагиба и одсјечака. Изглед дијалог – прозора за упозорење о погрешном уносу приказан је на слици 4:



Слика 4: Прозор за испис поруке о погрешном уносу података

Порука о погрешном уносу ће бити исписана у лабели „labelWarning“ чији је садржај у почетку празан стринг, али у случају појаве погрешног уноса његов садржај ће се подесити на вриједност поруке (стринг) о грешци која се појавила. Конкретно у горњем случају упозорење се појавило након остављања свих празних поља за унос координата.

Кликом на дугме „OK“ дијалог – прозор са упозорењем се затвара.

Извршни дио датотеке (и цјелокупне апликације):

```
257 if __name__ == '__main__':
258     app = QApplication(sys.argv)
259     windowChoose = WindowChoose()
260     windowChoose.show()
261     windowCoord = WindowCoordinates()
262     windowSlopeIntercept = WindowSlopeIntercept()
263     dialogInvalidInput = DialogInvalidInput()
264     sys.exit(app.exec_())
```

На почетку се инстанцира објекат типа „QApplication()“ уграђеног у „PyQt5“ библиотеку. Затим се инстанцирају и објекти направљених типова „WindowChoose()“, „WindowCoordinates()“, „WindowSlopeIntercept()“ и „DialogInvalidInput()“али се приказује само прозор имплементиран кроз класу „WindowChoose()“ јер је то почетни прозор за

одабир начина уноса података, док ће остали прозори да се отворе у зависности од одабира или уноса од стране корисника апликације.

3.2 Датотека „globalFunctions.py“

Ова датотека садржи само глобалне функције које се позивају у извршној датотеци апликације и неопходне су за правилно извршавање апликације. Свака од њих има своју намјену што ће бити објашњено постепено проласком кроз код.

```
1  from math import *
2  from sympy import *
3  from sympy.plotting import plot, plot_parametric
```

Претходни код се односи на увезене библиотеке неопходне за израчунавања и исцртавање графика функција.

Функција „invalidInputMessage(dictInputs, inputType)“:

```
5  def invalidInputMessage(dictInputs, inputType):
6
7      inputsCopy = dictInputs
8
9      warningMessage = ''
10
11     errInputs = []
12     for i in dictInputs:
13         try:
14             float(dictInputs[i])
15             dictInputs[i] = float(dictInputs[i])
16         except:
17             errInputs.append(i)
18
19     if errInputs != []:
20         dictInputs = inputsCopy
21         warningMessage = "Neispravan unos za:\n"
22         for i in errInputs:
23             warningMessage = warningMessage + i + ' '
24
25         return warningMessage
26
27     if inputType == 'coordinates':
28
29           if samePoints(dictInputs['x11'], dictInputs['y11'], dictInputs['x12'],
dictInputs['y12']):
30              warningMessage = 'Neispravan unos za pravu 1:\nZa iscrtavanje prave
potrebno je unijeti 2 različite tačke!'
31             return warningMessage
32
```

```
33          if samePoints(dictInputs['x21'], dictInputs['y21'], dictInputs['x22'],
dictInputs['y22']):
34              warningMessage = 'Neispravan unos za pravu 2:\nZa iscrtavanje prave
potrebno je unijeti 2 različite tačke!'
35              return warningMessage
36
37      elif inputType == 'slope_intercept':
38
39          if (dictInputs['ugao1'] < 0 or dictInputs['ugao1'] > 360):
40              warningMessage = 'Neispravan unos za ugao1!\nVrijednost ugla mora
biti izmedju 0° i 360°'
41              return warningMessage
42
43          if (dictInputs['ugao2'] < 0 or dictInputs['ugao2'] > 360):
44              warningMessage = 'Neispravan unos za ugao2!\nVrijednost ugla mora
biti izmedju 0° i 360°'
45              return warningMessage
46
47      return warningMessage
```

Функција „invalidInputMessage(dictInputs, inputType)“ служи за провјеру исправноти унесених података за праве.

Она је дефинисана са два параметра: „dictInputs“ и „inputType“. Први представља ријечник (објекат типа dict) који садржи унесене вриједности из свих поља за унос. Други је типа стринг и говори нам да ли треба да провјеравамо податке унијете као координате тачака или као нагиб и одсјечак тј. да ли први аргумент садржи координате тачака или негибе и одсјечке. У првом случају вриједност прослијеђеног другог аргумента биће „coordinates“, а у другом „slope_intercept“.

Функција враћа стринг који може бити празан ако није препознат неисправан унос, или садржи поруку о пронађеном неисправном уносу. Неисправан унос биће сваки онај који није реалан број тј. који не може да се конвертује у тип „float“, затим ако унесемо двије исте тачке за једну праву при уносу координата (позива се функција „samePoints(x1, y1, x2, y2)“) и ако унесемо вриједност која није између 0 и 360 (инклузивно) при уносу угла за одређивање нагиба праве у односу на x осу.

Функција „samePoints(x1, y1, x2, y2)“:

```
50 def samePoints(x1, y1, x2, y2):
51
52      if (x1 == x2) and (y1 == y2):
53          return 1
54
55      return 0
```

Ова функција је дефинисана са четири параметра који представљају координате двије

тачке. Функција враћа 0 уколико су унесене координате за двије различите тачке, док враћа 1 уколико су унесене координате двије исте тачке тј. уколико су x и y координате тачака исте.

Функција „setLinFuncFromPoints(x1, y1, x2, y2)“:

```
58 def setLinFuncFromPoints(x1, y1, x2, y2):
59
60     if (x2 - x1) == 0:
61         Xindex = 1
62         Yindex = 0
63         freeIndex = -x1
64     else:
65         Xindex = ((y2 - y1)/(x2 - x1))
66         Yindex = -1
67         freeIndex = (- ((y2 - y1)/(x2 - x1)) * x1 + y1)
68
69     return (Xindex, Yindex, freeIndex)
```

Ова функција служи за одређивање параметара линеарне функције која представља праву и ти параметри се одређују преко координата двије тачке.

Оно што желимо да одредимо јесу параметри „*a*“, „*b*“ и „*c*“ из општег облика једначине праве:

$$a * x + b * y + c = 0.$$

Ако имамо дате координате двије тачке онда општи облик можемо одредити преко формуле:

$$y * y1 = \frac{y2 - y1}{x2 - x1} * (x - x1)$$

односно:

$$\frac{y2 - y1}{x2 - x1} * x - y + \left(-\frac{y2 - y1}{x2 - x1} * x1 + y1\right) = 0$$

па за „*a*“, „*b*“ и „*c*“ имамо:

$$a = \frac{y2 - y1}{x2 - x1}$$

$$b = -1$$

$$c = \left(-\frac{y2 - y1}{x2 - x1} * x1 + y1\right)$$

Међутим, потребно је предвидјети случај када су координате x1 и x2 једнаке тј. када је права паралелна са y осом јер у том случају не можемо искористити претходне формуле за параметре јер се јавља дијељење са нулом. Права тада има следећи облик:

$$x + (-n) = 0,$$

гдје нам је $n = x1$ или $n = x2$, $x1 = x2$,

па за „*a*“, „*b*“ и „*c*“ имамо:

$$a = 1$$

$$b = 0$$

$$c = -x1 \text{ или } c = -x2$$

Функција „setLinFuncFromPoints(x1, y1, x2, y2)“ је дефинисана са четири параметра који представљају координате двије тачке, а враћа поворку тј. објекат типа „tuple“ који садржи редом параметре „*a*“, „*b*“ и „*c*“ као промјенљиве „Xindex“, „Yindex“ и „freeIndex“.

Функција „setLinFuncFromSlopeIntercept(angle, intercept)“:

```
72 def setLinFuncFromSlopeIntercept(angle, intercept):
73
74     if angle == 90 or angle == 270:
75         Xindex = 1
76         Yindex = 0
77         freeIndex = -intercept
78     else:
79         Xindex = tan(radians(angle))
80         Yindex = -1
81         freeIndex = intercept
82
83     return (Xindex, Yindex, freeIndex)
```

Ова функција такође служи за одређивање параметара линеарне функције која представља праву али се ти параметри одређују преко угла и одсјечка на y оси.

Оно што желимо да одредимо јесу параметри „*a*“, „*b*“ и „*c*“ из општег облика једначине праве:

$$a * x + b * y + c = 0.$$

Ако имамо дат нагиб *k* и одсјечак на *y* оси *n*, онда имамо следећи облик једначине праве:

$$y = k * x + n$$

односно:

$$k * x - y + n = 0$$

па за „*a*“, „*b*“ и „*c*“ имамо:

$$a = k = \tan(radinas(angle))$$

$$b = -1$$

$$c = n = intercept$$

Међутим, потребно је предвидјети случај када задати угао једнак 90° или 270° у односу на *x* осу тј. када је права паралелна са *y* осом јер у том случају не можемо искористити претходне формуле за параметре јер тангенс тих углова тежи бесконачно. Права тада има следећи облик:

$$x + (-n) = 0,$$

гдје нам је $n = intercept$,

па за „*a*“, „*b*“ и „*c*“ имамо:

$$a = 1$$

$$b = 0$$

$$c = -n = -intercept$$

Функција „setLinFuncFromSlopeIntercept(angle, intercept)“ је дефинисана са два параметра који представљају угао и одсјечак, а враћа поворку тј. објекат типа „tuple“ који садржи редом параметре „*a*“, „*b*“ и „*c*“ као промјенљиве „Xindex“, „Yindex“ и „freeIndex“.

Функција „stringGeneralForm(line)“:

```
86 def stringGeneralForm(line):
87
88       return '{:.2f}'.format(line[0])+'*x + '+'({:.2f})'.format(line[1])+' *y + 
'+'({:.2f})'.format(line[2])+' = 0\n'
```

Функција прима један аргумент и то поворку (објекат типа „tuple“) која је у имплементацији иначе резултат неке од двије претходно дефинисане функције тј. та поворка садржи параметре „*a*“, „*b*“ и „*c*“ општег облика једначине праве:

$$a * x + b * y + c = 0.$$

Функција враћа стринг који садржи испис општег облика једначине праве чији су параметри прослијеђени као аргумент типа „tuple“. Параметри „*a*“, „*b*“ и „*c*“ у стрингу који враћа функција су заокружени на двије децимале.

Функција „plotLine(line, data, intersectXabs, intersectYabs)“:

```
91 def plotLine(line, data, intersectXabs, intersectYabs):
92
93     x = symbols('x')
94     y = symbols('y')
95
96     if line[1] == -1:
97             p = plot_parametric((x, line[0]*x + line[2], (x, -intersectXabs-10,
intersectXabs+10)), show = False, xlabel="x", ylabel="y", title=data)
98     else:
99                 p   =   plot_parametric((-line[2],   x,   (x,   -intersectYabs-10,
intersectYabs+10)), show = False, xlabel="x", ylabel="y", title=data)
100
101    return p
```

Функција прима четири параметра:

1. line – поворка параметара „*a*“, „*b*“ и „*c*“ општег облика једначине праве
2. data – стринг који ће бити исписан у наслову графика
3. intersectXabs – апсолутна вриједност x координате тачке пресјека двије праве (служи за скалирање приказа x осе)
4. intersectYabs – апсолутна вриједност y координате тачке пресјека двије праве (служи за скалирање приказа y осе)

У зависности да ли је права паралелна са y осом, вриједност промјенљивој p биће додијељена на различите начине и представљаће објекат типа „Plot“ (увезен из SymPy библиотеке) који ће бити и повратна вриједност функције.

Функција „drawPlot(line1, line2)“:

```
104 def drawPlot(line1, line2):
105
106    x = symbols('x')
107    y = symbols('y')
108
109     systemLinEqResult = solve([line1[0]*x + line1[1]*y + line1[2], line2[0]*x +
line2[1]*y + line2[2]], (x, y))
110
111    strFromResult = 'Opšti oblik jednačina pravih:\n'+stringGeneralForm(line1) +
stringGeneralForm(line2)
112
113    intersectX = 0
```

```
114     intersectY = 0
115     if len(systemLinEqResult) == 2:
116                 strFromResult  +=  'Prave  se  sijeku  u  tački  sa  koordinatama
x='+'{:.2f}'.format(systemLinEqResult[x])+',
y='+'{:.2f}'.format(systemLinEqResult[y])+'\n'
117         intersectX = float(systemLinEqResult[x])
118         intersectY = float(systemLinEqResult[y])
119     elif len(systemLinEqResult) == 1:
120         strFromResult += 'Prave se poklapaju.\n'
121     else:
122         strFromResult += 'Prave su paralelne.\n'
123 
124     p1 = plotLine(line1, strFromResult, abs(intersectX), abs(intersectY))
125 
126     p2 = plotLine(line2, strFromResult, abs(intersectX), abs(intersectY))
127 
128     p1.extend(p2)
129     p1.show()
130 
131     return 1
```

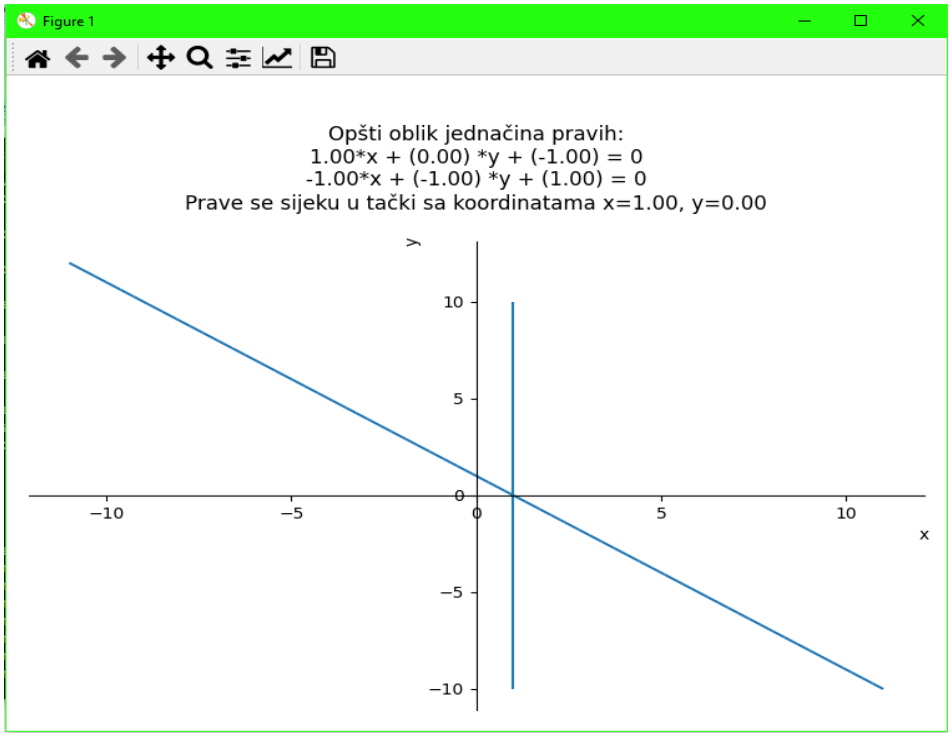
Функција служи за исцртавање графика на којем ће бити приказане обје праве чији су параметри „*a*“, „*b*“ и „*c*“ општег облика једначине праве: $a * x + b * y + c = 0$ прослијеђени као аргументи функције и то у облику поворке *(a, b, c)* за сваку од двије праве.

Најприје се у овој функцији ријешава систем двије линеарне једначине чији су параметри прослијеђени као аргументи функције и рјешење система се одређује помоћу функције „solve“ из „SymPy“ библиотеке која враћа објекат типа „dict“ чија је дужина 2 ако се праве сијеку (садржи x и y координату тачке пресјека), затим 1 ако се праве поклапају (садржи x у зависности од y), или 0 ако су праве паралелне (не постоји рјешење система).

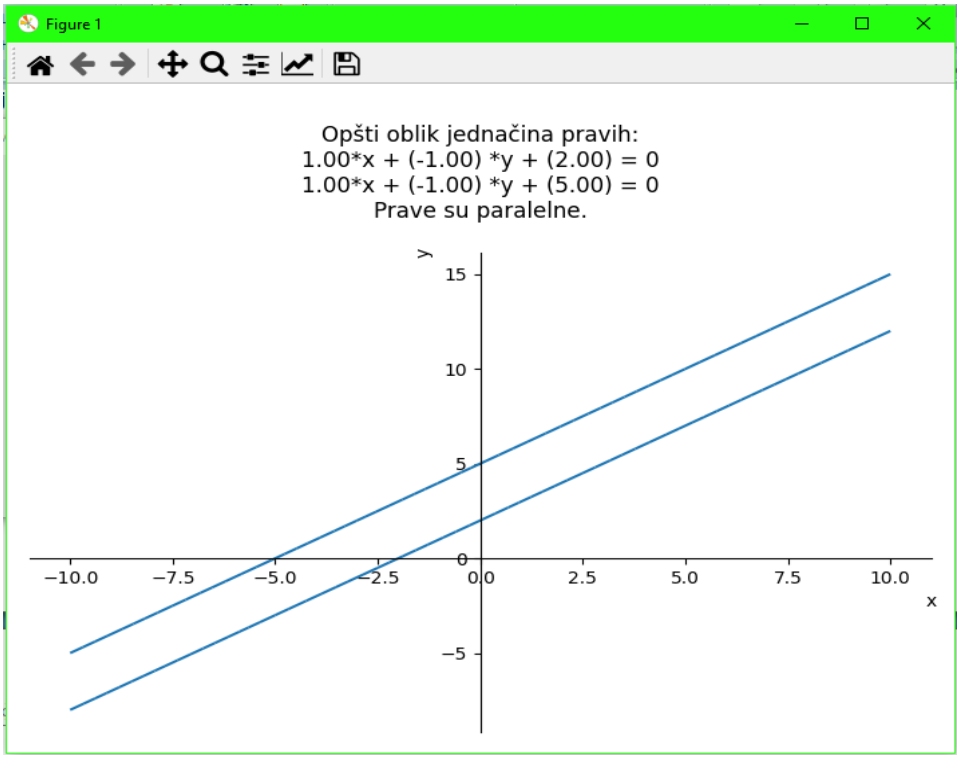
Затим се дужина добијеног рјешења користи за подешавање одговарајућих вриједности које ће бити исписане на графику и неких вриједности које служе за скалирање x и y оса координатног система.

На крају се преко функције „plotLine(line, data, intersectXabs, intersectYabs)“ која је претходно дефинисана додјељују вриједности типа „Plot“ за двије праве које ће бити исцртане на графику. Да бисмо приказали обје праве на истом графику користимо функцију „extend(arg)“ из „SymPy“ библиотеке, док за приказ прозора са графиком двије праве користимо функцију „show()“ такође из „SymPy“ библиотеке.

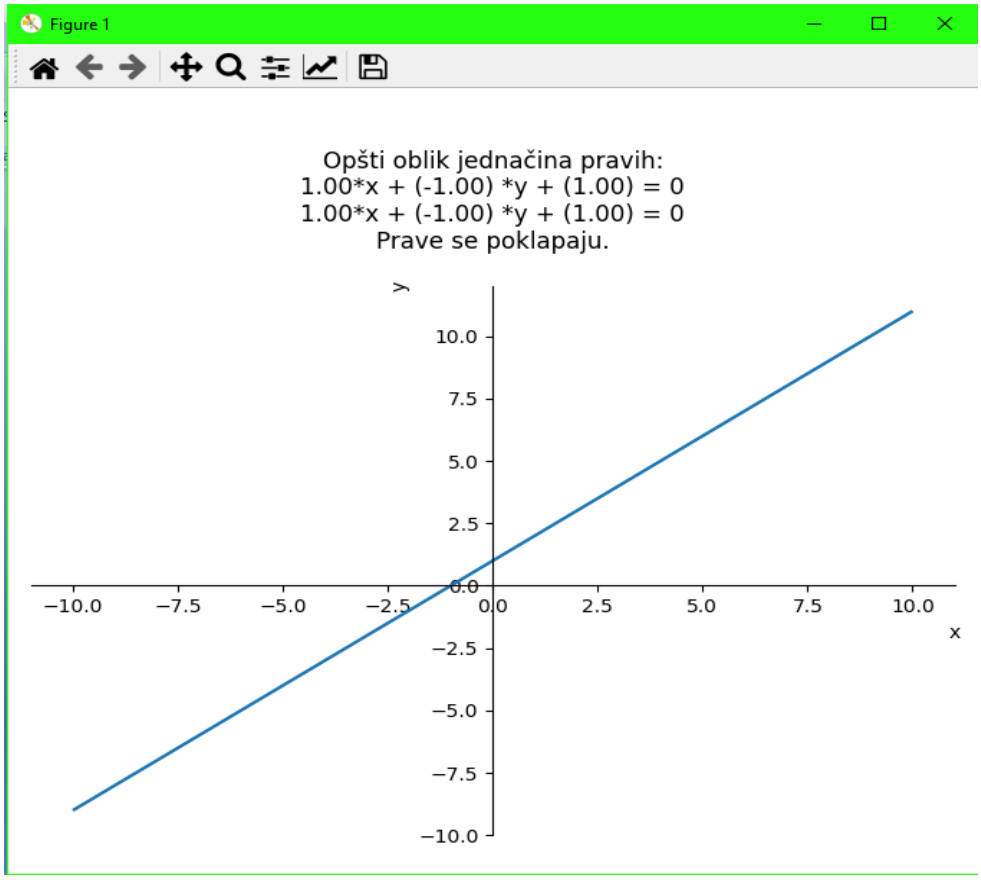
На сликама 5, 6 и 7 приказна су три графика на којима су различите врсте међусобног положаја двије праве:



Слика 5: График са правама које се сијеку



Слика 6: График са правама које су паралелне



Слика 7: График са правама које се поклапају

4. UML дијаграми

UML (Unified Modeling Language) је стандардни графички језик за моделовање објектно-оријентисаног софтвера.

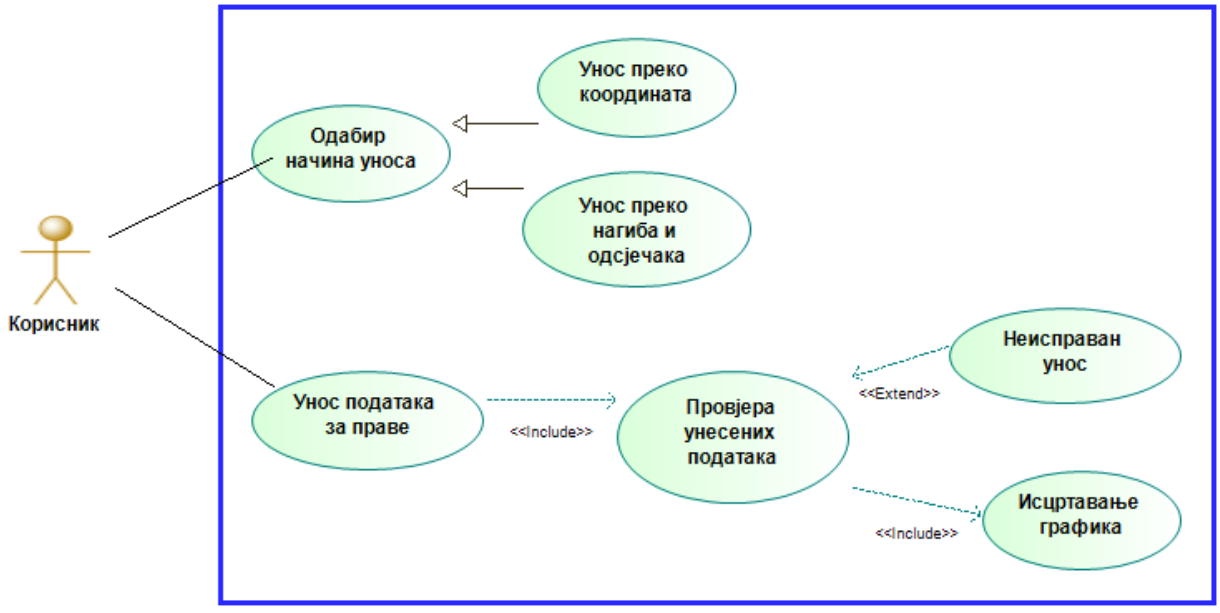
UML је језик који обухвата велики број дијаграма. Они дијаграми који се најчешће користе и који ће овде бити приказани за имплементирану апликацију су:

- Дијаграм случајева коришћења
- Дијаграм секвенци
- Дијаграм активности
- Дијаграм стања
- Дијаграм класа

За приказ дијаграма биће коришћен софтвер Modelio 4.1

4.1. Дијаграм случајева коришћења

Дијаграм случајева коришћења (use-case diagrams) приказује скуп случајева коришћења и актера као и међусобни однос између њих. Случај коришћења специфира шта субјект ради, а не како ради. Случајеви коришћења се приказују елипсама, а актери скицом људског лика. Дијаграм случајева коришћења приказан је на слици 8:



Слика 8: Дијаграм случајева коришћења (Use Case дијаграм)

Случај коришћења: „Одабир начина уноса“:

- Актери: Корисник
- Предуслови: Приступ апликацији
- Опис: Корисник, након покретања апликације, врши одабир начина уноса података за праве гдје бира између уноса преко координата или преко нагиба и одсјечка
- Изузеци: -
- Посљедице: Одабран начин уноса података за двије праве

Случај коришћења: „Унос преко координата“:

- Актери: -
- Предуслови: Одабран начин уноса преко координата
- Опис: Омогућава отварање прозора за унос података за праве преко координата двије тачке за сваку праву. Овај случај коришћења је повезан релацијом генерализације са случајем коришћења „Одабир начина уноса“
- Изузеци: -
- Посљедице: Могућност уноса података преко координата тачака за двије праве

Случај коришћења: „Унос преко нагиба и одсјечака“:

- Актери: -
- Предуслови: Одабран начин уноса преко нагиба и одсјечака
- Опис: Омогућава отварање прозора за унос података за праве преко нагиба и одсјечка за сваку праву. Овај случај коришћења је, заједно са претходним, повезан релацијом генерализације са случајем коришћења „Одабир начина уноса“
- Изузеци: -
- Посљедице: Могућност уноса података преко нагиба и одсјечка за двије праве

Случај коришћења: „Унос података за праве“:

- Актери: Корисник
- Предуслови: Одабран неки од начина уноса података за праве
- Опис: Омогућава унос података за праве преко одабраног начина уноса и потврду унесених података или повратак на прозор са могућношћу одабира начина за унос података за праве
- Изузеци: -
- Посљедице: Провјера унесених података након потврде уноса или поновни одабир начина уноса података за праве

Случај коришћења: „Провјера унесених података“:

- Актери: -
- Предуслови: Унесени подаци и потврђен унос
- Опис: Провјерава исправност унесених података за праве. Овај случај коришћења повезан је везом <<include>> са случајем коришћења „Унос података за праве“
- Изузеци: Неисправан унос – корисник се обавјештава о неисправном уносу
- Посљедице: Приказ графика ако је унос исправан, у супротном приказ обавјештења о неисправном уносу

Случај коришћења: „Неисправан унос“:

- Актери: -
- Предуслови: Унесени неисправни подаци за праве
- Опис: Приказује прозор са обавјештењем о неисправном уносу и о томе гдје се такав унос тачно налази. Овај случај коришћења повезан је везом <<Extend>> са случајем коришћења „Провјера унесених података“
- Изузеци: -
- Посљедице: Корисник је обавјештен о неисправном уносу и у стању је да то исправи

Случај коришћења: „Исцртавање графика“:

- Актери: -
- Предуслови: Унесени исправни подаци за обје праве
- Опис: Отворен прозор са исцртаним графиком на коме су приказане обје праве и подаци о њима и њиховом међусобном положају. Овај случај коришћења повезан је везом <<include>> са случајем коришћења „Провјера унесених података“
- Изузеци: -
- Посљедице: Корисник је дошао до жељеног исхода тј. приказа правих на графику

4.2. Дијаграм секвенци

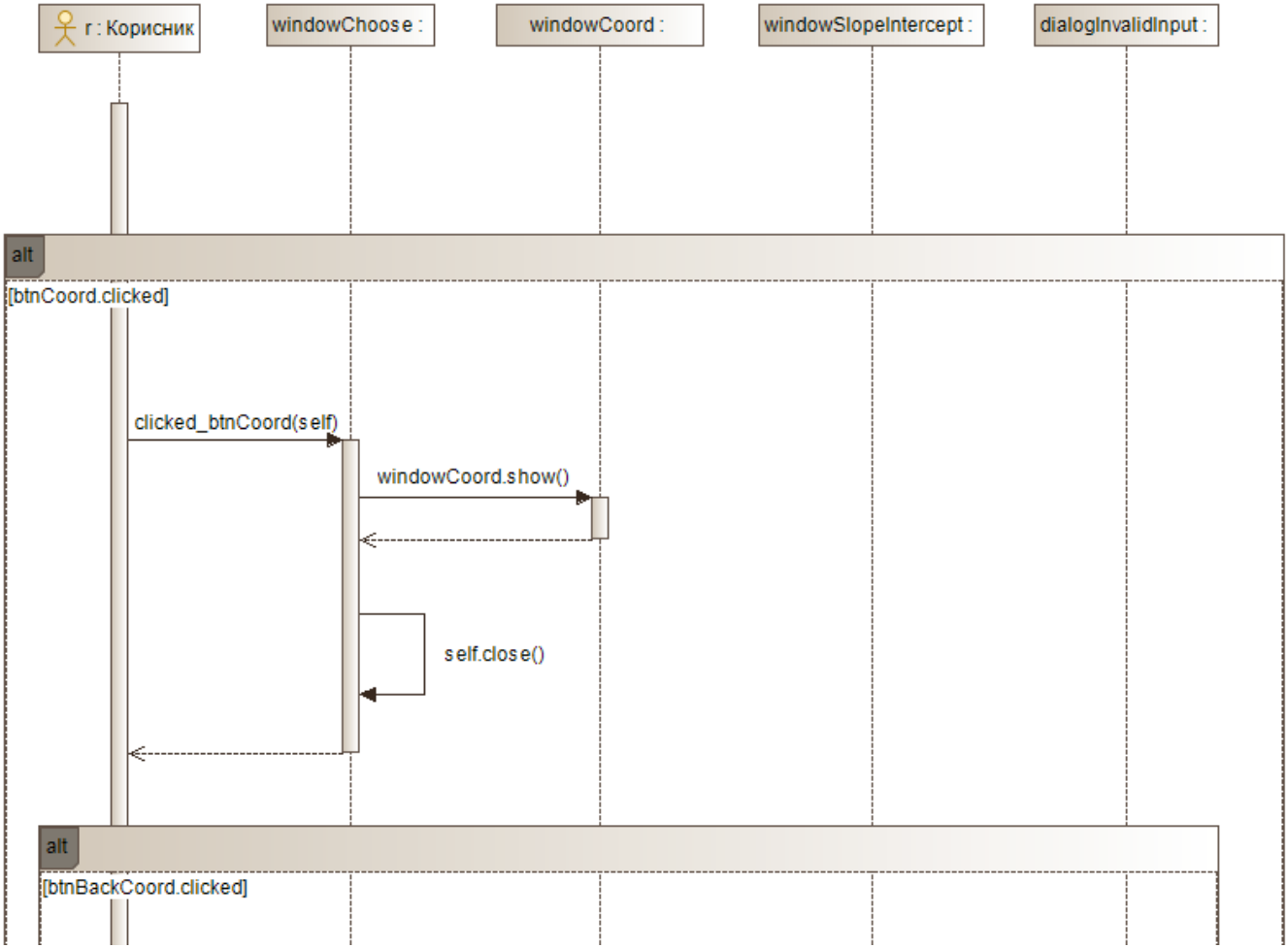
Дијаграми секвенци представљају један од четири типа дијаграма интеракције. Дијаграми интеракције показују следеће:

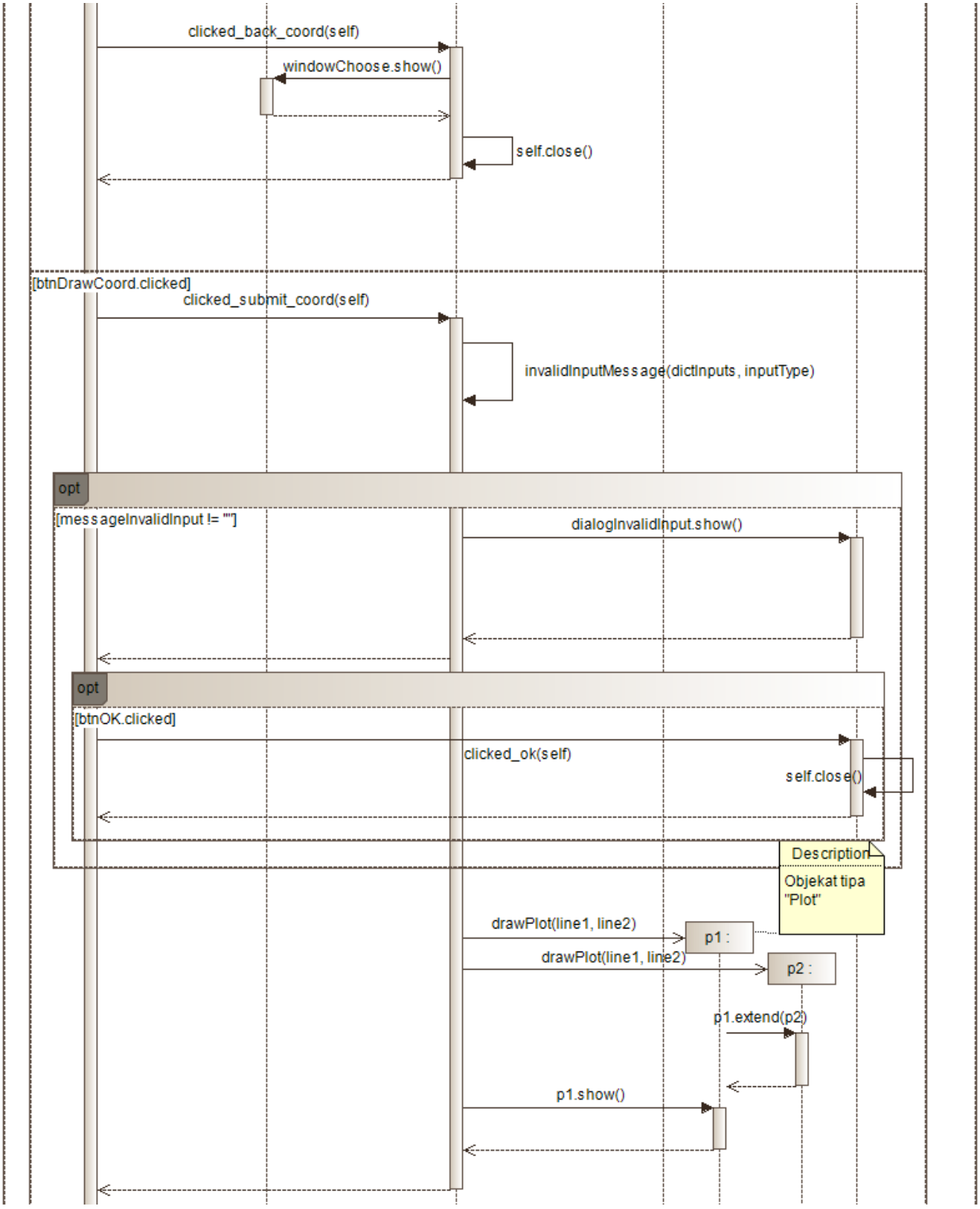
- Интеракција система са околином

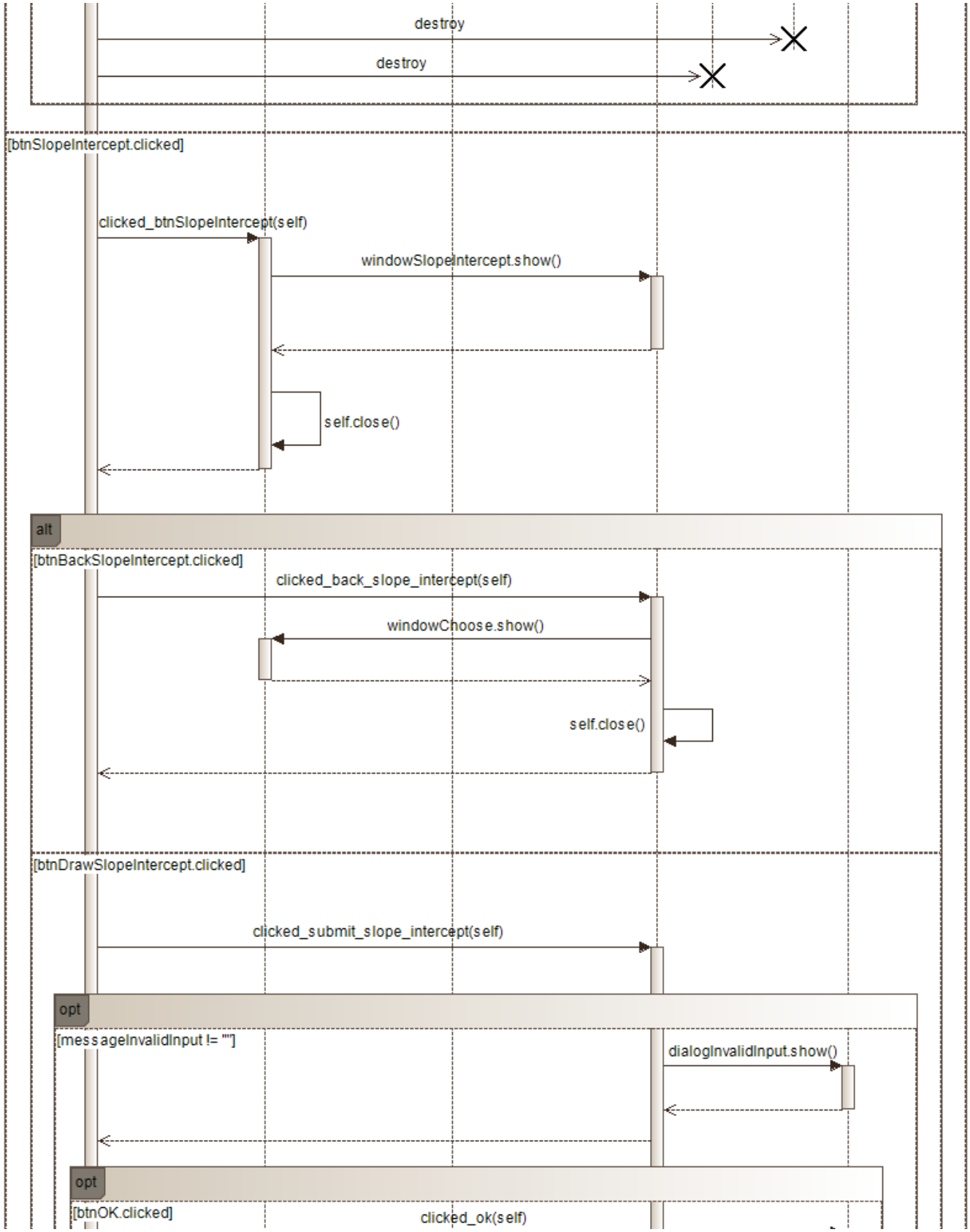
- Интеракција између дијелова система како би се показало како се може примјенити одређени случај коришћења (use - case)
- Међупроцесна комуникација у којој партнери морају да се придржавају одређених протокола
- Комуникација на нивоу класе (оперативни позиви, понашање међу објектима)

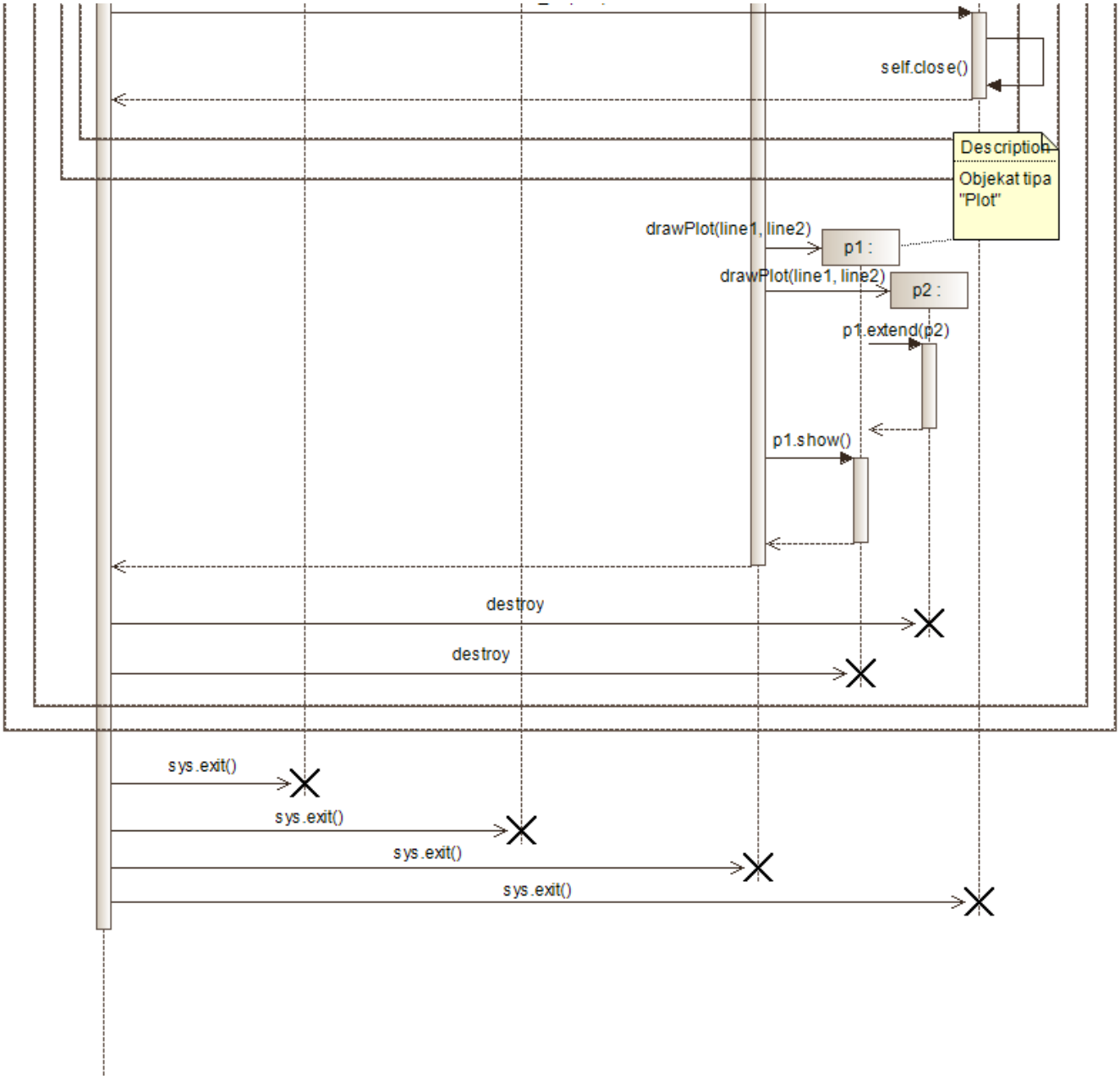
Дијаграм секвенци је заснован на истим концептима. Вертикална оса представља хронолошки ред, а хоризонтална интеракцију. У дијаграму секвенци вријеме тече одозго на доле тј. догађај изнад се десио раније. Обично се објекти који започињу интеракцију стављају с лијеве стране. Показује везе између класа које се успостављају тиме што објекти једне класе покрећу операције друге класе. Дијаграми секвенци могу да садрже актере, објекте и поруке.

Дијаграм секвенци за апликацију исцртавања правих је приказан на слици 9:




clicked_back_coord(self)
windowChoose.show()
self.close()
[btnDrawCoord.clicked]
clicked_submit_coord(self)
invalidInputMessage(dictInputs, inputType)
opt
[messageInvalidInput != ""]
dialogInvalidInput.show()
opt
[btnOK.clicked]
clicked_ok(self)
self.close()
Description
Objekat tipa "Plot"
drawPlot(line1, line2)
p1 :
drawPlot(line1, line2)
p2 :
p1.extend(p2)
p1.show()


destroy
destroy
[btnSlopeIntercept.clicked]
clicked_btnSlopeIntercept(self)
windowSlopeIntercept.show()
self.close()
alt
[btnBackSlopeIntercept.clicked]
clicked_back_slope_intercept(self)
windowChoose.show()
self.close()
[btnDrawSlopeIntercept.clicked]
clicked_submit_slope_intercept(self)
opt
[messageInvalidInput != ""]
dialogInvalidInput.show()
opt
[btnOK.clicked]
clicked_ok(self)



Слика 9: Дијаграм секвенци

Из дијаграма случајева коришћења преузимају се актери, а у овом случају то је један актер – Корисник. Дијаграми се креирају на нивоу објеката, а не на нивоу класе, што дозвољава сценарију једну или више инсатнци исте класе као што је то случај са „p1“ и „p2“ објектима у нашем дијаграму секвенци који су оба инстанце класе „Plot“ која је уграђена у „SymPy“ библиотеку.

Правоугаоници на дијаграму представљају објекте, вертикална испрекидана линија представља животни вијек објекта, док хоризонталне линије представљају поруке које. Објекат може упућивати поруке самом себи и такве поруке се називају ракурзивне

(рефлексивне) поруке и срећемо их у претходно приказаном дијаграму секвенци. Вертикални провоугаоници на линијама живота представљају трајање одређене активности.

4.3. Дијаграм активности

Дијаграми активности су намјењени моделирању динамичких аспеката (понашања) система.

Дијаграм активности приказује:

- Ток активности коју извршавају објекти
- Евентуално и ток објеката између корака активности

Активност је спецификација параметризованог понашања које се изражава кроз ток извршења преко секвенцирања и конкурисања подактивности.

Елементарне јединице подактивности су поједине акције.

Активност репрезентује неатомску обраду која се декомпонује на јединице.

Акција је основна јединица спецификације понашања која репрезентује неку трансформацију или обраду у моделираном систему.

- Акција је основни извршни елемент активности
- Акција представља један корак у активности који се обично даље не компонује
- Активност представља контекст акције
- Активност се може понављати на више мјеста у дијаграму
- Акција се дешава само једном (на датом мјесту унутар дате активности)

Дијаграми активности су графови који садрже **чворове** и **гране**.

Гране:

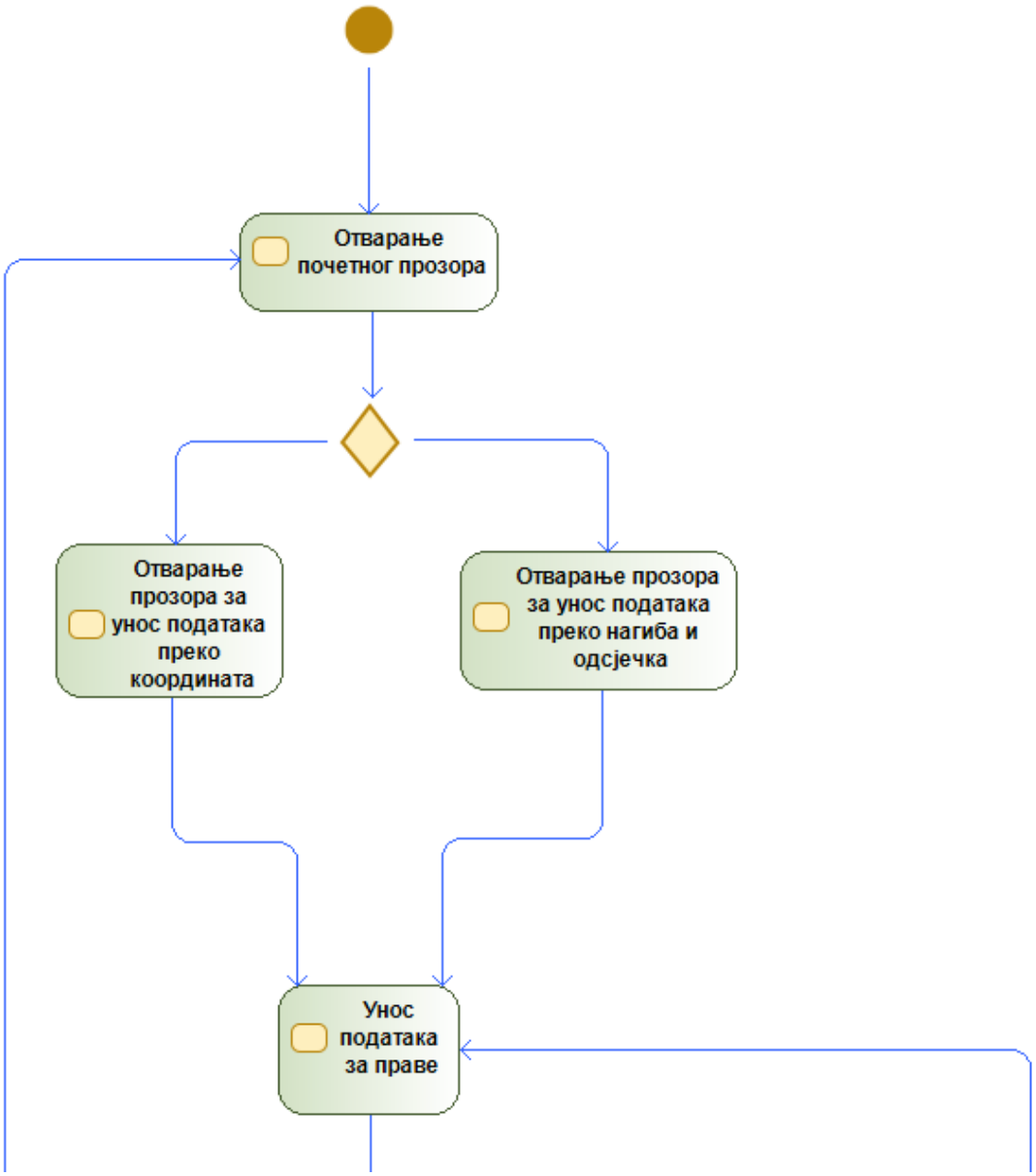
- Прелази (транзиције) између акција
- Ток објекта

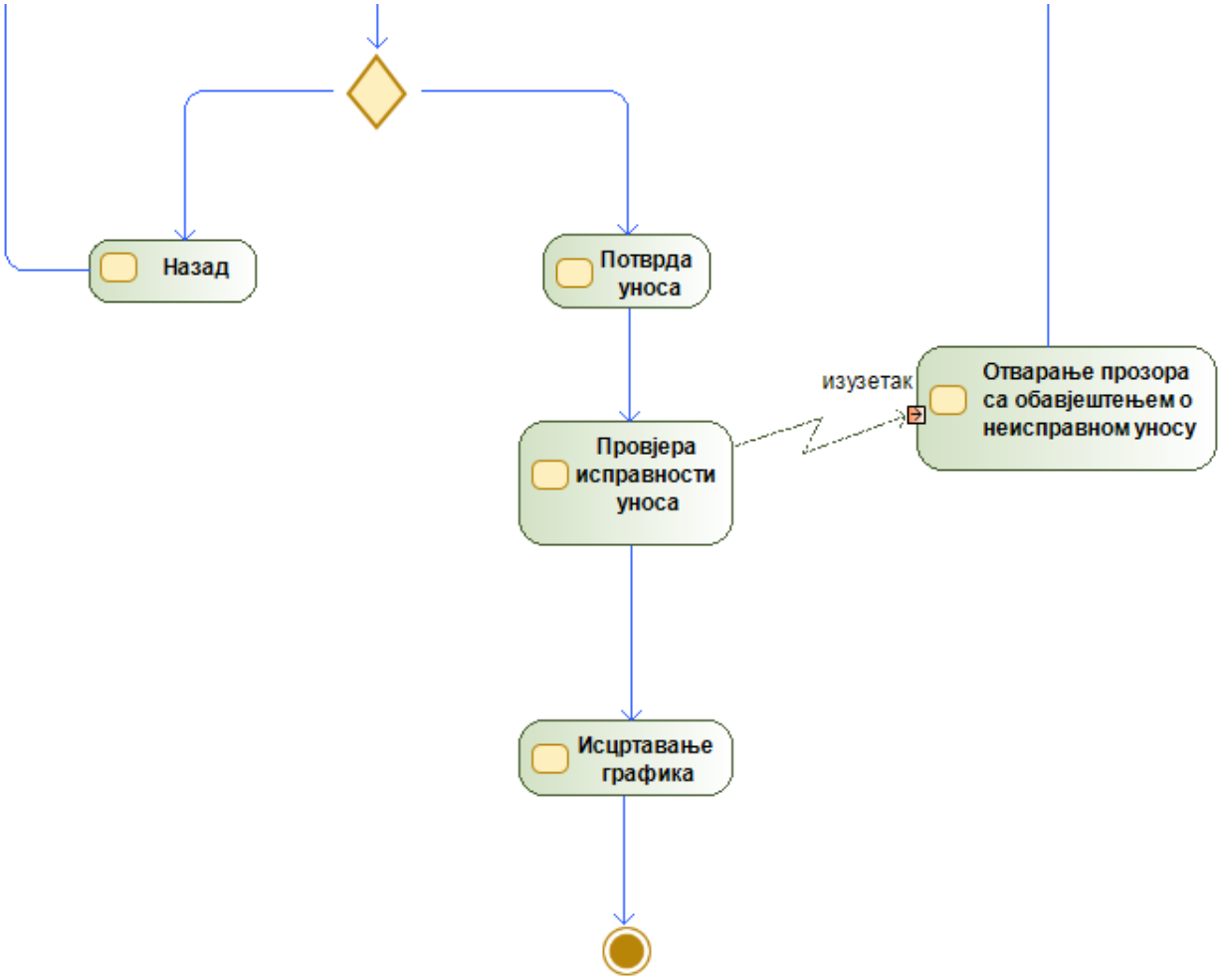
Чворови:

- Акције и активности
- Објекти
- Слање сигнала (send signal)
- Прихватање догађаја (accept event)

- Прихватање временског догађаја (accept time event)
- Контролни чворови
  - Секвенцијална гранања и спајања у току контроле (decision и merge)
  - Конкурентна гранања и спајања у току контроле (fork и join)
- Псеудочворови: почетни, завршни и крај тока
- Конектори

Дијаграм активности везан за овај пројекат је илустративно приказан на слици 10:





Слика 10: Дијаграм активности

4.4. Дијаграм стања

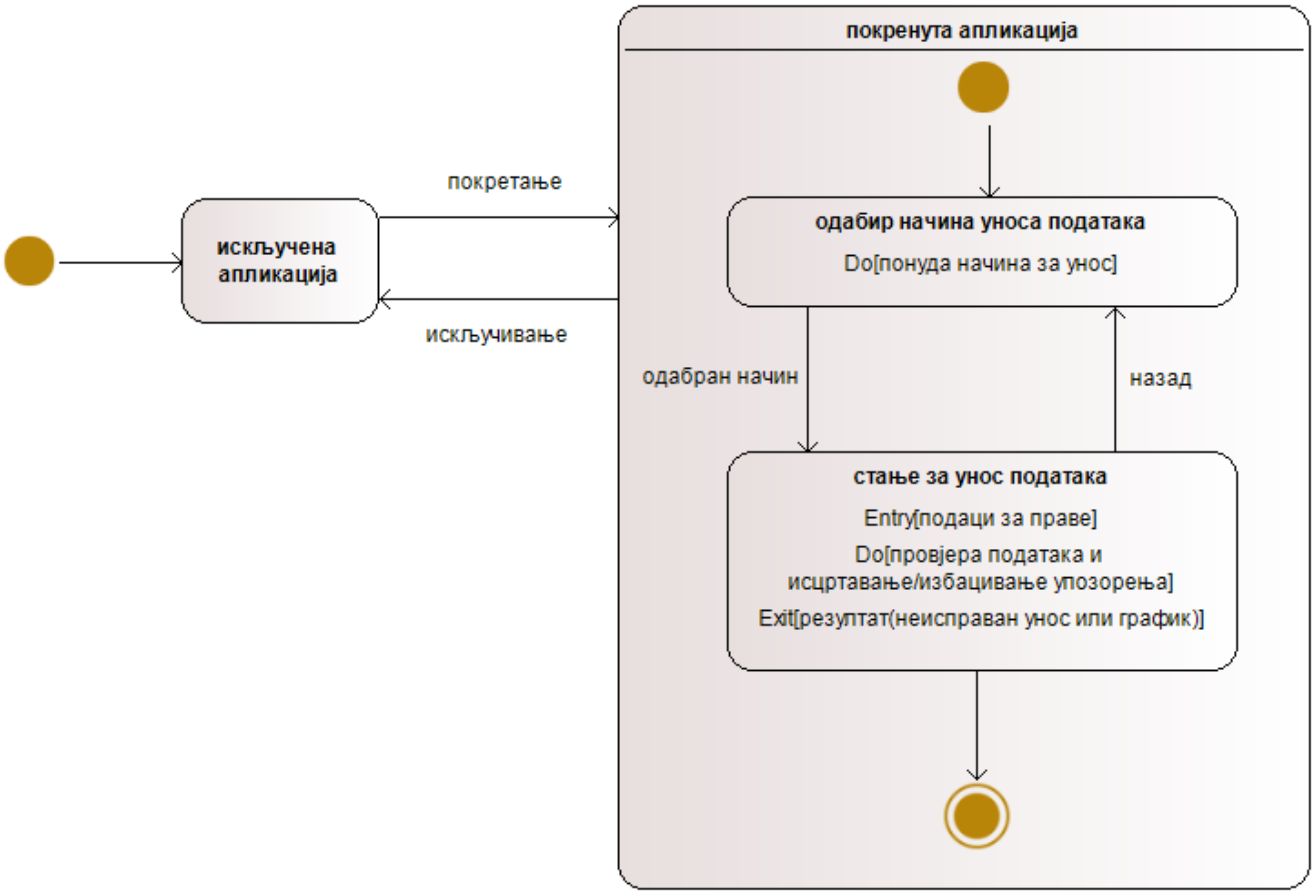
Аутомат стања/Машина стања (state machine) је понашање које специфира секвенце стања кроз која пролази и моделира понашање неког енитета или протокол интеракције. Ентитет реагује на догађај промјеном стања, а промјена стања изазива нове догађаје и акције.

Дијаграм стања је граф који приказује аутомат стања гдје су чворови стања, а гране су прелази. Дијаграми стања се фокусирају на ток активности и на догађајима вођено понашање. Аутомат стања се примјењује да спацифира понашање елемената модела чије текуће понашање зависи од историје.

Дијаграм стања приказује:

- стања и псеудостања (чворови графа)
- прелазе (транзиције) између стања (гране графа)
- догађаје који проузрокују промену стања
- акције које разултују из промене стања

Дијаграм стања ове апликације приказан је на слици 11:



Слика 11: Дијаграм стања

Почетно стање на дијаграму је стање искључене апликације из којег се покретањем прелази у сктивно тј. покренуто стање апликације. Такође се из стања укључене апликације може прећи на стање искључене апликације, искључивањем у било којем моменту. Стање покренуте апликације се састоји из два подстања од којих је прво стање оно у коме је могуће одабрати начин уноса података за праве. Из овог стања се након одабира прелази у стање за унос података. Из тог стања је такође могуће вратити се у стање за одабир начина уноса кликом на дугме „назад“. Да би стање за унос података одрадило посао потребно је унијети податке који се провјеравају и на основу добијеног резултата исцртава се график или се избацује упозорење о неисправном уносу уколико такав унос постоји за неки од података и то је последње стање у које апликација долази.

4.5. Дијаграм класа

Дијаграм класа приказује скуп класа, интерфејса, сарадњи, и других ствари структуре, повезаних ралацијама.

Дијаграм класа је граф образован од тјемена (ствари) повезаних гранама (релацијама). Овај дијаграм специфицира логичке и статичке аспекте модела и најчешћа је врста дијаграма у објектном моделирању.

Елементи дијаграма класа су:

- Ствари: класа, интерфејси, типови, изузеци, шаблони, сарадње, пакети
- Релације: зависности, генерализације, асоцијације, реализације

Класа је скуп објеката са сличним атрибутима, заједничким операцијама (методама) и везама са другим објектима. Симбол класе у дијаграму је правоугаоник подијељен хоризонталним линијама на одјељке.

Атрибути су именована својства класе која описују опсеге вриједности које појаве тог својства могу садржати. Сваки атрибут има свој тип.

Операције су сервиси који се могу захтјевати од неког објекта класе. Нотација: потпис који садржи листу аргумената са евентуалним типовима и подразумјеваним вриједностима, као и типом резултата.

Методе су акције или функције које класа може изводити и може манипулисати атрибутима.

Атрибути и методе могу се означити на следећи начин:

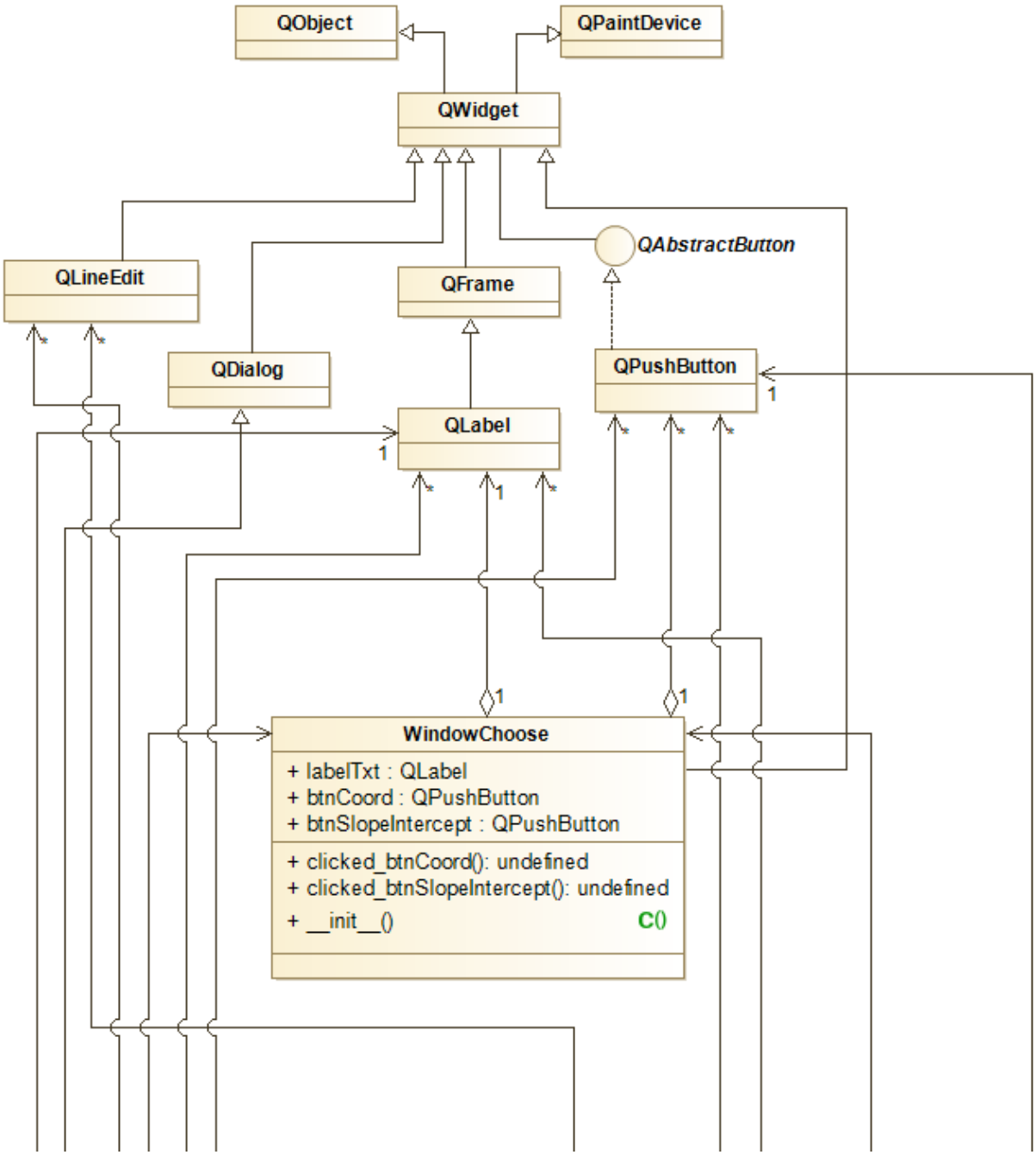
- Заштићени – видљиви само подкласама одређене класе (#)
- Приватни – није видљиво изван класе (-)
- Јавни – видљиво је свима (+)

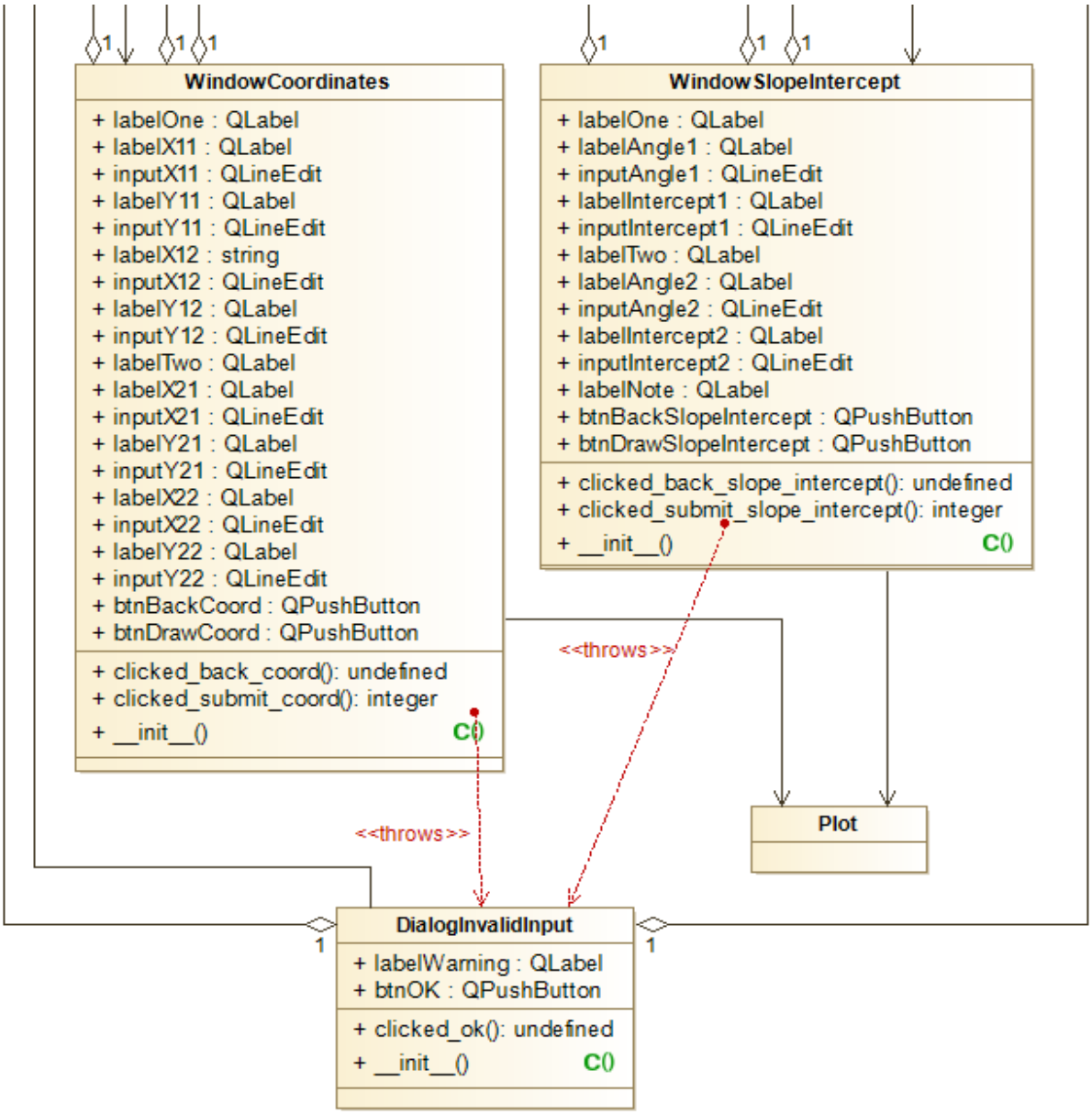
Релације се представљају линијама између класа. На класним дијаграмима се појављују све четири врсте релација:

- Зависност – повезује ствари код којих измјена независне ствари утиче на понашање зависне ствари
- Асоцијација – специфира да ли су примјерци једне ствари повезани са примјерцима друге ствари
- Генерализација – повезује општије са специјализованим стварима (однос надкласе и подкласе)

- Реализација – веза између класа код које један елемент (клијент) реализује понашање другог (добављача)
- Навигабилност – стрелицом означава навигабилност у одређеном правцу

Дијаграм класа за имплементирану апликацију исцртавања правих приказан је на слици 12:





Слика 11: Дијаграм класа

Приказани дијаграм класа садржи и неке класе које нису дефинисане у самом пројекту већ у коришћеним уграђеним Пајтон библиотекама, а оне су у дијаграм убачене због лакшег разумјевања односа међу класама и дефинисаних типова података. Такве уграђене класе приказане су без атрибута и операција, што не значи да их немају већ су само изостављени ради поједностављења изгледа дијаграма.

Везама генерализације приказани су односи надкласа – подкласа и илустративно је приказано које класе су наслијеђене из којих. Чак је присутна и једна реализација интерфејса из уграђене библиотеке.

Присутне су и везе навигабилности (у једном или оба правца) између појединих класа које означавају прозоре за графички приказ.

Такође имамо и двије везе подизања изузетка од старане операција „clicked_submit_coord()“ и „clicked_submit_slope_intercept()“, подизање изузетка резултује интеракцијом са класом „DialogInvalidInput(QDialog)“ тј. приказом прозора чије су карактеристике имплементиране у поменутој класи.

Веза слабе агрегације присутна је између класа у којима је имплементиран приказ одређених елемената (лабеле, поља за унос и дугмад) и класа у којима су имплементиране карактеристике тих елемената. Сваки од тих елемената повезан је са тачно једним примјерком класе са којом је у вези, док та класа са којом су одређени елементи у вези може да иницира инстанцирање (у зависноси која је класа) један или више примјерака класе неког од елемента (због карактеристика софтвера за исцртавње дијаграма више примјерака је означено са * иако се зна тачан број примјерака).

5. Литература

1. http://moodle.fink.rs/ - Портал за електронско учење – курс Софтверски инжењеринг (презентације и видео матријал)
2. https://www.youtube.com/watch?v=Fk1TBoBcrR4&t=1410s - PyQt5 туторијал
3. https://www.riverbankcomputing.com/static/Docs/PyQt5/ - PyQt5 документација
4. https://docs.sympy.org/latest/index.html - SymPy документација
5. Др Владимир М. Миловановић, Компоновање рачунарских програма, Факултет инжењерских наука, Крагујевац, 2021