Универзитет у Крагујевцу

Факултет инжењерских наука

Пројектни рад

Предмет:

Софтверско инжењерство

Тема:

Графичка симулација Соларног система

Студент:

Давид Јеремић 624/2017

Професор:

Ненад Филиповић

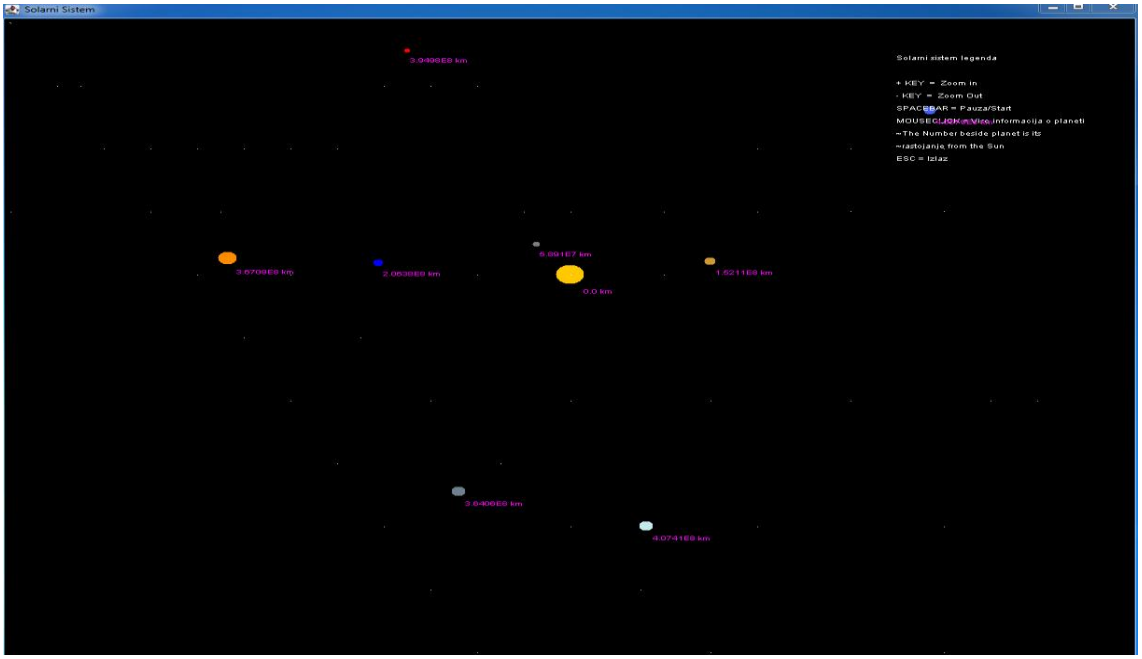
Тијана Шуштершич

САДРЖАЈ

ПОСТАВКА И ОПИС ЗАДАТКА

Тема овог пројектног задатка јесте графичка симулација Соларног система. Све планете треба да су у реалном односу брзина и величина.

Апликација је рађена тј. написана у програмском језику **Java**. Писана је у окружењу *Visual Studio Code*, а најбоље је покренути апликацију у окружењима као што су *Eclipse* и *Intellij Idea.* Да би покренули апликацију, потребно је имати **Java-ину** библиотеку ***Swing*** која служи за креирање GUI.



Слика 1. Покренута апликација

Корисник може мануелно да користи опције које су омогућене у апликацији преко пречица које се налазе на тастатури. Кликом на тастер *SPACEBAR*, корисник добија опцију *Пауза* за паузирање наше апликације и опцију *Старт* за стартовање паузиране апликације. Потом, кликом на тастере *PLUS KEY* и *MINUS KEY,* можемо зумирати и одзумирати наше планете на апликацији. Ако кликнемо директно на неку од планета или на сунце левим тастером миша можемо добити више информација о планети на коју смо кликнули тастером. За крај, кликом на тастер *ESC* излазимо из наше апликације.

Звезде су беле тачке које се насумично крећу по екрану.

ОПИС ИЗВОРНОГ КОДА

Централна класе наше апликације су:

- Небеско тело
- Соларни Систем

КЛАСА *НЕБЕСКО ТЕЛО*

Коришћењем класе Небеско Тело, апликација исцртава и конструише планете и звезде.Све планете имају своју масу, брзину, растојање, дијаметар. На почетку класе смо дефинисали објекте, то можемо приметити на следећим линијама кода(слика 2):

```
1  import java.awt.*;
3
4  public class NebeskoTelo
5  {
6
7      private int masa = 0;
8      private int  dijametar = 0;
9      private double xLoc = 0;
10     private double yLoc = 0;
11     private double velX = 0;
12     private double velY = 0;
13     private double brzina = 0;
14     Color boja;
15     private double ubrzanje =0;
16     private double dirX = 0;
17     private double dirY = 0;
18     private double rastojanje = 0;
19     private double initial=1000;
20     private double max=0;
21     boolean vidljivost;
22     int tackeOrbite[][] = new int[1000][2];
23     int brojac = 0;
```

Слика 2. Класа Небеско Тело

Након тога имамо конструктор за објекте планета(слика 3).

```
27  public NebeskoTelo(double x, double y, double xVelocity, double yVelocity, int masaTela, int dijametarTela, Color bojaTela, double brzinaTela)
28  {
29     xLoc = x;
30     yLoc = y;
31     velX = xVelocity;
32     velY = yVelocity;
33     masa = masaTela;
34      dijametar = dijametarTela;
35     boja = bojaTela;
36      brzina = brzinaTela;
```

Слика 3. Конструктор класе Небеско Тело

Јако је битно напоменути функције које служе за цртање и конструисање планета и звезда(слика 4).

```
96  public void draw(Graphics g, double velicina)
97  {
98      g.setColor(boja);
99      g.fillOval((int)(650+(xLoc- dijametar/2-650)*velicina), (int)(500+(yLoc- dijametar/2-500)*velicina),
100                 (int)( dijametar*velicina), (int)( dijametar*velicina));
101 }
102 public void disDes(Graphics g, double scale)
103 {
104     g.setColor(boja);
105     for (int[] orbit : tackeOrbite)
106         g.drawLine(orbit[0],orbit[1],orbit[0],orbit[1]);
107     g.setFont(new Font("Arial", Font.PLAIN, 10));
108     g.setColor(Color.MAGENTA);
109
110     g.drawString((Math.round(rastojanje*100.0)/100.0) * 1000000 + " km",
111                 dijametar+(int)(600+(xLoc- dijametar/2-600)*scale), 16+(int)(400+(yLoc- dijametar/2-400)*scale)+ dijametar);
112
113 }
```

Слика 4. Функција за исртавање планета

Такође важна функција *update* за конструисање звезда(слика 5).

```
71  public void update(double ZvezdaX, double ZvezdaY, int MasaZvezde)
72  {
73      if (vidljivost){
74          tackeOrbite[brojac][0]=(int)(xLoc+.5);
75          tackeOrbite[brojac][1]=(int)(yLoc+.5);
76          brojac = (brojac+1)%1000;
77      }
78      else{
79          tackeOrbite = new int[1000][2];
80          brojac = 0;
81      }
82      rastojanje = Math.sqrt((ZvezdaX - xLoc)*(ZvezdaX - xLoc) + (ZvezdaY - yLoc)*(ZvezdaY - yLoc));
83      initial = Math.min(rastojanje,initial);
84      max = Math.max(rastojanje,max);
85
86      ubrzanje = MasaZvezde/rastojanje/rastojanje;
87
88      dirX = (ZvezdaX-xLoc)/rastojanje;
89      dirY = (ZvezdaY-yLoc)/rastojanje;
90
91      velX += dirX * ubrzanje;
92      velY += dirY * ubrzanje;
93      move();
94
95  }
```

Слика 5. Функција за конструисање звезда

Аргументи ове функције су координате Звезда и њихова маса. У функцији рачунамо растојање звезда, њихово убрзање и позиције.

КЛАСА *СОЛАРНИ СИСТЕМ*

У класи Соларни Систем убацују се сви параметри небеског тела, као и слике(слика 6). У бесконачној петљи се стално ажурирају координате сваке од планета.

У овој класи смо користили два низа, један који служи за учитавање слика планета, а други који служи за планете.

```
11      Model model;
12      NebeskoTelo[] nebeskaTela = new NebeskoTelo[9];
13      boolean[] descriptionSeen = new boolean[9];
14
15      final static int DELAY = 50;
16      double velicina = 1;
17      BufferedImage[] buffimgs = new BufferedImage[9];
18      String[][] description;
19      boolean stop = false;
20      int clicked = -1;
```

Слика 6. Атрибути и низови

Као што смо рекли један низ садржи типове података *buffimgs,* а други инстанце класе Небеско Тело.

Класа Соларни Систем садржи два конструкотора:

1. Један који се користи за подразумеване величине и параметре
2. Други који се користи за мануелно коришћење

```
    public SolarniSistemMain()
    {
          model = new Model();
          model.setPreferredSize(new Dimension(1200, 1200));
          add(model);

           nebeskaTela[0] = new NebeskoTelo(600, 450, -4.7, 0, 9, 8, Color.GRAY, 1000); //Merkur
           nebeskaTela[1] = new NebeskoTelo(752, 400, 0, 2.5, 900, 12, new Color(207,153,52), 1000); //Venera
           nebeskaTela[2] = new NebeskoTelo(600, 150, 1.8, 0, 900, 11, Color.BLUE, 2000); //Zemlja
           nebeskaTela[3] = new NebeskoTelo(650, -50, 1.2, 0, 900, 7, Color.RED, 2000); //Mars
           nebeskaTela[4] = new NebeskoTelo(600, -100, 1.2, 0, 900, 20, new Color(255,140,0), 2000); //Jupiter
           nebeskaTela[5] = new NebeskoTelo(600, -150, 1.2, 0, 900, 15, new Color(112,128,144), 2000); //Saturn
           nebeskaTela[6] = new NebeskoTelo(600, -175, 1.2, 0, 900, 15, new Color(196,233,238), 2000); //Uran
           nebeskaTela[7] = new NebeskoTelo(0, 400, 0, -1.2, 900, 13, new Color(66, 98, 243), 2000);//Neptun

           nebeskaTela[8] = new NebeskoTelo(600, 400, .1, 0, 1000, 30, Color.ORANGE, 0);//Sunce



          setBackground(Color.BLACK);

          description = new String[][]{
              {"Merkur"," dijametar: " + nebeskaTela[0].getDijametar()*1058 + "kilometara",
                  "masa: 0.330 x 10^(24) kg",
                  "Tip atmosfere: Tanak",
                  "Prosecna temperatura: 167  C",
                  "Prosecno trajanje dana: 3.1 dan planete Zemlje",
                  "Najbliza suncu"},
              {"Venera"," dijametar: " + nebeskaTela[1].getDijametar()*1058 + "kilometara",
                    "masa: 4.87 x 10^(24) kg",
                    "Tip atmosfere: Srednje Tanak",
                    "Prosecna temperatura: 464  C",
                    "Prosecno trajanje dana: 9 dana planete Zemlje",
                    "Poznata kao blizanac planete zemlje"
                  },
              { "Planeta Zemlja"," dijametar: " + nebeskaTela[2].getDijametar()*1058 + "kilometara",
                      "masa: 5.97 x 10^(24) kg",
                      "Tip atmosfere: Tanak",
                      "Prosecna temperatura: 15  C",
                      "Prosecno trajanje dana: 1 dan planete Zemlje",
                      "Nas dom"
                 },
              { "Mars"," dijametar: " + nebeskaTela[3].getDijametar()*1058 + "kilometara",
                  "masa: 0.642 x 10^(24) kg",
                  "Tip atmosfere: Srednje debeo",
                  "Prosecna temperatura: -65  C",
                  "Prosecno trajanje dana: 8 dana planete Zemlje",
                  "Poznata kao crvena planeta"
                  },
```

Слика 7. Конструктор

```
70                  },
71              {"Jupiter"," dijametar: " + nebeskaTela[4].getDijametar()*1058 + "kilometara",
72                      "masa: 1898 x 10^(24) kg",
73                      "Tip atmosfere: Debeo",
74                      "Prosecna temperatura: -110  C",
75                      "Prosecno trajanje dana: 6 dana planete Zemlje",
76                      "Najveca planeta u solarnom sistemu"},
77                  {"Saturn"," dijametar: " + nebeskaTela[5].getDijametar()*3058 + "kilometara",
78                      "masa: 568 x 10^(24) kg",
79                      "Prosecna temperatura: -140  C",
80                      "Tip atmosfere: Debeo",
81                      "Poznata po svojim prstenovima"},
82                  {"Uran"," dijametar: " + nebeskaTela[6].getDijametar()*3058 + "kilometara",
83                          "masa: 86.8 x 10^(24) kg",
84                          "Tip atmosfere: Debeo",
85                          "Prosecna temperatura: -195  C",
86                          "Sastavljena od leda i stena"},
87              {"Neptun"," dijametar: " + nebeskaTela[7].getDijametar()*1058 + "kilometara",
88                  "masa: 102 x 10^(24) kg",
89                  "Tip atmosfere: Tanak",
90                  "Tip atmosfere: Debeo",
91                  "Prosecna temperatura: -200  C",
92                  "Prosecno trajanje dana: 6 dana planete Zemlje",
93                  "Jedina planeta pronadjena matematickim predvidjanjem"},
94              {"Sunce"," dijametar: " + nebeskaTela[8].getDijametar()*3058 + " kilometara",
95                  "masa: 1.989 Ã— 10^30 kg",
96                  "Tip atmosfere: Debeo",
97                  "Prosecna temperatura: 5505  C",
98                  "Najvece nebesko telo u solarnom sistemu"},
99           };
100
101          buffimgs[0] = getSlika("Merkur.jpg");
102          buffimgs[1] = getSlika("Venera.jpg");
103          buffimgs[2] = getSlika("PlanetaZemlja.jpg");
104          buffimgs[3] = getSlika("Mars.jpg");
105          buffimgs[4] = getSlika("Jupiter.jpg");
106          buffimgs[5] = getSlika("Saturn.jpg");
107          buffimgs[6] = getSlika("Uran.jpg");
108          buffimgs[7] = getSlika("Neptun.jpg");
109          buffimgs[8] = getSlika("Sunce.jpg");
110
111
112         Thread thread =  new Thread() {
113
114           @Override
115           public void run() {
116              gameLoop();
117           }
118         };
119
120         thread.start();
121     }
```

Слика 8. Учитавање слика и текста

Модел је постављен на величину 1200 пиксела пута 1200 пиксела. Између линија 28 и 37, иницијализују се планете и сунце. Линија 41 поставља црну позадину. Након тога између линија 43 и 98, налази се опис за сваку планету и сунце који се касније појављује на екрану апликације. Касније, између линија 101 и 109 се учитавају слике планета.

```
134  private void gameLoop() {
135
136    while (true) {
137          if (!stop)
138          {
139              for(int i = 0; i <nebeskaTela.length-1; i++)
140              {
141                   nebeskaTela[i].update( nebeskaTela[8].getXPozicija(), nebeskaTela[8].getYPozicija(), nebeskaTela[8].getMasa());
142              }
143          }
144       repaint();
145
146       try {
147          Thread.sleep(DELAY);
148       } catch (InterruptedException ex) { }
149    }
150  }
151
152
```

Слика 9. Петља

Функција *gameLoop* је бесконачна петља, где се стално ажурирају координате планета. А линија 144 је намењена за поновно цртање планета.

```
153    class Model extends JPanel implements KeyListener, MouseListener {
154      public Model() {
155
156         setFocusable(true);
157         requestFocus();
158         addKeyListener(this);
159         addMouseListener(this);
160      }
161
162
163      public void paintComponent(Graphics g) {
164
165
166         for(NebeskoTelo body :nebeskaTela)
167            body.draw(g,velicina);
168
169
170
171         for(int count=0;count<=1000;count++) {
172             g.setColor(Color.WHITE);
173
174             g.drawOval(50*count,100*count,1,1);
175            g.drawOval(75*count,100*count,1,1);
176
177             g.drawOval(100*count,200*count,1,1);
178             g.drawOval(150*count,200*count,1,1);
179             g.drawOval(200*count,200*count,1,1);
180             g.drawOval(250*count,200*count,1,1);
181             g.drawOval(300*count,200*count,1,1);
182             g.drawOval(350*count,200*count,1,1);
183             g.drawOval(400*count,100*count,1,1);
184             g.drawOval(450*count,100*count,1,1);
185             g.drawOval(500*count,100*count,1,1);
186             g.drawOval(550*count,300*count,1,1);
187             g.drawOval(600*count,300*count,1,1);
188             g.drawOval(700*count,300*count,1,1);
189             g.drawOval(800*count,300*count,1,1);
190             g.drawOval(900*count,300-count,1,1);
191             g.drawOval(1000*count,300-count,1,1);
192
193
194
195             }
```

Слика 10. Исцртавање збезда

Класа модел служи за цртање планета и сунца на линији 163. У *for* петљи се исртавају звезде на насумичним местима.

UML ДИЈАГРАМИ

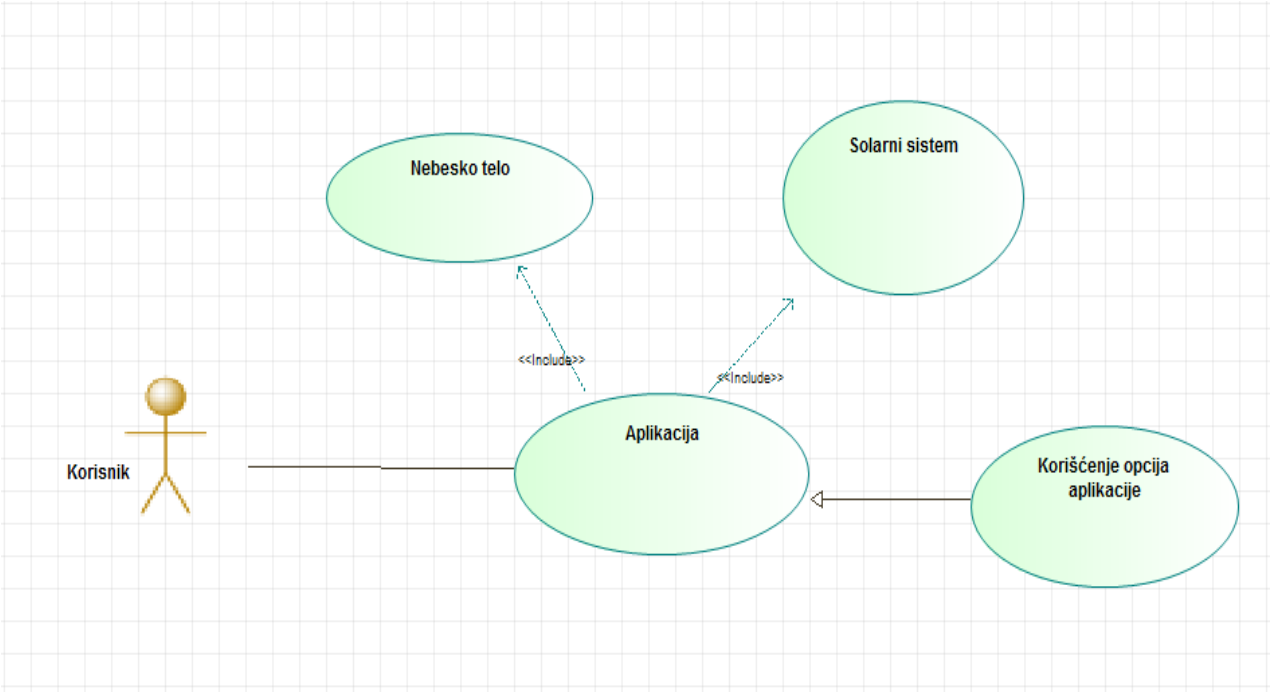
USE CASE DIAGRAM

- Дијаграм случајева коришћења (use -case) приказује скуп случајева коришћења и актера
- Типично се користи да специфира неку фукционалност и понашање неког субјекта
- Дијаграм визуелизује понашање система, подсистема или чак класе и интерфејса

Елементи дијаграма су:

- случајеви коришћења
- актери
- релације
- пакети

Дијаграм случајева коришћења на нашем пројекту изгледа овако(слика 11):



Слика 11. Use case diagram

ДИЈАГРАМ СЕКВЕНЦИ

Дијаграм секвенци приказује комуникацију између скупа објеката, која се остварује порукама које објекти међусобно размењују у циљу остваривања очекиваног понашања. Детаљно описује како се операције изводе – које поруке се шаљу и када.

Дијаграм секвенци је један од дијаграма интеракције

• Интеракција – понашање које обухвата скуп порука које се размењују

између скупа објеката у неком контексту са неком наменом

• Порука – спецификација комуникације између објеката која преноси

информацију

• Пријем поруке изазива акцију – извршење наредбе

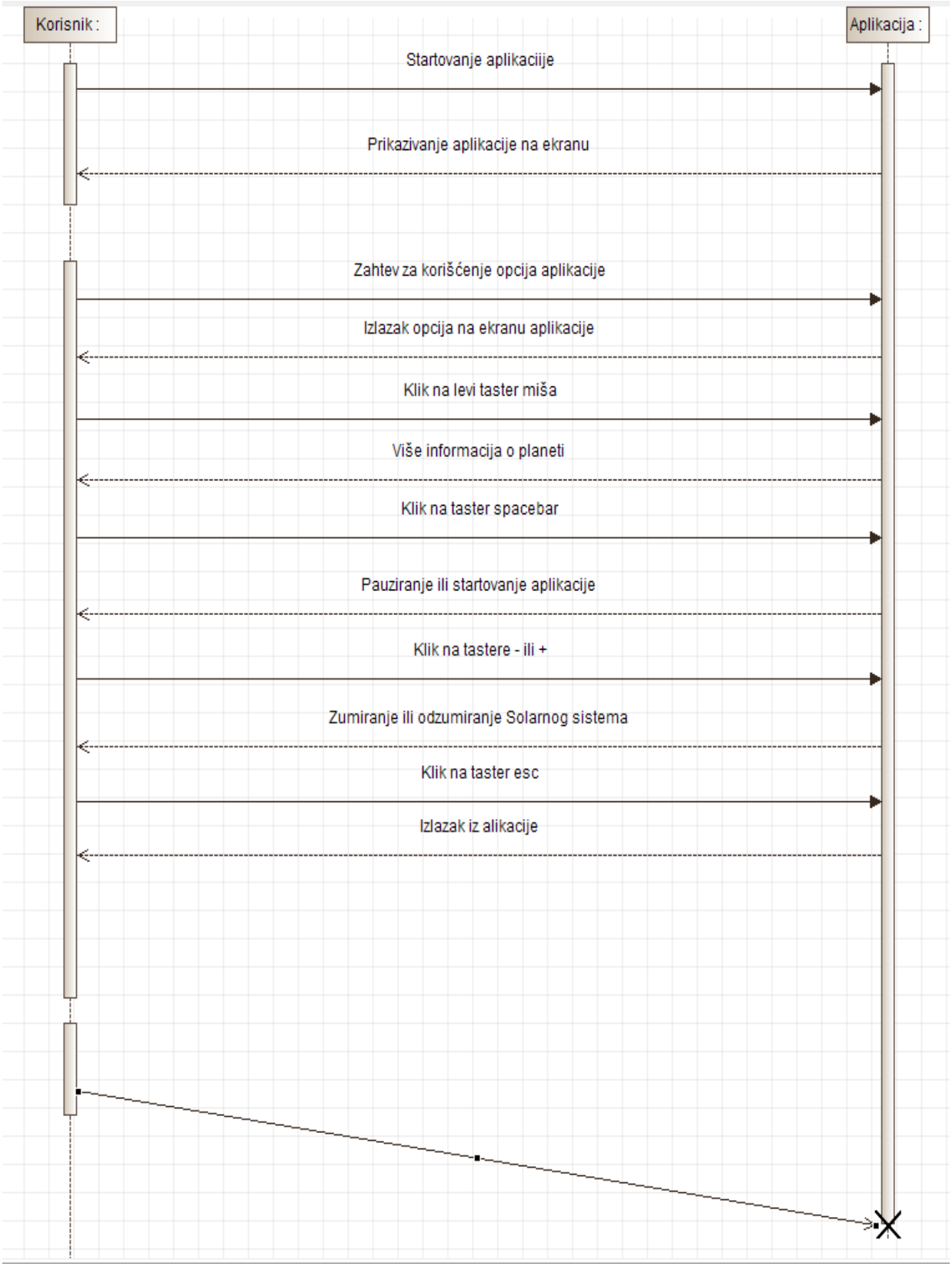
Ако су дијаграми случајева употребе претходно дефинисани – дијаграм секвенци је

једна од његових реализација – показује редослед:

• Догађаја – спољашњи улазни догађај – генерише учесник

• Операција – оџив на догађај у систему

На следећој слици можемо видети овај дијаграм(слика 12):

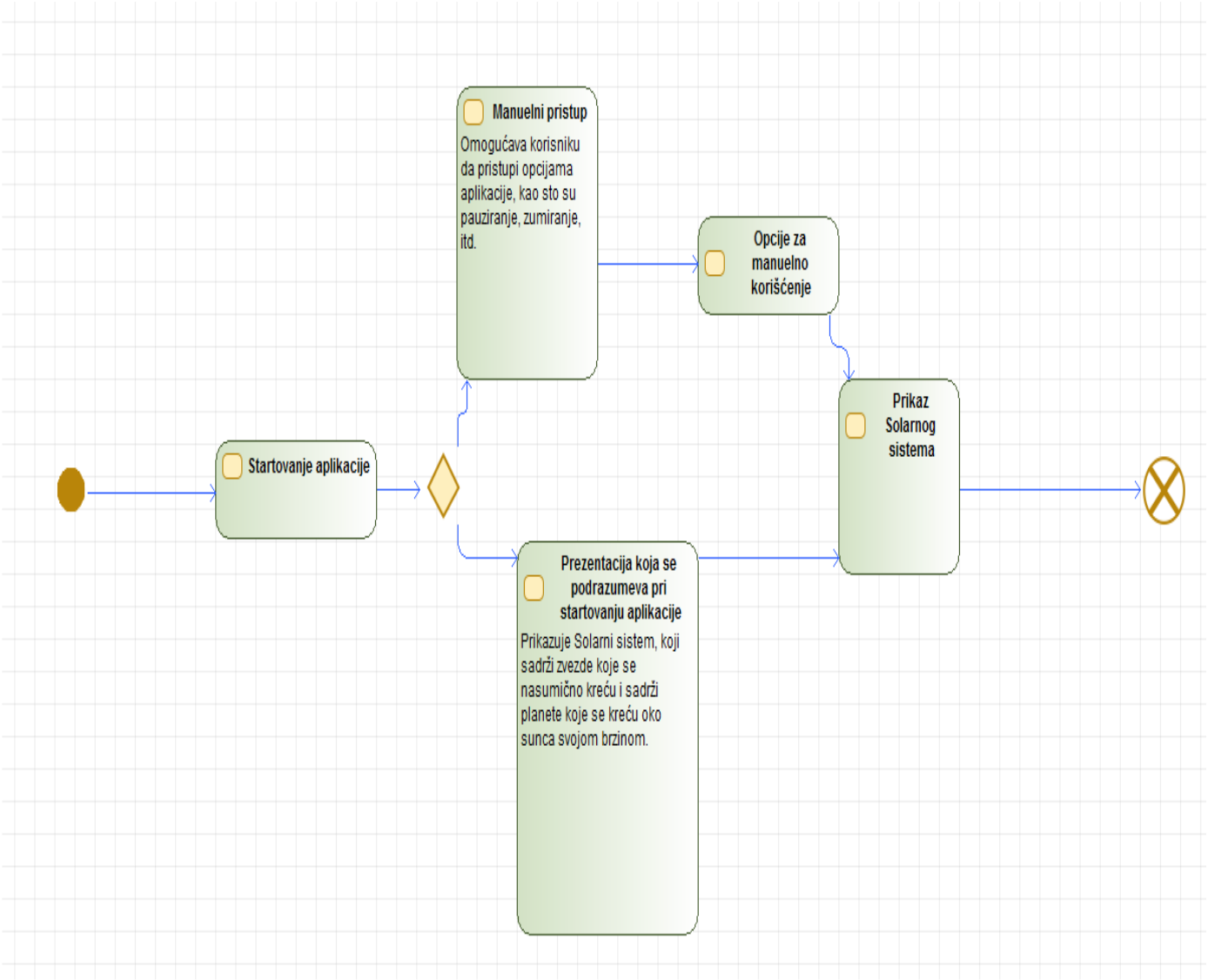


Слика 12. Дијаграм секвенци

ДИЈАГРАМ АКТИВНОСТИ

Дијаграм активности је намењен моделирању динамичких аспеката система. Дијаграм активности приказује ток активности коју извршавају објекти и евентуално и ток објеката између корака активности.

Активност је спецификација параметризованог понашања које се изражава кроз ток извршења преко секвенцирања и конкурисања подактивности. Дијаграм активности садржи чворове и гране.



Слика 13. Дијаграм активности

ДИЈАГРАМ СТАЊА

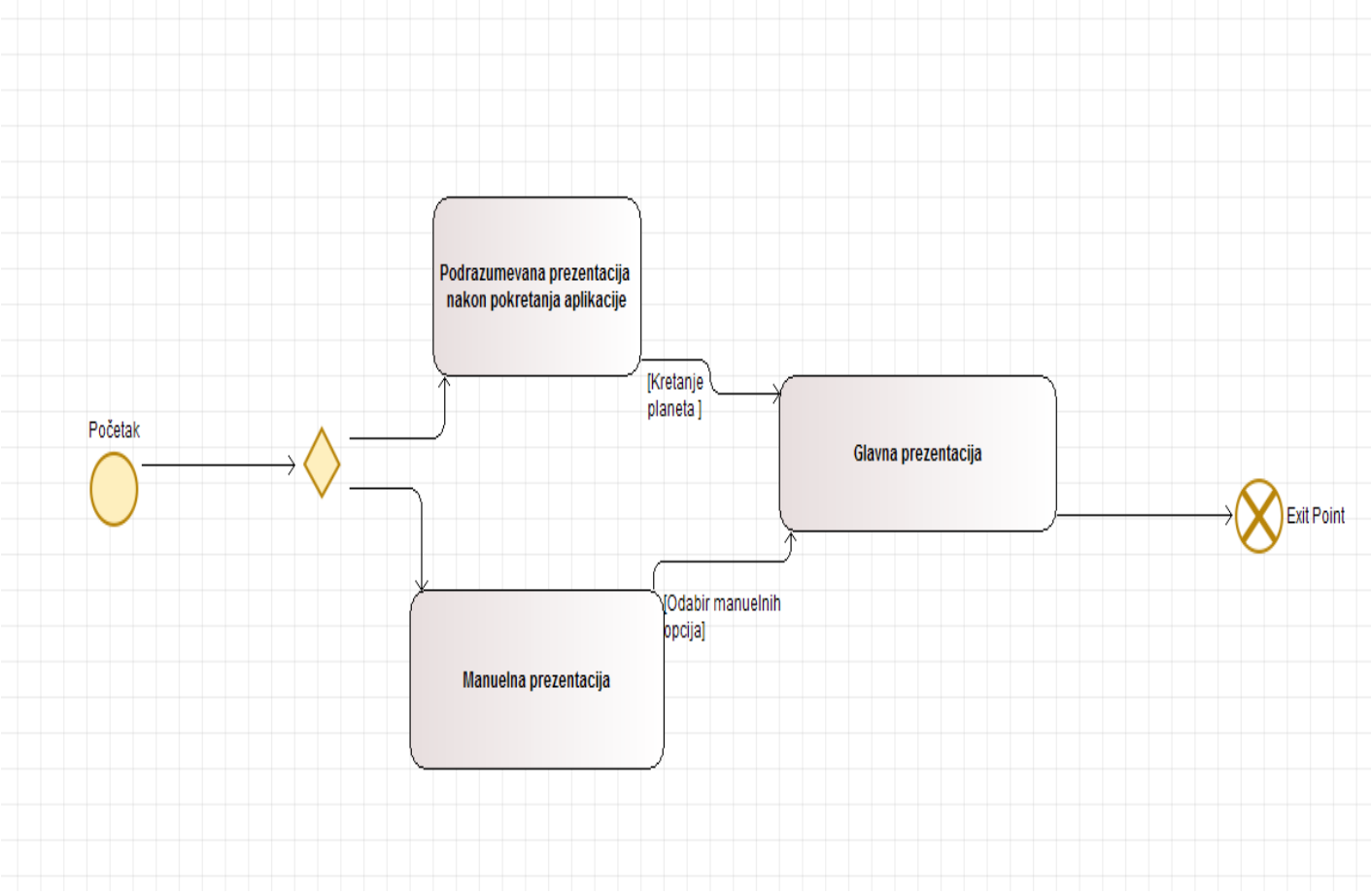
Аутомат стања:

- Понашање које специфицира секвенце стања кроз која пролази.
- Моделира понашање неког ентитета или протокол интеракције.

Дијаграм стања је граф који приказује аутомат стања:

- Чворови су стања.
- Гране су прелази.

Дијаграм стања, за дати пројектни задатак, приказан је на слици 14.



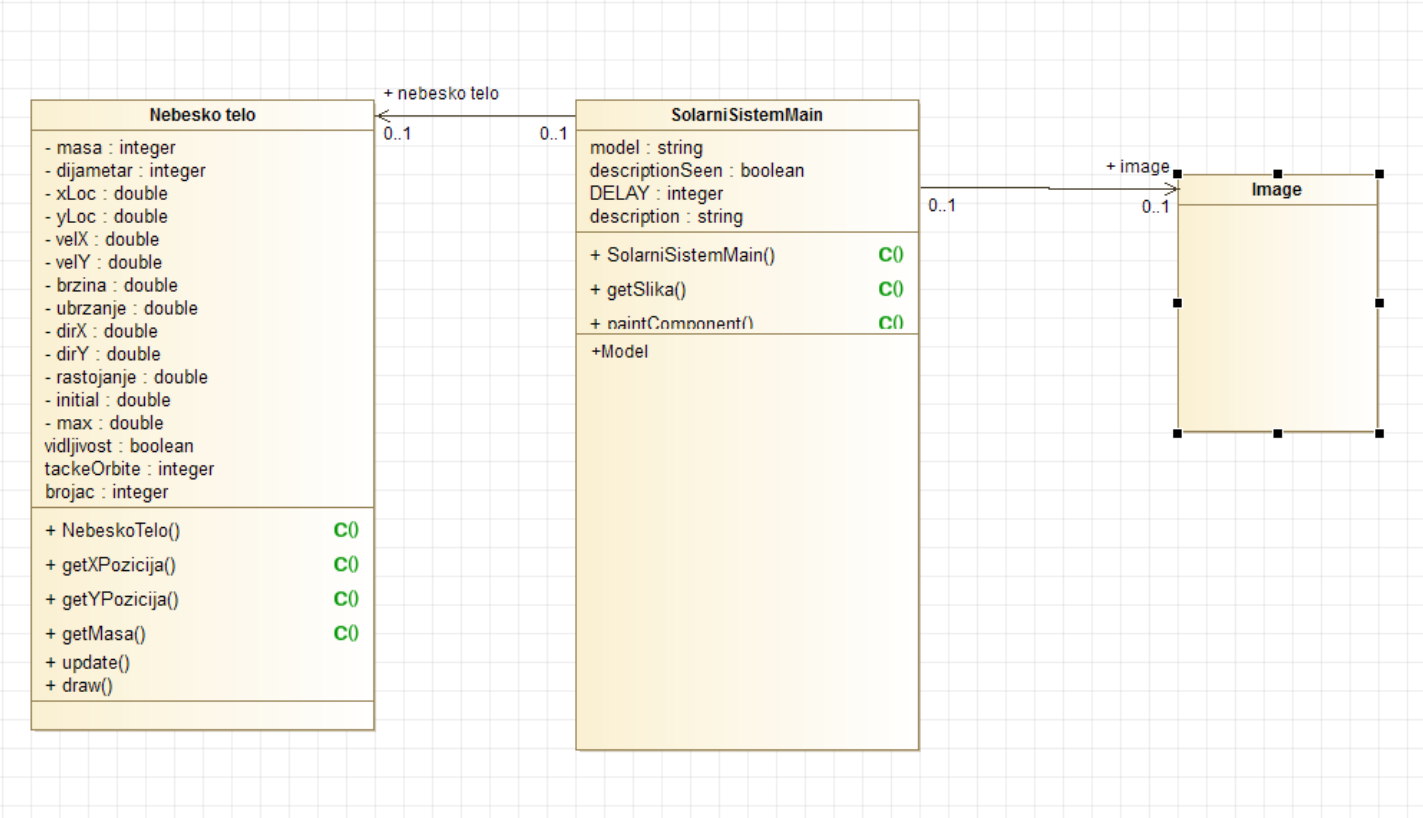
Слика 14. Дијаграм стања

ДИЈАГРАМ КЛАСА

Дијаграм класа показује скуп класа, интерфејса, сарадњи и других ствари структуре, повезаним релацијама.

Елементи дијаграма класа:

- Ствари: класа, интерфејс, типови, изузеци, шаблони, сарадње, пакети
- Релације: зависност, генерализације, асоцијације, реализације



Слика 15. Дијаграм класа

ЛИТЕРАТУРА

I. http://moodle.fink.rs/
II. https://docs.oracle.com/javase/tutorial/uiswing/
III. https://www.modelio.org/documentation-menu/tutorials.html
IV. https://www.eclipse.org/
V. https://code.visualstudio.com/
VI. https://sr.wikipedia.org/sr-ec/%D0%A1%D1%83%D0%BD%D1%87%D0%B5%D0%B2_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC