

**Šolski center Novo mesto**

**Srednja elektro šola in tehniška gimnazija**

**Šegova ulica 112**

**8000 Novo mesto**

**SPLOŠNA MATURA**

# **IZDELAVA IGER - KAČA**

(Maturitetna seminarska naloga)

Predmet: Računalništvo

Avtor: Gašper Gorše, T4C

Mentor: dr. Albert Zorko, univ. dipl. inž. el.

Novo mesto, marec 2022



## Povzetek in ključne besede

Java je objektno usmerjen programski jezik visoke ravni, ki temelji na razredih in je zasnovan tako, da ima čim manj izvedbenih odvisnosti. Gre za splošni programski jezik, ki programerjem omogoča, da napišejo enkrat in zaženejo kjerkoli, kar pomeni, da lahko sestavljena koda v Javi teče na vseh platformah, ki podpirajo Javo, ne da bi jo bilo treba ponovno sestaviti. Poznamo tri temeljne tehnološke pakete v Javi, med katerimi je tudi JDK, ki je okolje za razvijanje Java aplikacij. JRE je pa zbirka komponent za izdelavo in zagon Java aplikacije in je del JDK.

Kača je žanr videoigre, v kateri igralec manevrira z rastočo linijo, ki postane glavna ovira sama sebi. Koncept izvira iz arkadne igre za dva igralca Blockade podjetja Gremlin Industries iz leta 1976, zaradi enostavne izvedbe pa je nastalo na stotine različic (nekateri imajo v naslovu besedo kača ali črv) za številne platforme.

### ***Ključne besede:***

Razred

Metoda

Objekti

Konstruktor

Kača

Igralna plošča

Igralno polje

Java



# Kazala

## Vsebina

1. Uvod .....	1
2. Teoretični del naloge .....	2
2.1 Programski jezik Java .....	2
2.1.1 JDK – Java development kit .....	2
2.1.2 JRE – Java runtime enviroment.....	2
2.1.3 Knjižnice .....	3
2.2 Programsko okolje – Eclipse .....	4
2.3 Kača – igra (Snake game) .....	5
2.3.1 Zgodovina igre.....	6
2.3.2 Koncept igre .....	6
3. Praktični del .....	7
3.1 Razredi .....	7
3.1.1 Razred Kaca_igra.....	7
3.1.2 Razred Igralno_polje .....	7
3.1.3 Razred Igralna_plosca .....	8
4. Zaključek .....	14
5. Zahvala .....	15
6. Viri in literatura .....	16
Bibliografija .....	16
7. Stvarno kazalo .....	17
8. Priloge .....	18

## Kazalo slik

Slika 1: Razvojno okolje Eclipse.....	5
Slika 2: Igra - kača.....	5
Slika 3: Razred Kaca_igra .....	7
Slika 4: Razred Igralno_polje.....	8
Slika 5: Uporabljene knjižnice v razredu Igralna_plosca.....	8
Slika 6: Spremenljivke in konstante igre .....	9
Slika 7: Konstruktor Igralna_plosca.....	9
Slika 8: Metoda Pricetek_igre .....	10
Slika 9: Metoda BarvniElementi.....	10
Slika 10: Metoda Risanje.....	10
Slika 11: Metoda NovoJabolko.....	11
Slika 12: Metoda Premikanje .....	11
Slika 13: Metoda Preveri_za_jabolko .....	11
Slika 14: Metoda Preveri_za_trke.....	12
Slika 15: Metoda actionPerformed.....	12
Slika 16: Metoda paint.....	13
Slika 17: Razred MyKeyAdapter.....	13
Slika 18: Metoda Konec_igre .....	13

## 1. Uvod

Za maturitetno seminarsko nalogo sem si pri predmetu računalništvo izbral temo izdelava iger. V programskem okolju Eclipse in programskem jeziku Java sem izdelal preprosto in zelo znano igro kače. Princip igre je zelo enostaven. Imamo strukturo, ki se premika po določenem polju in nad katero ima uporabnik popoln nadzor. Cilj igre je, da igralcu uspe "pojesti" čim večje število jabolk, ki se naključno pojavljajo na zaslonu.

Programsko kodo sem ustvaril s pomočjo spletnih virov za programiranje ter nekatere objavljene videoposnetke na to temo. V seminarski nalogi bom predstavil programski jezik, programsko okolje Eclipse ter samo kodo igre.

## 2. Teoretični del naloge

Seminarsko nalogo sem pisal v programskem okolju Eclipse in uporabljal sem programski jezik Java.

### 2.1 Programski jezik Java

Java je objektno usmerjen programski jezik visoke ravni, ki temelji na razredih in je zasnovan tako, da ima čim manj izvedbenih odvisnosti. Gre za splošni programski jezik, ki programerjem omogoča, da napišejo enkrat in zaženejo kjerkoli, kar pomeni, da lahko sestavljena koda v Javi teče na vseh platformah, ki podpirajo Javo, ne da bi jo bilo treba ponovno sestaviti[18] Aplikacije Java so običajno sestavljene v bitno kodo, ki se lahko zažene v katerem koli virtualnem stroju Java (JVM) ne glede na arhitekturo računalnika, na katerem je zasnovana. Sintaksa Jave je podobna sintaksi jezikov C in C++, vendar ima manj nizkonivojskih zmogljivosti kot oba. Zagonski čas Jave omogoča dinamične zmožnosti (kot sta refleksija in spreminjanje kode med izvajanjem), ki običajno niso na voljo v tradicionalnih sestavljenih jezikih. Leta 2019 je bila Java po podatkih GitHuba eden najbolj priljubljenih programskih jezikov v uporabi, zlasti za odjemalsko-strežniške spletne aplikacije, saj jo je uporabljalo 9 milijonov razvijalcev. [1] [2]

#### 2.1.1 JDK – Java development kit

Razvojni komplet Java (JDK) je distribucija tehnologije Java podjetja Oracle Corporation. Izvaja specifikacijo jezika Java (JLS) in specifikacijo virtualnega stroja Java (JVMS) ter zagotavlja standardno izdajo (SE) aplikacijskega programskega vmesnika Java (API). Je izpeljanka skupnostnega OpenJDK, ki ga upravlja Oracle. [5] Zagotavlja programsko opremo za delo z aplikacijami Java. Primeri vključene programske opreme so virtualni stroj, prevajalnik, orodja za spremljanje zmogljivosti, razhroščevalnik in drugi pripomočki, za katere Oracle meni, da so koristni za programerja Jave. [3]

#### 2.1.2 JRE – Java runtime environment

Zagonsko okolje Java (JRE), ki ga je izdal Oracle, je prosto dostopna distribucija programske opreme, ki vsebuje samostojni JVM (HotSpot), standardno knjižnico Java (Java Class Library), orodje za konfiguracijo in - do ukinitve v JDK 9 - vtičnik za brskalnik. To je najpogostejše okolje Java, ki je nameščeno v osebnih računalnikih v obliki prenosnih in namiznih računalnikov.



Mobilni telefoni, vključno s funkcijskimi telefoni in zgodnjimi pametnimi telefoni, ki so opremljeni z JVM, najverjetneje vključujejo JVM, namenjen izvajanju aplikacij, ki so namenjene Micro Edition platforme Java. Večina sodobnih pametnih telefonov, tabličnih računalnikov in drugih ročnih računalnikov, ki poganjajo aplikacije Java, jih najverjetneje poganja s podporo operacijskega sistema Android, ki vključuje odprtokodni virtualni stroj, nezdružljiv s specifikacijo JVM. (Namesto tega Googlova razvojna orodja za Android kot vhodne podatke sprejemajo programe Java in izpisujejo Dalvik bajtokodo, ki je izvirni vhodni format za virtualni stroj v napravah Android). [4]

### 2.1.3 Knjižnice

Knjižnica razredov Java (JCL) je niz dinamično naložljivih knjižnic, ki jih lahko jeziki virtualnega stroja Java (JVM) pokličejo ob zagonu. Ker platforma Java ni odvisna od določenega operacijskega sistema, se aplikacije ne morejo zanašati na nobeno od knjižnic, ki so značilne za platformo. Namesto tega platforma Java zagotavlja obsežen nabor standardnih knjižnic razredov, ki vsebujejo funkcije, značilne za sodobne operacijske sisteme. [5]

V seminarski nalogi sem uporabil naslednje knjižnice:

#### ***Package java.awt.\****

Vsebuje vse razrede za ustvarjanje uporabniških vmesnikov za slikanje grafike in slik. Objekt uporabniškega vmesnika, kot je gumb ali drsna vrstica, se v terminologiji AWT imenuje komponenta. Razred Komponenta je koren vseh komponent AWT. Nekatere komponente sprožijo dogodke, ko uporabnik sodeluje s komponentami. Razred AWTEvent in njegovi podrazredi se uporabljajo za predstavitev dogodkov, ki jih lahko sprožijo komponente AWT. [6]

#### ***Package java.awt.event.\****

Zagotavlja vmesnike in razrede za obravnavo različnih vrst dogodkov, ki jih sprožijo komponente AWT. [6]

#### ***Package javax.swing.\****

Zagotavlja nabor "lahkih" komponent (vse v jeziku Java), ki v največji možni meri delujejo enako na vseh platformah. Tipične aplikacije Swing izvajajo obdelavo kot odziv na dogodek, ki ga ustvari gesta uporabnika. [7]

***Package java.util.Random***

Primerek razreda `java.util.Random` se uporablja za generiranje toka psevdonaključnih števil. Razred uporablja 48-bitno seme, ki je spremenjeno z linearno kongruenčno formulo. Algoritmi, ki jih izvaja razred `Random`, uporabljajo zaščiteno uporabno metodo, ki lahko ob vsakem klicu zagotovi do 32 psevdorazporedno generiranih bitov. [8]

***Package javax.swing.JPanel***

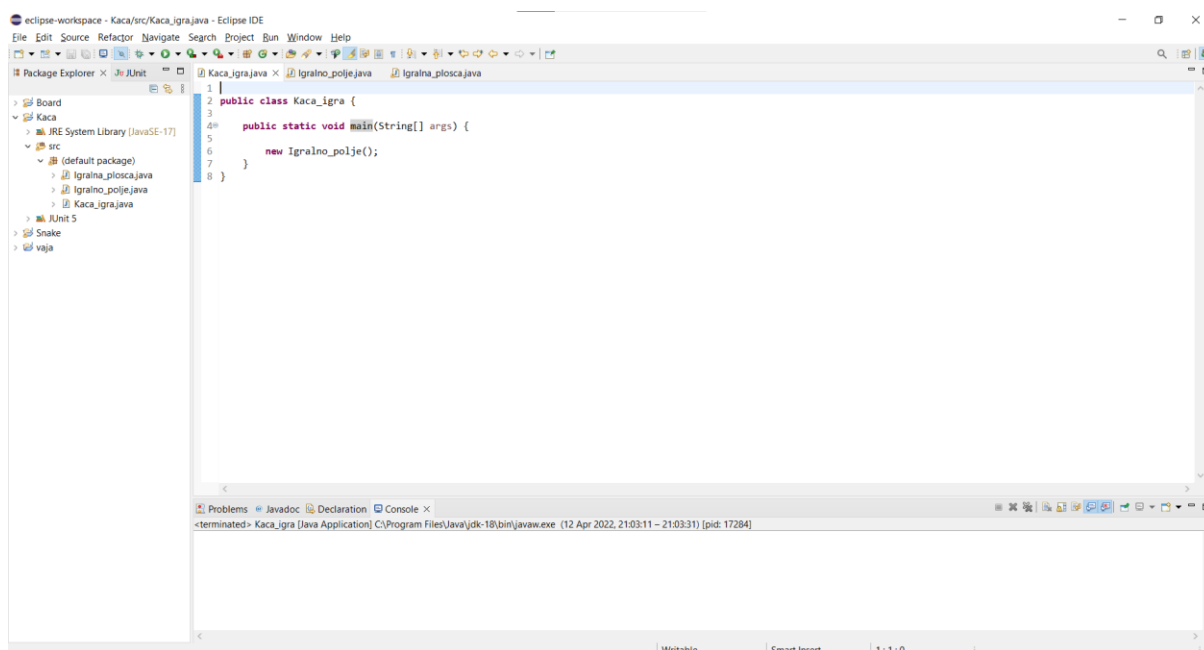
`JPanel`, ki je del paketa `Java Swing`, je vsebnik, v katerem lahko shranite skupino komponent. Glavna naloga panela `JPanel` je organizirati komponente, v njem je mogoče nastaviti različne postavitve, ki omogočajo boljšo organizacijo komponent, vendar nima naslovne vrstice. [9]

***Package javax.swing.JFrame***

Razred `javax.swing.JFrame` je vrsta vsebnika, ki podeduje razred `java.awt.Frame`. Okvir `JFrame` deluje kot glavno okno, v katerem so dodane komponente, kot so oznake, gumbi in besedilna polja, s čimer se ustvari grafični uporabniški vmesnik. [10]

## 2.2 Programsko okolje – Eclipse

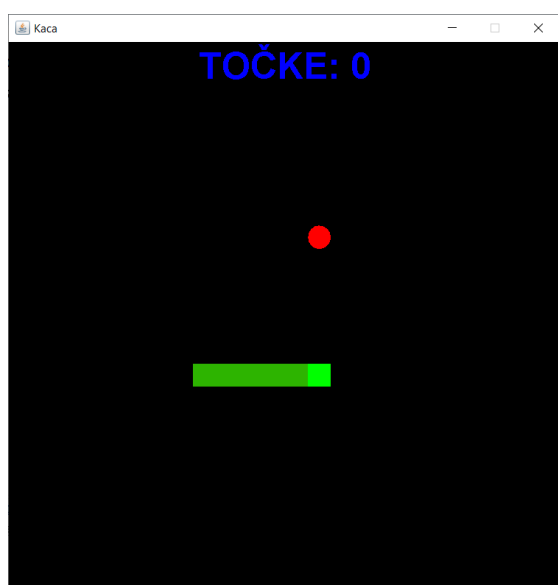
Eclipse je integrirano razvojno okolje (IDE), ki se uporablja pri računalniškem programiranju. Vsebuje osnovni delovni prostor in razširljiv sistem vtičnikov za prilagajanje okolja. Je drugi najbolj priljubljen IDE za razvoj Jave, do leta 2016 pa je bil tudi najbolj priljubljen. Eclipse je večinoma napisan v Javi in se primarno uporablja za razvoj aplikacij v Javi, vendar ga je mogoče z vtičniki uporabljati tudi za razvoj aplikacij v drugih programskih jezikih, vključno z Ada, ABAP, C, C++, C#, Clojure, COBOL, D, Erlang, Fortran, Groovy, Haskell, JavaScript, Julia, Lasso, Lua, NATURAL, Perl, PHP, Prolog, Python, R, Ruby (vključno z ogrođjem Ruby on Rails), Rust, Scala in Scheme. Uporabite ga lahko tudi za pripravo dokumentov s programom LaTeX (z vtičnikom TeXlipse) in paketov za programsko opremo Mathematica. Razvojna okolja med drugim vključujejo razvojna orodja Eclipse Java (JDT) za Javo in Scalo, Eclipse CDT za C/C++ in Eclipse PDT za PHP. [11] [12] [13]



Slika 1: Razvojno okolje Eclipse

## 2.3 Kača – igra (Snake game)

Kača je žanr videoigre, v kateri igralec manevrira z rastočo linijo, ki postane glavna ovira sama sebi. Koncept izvira iz arkadne igre za dva igralca Blockade podjetja Gremlin Industries iz leta 1976, zaradi enostavne izvedbe pa je nastalo na stotine različic (nekatero imajo v naslovu besedo kača ali črv) za številne platforme. Arkadna igra Tron iz leta 1982, ki temelji na filmu, vključuje igranje kač v segmentu Light Cycles za enega igralca. Ko je bila leta 1998 na mobilne telefone Nokia pred naložena njena različica, je zanimanje za igre s kačami ponovno naraslo, saj so našle širše občinstvo. Samo za iOS je na voljo več sto iger s kačami. [14]



Slika 2: Igra - kača

### 2.3.1 Zgodovina igre

Zasnova kače izvira iz arkadne igre Blockade, ki jo je leta 1976 razvil in izdal Gremlin. Istega leta je bila klonirana kot Bigfoot Bonkers. Leta 1977 je podjetje Atari, Inc. izdalo dve igri, navdihnjeni z igro Blockade: arkadno igro Dominos in igro Surround za Atari VCS. Surround je bila ena od devetih iger za Atari VCS v ZDA in jo je prodajal Sears pod imenom Chase. Istega leta je bila podobna igra pod imenom Checkmate predstavljena tudi za igralno ploščo Bally Astrocade. [14]

### 2.3.2 Koncept igre

Igralec nadzira piko, kvadrat ali predmet na omejeni ravnini. Ko se premika naprej, za seboj pušča sled, ki spominja na premikajočo se kačo. V nekaterih primerih je konec poti na fiksnem mestu, zato se kača med premikanjem nenehno daljša. Pri drugem primeru ima kača določeno dolžino, tako da se giblje rep, ki je od glave oddaljen za določeno število enot. Igralec izgubi, če kača naleti na rob zaslona, drugo oviro ali samo sebe. [15]

Igra je bila narejena v dveh različicah:

1. V prvi, ki je najpogostejše igra za dva igralca, je na igralnem polju več kač. Vsak igralec poskuša blokirati drugega igralca, tako da nasprotnik naleti na obstoječo pot in izgubi. Surround za Atari VCS je primer te vrste. Segment Light Cycles arkadne igre Tron je različica za enega igralca, v kateri druge "kače" upravlja umetna inteligenca. [15]
2. V drugi različici poskuša eden igralec pojesti predmete tako, da z glavo kače udari vanje. Vsak zaužiti predmet podaljša kačo, zato je izogibanje trku s kačo vedno težje. Igralec je pri tej varianti sam na polju. [15]

### 3. Praktični del

Kot praktični del naloge sem idejo za igro realiziral z programskim orodjem Eclipse. V grobem je igra sestavljena iz treh razredov, v katerih je v enem razredu napisana glavna in najbolj pomembna koda. V kodi sem večino kode napisal v obliki metod, ki sem jih nato klical ob določenem dogodku v igri. Za pravilno delovanje igre sem moral uporabiti različne knjižnice, ki sem jih opisal v teoretičnem delu.

#### 3.1 Razredi

V kodi imam 3 razrede:

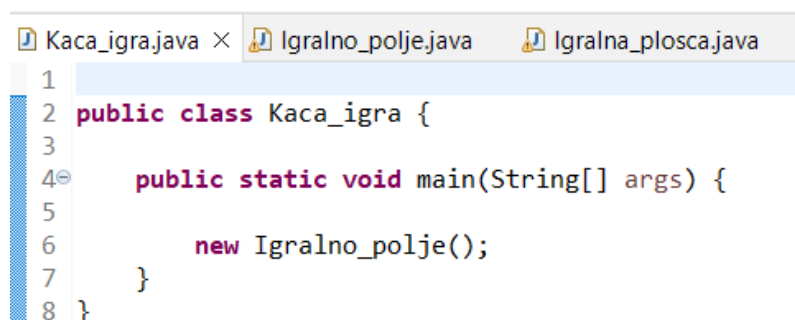
**Kaca\_igra**

**Igralno\_polje**

**Igralna\_plosca**

##### 3.1.1 Razred Kaca\_igra

V razredu Kaca\_igra sem ustvaril objekt razreda Igralno\_polje, ki je pomemben za zagon okna igre.



```

1
2 public class Kaca_igra {
3
4     public static void main(String[] args) {
5
6         new Igralno_polje();
7     }
8 }

```

Slika 3: Razred Kaca\_igra

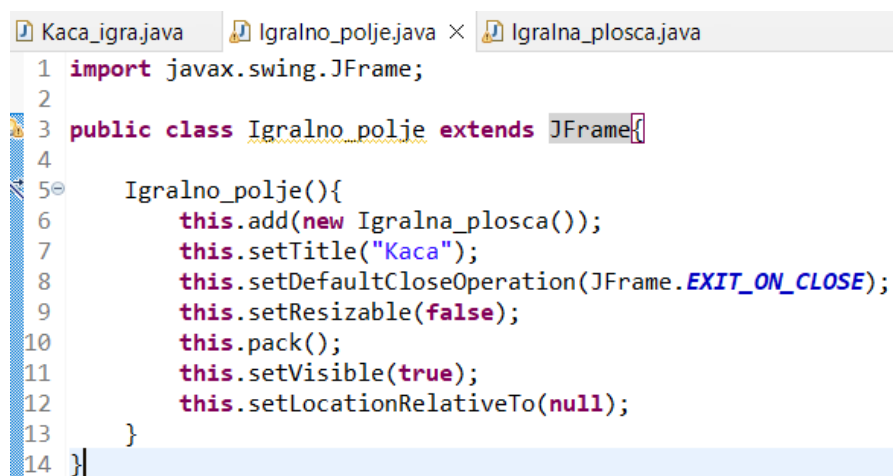
##### 3.1.2 Razred Igralno\_polje

Ta razred je odgovoren za pravilno delovanje GUI – pojavnega okna v katerem je igra.

Na začetku sem uporabil knjižnico javax.swing.JFrame, ki mi omogoča izvajanje ukazov *this*. Nato sem ustvaril konstruktor v katerem sem uporabljal te ukaze.

1. Začel sem z dodajanjem objekta Igralna\_plosca v konstruktor. `this.add(new Igralna_plosca());`
2. Za tem sem dodal naslov igre, ki ga vidimo na pojavnem oknu. `this.setTitle("Kaca");`

3. Nato sem dodal še `this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);`, zato da lahko zapremo aplikacijo kadar želimo.
4. Z ukazom `this.setResizable(false);`, sem določil, da okna ne moremo spreminjati po velikosti.
5. `this.pack();`, omogoči, da je vsebina okna enaka ali nad želeno vrednostjo.
6. Da se nam okno sploh prikaže, sem nato dodal še ukaz `this.setVisible(true);`
7. Končal sem z ukazom `this.setLocationRelativeTo(null);`, ki pojavno okno odpre na sredini monitorja.



```

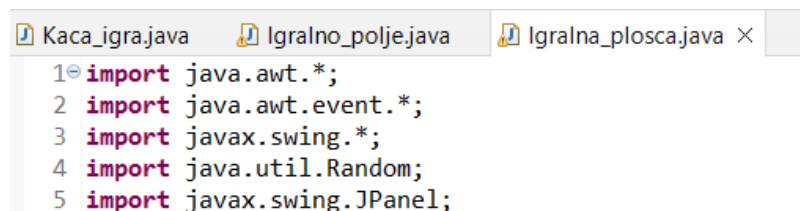
1 import javax.swing.JFrame;
2
3 public class Igralno_polje extends JFrame{
4
5     Igralno_polje(){
6         this.add(new Igralna_plosca());
7         this.setTitle("Kaca");
8         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
9         this.setResizable(false);
10        this.pack();
11        this.setVisible(true);
12        this.setLocationRelativeTo(null);
13    }
14 }

```

Slika 4: Razred Igralno\_polje

### 3.1.3 Razred Igralna\_plosca

Ta razred je najbolj obsežen. Imam uporabljene različne knjižnice, katere mi omogočajo lažje oblikovanje in programiranje igre, ki pa sem jih opisal v teoretičnem delu, zato jih bom tukaj samo naštel.



```

1 import java.awt.*;
2 import java.awt.event.*;
3 import javax.swing.*;
4 import java.util.Random;
5 import javax.swing.JPanel;

```

Slika 5: Uporabljene knjižnice v razredu Igralna\_plosca

1. Začel sem z `public class Igralna_plosca extends JPanel implements ActionListener`, ki deduje razred razreda JPanel in implementira ActionListener.

2. Nato sem v razredu deklariral skoraj vse uporabljene konstante in spremenljivke, ki jih bom uporabil v spodnjih metodah.

```

9      static final int SIRINA_ZASLONA = 600;
10     static final int VISINA_ZASLONA = 600;
11     static final int VELIKOST_ELEMENTOV = 25;
12     static final int STEVILO_ELEMENTOV = (SIRINA_ZASLONA*VISINA_ZASLONA)/VELIKOST_ELEMENTOV;
13     static final int HITROST = 85;
14     final int x[] = new int[STEVILO_ELEMENTOV]; //x koordinata
15     final int y[] = new int[STEVILO_ELEMENTOV]; //y koordinata
16     int deliTelesa = 6; //koliko je kaca dolga, zacetna dolzina
17     int pojedenaJabolka;
18     int xJabolko; //x koordinata jabolka
19     int yJabolko; //y koordinata jabolka
20     char smer = 'R'; //zacetna smer, R-desno, L-levo, U-gor, D-dol
21     boolean izvajanje = false; //ali je igra v teku ali ne
22     Timer timer;
23     Random random;
24

```

Slika 6: Spremenljivke in konstante igre

3. Sledi konstruktor razreda Igralna\_plosca, kateri bo odgovoren za igralno ploščo, na kateri se bo igra odvijala. Pričnemo z novim razredom Random, ki ga bomo uporabili kasneje. Nato določimo širino ter višino našega zaslona, ki se ne bo spreminjala. Določimo barvo podlage, za katero sem izbral črno. Ukaz `this.setFocusable(true);`, bo storil, da program lahko bere stisnjene tipke na tipkovnici in ukaz `this.addKeyListener(new MyKeyAdapter());`, da se program odzove na stisnejo tipko. Nato kličemo metodo za začetek igre `Pricetek_igre();`.

```

24
25     Igralna_plosca(){
26         random = new Random();
27         this.setPreferredSize(new Dimension(SIRINA_ZASLONA,VISINA_ZASLONA));
28         this.setBackground(Color.black);
29         this.setFocusable(true);
30         this.addKeyListener(new MyKeyAdapter());
31         Pricetek_igre();
32     }

```

Slika 7: Konstruktor Igralna\_plosca

4. S to metodo na začetku, kličemo metodo `NovoJabolko();`, ki generira jabolko na igralno ploščo. Naš parameter `izvajanje = true;`, postavimo na true, da imamo pogoj, da se igra izvaja. Nato se izvede ukaz `timer = new Timer(HITROST,this);`, ki ima parameter `HITROST`, ki določa kako hitro se igra odvija. Nato samo pokličemo `timer.start();`, da se timer začne.

```

33 public void Pricetek_igre(){
34     NovoJabolko();
35     izvajanje = true;
36     timer = new Timer(HITROST,this);
37     timer.start();
38 }

```

Slika 8: Metoda Pricetek\_igre

5. Ta metoda nam omogoča uporabo ukazov Graphics g, kateri nam bodo v pomoč, ko bomo ustvarili elemente igre.

```

39 public void BarvniElementi(Graphics g){
40     super.paintComponent(g);
41     Risanje(g);
42 }

```

Slika 9: Metoda BarvniElementi

6. Z metodo Risanje ustvarjamo elemente naše igre. 45 in 46 vrstica ustvarita jabolko, ter določita barvo jabolka na rdečo. Od 48 do 52 vrstice imamo for zanko, ki je namenjena delom telesa, iz katerih je kača sestavljena. Poskrbi tudi za to, da kača pridobi 1 del telesa, če poje jabolko. Celoten postopek risanja je v if zanki, katera se izvaja, v primeru, da je `izvajanje = true`; . Če ta pogoj ni izpolnjen, se igra konča, kliče se metoda `Konec_igre(g)`; . Vmesna if zanka, pa riše dele telesa kače na prave koordinate v mreži.

```

43 public void Risanje(Graphics g){
44     if(izvajanje) {
45         g.setColor(Color.red);
46         g.fillOval(xJabolko, yJabolko, VELIKOST_ELEMENTOV, VELIKOST_ELEMENTOV);
47
48         for(int i=0; i<deliTelesa; i++) {
49             if(i==0) {
50                 g.setColor(Color.green);
51                 g.fillRect(x[i], y[i], VELIKOST_ELEMENTOV, VELIKOST_ELEMENTOV);
52             }
53             else {
54                 g.setColor(new Color(45, 180, 0));
55                 //g.setColor(new Color(random.nextInt(255), random.nextInt(255), random.nextInt(255))); //spreminjanje barve
56                 g.fillRect(x[i], y[i], VELIKOST_ELEMENTOV, VELIKOST_ELEMENTOV);
57             }
58         }
59         g.setColor(Color.blue);
60         g.setFont(new Font("SansSerif", Font.BOLD, 40));
61         FontMetrics metrics = getFontMetrics(g.getFont());
62         g.drawString("TOČKE: " + pojedenaJabolka, (SIRINA_ZASLONA - metrics.stringWidth("TOČKE: " + pojedenaJabolka)) / 2, g.getFont().getSize());
63     }
64     else {
65         Konec_igre(g);
66     }
67 }

```

Slika 10: Metoda Risanje



7. Metoda NovoJabolko na naključnih koordinatah ustvari novo jabolko.

```

68 public void NovoJabolko(){
69     xJabolko = random.nextInt((int)(SIRINA_ZASLONA/VELIKOST_ELEMENTOV))*VELIKOST_ELEMENTOV;
70     yJabolko = random.nextInt((int)(VISINA_ZASLONA/VELIKOST_ELEMENTOV))*VELIKOST_ELEMENTOV;
71
72 }

```

Slika 11: Metoda NovoJabolko

8. Na začetku metode s pomočjo for zanke določamo, dele telesa naše kače. Nato uporabimo switch case kateri bo spreminjal smer naše kače.

```

73 public void Premikanje(){
74     for(int i = deliTelesa; i>0; i--) {
75         x[i] = x[i-1];
76         y[i] = y[i-1];
77     }
78     switch(smer) {
79         case 'U':
80             y[0] = y[0] - VELIKOST_ELEMENTOV;
81             break;
82         case 'D':
83             y[0] = y[0] + VELIKOST_ELEMENTOV;
84             break;
85         case 'L':
86             x[0] = x[0] - VELIKOST_ELEMENTOV;
87             break;
88         case 'R':
89             x[0] = x[0] + VELIKOST_ELEMENTOV;
90             break;
91     }
92
93 }

```

Slika 12: Metoda Premikanje

9. Z if stavkom preverimo koordinate jabolka in če pogoj drži, se kača poveča za 1 del. 97 je namenjena spremenljivki, ki šteje koliko jabolk je kača pojedla. Z njim v eni izmed naslednjih metod izpisujemo rezultat. Na koncu kličemo metodo `NovoJabolko()`, da se za pojedenim jabolkom ustvari novo.

```

94 public void Preveri_za_jabolko(){
95     if((x[0] == xJabolko) && (y[0] == yJabolko)) {
96         deliTelesa++;
97         pojedenaJabolka++;
98         NovoJabolko();
99     }
100 }

```

Slika 13: Metoda Preveri\_za\_jabolko

10. Prva for zanka preverja ali so koordinate glave in telesa identične, če so se spremenljivki **izvajanje** pripiše **false**. To pomeni, da se igra prekine. Nato z ostalimi if stavki preverjamo koordinate in ali je prišlo do trka med kačo in mejami igralnega polja.

```

101 public void Preveri_za_trke(){
102
103     //preverimo ce se glava trci z telesom
104     for(int i=deliTelesa; i>0; i--) {
105         if( (x[0] == x[i] && y[0] == y[i]) ) {
106             izvajanje = false;
107         }
108     }
109     //L meja - trk
110     if(x[0] < 0) {
111         izvajanje = false;
112     }
113     //D meja - trk
114     if(x[0] > SIRINA_ZASLONA) {
115         izvajanje = false;
116     }
117     //U meja - trk
118     if(y[0] < 0) {
119         izvajanje = false;
120     }
121     //D meja - trk
122     if(y[0] > VISINA_ZASLONA) {
123         izvajanje = false;
124     }
125
126     if(!izvajanje) {
127         timer.stop();
128     }
129
130 }

```

Slika 14: Metoda Preveri\_za\_trke

11. S to metodo pokličemo 3 metode, s katerimi se naša kača začne premikati po polju. Z **repaint()**; pa to narišemo.

```

143 @Override
144 public void actionPerformed(ActionEvent e) {
145     if (izvajanje) {
146         Premikanje();
147         Preveri_za_jabolko();
148         Preveri_za_trke();
149     }
150     repaint();
151 }
152

```

Slika 15: Metoda actionPerformed

12. Metoda kliče BarvneElemente ter jih nariše.

```
153 @Override
154 public void paint(Graphics g) {
155     BarvniElementi(g);
156 }
```

Slika 16: Metoda paint

13. S tem razredom smo uporabili razširitev KeyAdapter katera nam je omogočila uporabljanje tip na tipkovnici. S pomočjo switch case sem ustvaril premikanje s pomočjo smernih puščic in onemogočil premikanje za več kot 90 ° (kača se ne more obrniti sama vase).

```
158 public class MyKeyAdapter extends KeyAdapter{
159     @Override
160     public void keyPressed(KeyEvent e){
161         switch(e.getKeyCode()) {
162             case KeyEvent.VK_LEFT:
163                 if (smer != 'R') {
164                     smer = 'L';
165                 }
166                 break;
167             case KeyEvent.VK_RIGHT:
168                 if (smer != 'L') {
169                     smer = 'R';
170                 }
171                 break;
172             case KeyEvent.VK_UP:
173                 if (smer != 'D') {
174                     smer = 'U';
175                 }
176                 break;
177             case KeyEvent.VK_DOWN:
178                 if (smer != 'U') {
179                     smer = 'D';
180                 }
181                 break;
182         }
183     }
184 }
185 }
```

Slika 17: Razred MyKeyAdapter

13. S to metodo izpišemo število doseženih točk ter napis "KONEC IGRE". S temi vrsticami kode ter posebnimi ukazi, sem lahko določil kje na ekranu se bo rezultat izpisal.

```
131 public void Konec_igre(Graphics g){
132     //Izpis dosezenih točk
133     g.setColor(Color.blue);
134     g.setFont(new Font("SansSerif", Font.BOLD, 40));
135     FontMetrics metrics1 = getFontMetrics(g.getFont());
136     g.drawString("TOČKE: " + pojedenaJabolka, (SIRINA_ZASLONA - metrics1.stringWidth("TOČKE: " + pojedenaJabolka)) / 2, g.getFont().getSize());
137     //Izpis KONEC IGRE
138     g.setColor(Color.blue);
139     g.setFont(new Font("SansSerif", Font.BOLD, 75));
140     FontMetrics metrics2 = getFontMetrics(g.getFont());
141     g.drawString("KONEC IGRE", (SIRINA_ZASLONA - metrics2.stringWidth("KONEC IGRE")) / 2, VISINA_ZASLONA / 2);
142 }
```

Slika 18: Metoda Konec\_igre

## 4. Zaključek

Za izdelavo programa igre Kača sem na začetku najprej naredil načrt kako se lotiti naloge. Sprva sem imel veliko različnih razredov, vendar se mi je ideja zdela preveč zapletena in sem razrede tako spremenil v metode v enem razredu, ki jih skozi program kličem in uporabljam. Tako sem ostal z razredom, ki ima samo 1 konstruktor, razred za uvedbo in nadzor GUI igre ter glavni razred z vsemi metodami.

Težave, ki so se mi pojavljale pri pisanju programa, so bile samo na začetku zaradi stare programske opreme. Samo kodiranje mi ni delalo precejšnjih težav, saj sem delal postopoma in si ob kodiranju sproti zaganjal program ter preverjal ali koda deluje.

Cilj maturitetne seminarske naloge sem dosegel, saj sem sprogramiral delujočo igro v programskem jeziku Java. Igro bi lahko izboljšal z grafičnimi dodatki in začetnim menijem. Lahko bi dodal stopnje težavnosti, ker bi se kača premikala hitreje kot se sicer.

## 5. Zahvala

V zahvali bi se na začetku rad zahvalil profesorjem za računalništvo. In sicer profesorju dr. Albertu Zorku in univ. dipl. inž. Gregorju Medetu, profesorju za laboratorijske vaje univ. dipl. inž. Tomažu Ferbežarju in vsem ostalim profesorjem, ki so me poučevali predmet računalništva. Profesorji so mi bili v veliko pomoč, ne samo pri izdelavi maturitetne seminarske naloge, vendar tudi pri ostalih stvareh.

## 6. Viri in literatura

### Bibliografija

- [1] Arcade-history.com, „Blockade video game, Gremlin Ind, inc. (1976),“ *Blockade video game, Gremlin Ind, inc. (1976)*, p. 2011, 11 junij 2011.
- [2] T. E. Foundation, „Eclipse desktop & web IDEs,“ The Eclipse Foundation, 2022.
- [3] GeeksforGeeks, „Java Swing – JPanel With Examples,“ GeeksforGeeks, 2021.
- [4] javatpoint, „Java JFrame,“ javatpoint, 2021.
- [5] JuliaDT, „GitHub - JuliaComputing/JuliaDT: Julia Development Toolkit for Eclipse,“ github.com, 2018.
- [6] S. Microsystems, „JavaHotSpot Virtual Machine,“ Oktober 2018.
- [7] OpenJDK, „OpenJDK FAQ,“ 2021.
- [8] Oracle, „Design Goals of the Java™ Programming Language,“ Oracle, 1999.
- [9] Oracle, „How Classes are Found,“ Oracle Corporation, 2015.
- [10] Oracle, „Package java.awt,“ Oracle, 2015.
- [11] Oracle, „Package javax.swing,“ Oracle, 2015.
- [12] M.-H. Professional, „High score!: the illustrated history of electronic games,“ *High score!: the illustrated history of electronic games*, 7 April 2011.
- [13] SNYK, „IntelliJ IDEA dominates the IDE market with 62% adoption among JVM developers,“ 30 Januar 2017.
- [14] Tutorialspoint, „Java.util.Random Class,“ *Introduction*, December 2021.
- [15] C. Weekly, „Write once, run anywhere?,“ 5 Maj 2002.
- [16] Wikipedija, „[https://sl.wikipedia.org/wiki/Programski\\_jezik\\_java](https://sl.wikipedia.org/wiki/Programski_jezik_java),“ Wikipedija, 2021.

## 7. Stvarno kazalo

iger, 1  
Igralna plošča, 9  
Igralna\_plosca, 9  
igralno ploščo, 6  
Igralno polje, 9  
igre, 1  
Java, 9, 1, 2, 3, 14  
Jave, 4  
Javi, 9, 2  
kača, 11, 12, 14  
Kača, 9, 5, 14  
kače, 11  
kačo, 6, 12  
knjižnic, 3  
knjižnice, 7  
knjižnico, 7  
konstruktor, 7, 14  
Konstruktor, 9  
metoda, 10  
Metoda, 9, 11, 13  
metodami, 14  
metode, 11, 12  
metodo, 9, 10, 11, 13  
Objekti, 9  
objektno, 9  
razred, 14  
Razred, 9, 4  
razreda, 4  
razrede, 7, 14  
razredih, 9  
razredom, 13, 14  
razredov, 3, 7, 14  
razredu, 7, 14

## 8. Priloge

Priloga 1 – celotna koda programa:

<https://github.com/GasperGorse/Maturitetna-seminarska-naloga>