Šolski center Novo mesto

Srednja elektro šola in tehniška gimnazija

Šegova ulica 112

8000 Novo mesto

Maturitetna seminarska naloga pri računalništvu

**IZDELAVA IGER - MINOLOVEC**

Avtor: Matevž Sladič, T4C

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

Šolsko leto: 2020/2021

Novo mesto, april 2021

**Povzetek in ključne besede**

S seminarsko nalogo želim predstaviti svojo pot do končnega izdelka, igre Minolovec. V nalogi sta predstavljena moj teoretični kot tudi praktični pristop k reševanju naloge in doseganju cilja izdelave funkcionalne računalniške igre Minolovec. Predstavljena sta tako načrtovanje in dekompozicija problema, kot tudi njegove rešitve v obliki kode v programskem jeziku Java in končne aplikacije. Napisano kodo sem po delih tudi predstavil in besedilu dodal še izseke iz izvorne kode, na podlagi katerih sem predstavil svoje razmišljanje in pristop k reševanju naloge. Največji del kode zagotovo zavzema grafični vmesnik, saj je ta najpomembnejši element, ker predstavlja povezavo med uporabnikom - igralcem na eni in kodo na drugi strani.

Ključne besede: Java, grafični vmesnik, koda, spremenljivka, tabela, metoda, razred, Minolovec, igra

**Abstract and keywords**

In this paper I want to present my journey to the final product, the Minesweeper game. I present my theoretical and practical approach to solving the problem and reaching the goal of making a functional computer game Minesweeper. Both planning and functional decomposition are presented in the paper, as well as the solution in the shape of a program code, written in the programming language Java, and the final application. I presented the final code in parts and added some sections of the source code, which helped me to explain my way of thinking and my approach to solving the problem. The graphical user interface (GUI) presents the biggest part of the final code, since it is the most important part of any application. It is the connection between the user – player on one and the source code on the other side.

Keywords: Java, graphical user interface, code, variable, table, method, class, Minesweeper, game

KAZALO

[1 Uvod 1](#_Toc69277681)

[1.1 Osnovni pojmi 1](#_Toc69277682)

[2 Igra Minolovec 3](#_Toc69277683)

[2.1 Načrtovanje naloge 3](#_Toc69277684)

[2.2 Programska rešitev naloge 6](#_Toc69277685)

[2.2.1 Glavni razred Main 6](#_Toc69277686)

[2.2.2 Razred Gui in njegovi podrazredi ter podprogrami 8](#_Toc69277687)

[2.2.3 Razred Gui2 in njegovi podrazredi ter podprogrami 13](#_Toc69277688)

[2.2.4 Končni izgled igre 14](#_Toc69277689)

[PRILOGA 4: 15](#_Toc69277690)

[3 ZAKLJUČEK 16](#_Toc69277691)

[4 ZAHVALA 17](#_Toc69277692)

[5 LITERATURA 18](#_Toc69277693)

[6 PRILOGE 20](#_Toc69277694)

[PRILOGA 1: Koda glavnega razreda aplikacije z neskončno zanko 20](#_Toc69277695)

[PRILOGA 2: Koda razreda glavnega grafičnega vmesnika in s funkcijami igre 20](#_Toc69277696)

[PRILOGA 3: Koda razreda grafičnega vmesnika z navodili 20](#_Toc69277697)

[PRILOGA 4: Aplikacija igre Minolovec 20](#_Toc69277698)

[PRILOGA 5: Seminarska naloga v elektronski obliki in predstavitev za zagovor seminarske naloge 20](#_Toc69277699)

**KAZALO SLIK**

[Slika 1: Dekompozicija načrtovanja naloge 3](file:///C:\Users\matev\OneDrive%20-%20Šolski%20center%20Novo%20mesto\Desktop\Maturitetna\Izdelava%20iger%20-%20Minolovec_Sladič_Matevž.docx#_Toc69277700)

[Slika 2: Skica končnega izgleda grafičnega vmesnika 5](file:///C:\Users\matev\OneDrive%20-%20Šolski%20center%20Novo%20mesto\Desktop\Maturitetna\Izdelava%20iger%20-%20Minolovec_Sladič_Matevž.docx#_Toc69277701)

[Slika 3: Uvažanje programskih knjižnic 6](file:///C:\Users\matev\OneDrive%20-%20Šolski%20center%20Novo%20mesto\Desktop\Maturitetna\Izdelava%20iger%20-%20Minolovec_Sladič_Matevž.docx#_Toc69277702)

[Slika 4: Razred Main in definicija niti 6](file:///C:\Users\matev\OneDrive%20-%20Šolski%20center%20Novo%20mesto\Desktop\Maturitetna\Izdelava%20iger%20-%20Minolovec_Sladič_Matevž.docx#_Toc69277703)

[Slika 5: Metoda run() 7](file:///C:\Users\matev\OneDrive%20-%20Šolski%20center%20Novo%20mesto\Desktop\Maturitetna\Izdelava%20iger%20-%20Minolovec_Sladič_Matevž.docx#_Toc69277704)

[Slika 6: Glavne spremenljivke 8](file:///C:\Users\matev\OneDrive%20-%20Šolski%20center%20Novo%20mesto\Desktop\Maturitetna\Izdelava%20iger%20-%20Minolovec_Sladič_Matevž.docx#_Toc69277705)

[Slika 7: Glavne tabele nujne za delovanje programa 9](file:///C:\Users\matev\OneDrive%20-%20Šolski%20center%20Novo%20mesto\Desktop\Maturitetna\Izdelava%20iger%20-%20Minolovec_Sladič_Matevž.docx#_Toc69277706)

[Slika 8: Konstruktor razreda Gui 9](file:///C:\Users\matev\OneDrive%20-%20Šolski%20center%20Novo%20mesto\Desktop\Maturitetna\Izdelava%20iger%20-%20Minolovec_Sladič_Matevž.docx#_Toc69277707)

[Slika 9: Del metode paintComponent 10](file:///C:\Users\matev\OneDrive%20-%20Šolski%20center%20Novo%20mesto\Desktop\Maturitetna\Izdelava%20iger%20-%20Minolovec_Sladič_Matevž.docx#_Toc69277708)

[Slika 10: Metodi za računanje koordinat polja v mreži 11](file:///C:\Users\matev\OneDrive%20-%20Šolski%20center%20Novo%20mesto\Desktop\Maturitetna\Izdelava%20iger%20-%20Minolovec_Sladič_Matevž.docx#_Toc69277709)

[Slika 11: Del rekurzivne funkcije find\_empty\_cells 12](file:///C:\Users\matev\OneDrive%20-%20Šolski%20center%20Novo%20mesto\Desktop\Maturitetna\Izdelava%20iger%20-%20Minolovec_Sladič_Matevž.docx#_Toc69277710)

[Slika 12: Spremenljivke razreda Gui2 13](file:///C:\Users\matev\OneDrive%20-%20Šolski%20center%20Novo%20mesto\Desktop\Maturitetna\Izdelava%20iger%20-%20Minolovec_Sladič_Matevž.docx#_Toc69277711)

[Slika 13: Del konstruktorja Gui2() 13](file:///C:\Users\matev\OneDrive%20-%20Šolski%20center%20Novo%20mesto\Desktop\Maturitetna\Izdelava%20iger%20-%20Minolovec_Sladič_Matevž.docx#_Toc69277712)

[Slika 14: Metoda, ki določa funkcijo gumbu myButton 14](file:///C:\Users\matev\OneDrive%20-%20Šolski%20center%20Novo%20mesto\Desktop\Maturitetna\Izdelava%20iger%20-%20Minolovec_Sladič_Matevž.docx#_Toc69277713)

[Slika 15: Igralno okno aplikacije 14](file:///C:\Users\matev\OneDrive%20-%20Šolski%20center%20Novo%20mesto\Desktop\Maturitetna\Izdelava%20iger%20-%20Minolovec_Sladič_Matevž.docx#_Toc69277714)

[Slika 16: Okno z informacijami in navodili 15](file:///C:\Users\matev\OneDrive%20-%20Šolski%20center%20Novo%20mesto\Desktop\Maturitetna\Izdelava%20iger%20-%20Minolovec_Sladič_Matevž.docx#_Toc69277715)

# Uvod

Igra Minolovec je prvič izšla leta 1990, od takrat pa lahko zasledimo mnogo izdaj in različnih pristopov k izdelavi igre. Govori se, da ima vsak programer svoj stil programiranja, ki mu najbolj ustreza in po katerem se zgleduje. V seminarski nalogi bom predstavil svoj stil programiranja in svoj postopek razmišljanja ter raziskovanja literature za doseganje glavnega cilja, to je izdelava svoje verzija funkcionalne igre Minolovec.

## Osnovni pojmi

**Java** je objektno orientiran programski jezik, ki nam omogoča razvoj programov in grafičnih vmesnikov. Glavni principi Jave so, da je jezik varen, da je prenosen ter da je izvajanje programov hitro. Za uporabo tega programskega jezika je nujna predhodna inštalacija javinega razvojnega orodja (angl. JDK), ki je javno dostopen in brezplačen na spletni povezavi *https://www.java.com/en/* (1) (2). Je eden izmed najbolj priljubljenih programskih jezikov na svetu, prav posebej pa je Java priljubljena v podjetniškem programiranju. Osnovne funkcije programskega jezika so razširljive z uvažanjem knjižnic s funkcijami v samo kodo (3).

**Eclipse** je najbolj priljubljeno javno dostopno razvojno okolje, v katerem lahko z uporabo programskega jezika Java razvijamo od enostavnejših, do zapletenejših in zahtevnejših programov in aplikacij (4) (5).

**Igra Minolovec** je nastala kot originalno delo podjetja Microsoft iz leta 1990, napisala pa sta jo Robert Donner in Curt Johnson. Skozi leta se njena oblika in način delovanja nista pretirano spreminjala, je pa igra dobila številne izpeljanke za različne operacijske sisteme in platforme (6).

**Programske knjižnice** so zbirke razredov in funkcij, ki jih je napisal nekdo pred nami in jih naložimo skupaj z javinim razvojnim orodjem (7). Njihova izvorna koda je zapisana v isti mapi kamor smo shranili JDK in so zapisane s podaljškom .jar (8).

**Metoda** ali funkcija je podprogram, ki vsebuje del kode, ki se izvede zgolj, ko jo »pokličemo«. To pomeni, da moramo v glavni kodi programa specifično klicati ime metode z vsemi parametri, da se ta izvede, npr. resetAll(). Metode so lahko različnih podatkovnih tipov kot so int, String, boolean,…, ki morajo vrniti podatek tega podatkovnega tipa ali pa tipa void, ki ne rabi vrniti ničesar (9).

**Razred** je načrt, ki določa skupne značilnosti sorodnih predmetov (9). »V predmetnih programskih jezikih v resnici programiramo razrede. Z njimi določimo skupne značilnosti večjega števila predmetov. Ko je načrt dokončan, lahko po njem izdelamo poljubno število izvodov razreda (angl. *class instances*) oz. predmetov.« (10)

# Igra Minolovec

## Načrtovanje naloge

**Prvotna ideja** za izdelavo igre Minolovec kot maturitetno nalogo je nastala iz zanimanja za delovanje in izdelovanje iger v Javi. Zavedal sem se, da je cilj dosegljiv, primeren za pridobivanje novih znanj s področja programiranja, hkrati pa odpira tudi možnosti za izboljševanje in dopolnjevanje v prihodnosti. Načrtovanja sem se lotil z dekompozicijo ideje, ki jo grafično prikazuje slika 1.

Sprva sem glavni cilj razdelil na manjša podproblema, ugotavljanja in programiranja pravil delovanja igre in programiranje grafičnega vmesnika, ki je bilo zame še nedotaknjeno področje. Pri načrtovanju sem si pomagal s programom Trello, ki je brezplačna grafična platforma za načrtovanje projektov (11).

Slika 1: Dekompozicija načrtovanja naloge

**Pravila delovanja igre** sem dodatno razdelil na generiranje naključnega minskega polja določene velikosti in na pogoj za konstantno preverjanje stanja igre, torej ali je igralec zmagal ali izgubil. Pri Minolovcu je potrebno preverjati ali je igralec:

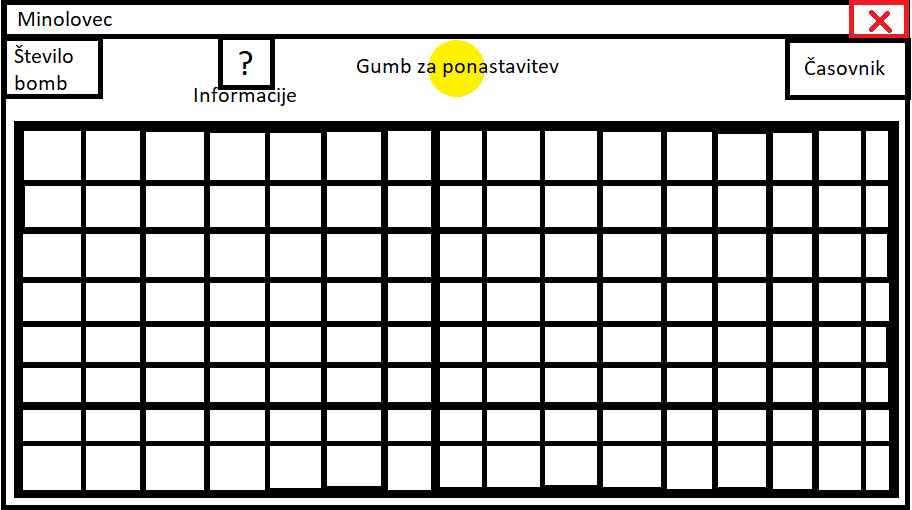
* kliknil na polje pod katerim se nahaja bomba in ga s tem odpre,
* vsa polja pod katerimi se skrivajo bombe označil z zastavico,
* kliknil na gumb za predčasno ponastavitev igre.

V primeru, da igralec s klikom odpre polje pod katerim se nahaja bomba, igralec izgubi igro in čas igranja neha teči, prav tako pa se odprejo vsa polja, ki niso označena z zastavico. V primeru, da igralec vsa polja z bombami označi z zastavico in so vsa ostala polja odprta, igralec zmaga igro in se ustavi štetje igralnega časa. V primeru, da igralec klikne na gumb za predčasno ponastavitev igre, pa se igra v celoti ponastavi.

Ob generiranju naključnega minskega polja moramo biti pozorni, da polje ob vsaki novi igri predstavlja približno enako težavnost, torej, da polje vedno vsebuje podobno število bomb in da se generira naključno. Potrebno je biti pozoren tudi na velikost igralne mreže. V svojem primeru sem izbral mrežo s 16 stolpci in 9 vrsticami.

**Grafični vmesnik** je pomemben sestavni del končnega izdelka, saj ta predstavlja povezavo med kodo in končnim uporabnikom, v mojem primeru igralcem. Načrtovanja vmesnika sem se lotil sprva s študijem strokovne literature na to temo, kasneje pa s praktičnim preizkušanjem in učenjem na primerih. Izdelovanje grafičnega vmesnika sem razdelil na vmesnik z navodili in glavni vmesnik, v katerem bi se igrala igra. V igralnem oknu je bilo potrebno načrtovati polje za orodno vrstico ter polje z mrežo minskega polja. Ob načrtovanju izdelovanja naloge sem se prav tako lotil raziskovanja in študiranja strokovne literature ter forumov, v katerih sem iskal nasvete, kako se lotiti naloge in na kaj je potrebno biti pozoren pri izdelovanju te igre.

Ko sem izdelal načrt za glavni del naloge, pa je bilo potrebno narediti tudi načrt, kako izgled igre prirediti po svojem okusu, da se bo malce razlikovala od ostalih različic in jim ne bo preveč podobna. V spletnem brskalniku sem poiskal originalno izdajo igre, ki je bila še del Microsoftovega paketa iger, te pa so bile del vsakega računalnika z operacijskim sistemom Windows. Ob preizkušanju njihove igre sem si sproti zapisoval, katere lastnosti so mi všeč in bi jih rad obdržal in katere bi si želel izdelati drugače. Tako sem si izdelal skico končnega izgleda grafičnega vmesnika igre, ki jo prikazuje slika 2.



Slika 2: Skica končnega izgleda grafičnega vmesnika

## Programska rešitev naloge

Ko sem si ustvaril želeno sliko o izgledu končnega izdelka, sem moral ugotoviti, kako bi se zadeve lotil programsko. Sprva sem uvozil grafično knjižnico SWING oziroma javax.swing.\*, ki je namenjena izdelavi grafičnih vmesnikov in dodajanju komponent, kot so gumbi, slike, tabele, grafi, risbe, ipd. (10). Prav tako je bilo potrebno uvoziti še ostale knjižnice. Knjižnica java.util.\* mi omogoča sledenje času (12), skupek knjižnic java.awt.\* pa so mi omogočile sledenje miški in nastavitve glede okna grafičnega vmesnika (13). Uvažanje teh knjižnic v kodi prikazuje slika 3.

**import** javax.swing.\*;

**import** java.util.\*;

**import** java.awt.\*;

**import** java.awt.event.MouseEvent;

**import** java.awt.event.MouseListener;

**import** java.awt.event.MouseMotionListener;

Slika 3: Uvažanje programskih knjižnic

### Glavni razred Main

Poleg uvažanja knjižnic je bilo potrebno napisati glavni razred, ki vsebuje metodo ter nit, ki bo pognala celoten program. Koda glavnega razreda aplikacije je dostopna v prilogi 1. Glavni razred Main vsebuje vmesnik Runnable, ki nam omogoča, da iz tega razreda ustvarimo nit (angl. *Thread*), ki je samostojen preprost proces, ki se odvija hkrati z ostalimi nitmi znotraj programa (14). S kodo, ki jo prikazuje slika 4, ustvarimo novo nit, ki jo poženemo z ukazom .start(). Ob tem se izvede vse, kar je definirano v metodi run(). Hkrati smo v glavnem razredu ustvarili tudi po eno instanco vsakega izmed razredov, ki vsebujeta kodo za grafična vmesnika.

Slika 4: Razred Main in definicija niti

public class Main implements Runnable {

// ustvarjanje instanc razredov grafičnega vmesnika

Gui gui = new Gui();

Gui2 infoWindow = new Gui2();

public static void main(String[] args) {

// nit nam omogoča da program izvaja več delov naenkrat

new Thread(new Main()).start();

}

@Override

public void run() {

}

}

S klicem metode run() se začne izvajati neskončna zanka, ki konstantno posodablja predvajano sliko na zaslonu, s čimer dosežemo spreminjanje igralne plošče s časom ter ob vsakem kliku. Prav tako je naloga neskončne zanke nenehno preverjanje stanja igre, torej zmage ali poraza. Skrbi tudi, da se okno z informacijami odpre, ko kliknemo na gumb z vprašajem. Metodo run() in neskončno zanko s pogojnimi stavki za preverjanje prikazuje slika 5.

Slika 5: Metoda run()

**public void** run() {

while (true) { // neskončna zanka

gui.frame.repaint();

if (gui.resetter == false) {

// preverjanje stanja igre zmaga/poraz

gui.checkVictroyStatus();

}

if (gui.info == true) {

// odpiranje okna z informacijami ob kliku na vprašaj

infoWindow.frame.setVisible(true);

infoWindow.open = true;

gui.info = false;

}

}

}

### Razred Gui in njegovi podrazredi ter podprogrami

Kot drugega izmed pomembnejših razredov sem definiral razred Gui, v katerem je zapisana večina kode, ki je odgovorna za delovanje igre. Najprej sem vanj zapisal glavne spremenljivke, ki definirajo velikost glavnega okna grafičnega vmesnika, lokacije gumbov za informacije o igri in ponastavitev igre ter lokacijo polij, kjer se izpisujeta čas ter število bomb, ki jih je potrebno označiti. Prav tako so med njimi spremenljivke, ki določajo stanje igre torej zmago ali poraz, in spremenljivke ki določajo odprtje okna z informacijami. Te prikazuje slika 6.

**public boolean** resetter = false; **public boolean** start = false;

**Date** startDate = **new** Date(); **public int** timeX = 574;

**Date** endDate; **public int** timeY = 5;

**public int** screenHeight = 410; **public int** vicMesX = 374;

**public int** screenWidth = 654; **public int** vicMesY = -50;

**int** spacing = 5; **public int** sec = 0;

**String** vicMes = "N"; **public boolean** info = false;

**public int** mx = -100; **public boolean** happiness = true;

**public int** my = -100; **int** neighs = 0;

**public int** number = 0; **public boolean** victory = false;

**public int** smileyX = 300; **public boolean** defeat = false;

**public int** smileyY = 2; **public boolean** prviKlik = false;

**public int** smileyCenterX = smileyX + 20;  **Random** rand = **new** Random();

**public int** smileyCenterY = smileyY + 20;

Slika 6: Glavne spremenljivke

Poleg glavnih spremenljivk je bilo potrebno definirati tudi 4 glavne tabele (angl. array), v katerih so zapisani vsi podatki o igralčevi interakciji z igralno mrežo in naključno postavljene mine. Zapis teh tabel prikazuje slika 7. Tabela mines je podatkovnega tipa int ali integer, kar pomeni, da se vanjo shranjujejo cela števila. V njej se ob prvem kliku v igralno mrežo na vsako polje zapiše število 0 v primeru, da na tem mestu ni bombe, v nasprotnem primeru pa število 1. V tabelo neighbours se prav tako zapisujejo cela števila, ki nam povedo, koliko min ima vsako polje v sosednjih 8 poljih. Tabela revealed hrani podatke tipa boolean oziroma logične vrednosti true ali false. V primeru, da igralec klikne na neko polje in se to odpre, se privzeta vrednost polja iz false spremeni na true. Posledica tega je, da se na grafičnem vmesniku pokaže kar je skrito pod tistim poljem, na katerega je igralec kliknil. Podobno funkcijo ima tabela flagged, ki je prav tako tipa boolean, njeni podatki pa določajo, katera polja so označena z zastavico in katera ne.

Slika 7: Glavne tabele nujne za delovanje programa

Konstruktor razreda Gui, ki ga prikazuje slika 8, glavnemu oknu grafičnega vmesnika nastavi naslov, definira njegovo velikost v povezavi z že prej omenjenima spremenljivkama, nastavi privzeto funkcija klika na gumb X za zapiranje okna, da se s tem aplikacija ustavi. Nastavlja tudi, da oknu ni mogoče spreminjati velikosti in da je okno stalno vidno. Prav tako poskrbi, da so ob zagonu aplikacije vsa polja v tabeli revealed označena kot zakrita, oknu pa določi, da svojo vsebino dobi iz podrazreda Board. Programu omogoča tudi sledenje miški in pritiskom na miškine gumbe z ustvaritvijo objektov[[1]](#footnote-1) razredov Move in Click.

Slika 8: Konstruktor razreda Gui

// Konstruktor razreda GUI, ki ustvari grafični vmesnik

**public** Gui() {

frame.setTitle("MINOLOVEC");

// oknu nastavi naslov

frame.setSize(screenWidth, screenHeight);

// oknu nastavi velikost

frame.setState(Frame.**NORMAL**);

// dovoli, da aplikacijo ikonificiramo frame.setDefaultCloseOperation(JFrame.**EXIT\_ON\_CLOSE**);

// programu določi, da se ob kliku na X ustavi

frame.setLocationRelativeTo(**null**);

// določi, da se okno vedno odpre na sredini ekrana

frame.setVisible(**true**);

// določi, da je okno vidno

frame.setResizable(**false**);

// določi, da oknu ne moramo spreminjati velikosti

// nastavi, da so vsa polja prvotno zakrita

**for** (**int** i = 0; i < 16; i++) {

**for** (**int** j = 0; j < 9; j++) {

revealed[i][j] = **false**;

}

}

Board board = **new** Board();

frame.setContentPane(board);

Move move = **new** Move();

frame.addMouseMotionListener(move);

Click click = **new** Click();

frame.addMouseListener(click); }

// tabela mines definira lokacijo min v polju 16x9

**int**[][] mines = new int[16][9];

// tabela neighbours definira število min v bližini določenega polja

**int**[][] neighbours = new int[16][9];

// tabela reveald definira stanje odprtosti določenega polja

**boolean**[][] revealed = new boolean[16][9];

// tabela flagged definira stanje označenosti z zastavico določenega polja

**boolean**[][] flagged = new boolean[16][9];

Razred Board, predstavlja JPanel, torej polje, kamor lahko rišemo grafiko. S podaljškom razreda in uvozom metode paintComponent sem si omogočil izrisovanje vseh potrebnih komponent v okno grafičnega vmesnika. Slika 9 prikazuje le manjši del kode za izrisovanje smeška, torej gumba, za ponastavitev igre. Zapisan je tudi if stavek, ki spremlja stanje igre in v primeru, da je igralec izgubil igro, se smeško spremeni v žalosten obraz. Poleg tega je ta metoda zadolžena tudi za izris igralne mreže, zastavice, bombe v primeru, da kliknemo na polje z bombo, ter obeh oken v katerih se izpisujeta čas ter število preostalih neuporabljenih zastavic. Celotno kodo metode paintComponent hrani https://github.com/MatevzSladic/Maturitetna-seminarska-naloga/blob/master/src/Main.java

Slika 9: Del metode paintComponent

**public void** paintComponent(Graphics g) {

// izris smeška

g.setColor(Color.**yellow**);

g.fillOval(smileyX, smileyY, 40, 40);

g.setColor(Color.**black**);

g.fillOval(smileyX + 7, smileyY + 10, 7, 7);

g.fillOval(smileyX + 25, smileyY + 10, 7, 7);

**if** (happiness == **true**) { // izris ust

g.fillRect(smileyX + 12, smileyY + 27, 17, 2);

g.fillRect(smileyX + 11, smileyY + 25, 2, 2);

g.fillRect(smileyX + 28, smileyY + 25, 2, 2);

} **else** {

g.fillRect(smileyX + 12, smileyY + 25, 17, 2);

g.fillRect(smileyX + 11, smileyY + 27, 2, 2);

g.fillRect(smileyX + 28, smileyY + 27, 2, 2);

}

}

PRILOGA 2: Koda razreda glavnega grafičnega vmesnika in s funkcijami igre.

Sledita razred Move in metoda checkVictoryStatus. Razred Move je zadolžen za sledenje miški. Dodal sem mu podaljšek MouseMotionListener, ki sledi premikom miške in ob vsakem premiku miške s pomočjo metode .getX() oz. .getY() pridobi koordinate v pikah (angl. pixel) kje v oknu se nahaja miška.

Metoda checkVictoryStatus je samostojna metoda, ki s pomočjo pogojnih stavkov in dvojnih zank preverja, ali so izpolnjeni pogoji za zmago, ali je morda igralec odkril bombo in s tem izgubil igro. Poskrbi tudi za to, da se ob dosežku katere izmed možnosti (zmage ali poraza) shrani končni čas, torej čas, ki ga je igralec porabil. Preverjanje metodi olajšata še 2 pomožni metodi, totalMines()in totalMinesFlagged(). Prva prešteje število vseh bomb v celotnem polju, druga pa tabelo mines primerja s tabelo flagged in ugotavlja, koliko bomb je pravilno označenih.

Razred, ki vsebuje največji del kode igre je razred Click. Ta z metodo mouseClicked(), ki je razdeljena na dva dela, spremlja vsak pritisk na gumba miške. Prvi del metode spremlja pritiske na levi gumb, s tem pa je povezana večina interakcije z igro. Ko igralec zažene igro in prvič pritisne v mrežo igralnega polja se izvede koda, ki ustvari naključno igralno mrežo in napolni prej omenjene tabeli mines in neighbours tako, da je polje prvega klika vedno prazno in se s pomočjo metode find\_empty\_cells, ki si jo bomo ogledali pozneje, vedno odpre več polij, da je igranje igre olajšano. Programska koda nam s pomočjo sledenja miški in podprogramov inBoxX() in inBoxY()ugotavlja v katerem polju igralne mreže se nahajamo. Slika 10 prikazuje ti dve metodi in dvojne zanke ter pogojna stavka, s pomočjo preračunavanja točk ugotavljata v katerem polju v mreži se nahaja miška. Če se ne nahaja v nobenem polju, metoda vrne vrednost -1.

Slika 10: Metodi za računanje koordinat polja v mreži

**public** **int** inBoxX() {

**for** (**int** i = 0; i < 16; i++) {

**for** (**int** j = 0; j < 9; j++) {

**if** (mx >= spacing + i \* 40 + 6 && mx < spacing + i \* 40 + 40 && my >= spacing + j \* 40 + 40 + 26 && my < spacing + j \* 40 + 40 + 26 + 40 - spacing) {

**return** i;

}

}

}

**return** -1;

}

**public** **int** inBoxY() {

**for** (**int** i = 0; i < 16; i++) {

**for** (**int** j = 0; j < 9; j++) {

**if** (mx >= spacing + i \* 40 + 6 && mx < spacing + i \* 40 + 40 && my >= spacing + j \* 40 + 40 + 26 && my < spacing + j \* 40 + 40 + 26 + 40 - spacing) {

**return** j;

}

}

}

**return** -1;

}

Razred Click prav tako s pomočjo drugih metod preverja, ali je morda igralec kliknil v polje za predčasno ponastavitev igre torej smeška oziroma ali je morda kliknil na gumb za informacije. Igralec lahko, preden začne igro igrati, prebere navodila, ne da bi se mu ob tem že začel prištevati porabljen čas, ko pa igralec klikne v polje na igralni mreži, čas začne teči. Kljub temu, da čas teče, lahko igralec še vedno odpre okno z navodili in si jih prebere kasneje.

V primeru, da igralec želi ponastaviti igro in klikne v smeška, se kliče metoda resetAll(), ki povzroči, da se vse spremenljivke ponovno postavijo na prvotno vrednost, vse tabele se izpraznijo, izgled uporabniškega vmesnika pa se postavi nazaj na prvoten izgled.

Največ težav sem imel z implementacijo pravila, ko igralec odpre polje, ki nima sosednjih bomb, program avtomatsko odpre vsa sosednja polja, ki niso bomba, oziroma toliko polj, dokler ne doseže prvega polja s sosednjo vsaj eno bombo. Rešitev sem našel na spletu (vira številka (15) in (16)), kodo pa priredil tako, da je ustrezala mojemu programu. Potrebno jo je bilo dodelati tudi tako, da je sploh pravilno delovala, saj mi iz neznanega razloga koda, ki sem jo našel na zgoraj omenjenih virih ni delovala pravilno. Metoda find\_empty\_cells je rekurzivna metoda[[2]](#footnote-2), ki ob kliku na prazno polje poskrbi, da se odprejo vsa sosednja polja oziroma polja, ki so v stiku spraznim poljem, ki so bodisi prazna bodisi imajo vsaj 1 sosednjo bombo in ob tem niso bomba. Del kode metode find\_empty\_cells prikazuje slika 11.

Slika 11: Del rekurzivne funkcije find\_empty\_cells

cellY = i + 1;

cellX = j + 1;

**if** (cellY < 9) {

**if** (!(mines[cellX][cellY] == 1)) {

**if** (neighbours[cellX][cellY] == 0

&& revealed[cellX][cellY] == **false**

&& flagged[cellX][cellY] != **true**) {

cellX8 = cellX;

cellY8 = cellY;

}

**if** (flagged[cellX][cellY] != **true**) {

revealed[cellX][cellY] = **true**;

}

}

}

**if** (cellX1 >= 0 && cellY1 >= 0) {

find\_empty\_cells(cellX1, cellY1);

}

### Razred Gui2 in njegovi podrazredi ter podprogrami

Zadnji razred, ki sem ga moral ustvariti, pa je Gui2. Ta definira grafični vmesnik z informacijami in vse njegove komponente, torej gumb, sliki in polja z besedilom. Spremenljivke tega razreda prikazuje slika 12. Spremenljivka frame predstavlja okno, ki se odpre ob kliku na gumb za informacije v oknu grafičnega vmesnika igre, ki ga definira razred Gui. Na tem oknu se prikažejo vse ostale komponente navedene med spremenljivkami. Gumb myButton je označen z napisom »NAZAJ V IGRO«, funkcionalnost pa mu določimo v konstruktorju razreda, ki si ga bomo ogledali kasneje.

Slika 12: Spremenljivke razreda Gui2

JFrame frame = **new** JFrame();

JButton myButton = **new** JButton("NAZAJ V IGRO");

JLabel label1 = **new** JLabel("");

JLabel label2 = **new** JLabel("");

JLabel informacije;

JLabel informacije\_1;

JLabel informacije\_2;

V konstruktorju razreda Gui() sem prej omenjenim spremenljivkam in komponentam določil lastnosti, kot so velikost, lokacijo, besedilo, ki se z njimi izpisuje ter jih vse dodal na okno z informacijami z ukazom:

frame.getContentPane().add(»ime\_komponente«)

Ta oknu frame doda komponento, katere ime smo navedli v zadnjem oklepaju. Slika 13 prikazuje del kode konstruktorja, ki določa lastnosti okna frame in lokacijo ter sledilec pritiska gumbu myButton.

**public** Gui2() {

myButton.setBounds(100, 340, 200, 40);

myButton.setFocusable(**false**);

myButton.addActionListener(**this**);

frame.setTitle("Navodila");

frame.setSize(420, 420);

frame.getContentPane().setLayout(**null**);

frame.setVisible(**false**);

frame.setResizable(**false**);

frame.setLocationRelativeTo(**null**);

frame.getContentPane().add(myButton);

…}

Slika 13: Del konstruktorja Gui2()

Navodila igre so v grafični vmesnik vpisana s pomočjo ukazov HTML, ki mi omogočajo enostavnejše urejanje besedila v vrstice z ukazom <br>, ki povzroči prelom v besedilu in kurzor postavi na začetek nove vrstice. Celotno besedilo navodil je zato zapisano znotraj polja omejenega z <html> … </html>.

Zadnja komponenta razreda Gui2 pa je metoda actionPerformed(ActionEvent e), ki gumbu myButton določa funkcijo, da se ob kliku nanj okno z informacijami zapre. Ta del kode prikazuje slika 14. Gumb myButton ne bi bil potreben, saj se zgodi enako, ko kliknemo na privzeti gumb X za zapiranje aplikacij, vendar pa je zaradi praktičnosti dodan, da igralec ne rabi skrbeti, da bi pomotoma zaprl celoten program.

Slika 14: Metoda, ki določa funkcijo gumbu myButton

@Override

**public** **void** actionPerformed(ActionEvent e) {

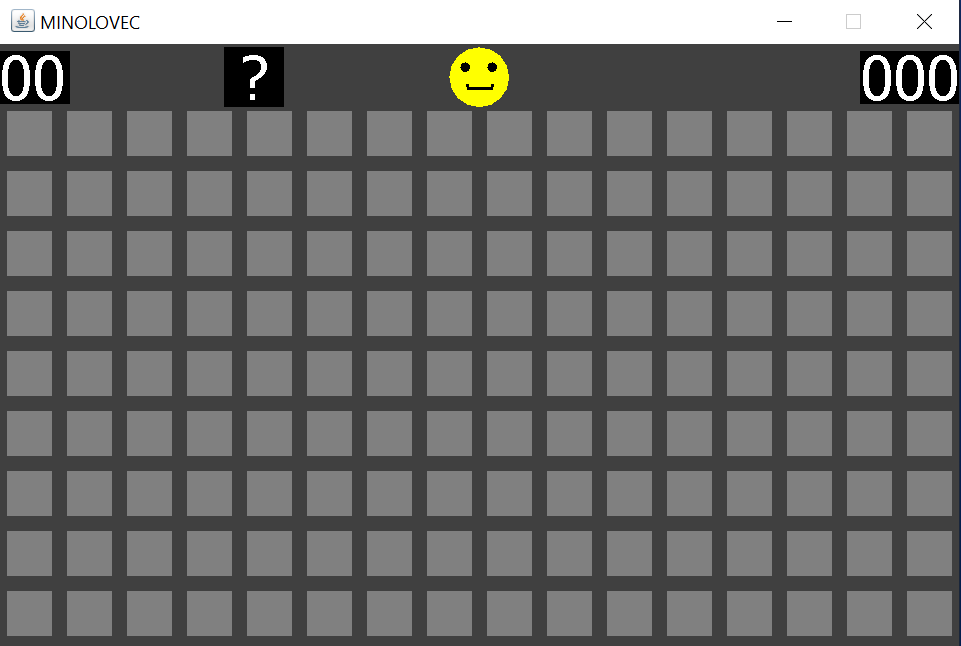
**if** (e.getSource() == myButton) {

frame.setVisible(**false**);

}

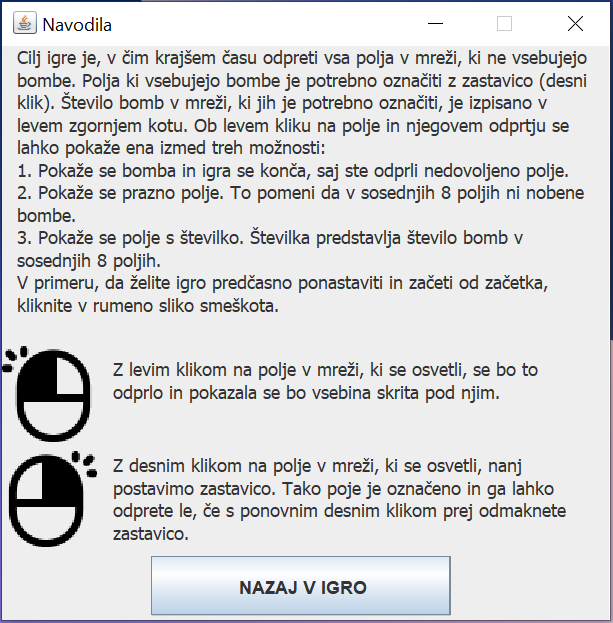
}

### Končni izgled igre

Ob zagonu igre se pojavi okno, ki ga prikazuje slika 15. To je glavno igralno okno, v katerem se nahajata igralna mreža in orodna vrstica, katere komponente so gumb za odpiranje okna z informacijami, gumb za ponastavitev igre in polji, kjer se izpisujeta čas ter število še neporabljenih zastavic. Zadovoljen sem tudi z izgledom končnega izdelka, saj sem dosegel grafične lastnosti kot sem jih predvidel med načrtovanjem v prvotni skici.

Slika 15: Igralno okno aplikacije

Ob kliku na gumb z vprašajem se odpre novo okno, ki ga prikazuje slika 16, na katerem so izpisana navodila za igranje igre.



Slika 16: Okno z informacijami in navodili

Aplikacija v EXE[[3]](#footnote-3) obliki je dostopna med prilogami: https://**github**.com/MatevzSladic/Maturitetna-seminarska-naloga/blob/master/src/Gui2.java

PRILOGA 4: Aplikacija igre Minolovec.

# ZAKLJUČEK

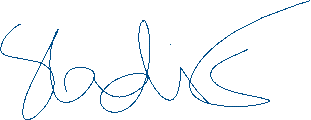
S seminarsko nalogo sem vam želel predstaviti moj postopek izdelovanja računalniške igre. Menim, da sem dosegel svoj glavni cilj, to je izdelati funkcionalno računalniško igro Minolovec oziroma njeno novo različico. Pri končnem izdelku pa ostaja še veliko prostora za izboljšave. Nalogo bi lahko nadgradil tako, da bi aplikacijo naredil prilagodljivo glede na resolucijo zaslona, na katerem je zagnana. Prav tako imam željo nalogo nadgraditi tako, da bo v prihodnosti imela več nivojev težavnosti in različne velikosti igralnih mrež, ter posledično večje ali manjše število bomb v mreži. Izdelovanje maturitetnega izdelka me je tudi približalo k programiranju, saj sem vanj vložil veliko časa, se naučil veliko novih metod in pridobil nova znanja na področju izdelovanja iger.

# ZAHVALA

Iskreno se zahvaljujem svojemu mentorju, prof. dr. Albertu Zorku, za vso pomoč in svetovanje pri izdelavi naloge ter čas, ki mi ga je posvetil. Zahvaljujem se Vam tudi za korektno predavano snov, ki je bila podlaga za izdelavo naloge.

Prav tako bi se rad zahvalil vsem, ki ste mi tekom izdelovanja maturitetne seminarske naloge kakorkoli pomagali, mi svetovali in mi namenili svoj čas. Zahvalil bi se rad tudi staršema, ki sta mi stala ob strani skozi lepe in slabe trenutke tekom mojega srednješolskega izobraževanja.

Matevž Sladič



# LITERATURA

1. **Oracle.** What is Java technology and why do I need it? [Elektronski] Oracle. [Navedeno: 3.. marec 2021.] https://www.java.com/en/download/help/whatis\_java.html.

2. —. The Java Language Environment. [Elektronski] [Navedeno: 3.. marec 2021.] https://www.oracle.com/java/technologies/introduction-to-java.html.

3. **Evans, Benjamin J. in Flanagan, David.** *Java in a Nutshell.* Sebastopol : O’Reilly Media, Inc., 2015.

4. **Fundation, Eclipse.** Eclipse. [Elektronski] Eclipse Foundation, Inc. [Navedeno: 3.. marec 2021.] https://www.eclipse.org/org/.

5. **Burnette, Ed.** *Eclipse IDE Pocket Guide.* s.l. : O'Reilly Media, Inc., 2005. 9780596100650.

6. **Cobbett, Richard.** The most successful game ever: a history of Minesweeper. [Elektronski] 5.. maj 2009. [Navedeno: 30.. marec 2021.] https://www.techradar.com/news/gaming/the-most-successful-game-ever-a-history-of-minesweeper-596504.

7. **Workman, Kevin.** [Elektronski] [Navedeno: 4.. april 2021.] https://happycoding.io/tutorials/java/libraries.

8. **Mesojedec, Uroš in Fabjan, Borut.** *Java 2: temelji programiranja.* Ljubljana : Pasadena, 2004. str. 24-37. 961-6361-30-9.

9. *Računalništvo: Interno gradivo.* **Zorko, Albert.** Novo mesto : s.n., 2020.

10. **Mesojedec, Uroš in Fabjan, Borut.** *Java 2: temelji programiranja.* Ljubljana : Pasadena, 2004. str. 105 -. 961-6361-30-9.

11. **Software, Fog Creek.** [Elektronski] Atlassian, september 2011. [Navedeno: 3.. marec 2021.] https://trello.com/about.

12. **Oracle.** Package java.util. [Elektronski] Oracle. [Navedeno: 4.. april 2021.] https://docs.oracle.com/javase/8/docs/api/java/util/package-summary.html.

13. **JavaTpoint.** [Elektronski] JavaTpoint. [Navedeno: 4.. april 2021.] https://www.javatpoint.com/java-awt.

14. **Davison, Andrew.** *Killer game programming in Java.* Sebastopol : O’Reilly Media, Inc., 2005. 978-0-596-00730-0.

15. **Bodnar, Jan.** Zetcode. *Java Minesweeper.* [Elektronski] 2007-2021. [Navedeno: 3.. marec 2021.] https://zetcode.com/javagames/minesweeper/.

16. **JON.** Java Tutorial - Creating a Minesweeper. [Elektronski] 28.. oktober 2016. [Navedeno: 4.. oktober 2020.] https://youtube.com/playlist?list=PLGxHvpw-PAk6QvPw0fYe8bks31GRKvymK.

17. **Mahnič, Viljan, Fürst, Luka in Rožanc¸, Igor.** *Java skozi primere.* Šenčur : Bi-Tim, 2008. 978-061-6046-10-7.

# PRILOGE

## PRILOGA 1: Koda glavnega razreda aplikacije z neskončno zanko

Glavni razred programa aplikacije je možno dostopati preko spodnje povezave:

<https://github.com/MatevzSladic/Maturitetna-seminarska-naloga/blob/master/src/Main.java>

## PRILOGA 2: Koda razreda glavnega grafičnega vmesnika in s funkcijami igre

Koda razreda v katerem sta definirana glavni grafični vmesnik in koda s funkcijami igre sta dostopna na povezavi:

<https://github.com/MatevzSladic/Maturitetna-seminarska-naloga/blob/master/src/Gui.java>

## PRILOGA 3: Koda razreda grafičnega vmesnika z navodili

Koda razreda v katerem je definiran grafični vmesnik z navodili igre je dostopna na povezavi:

<https://github.com/MatevzSladic/Maturitetna-seminarska-naloga/blob/master/src/Gui2.java>

## **PRILOGA 4:** Aplikacija igre Minolovec

Na spodnji povezavi se nahaja stran, s katere si lahko prenesete končno aplikacijo igre Minolovec v EXE obliki:

<https://github.com/MatevzSladic/Maturitetna-seminarska-naloga/blob/master/Minolovec.exe>

ali pa na povezavi:

<https://github.com/MatevzSladic/Maturitetna-seminarska-naloga/releases/tag/v1.0>

## PRILOGA 5: Seminarska naloga v elektronski obliki in predstavitev za zagovor seminarske naloge

<https://github.com/MatevzSladic/Maturitetna-seminarska-naloga/tree/master/Seminarska%20naloga%20in%20PPT>

1. Instanca razreda. [↑](#footnote-ref-1)
2. Metoda, ki za svoje delovanje kliče samo sebe. [↑](#footnote-ref-2)
3. Oblika zagonske datoteke, ki ni vezana na izvorno mesto kode. [↑](#footnote-ref-3)