



SVEUČILIŠTE U MOSTARU  
FAKULTET STROJARSTVA RAČUNARSTVA I ELEKTROTEHNIKE

## **Programiranje u Javi**

*Projektni zadatak: **NINJA: Crimson-Shadow 2D platformer***

**Marko Rezić**

Akadska godina 2017/2018.

## PROJEKTNI ZADATAK:

### **Opis problema:**

Detaljno opisati problem koji je potrebno riješiti izgradnjom programske podrške. Svaki student piše svoju temu minimalno 500 znakova.

Ovaj projekt obuhvaća sustav za upravljanje korisničkim podacima (administratora i običnih igrača), te sustav za upravljanje igricom NINJA: Crimson Shadow.

Sustav se sastoji od logina, registracije, baze podataka, administracije, glavnog izbornika za igru i samu igru.

Odvojeno je korisničko sučelje napravljeno posebno za administratore tako da imaju potpun nadzor nad povjerljivim korisničkim podacima, i mogu mijenjati po potrebi, također svaki korisnik za sebe podatke može mijenjati unutar glavnog izbornika za igru.

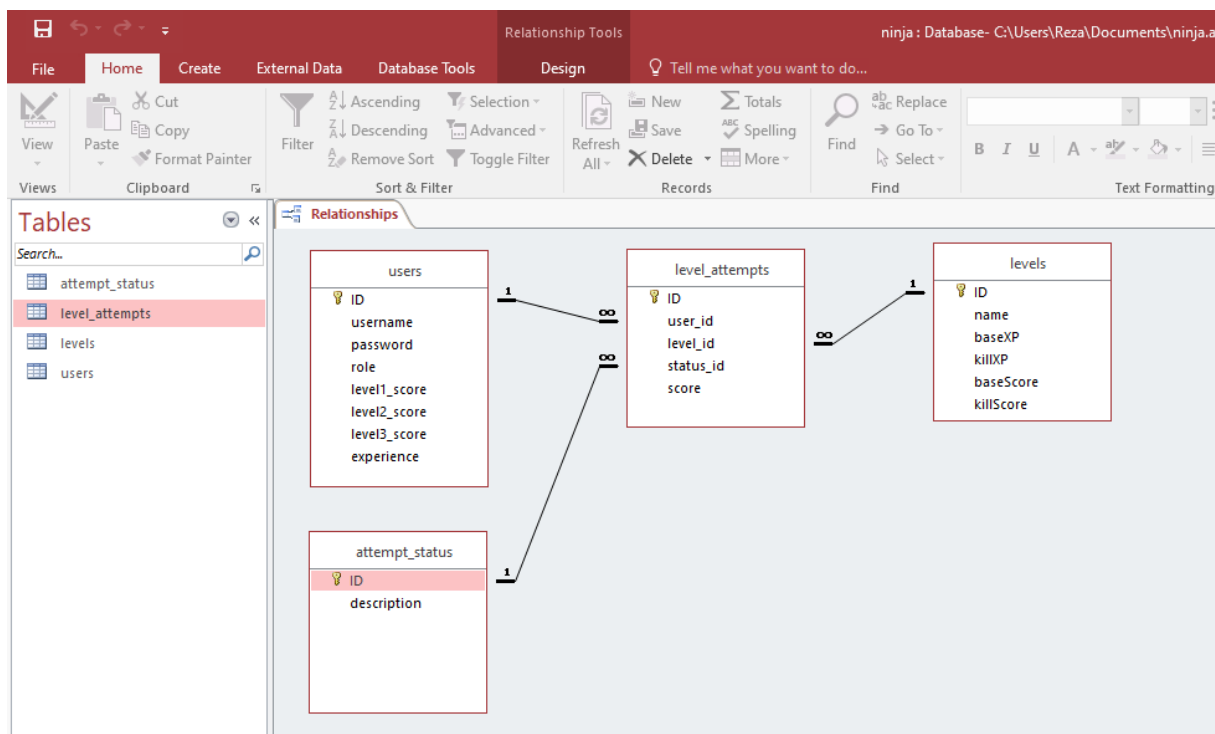
Sustav omogućava komunikaciju između baze i relevantnih statistika iz igrice koje treba pratiti kao što su sami igrači, njihovi rezultati, pojedine razine igrice itd.

# 1. Modeliranje podataka

Prvi dio zadatka se radi u .

- Poglavlje 1.1. (Relacijski model podataka) - potrebno je staviti sliku relacijskog modela kreiranog u MS Accessu.
- Poglavlje 1.2. (Opis tablica u relacijskom modelu) - potrebno je navesti i opisati sve tablice (i atribut unutar tablica) iz dobivenog modela te na kraju priložiti sliku
- Poglavlje 1.3. (Aplikacija u Javi) - sadrži screenshotove kreiranih formi Javi.

## 1.1. Relacijski model podataka



## 1.2. Opis tablica u relacijskom modelu

### Users

Naziv atributa	Integritet (PK/FK, NULL / NOT NULL)	Kratki opis atributa
id	PK NOT NULL AUTO INCREMENT	Id korisnika
username	NOT NULL	Korisničko ime
password	NOT NULL	Šifra korisnika
role	NOT NULL	Uloga (admin ili igrač)
level1_score	NOT NULL DEFAULT 0	Najveći rezultat postignut od korisnika na prvom nivou, 0 ako nije odigrao
Level2_score	NOT NULL DEFAULT 0	Najveći rezultat postignut od korisnika na drugom nivou, 0 ako nije odigrao
Level3_score	NOT NULL DEFAULT 0	Najveći rezultat postignut od korisnika na trećem nivou, 0 ako nije odigrao

## Tehnička dokumentacija projekta

experience	NOT NULL DEFAULT 0	Ukupno iskustvo (XP) koji je igrač stekao igrajući, što određuje i njegov Ninja Level
------------	--------------------	---

### Levels

Naziv atributa	Integritet (PK/FK, NULL / NOT NULL)	Kratki opis atributa
id	PK NOT NULL	Id nivoa
name	NOT NULL	Naziv nivoa
baseXP	NOT NULL	Experience poeni koje korisnik dobije ako samo završi nivo
killXP	NOT NULL	Experience poeni koje korisnik dobije za svakog neprijatelja ubijenog u nivou
baseScore	NOT NULL	Rezultat koji dobije korisnik ako samo završi nivo
killScore	NOT NULL	Dodatni rezultat za svakog ubijenog neprijatelja

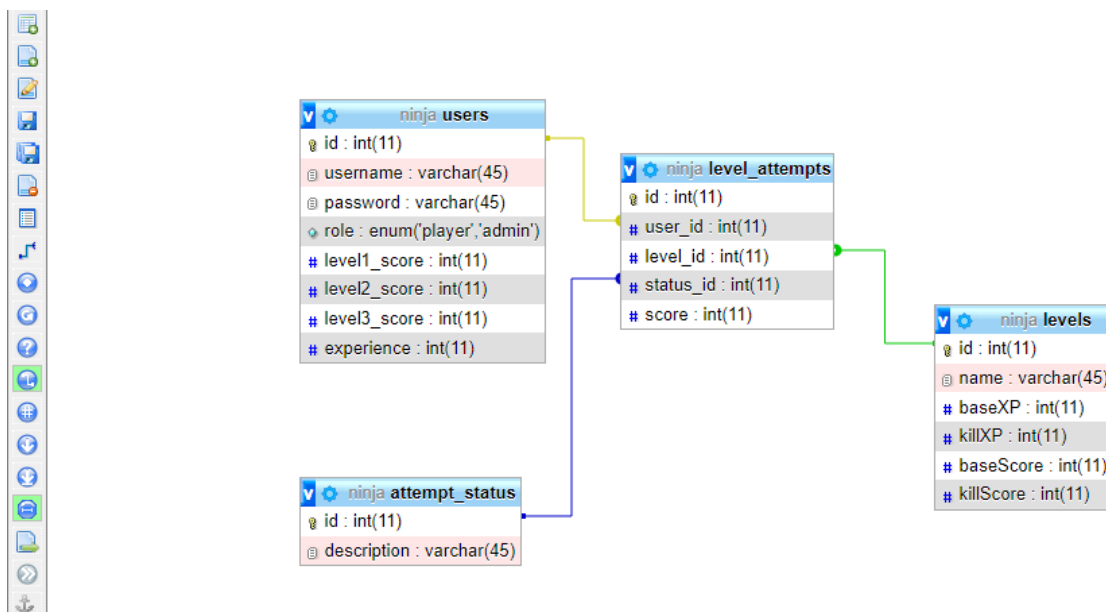
### Attempt status

Naziv atributa	Integritet (PK/FK, NULL / NOT NULL)	Kratki opis atributa
id	PK NOT NULL AUTO INCREMENT	Id statusa
description	NOT NULL	Opis statusa pokušaja ('Game over', 'Razina prijeđena') i sl.

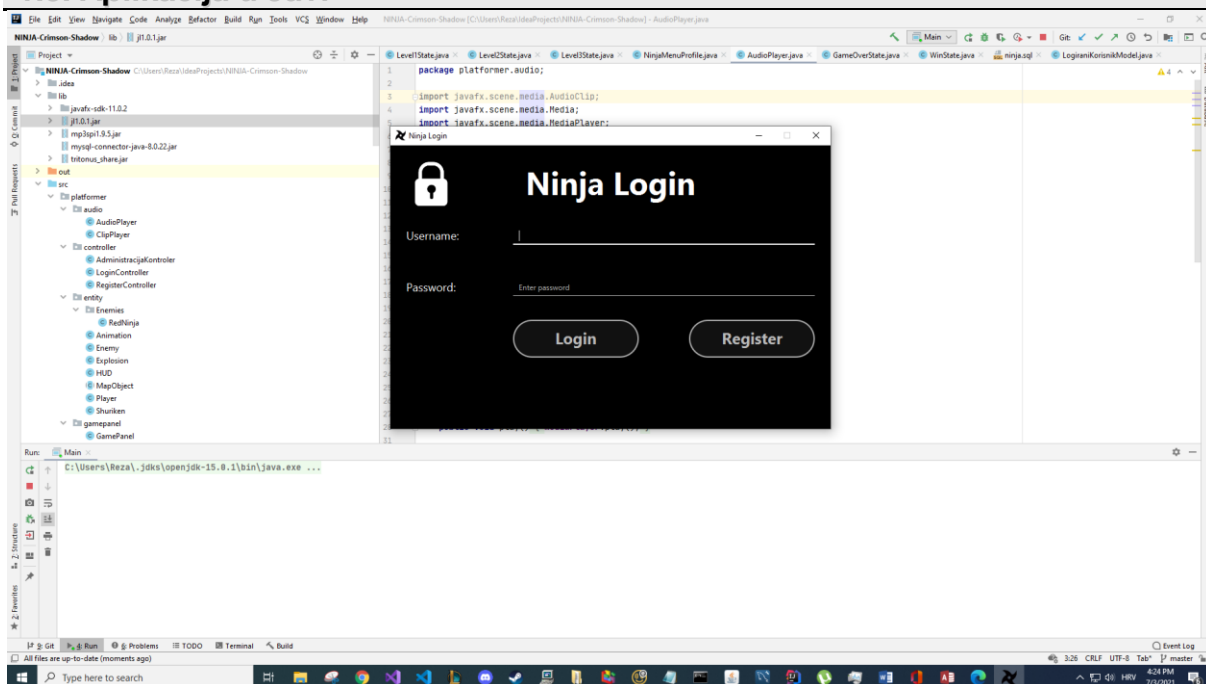
### Level attempts

Naziv atributa	Integritet (PK/FK, NULL / NOT NULL)	Kratki opis atributa
id	PK NOT NULL AUTO INCREMENT	Id pokušaja nivoa
user_id	FK NOT NULL	Id korisnika koji je pokušao preći nivo (referenca na tablicu users)
level_id	NOT NULL	Id nivoa koji je pokušao preći korisnik (referenca na tablicu levels)
status_id	NOT NULL	Id statusa pokušaja koji opisuje njegovu uspješnost (referenca na tablicu attempt_status)
score	NOT NULL	Rezultat koji je postigao korisnik na kraju pokušaja

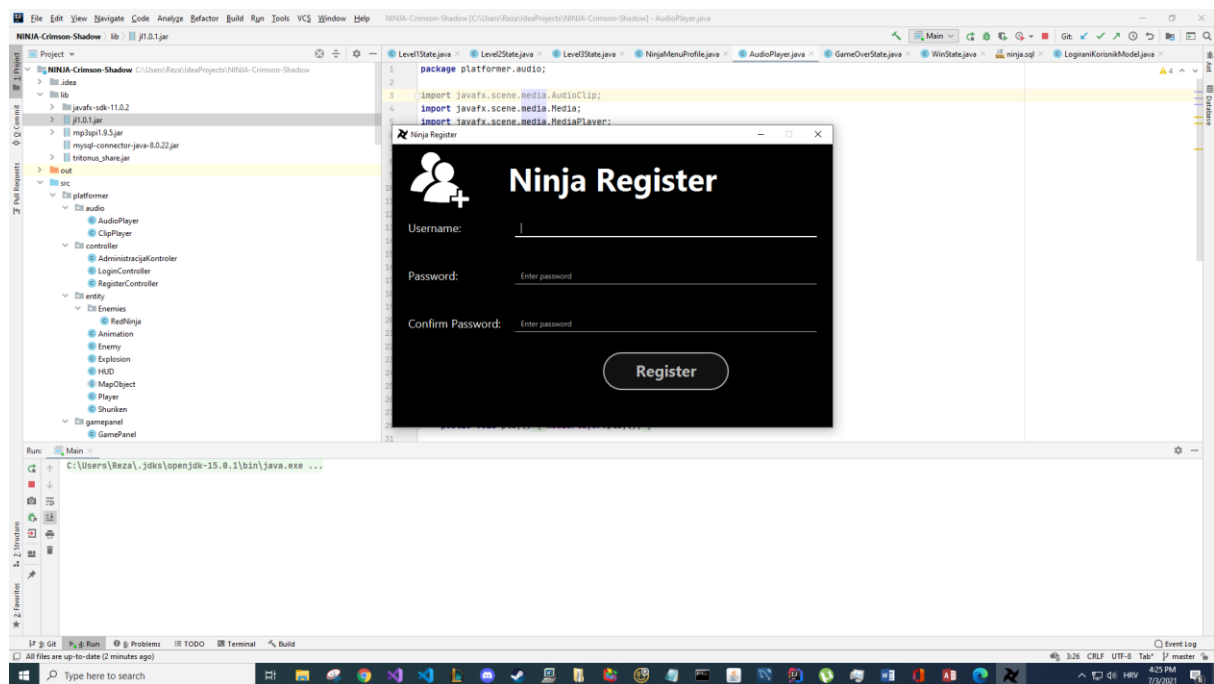
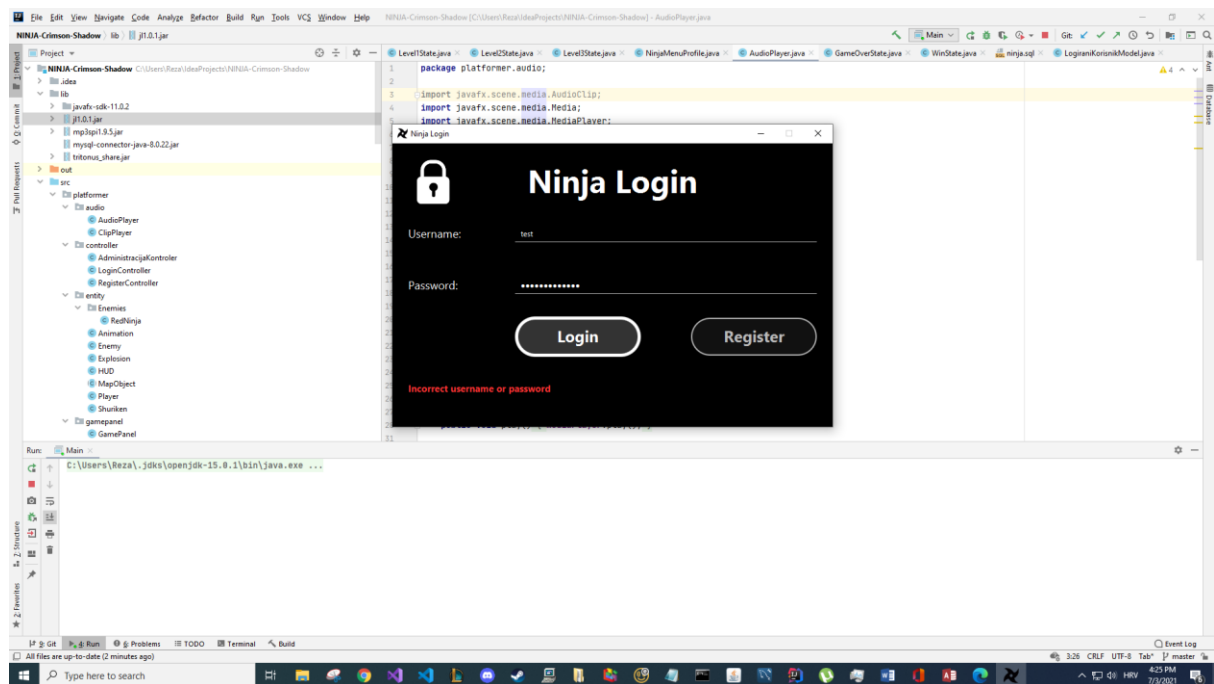
Table	Action	Rows	Type	Collation	Size	Overhead
<input type="checkbox"/> attempt_status	Browse  Structure  Search  Insert  Empty  Drop	3	InnoDB	utf8_unicode_ci	16.0 KiB	-
<input type="checkbox"/> levels	Browse  Structure  Search  Insert  Empty  Drop	3	InnoDB	utf8_unicode_ci	16.0 KiB	-
<input type="checkbox"/> level_attempts	Browse  Structure  Search  Insert  Empty  Drop	9	InnoDB	utf8_unicode_ci	64.0 KiB	-
<input type="checkbox"/> users	Browse  Structure  Search  Insert  Empty  Drop	3	InnoDB	utf8_unicode_ci	16.0 KiB	-
4 tables	Sum	18	InnoDB	utf8_unicode_ci	112.0 KiB	0 B



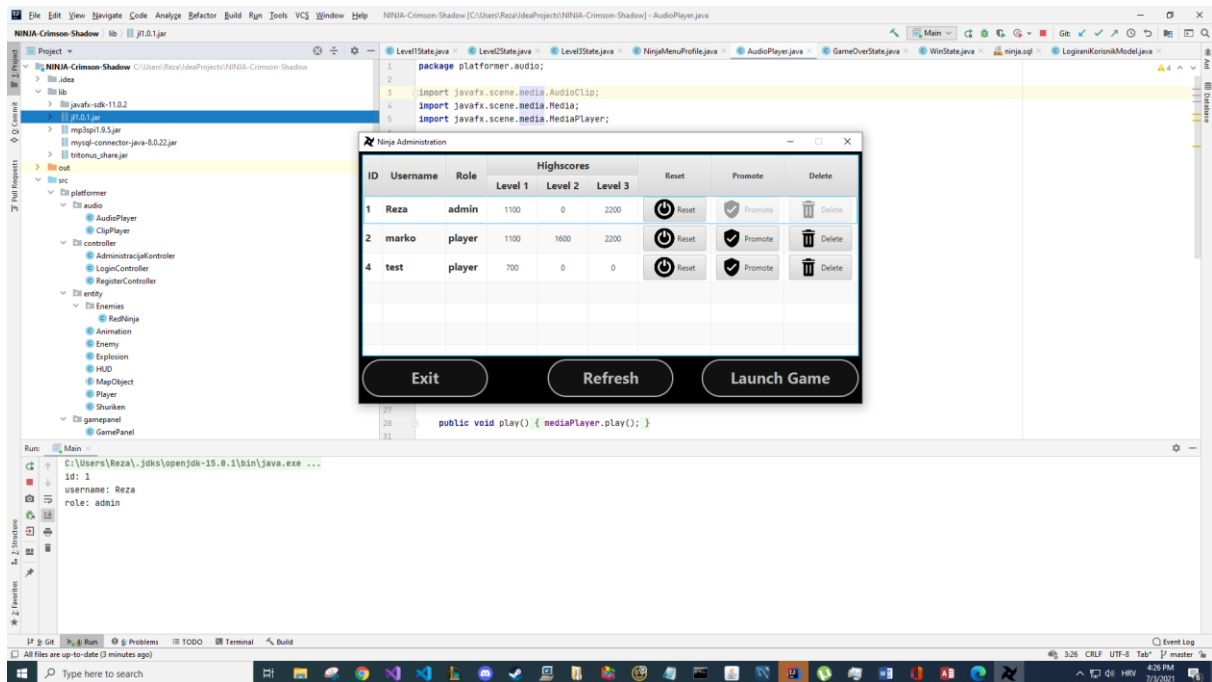
## 1.3. Aplikacija u Javi



## Tehnička dokumentacija projekta

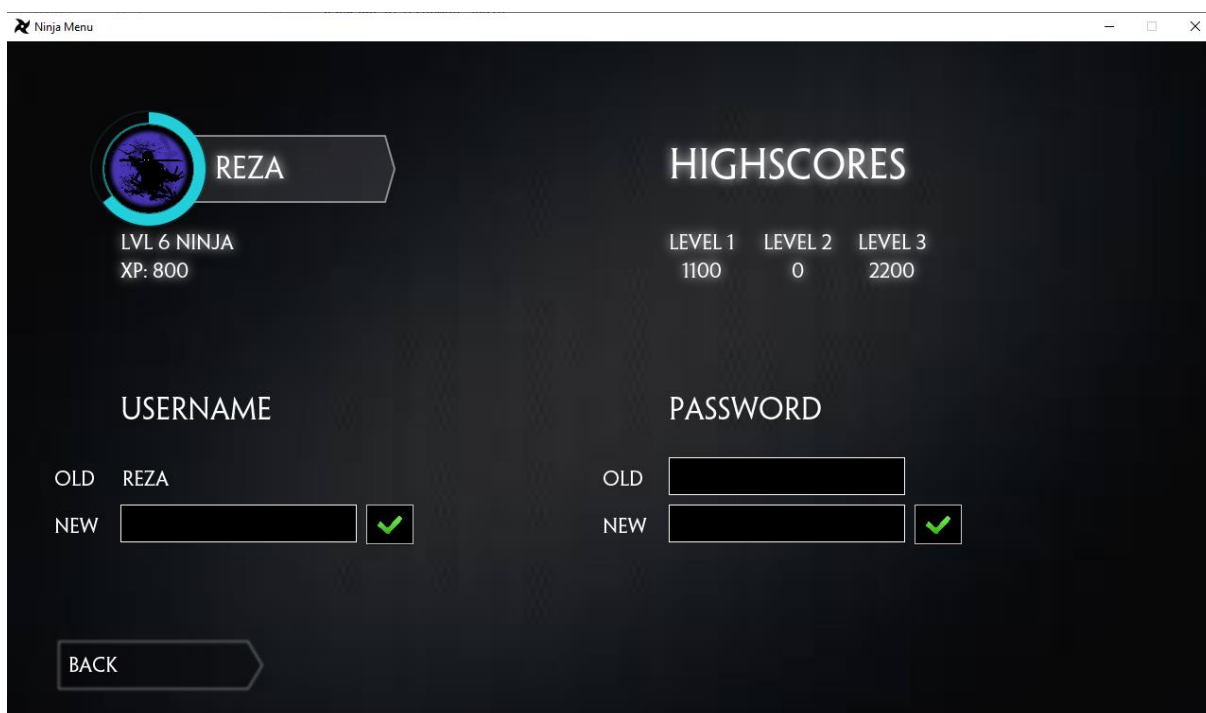
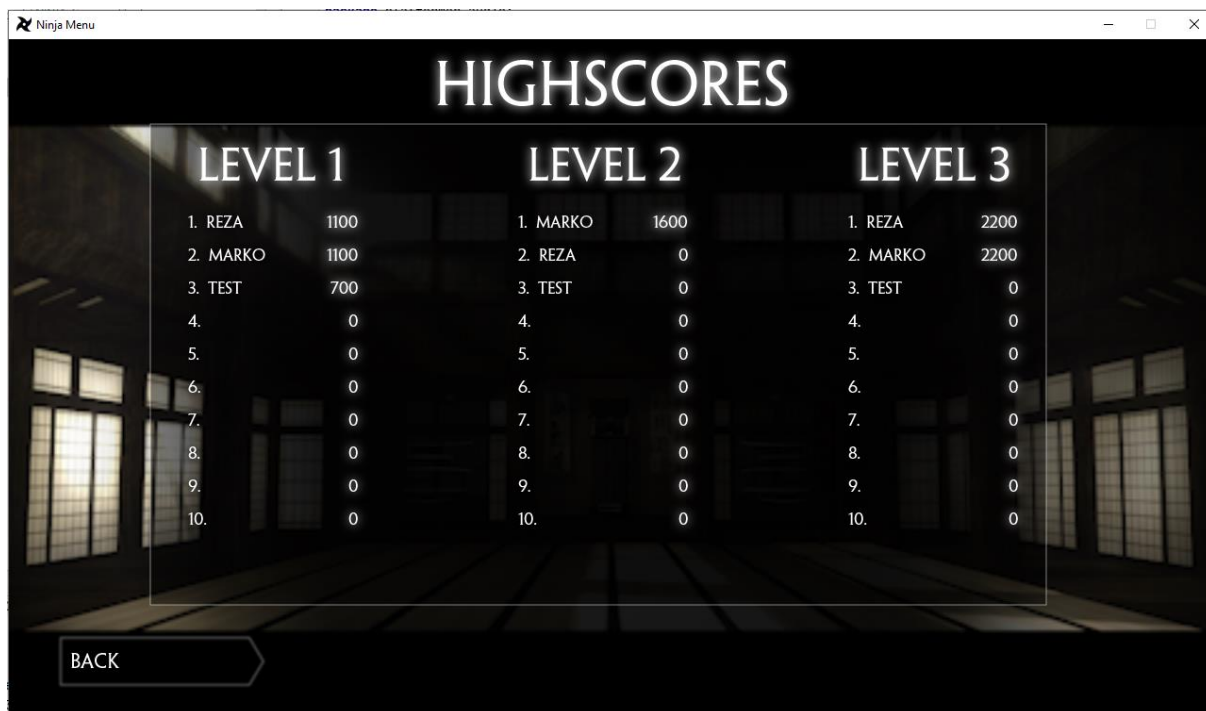


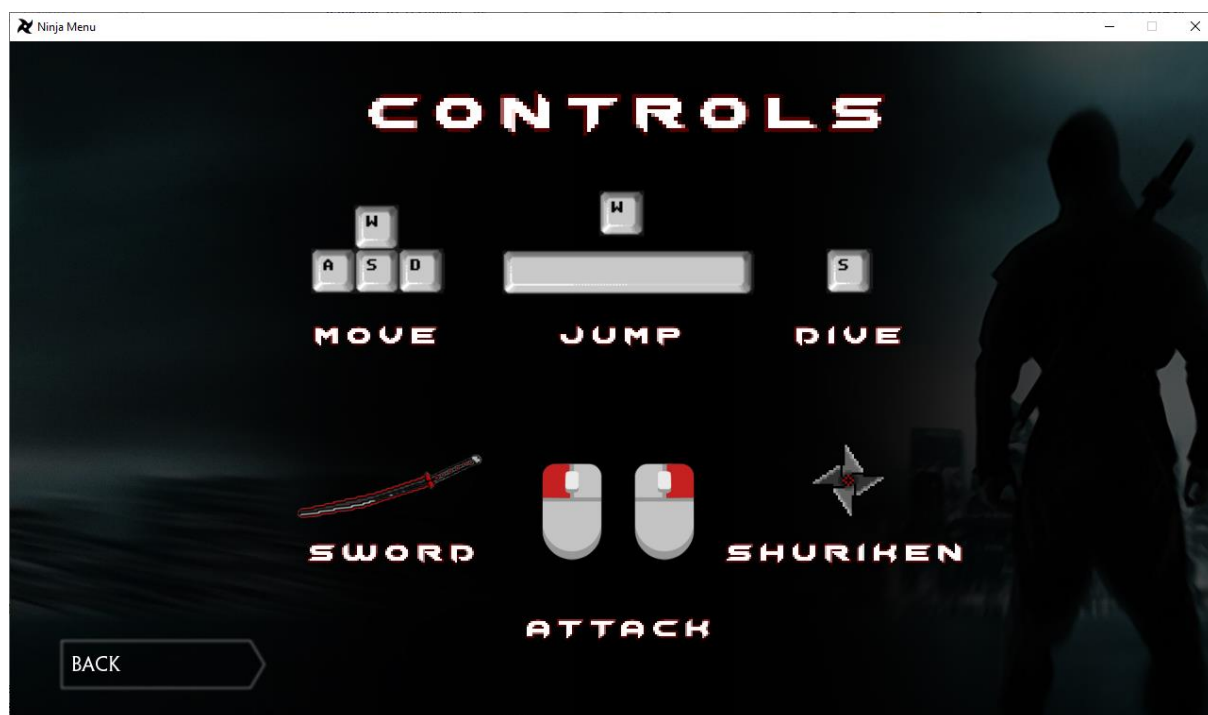
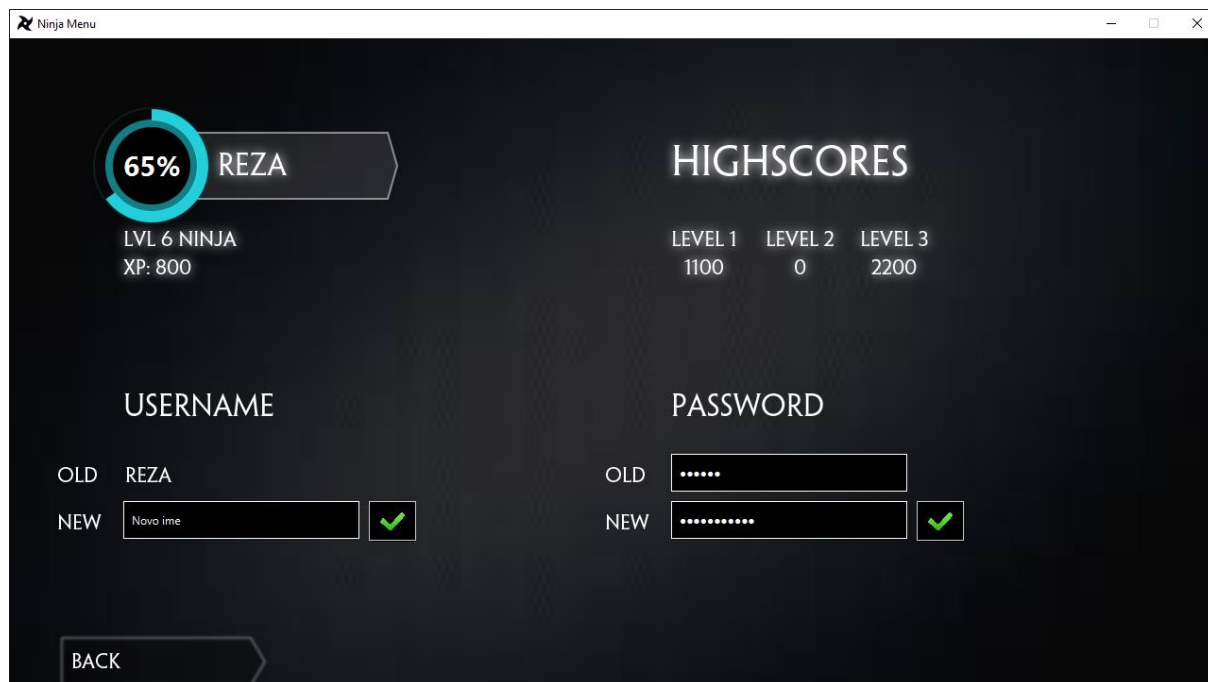
## Tehnička dokumentacija projekta

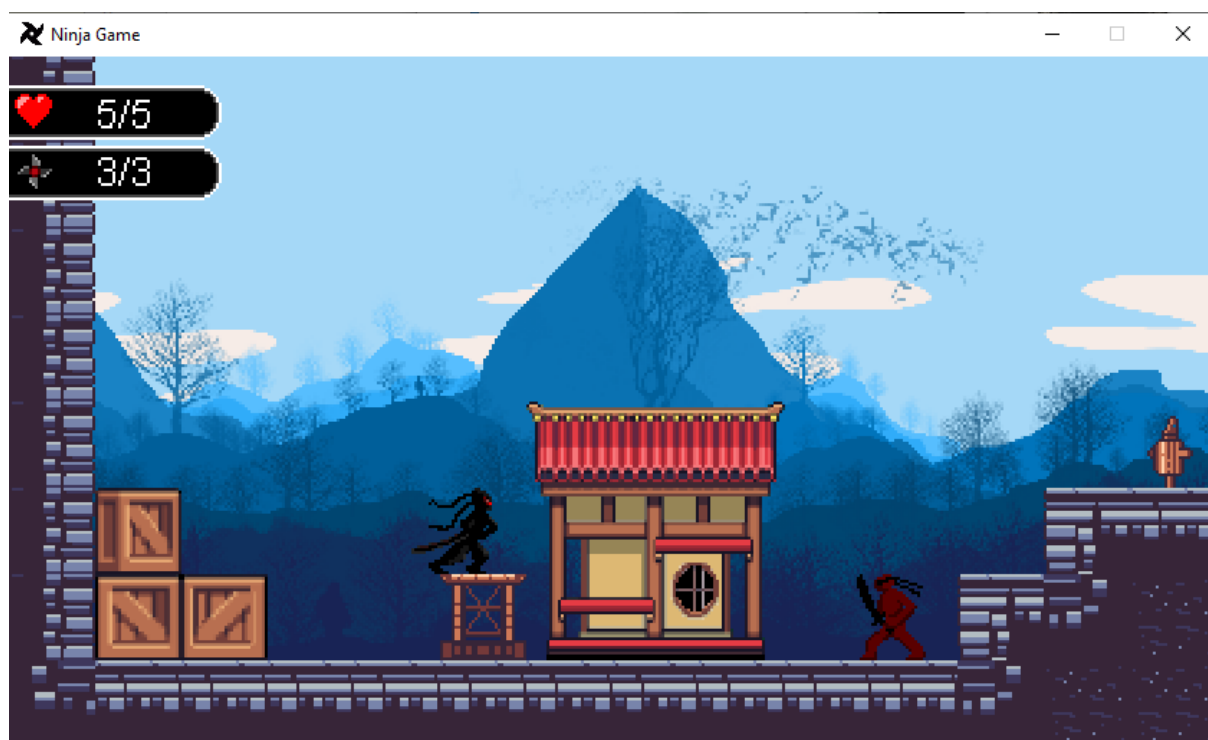
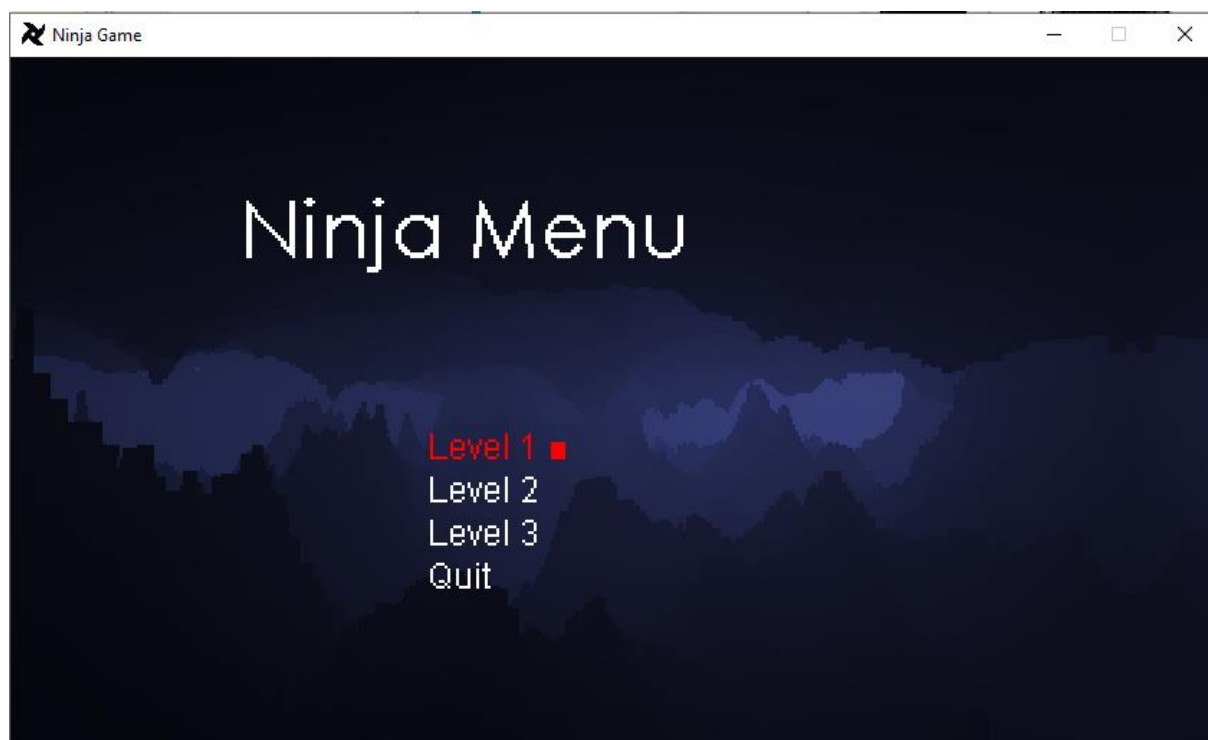


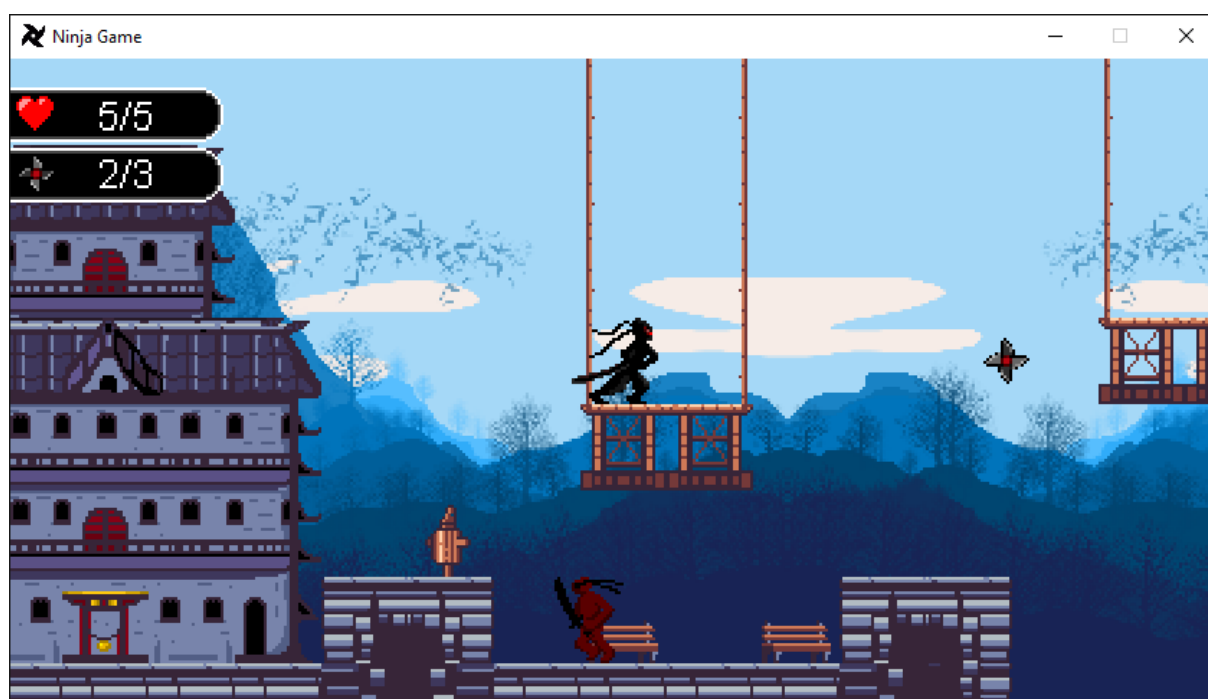
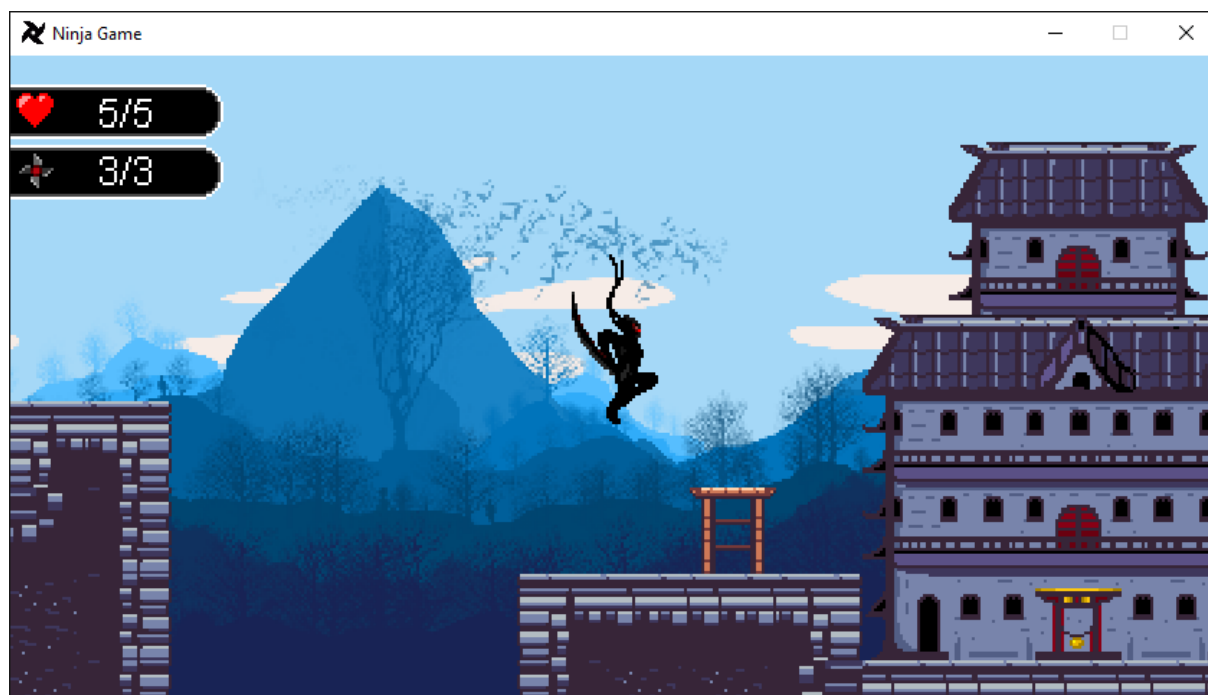


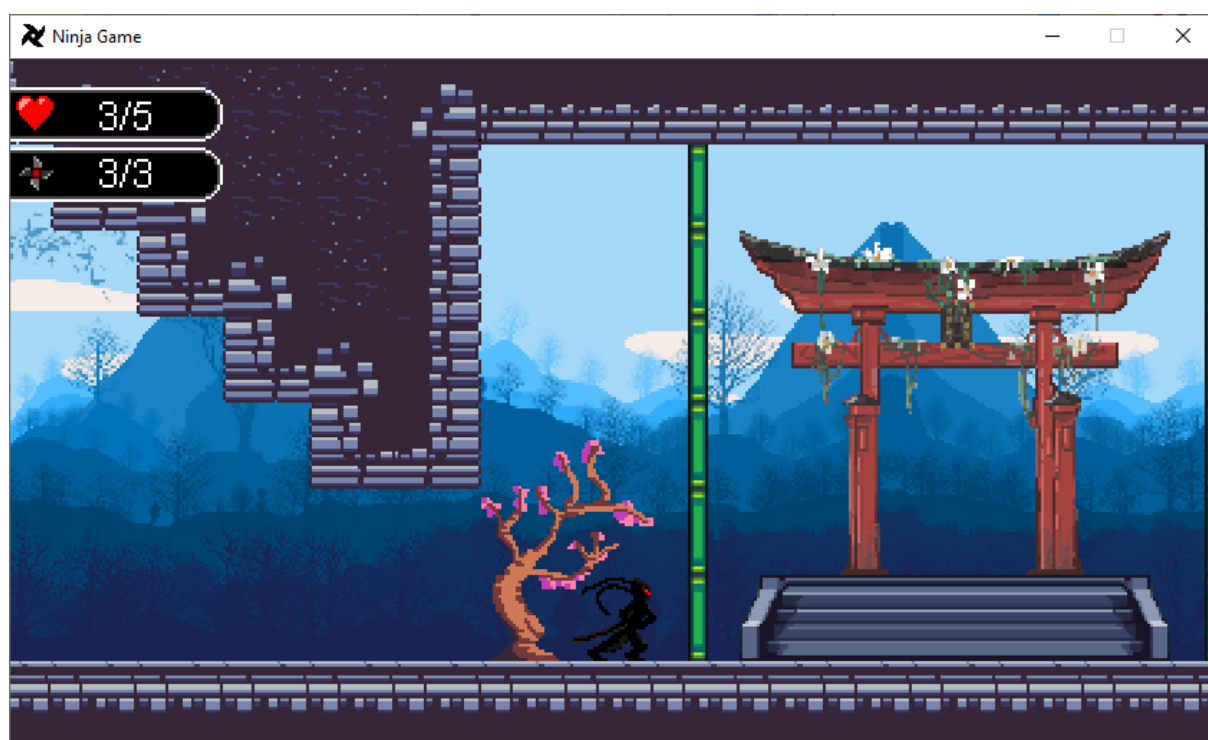
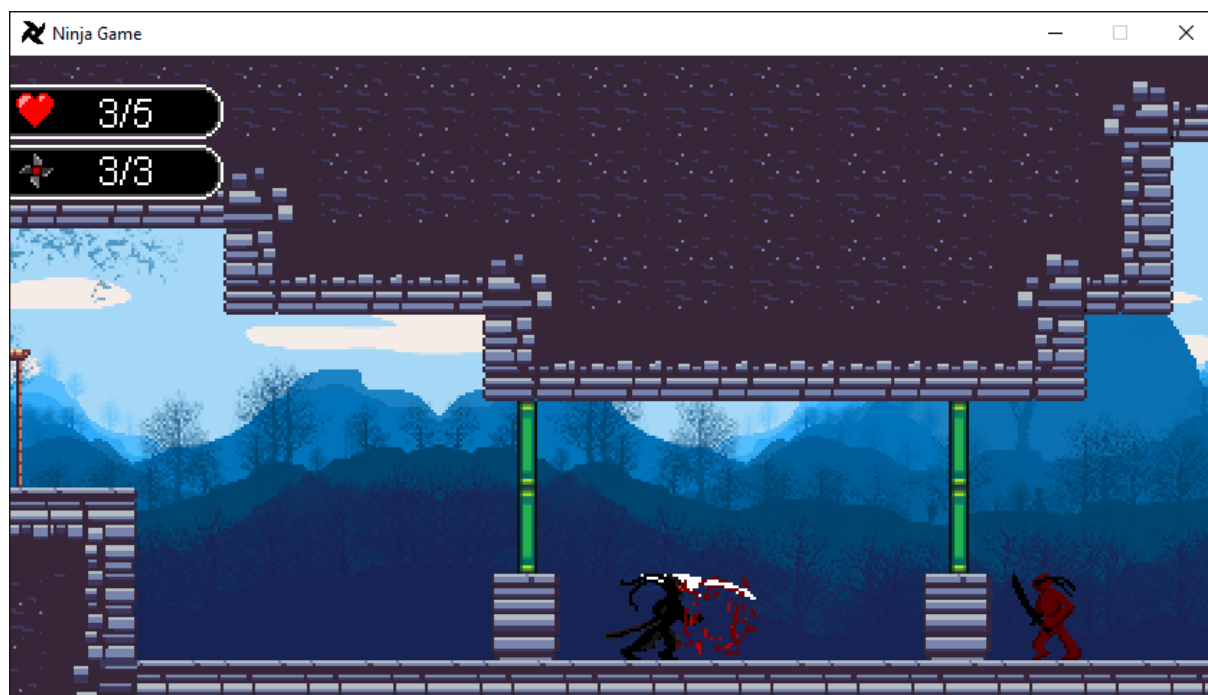


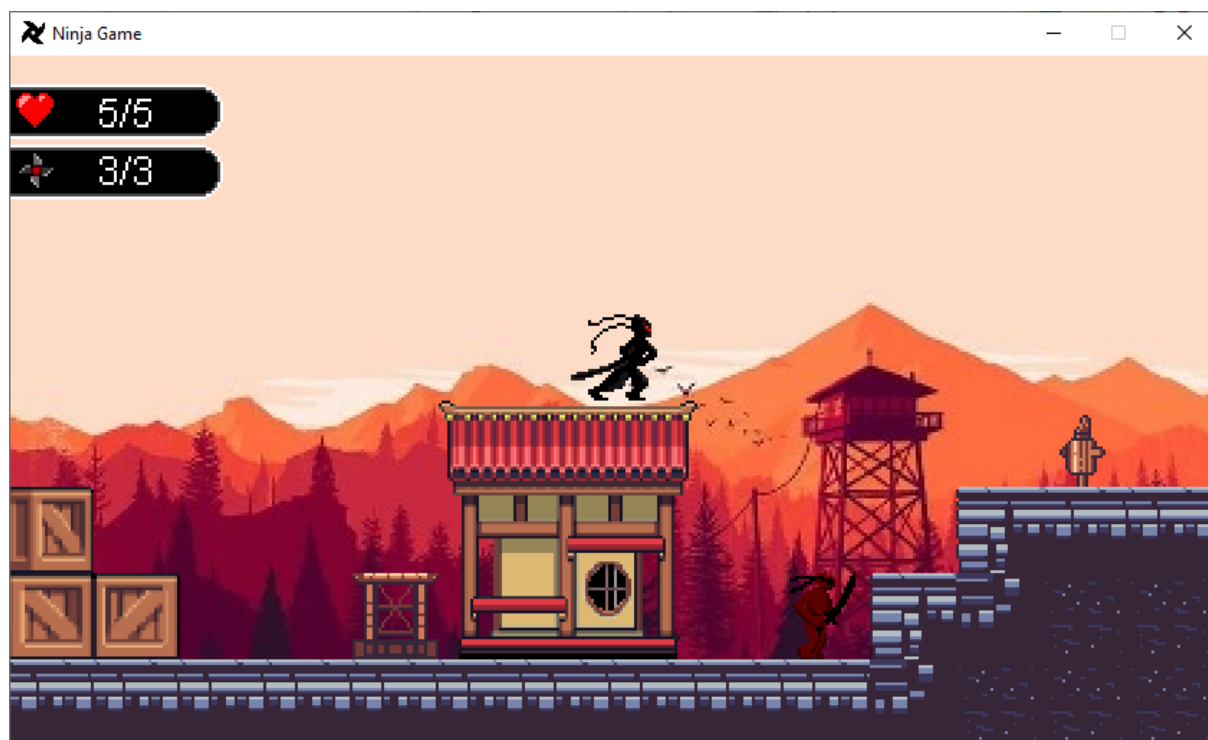
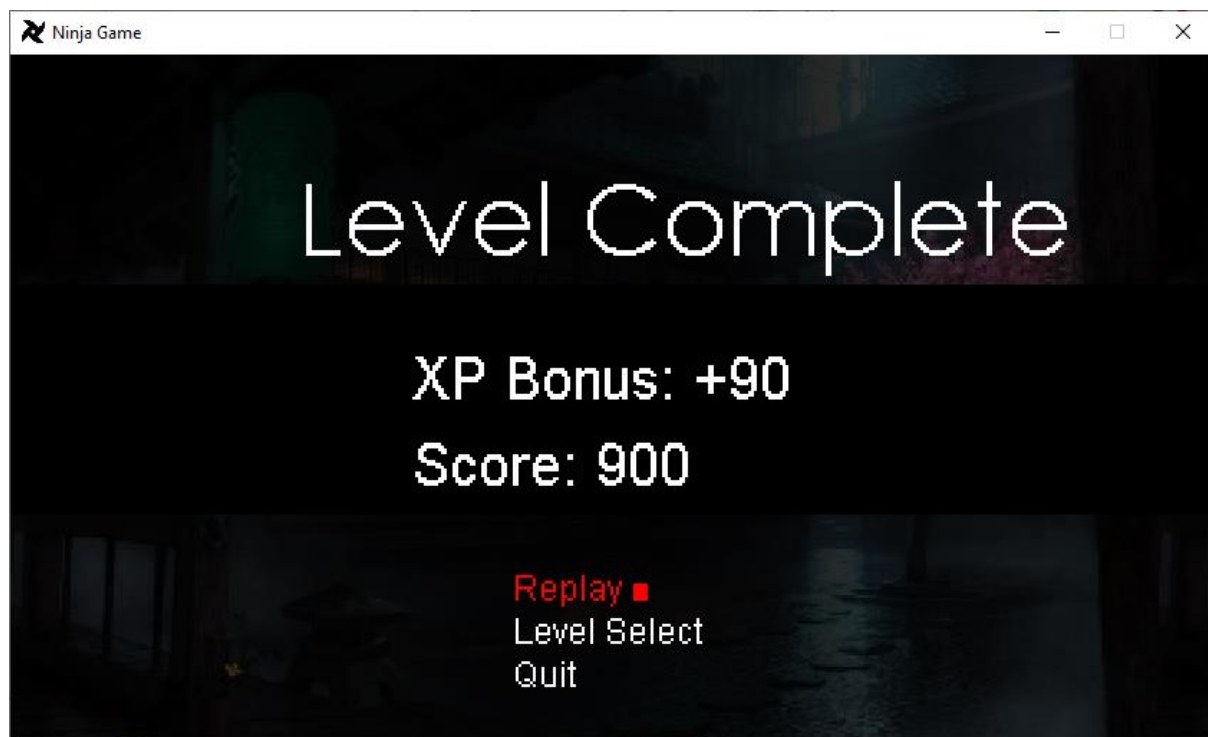


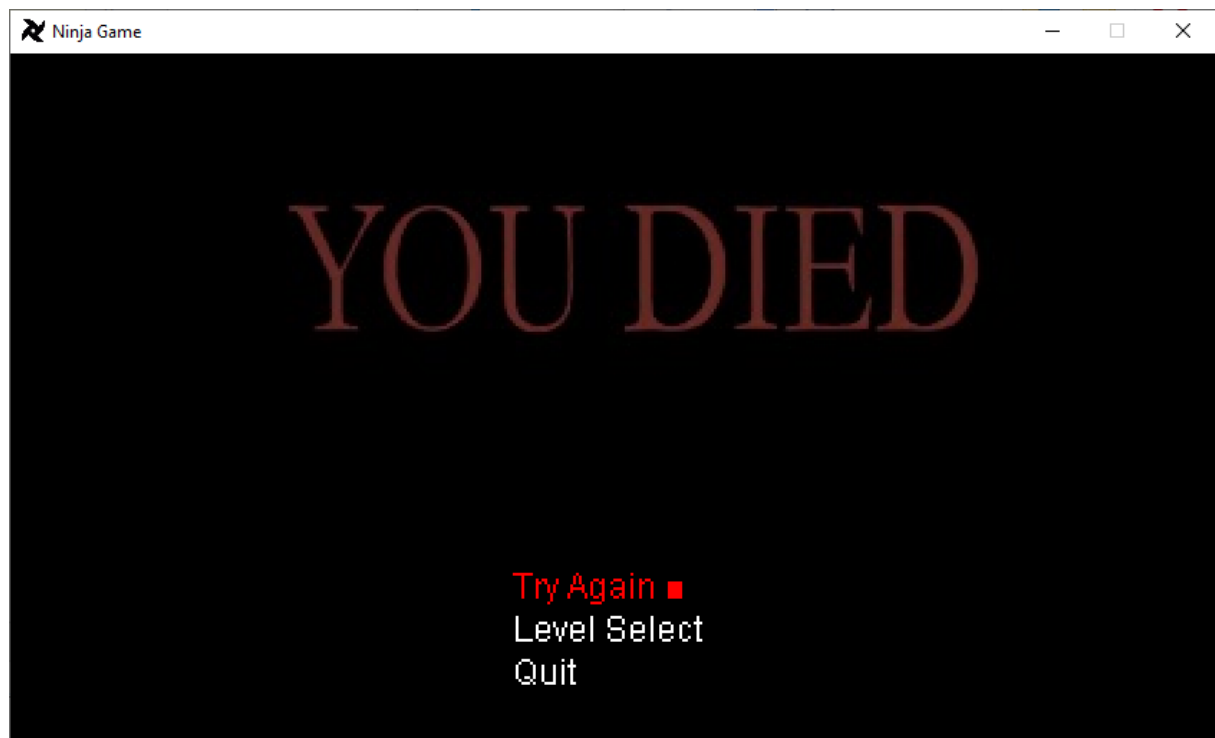














## 2. Model programske podrške

Drugi dio zadatka se odnosi na modeliranje programske podrške, detaljno ćete raspisati sve klase i na kraju priložiti dijagram klase koji opisuje izgled programske podrške:

- Napraviti class dijagram
- Opisati metode koje se nalaze unutar svake klase

U navedenim klasama potrebno je opisati :

- 1) Paketi – kratko opisati paket vaše programske podrške
- 2) Klase – kratko opisati klase koje koristite (ukratko za što će se klasa koristiti
  - a. Atribute – kratko pisati attribute koje se nalaze unutar klase;
  - b. Metode – kratko opisati metode koje se koriste za što služe i opišite njihov povratni tip;

### 2.1. Paketi

---

Opis paketa

Projekt Ninja Crimson Shadow sastoji se od 2 paketa: Platformer i Resources

Paket Platformer se sastoji od 9 paketa i 1 klase GameStart (main) koji pokreću i upravljaju svim funkcionalnim komponentama aplikacije

**Platformer paketi:**

1. **Audio:** 2 klase za učitavanje i sviranje zvukova i glazbe
2. **Controller:** 3 klase za upravljanje modelima i podacima iz baze
3. **Entity:** 1 paket Enemies koji sadrži sve klase igračevih neprijatelja (trenutno samo 1 klasa), i 7 klasa za opće objekte na mapi, grafičko sučelje, igrača, projekte i animacije
4. **Gamepanel:** 1 klasa koja upravlja prozorom od 2D platformer igrice
5. **Gamestate:** 9 klasa koje opisuju sva moguća stanja igre i upravljaju svim događanjima vezanim za objekte u igri
6. **Model:** 7 klasa za dohvatanje, spremanje, mijenjanje i brisanje podataka iz baze
7. **Ninjamenu:** 10 klasa koje zajedno čine čitav glavni izbornik od igrice
8. **Tilemap:** 3 klase za sve pozadinske i blokove objekte u igrici, i njihov razmještaj/mapiranje
9. **View:** 3 css i 3 fxml datoteke za pripadajuće kontrolere logina, registracije i administracije













Paket Resources se sastoji od 10 paketa koji sadrže sve potrebne multimedijske datoteke aplikacije





**Resources paketi:**

1. **Backgrounds:** sve slikovne pozadine
2. **Fonts:** svi korišteni nestandardni fontovi
3. **HUD:** grafičko sučelje za igricu
4. **Images:** slike manjih objekata u aplikaciji (ikone i slično)
5. **Maps:** tekstualne datoteke sa zapisom koji govori raspored elemenata mape za igranje
6. **Music:** pozadinska glazba
7. **SFX:** zvučni efekti
8. **Sprites:** slike (frameovi) korišteni za animiranje objekte
9. **Styles:** dodatni stilovi za manje elemente
10. **Tilesets:** skup elementarnih blokova od kojih se gradi mapa igre

## 2.2. Klase

---

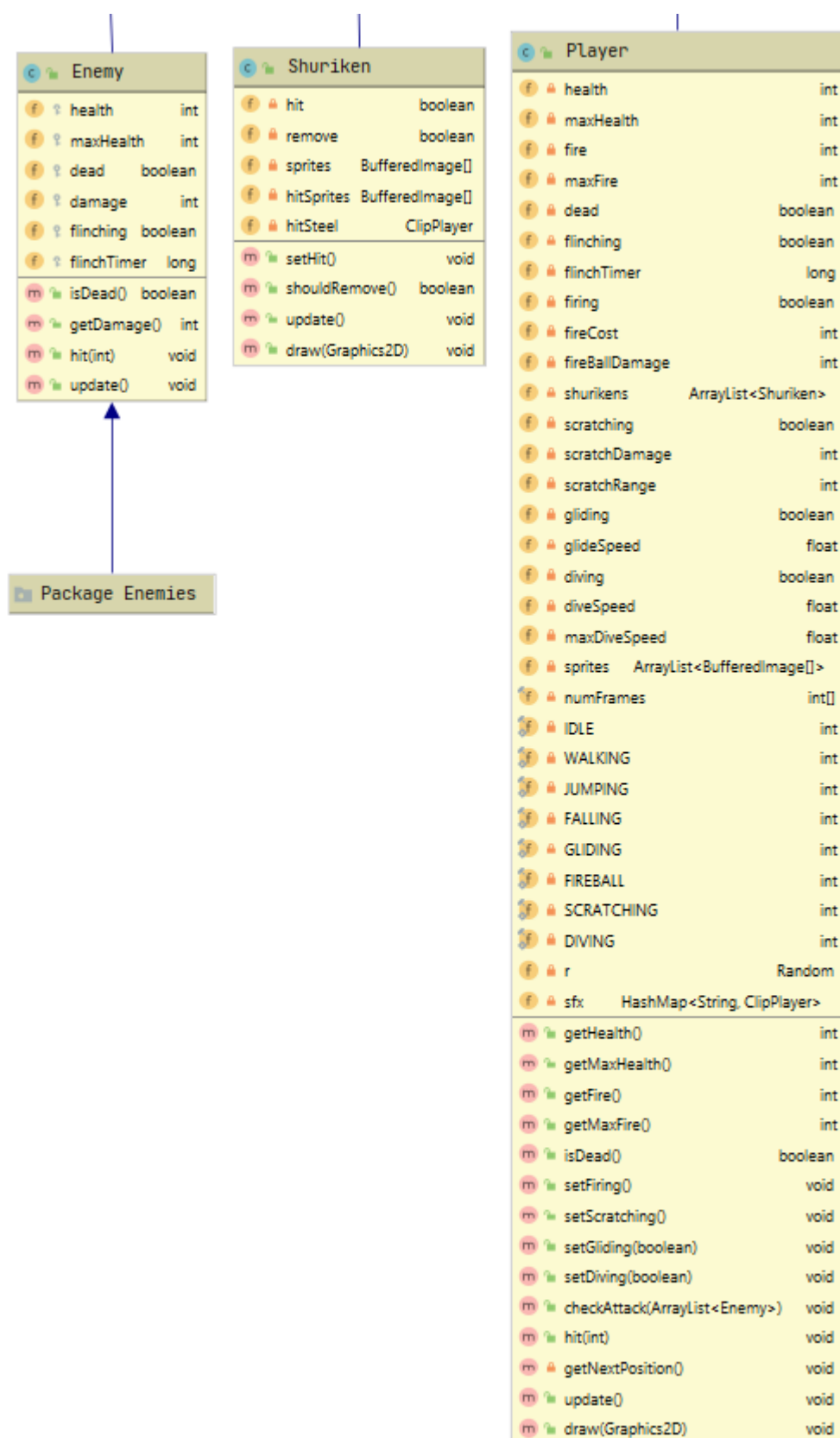
		AudioPlayer
		media Media
		mediaPlayer MediaPlayer
		play() void
		playLoop() void
		stop() void

		ClipPlayer
		clip AudioClip
		play() void

AdministracijaKontroler		
f	usersTable	TableView
f	idCol	TableColumn
f	usernameCol	TableColumn
f	roleCol	TableColumn
f	level1Col	TableColumn
f	level2Col	TableColumn
f	level3Col	TableColumn
f	resetCol	TableColumn
f	promoteCol	TableColumn
f	deleteCol	TableColumn
	initialize(URL, ResourceBundle)	void
	updateTable()	void
	openLogin(ActionEvent)	void
	launchGame(ActionEvent)	void
	letterbox(Scene, Pane)	void

LoginController		
f	statusLbl	Label
f	usernameTxt	TextField
f	passwordTxt	PasswordField
f	currentUser	LogiraniKorisnikModel
	login(ActionEvent)	void
	updateCurrentUser()	void
	openRegister(ActionEvent)	void
	letterbox(Scene, Pane)	void
	initialize(URL, ResourceBundle)	void

RegisterController		
f	statusLbl	Label
f	usernameTxt	TextField
f	passwordTxt	PasswordField
f	repasswordTxt	PasswordField
	register(ActionEvent)	void
	letterbox(Scene, Pane)	void
	initialize(URL, ResourceBundle)	void



MapObject		
f	tileMap	TileMap
f	tileSize	int
f	xmap	double
f	ymap	double
f	x	double
f	y	double
f	dx	double
f	dy	double
f	width	int
f	height	int
f	cwidth	int
f	cheight	int
f	currRow	int
f	currCol	int
f	xdest	double
f	ydest	double
f	xtemp	double
f	ytemp	double
f	topLeft	boolean
f	topRight	boolean
f	bottomLeft	boolean
f	bottomRight	boolean
f	animation	Animation
f	currentAction	int
f	previousAction	int
f	facingRight	boolean
f	left	boolean
f	right	boolean
f	up	boolean
f	down	boolean
f	jumping	boolean
f	falling	boolean
f	moveSpeed	double
f	maxSpeed	double
f	stopSpeed	double
f	fallSpeed	double
f	maxFallSpeed	double
f	jumpStart	double
f	stopJumpSpeed	double
m	intersects(MapObject)	boolean
m	getRectangle()	Rectangle
m	calculateCorners(double, double)	void
m	checkTileMapCollision()	void
m	getx()	int
m	gety()	int
m	getWidth()	int
m	getHeight()	int
m	getCWidth()	int
m	getCHeight()	int
m	setPosition(double, double)	void
m	setVector(double, double)	void
m	setMapPosition()	void
m	setLeft(boolean)	void
m	setRight(boolean)	void
m	setUp(boolean)	void
m	setDown(boolean)	void
m	setJumping(boolean)	void
m	notOnScreen()	boolean
m	draw(Graphics2D)	void

Explosion		
f	x	int
f	y	int
f	xmap	int
f	ymap	int
f	width	int
f	height	int
f	animation	Animation
f	sprites	BufferedImage[]
f	remove	boolean
m	update()	void
m	shouldRemove()	boolean
m	setMapPosition(int, int)	void
m	draw(Graphics2D)	void

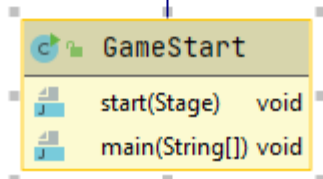
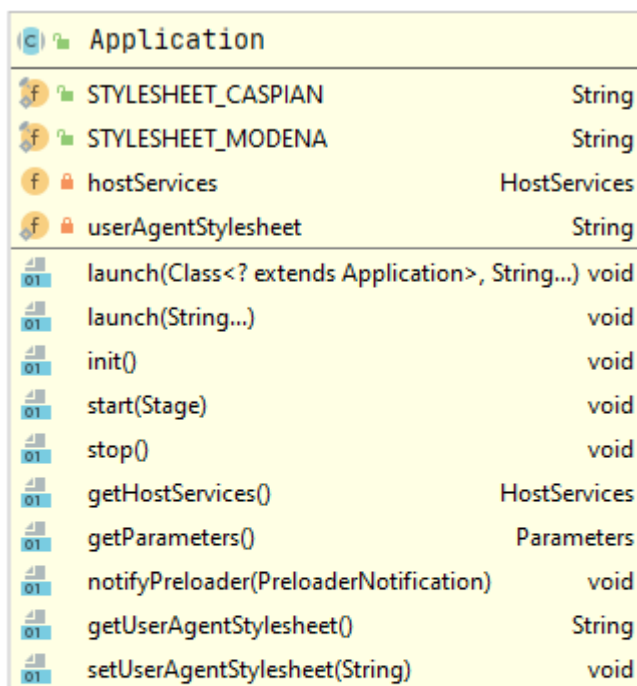
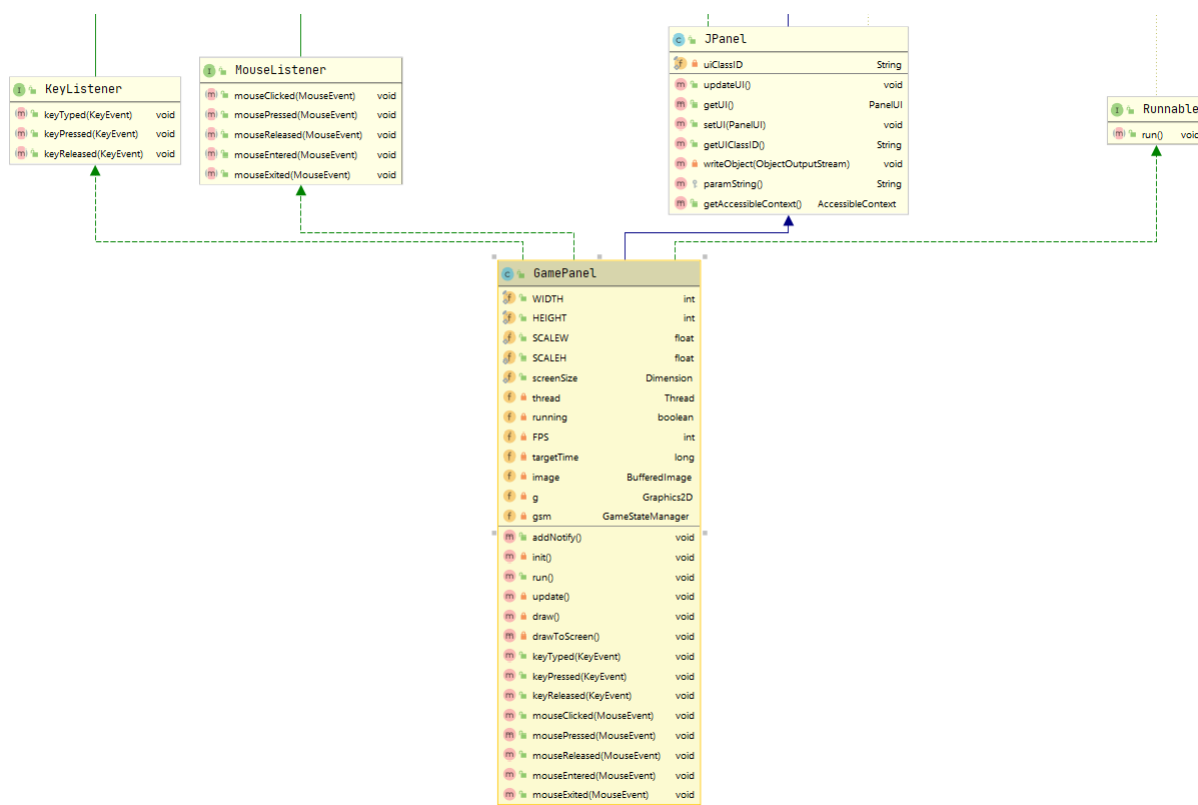
Animation		
f	frames	BufferedImage[]
f	currentFrame	int
f	startTime	long
f	delay	long
f	playedOnce	boolean
m	setFrames(BufferedImage[])	void
m	setDelay(long)	void
m	setFrame(int)	void
m	update()	void
m	getFrame()	int
m	getImage()	BufferedImage
m	hasPlayedOnce()	boolean

HUD		
f	player	Player
f	image	BufferedImage
f	font	Font
m	draw(Graphics2D)	void

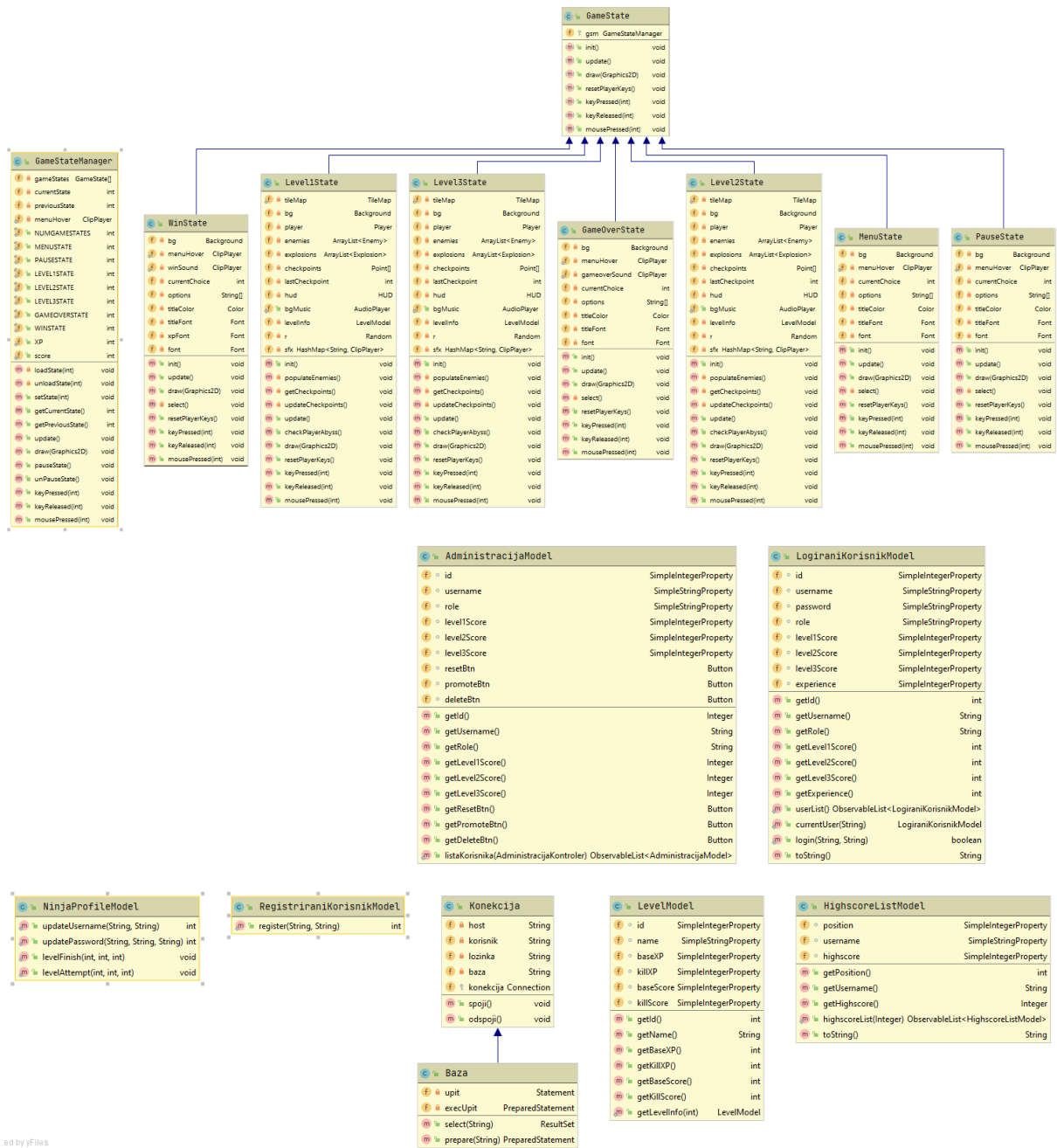
Enemy		
f	health	int
f	maxHealth	int
f	dead	boolean
f	damage	int
f	flinching	boolean
f	flinchTimer	long
m	isDead()	boolean
m	getDamage()	int
m	hit(int)	void
m	update()	void

RedNinja		
f	sprites	BufferedImage[]
m	getNextPosition()	void
m	update()	void
m	draw(Graphics2D)	void





# Tehnička dokumentacija projekta





NinjaMenuSlider	
text	Text
shadow	Effect
blur	Effect
bgblur	Effect
menuHover	ClipPlayer
slider	Slider
setOnAction(Runnable)	void

TileMap		
f	x	double
f	y	double
f	xmin	int
f	ymin	int
f	xmax	int
f	ymax	int
f	tween	double
f	map	int[][]
f	tileSize	int
f	numRows	int
f	numCols	int
f	width	int
f	height	int
f	tileset	BufferedImage
f	numTilesAcross	int
f	tiles	Tile[][]
f	rowOffset	int
f	colOffset	int
f	numRowsToDraw	int
f	numColsToDraw	int
m	loadTiles(String)	void
m	loadMap(String)	void
m	getTileSize()	int
m	getX()	double
m	getY()	double
m	getWidth()	int
m	getHeight()	int
m	getType(int, int)	int
m	setTween(double)	void
m	setPosition(double, double)	void
m	fixBounds()	void
m	draw(Graphics2D)	void

Background		
f	image	BufferedImage
f	diff	int
f	x	double
f	y	double
f	dx	double
f	dy	double
f	moveScale	double
m	setPosition(double, double)	void
m	setVector(double, double)	void
m	update()	void
m	draw(Graphics2D)	void

Tile		
f	image	BufferedImage
f	type	int
f	NORMAL	int
f	BLOCKED	int
m	getImage()	BufferedImage
m	getType()	int

### 2.2.1. Klasa 1- AudioPlayer

---

Ova klasa služi da učitavamo veće zvučne zapise koa što je pozadinska glazba i da tako imamo referencu na pozadinsku glazbu određene komponente, koju onda možemo pauzirati ili namještati glasnoću i sl.

#### 2.2.1.1. Atributi

---

**Private Media media** – učitana multimedijaska datoteka

**Public MediaPlayer mediaPlayer** – upravljač glazbe

#### 2.2.1.2. Metode

---

**public void play()** – svira učitanu glazbu od početka

**public void playLoop()** – svira učitanu glazbu i prilikom kraja ju ponovno pokrene

**public void stop()** – zaustavlja učitanu glazbu

### 2.2.2 Klasa 2 - ClipPlayer

---

Ima sličnu funkciju kao AudioPlayer samo za kraće zvučne zapise koji se često pokreću tijekom prijelaza mišem na tipku ili zvučne efekte u igrici

#### 2.2.2.1. Atributi klase 2

---

**Private AudioClip clip** – učitani zvučni zapis

#### 2.2.2.2. Metode klase 2

---

**Public void play()** – svira učitani zvučni zapis, ovi zvukovi se mogu nesmetano paralelno puštati

### 2.2.3 Klasa 3 - AdministracijaKontroler

---

Ova klasa upravlja korisničkim sučeljem za administratore, omogućava dohvaćanje potrebnih podataka iz baze i izmjenjivanje potrebnih prozora primjerice za login ili pokretanje igre

#### 2.2.3.1. Atributi klase 3

---

**TableView usersTable** – referenca na FXML tablicu korisnika

**TableColumn idCol** – referenca na FXML stupac u kojem je zapisan korisnikov id

**TableColumn usernameCol** – referenca na FXML stupac u kojem je zapisano korisničko ime

**TableColumn roleCol** – referenca na FXML stupac u kojem je zapisana korisnikova uloga

**TableColumn level1\_score** – referenca na FXML stupac u kojem je zapisan korisnikov najveći rezultat iz prve razine

**TableColumn level2\_score** – referenca na FXML stupac u kojem je zapisan korisnikov najveći rezultat iz druge razine

**TableColumn level3\_score** – referenca na FXML stupac u kojem je zapisan korisnikov najveći rezultat iz treće razine

**TableColumn resetCol** – referenca na FXML stupac u za resetiranje korisnikovih podataka

**TableColumn promoteCol** – referenca na FXML stupac u za unaprjeđivanje korisnika na admina

**TableColumn deleteCol** – referenca na FXML stupac u za brisanje korisnika

### 2.2.3.2. Metode klase 3

---

**Public void initialize(URL url, ResourceBundle rb)** – poziva se kada je objekt instanciran kako bi se dohvatili podaci iz baze i tablica popunila s tim podacima

**Public void updateTable()** – koristi se svaki put kada naknadno trebamo ažurirati podatke u tablici koja je već učitana

**Public void openLogin(ActionEvent e)** – funkcija za otvaranje logina prilikom zatvaranja administracijskog panela

**Public void launchGame(ActionEvent e)** – funkcija za pokretanje igrice iz administracijskog panela, budući da se adminima prvo otvori administracijski prozor nakon logina

**Private void letterbox(final Scene scene)** – funkcija koja omogućava da prilikom promjene veličine prozora skaliramo sadržaj prikladno

## 2.2.4 Klasa 4 - LoginController

---

Upravlja podacima i mijenjanjem prozora za login sučelje

### 2.2.4.1. Atributi klase 4

---

**Label statusLbl** – FXML referenca na dio teksta koji naznačava stanje podataka logina

**TextField usernameTxt** – FXML referenca na input za ime

**PasswordField passwordTxt** – FXML referenca na input za šifru

**Public static LogiraniKorisnikModel currentUser** – referenca na trenutnog logiranog korisnika, kako bi ti podaci bili pristupačni bilo kojoj drugoj klasi

**Public static class SceneSizeChangeListener()** – klasa specifično kreirana da bi ju mogla koristiti letterbox funkcija, ova klasa detektira promjenu veličine prozora i daje ispravan omjer za skaliranje

### 2.2.4.2. Metode klase 4

---

**Public void login(ActionEvent e)** – funkcija za obavljanje radnje prijavljivanja korisnika, u slučaju greške mijenja se text od statusLbl kako bi se korisniku ispisala greška u pitanju

**Public static void updateUser()** – funkcija koja se poziva kada trebamo najnovije podatke za trenutnog korisnika ažurirati, prilikom promjene neke vrijednosti i slično

**Public void openRegister()** – funkcija za otvaranje register prozora prilikom klika na tipku Register

**Private void letterbox()** – funkcija za prilagodljivo skaliranje sadržaja na promjenjivom prozoru

**Public void initialize(URL url, ResourceBundle rb)** – funkcija za inicijalizaciju, nije implementirano u konkretnoj klasi

## 2.2.5 Klasa 5 - RegisterController

---

Klasa za upravljanje sučeljem registracije, uključujući provjeru ispravnosti podataka, i funkcije za izmjenu prozora

### 2.2.5.1. Atributi klase 5

---

**Label statusLbl** – FXML referenca na dio teksta koji naznačava stanje podataka registracije

**TextField usernameTxt** – FXML referenca na input za ime

**PasswordField passwordTxt** – FXML referenca na input za šifru

**PasswordField repasswordTxt** – FXML referenca na input za potvrdu šifre

**Public static class SceneSizeChangeListener()** – klasa specifično kreirana da bi ju mogla koristiti letterbox funkcija, ova klasa detektira promjenu veličine prozora i daje ispravan omjer za skaliranje

### 2.2.5.2. Metode klase 5

---

**Public void register(ActionEvent e)** – funkcija za obavljanje radnje registriranja korisnika, u slučaju greške ili zauzetosti imena mijenja se text od statusLbl kako bi se korisniku ispisala greška u pitanju

**Private void letterbox()** – funkcija za prilagodljivo skaliranje sadržaja na promjenjivom prozoru

**Public void initialize(URL url, ResourceBundle rb)** – funkcija za inicijalizaciju, nije implementirano u konkretnoj klasi

## 2.2.6 Klasa 6 - RedNinja

---

Trenutno jedina klasa paketa Enemies u koje spadaju svi protivnički entiteti igraču. RedNinja je jednostavan u smislu da se samo kreće lijevo-desno uz promjene smjera kada naiđe na zid. U slučaju kontakta s igračem oduzima 1 život

### 2.2.6.1. Atributi klase 6

---

**Private BufferedImage[] sprites** – skup sličica za animaciju RedNinja objekata

### 2.2.6.2. Metode klase 6

---

**Private void getNextPosition()** – dohvaća sljedeću poziciju objekta, na osnovu smjera kretanja i jeli pada

**Public void update()** – ažurira objekt kako bi promjene bile vidljive, uključujući animaciju i poziciju

**Public void draw(Graphics2D g)** – prikazuje RedNinja objekta na GamePanel

## 2.2.7 Klasa 7 - Animation

---

Ova klasa sadrži svu logiku za izmjenjivanja slike sličica nekog objekta, sa mogućnošću postavljanja koji je trenutni niz sličica i brzina promjene

### 2.2.7.1. Atributi klase 7

---

**Private BufferedImage[] frames** – sličice koje treba animirati

**Private int currentFrame** – index trenutne sličice

**Private long startTime** – služi za označavanje početne točke prilikom mjerenja proteklog vremena

**Private long delay** – količina vremena koja će se čekati između svake sličice

**Private boolean playedOnce** – vrijednost koja naznačava ako je animacija barem jedno odrađena do kraja

### 2.2.7.2. Metode klase 7

**Public void setFrames(BufferedImage[] frames)** – postavlja sličice animacije

**Public void setDelay(long d)** – postavlja vremenski razmak sličica

**Public void setFrame(int i)** – postavlja index trenutne sličice

**Public void update()** – ažurira vrijednosti animacije

**Public int getFrame()** – vraća index trenutne sličice (int)

**Public boolean hasPlayedOnce()** – vraća vrijednost od playedOnce (boolean)

## 2.2.8 Klasa 8 - Enemy

*Klasa koja opisuje zajedničke generalne funkcionalnosti neprijateljevskih objekata*

### 2.2.8.1. Atributi klase 8

**Protected int health** – količina udarnih poena koje objekt može primiti dok ne umre

**Protected int maxHealth** – maksimalan broj udarnih poena

**Protected boolean dead** – vrijednost koja govori jeli objekt mrtav (true ako jeste)

**Protected int damage** – količina udarnih poena koje bi objekt nanio igraču u slučaju udarca

**Protected boolean flinching** – vrijednost naznačava jeli objekt nedavno udaren pa ima određeno vrijeme oporavka dok ne mogne opet biti udaren

**Protected long flinchTimer** – vremenski period nakon kojeg objekt može opet primiti udarce

### 2.2.8.2. Metode klase 8

**Public boolean isDead()** – vraća istinosnu vrijednost ako je objekt mrtav, inače false

**Public int getDamage()** – vraća količinu udarnih poena koje nanosi ovaj objekt

**Public void hit(int damage)** – funkcija koja ažurira relevantne vrijednosti prilikom primanja udarca

**Public void update()** – funkcija za ažuriranje

## 2.2.9 Klasa 9 - Explosion

*Klasa za vizualni efekt nestajanja protivnika nakon smrti*

### 2.2.9.1. Atributi klase 9

**Private int x** – x pozicija eksplozije

**Private int y** – y pozicija eksplozije

**Private int xmap** – x pozicija mape trenutnog nivoa

**Private int ymap** – y pozicija mape trenutnog nivoa

**Private Animation animation** – animacija objekta

**Private BufferedImage[] sprites** – sličice za animaciju

**Private boolean remove** – vrijednost koja govori da li treba ukloniti eksploziju

### 2.2.9.2. Metode klase 9

**Public void update()** – ažurira vrijednosti

**Public void setMapPosition(int x, int y)** – postavlja lokalnu poziciju mape

**Public void draw(Graphics2D g)** – prikazuje eksploziju na ekran

## 2.2.10 Klasa 10 - HUD

**Klasa za prikaz korisničkog sučelja u igri, sadrži broj života i shurikena raspoloživih igraču**

#### **2.2.10.1. Atributi klase 10**

**Private Player player** – referenca na igrača

**Private BufferedImage image** – slika korisničkog sučelja

**Private Font font** – font teksta sučelja

#### **2.2.10.2. Metode klase 10**

**Public void draw()** - prikaz sučelja na ekran

### **2.2.11 Klasa 11 - MapObject**

**Superklasa za skoro sve objekte koji se pojavljuju u aktivnoj igrici, sadrži sve osnovne podatke potrebne da bi se odredila pozicija, okvir za koliziju, dimenzije i kretnja objekata**

#### **2.2.11.1. Atributi klase 10**

**protected TileMap tileMap** - referenca na trenutno korišteni set blokova u nivou

**protected int tileSize** – veličina blokova

**protected double xmap** – x pozicija mape

**protected double ymap** – y pozicija mape

**protected double x** – x pozicija objekta

**protected double y** – y pozicija objekta

**protected double dx** – promjena x pozicije

**protected double dy** – promjena y pozicije

**protected int width** - širina

**protected int height** - visina

**protected int cwidth** – širina za koliziju

**protected int cheight** – visina za koliziju

**protected int currRow** – trenutni redak

**protected int currCol** – trenutni stupac

**protected double xdest** – destinacija x pozicije za iduću iteraciju

**protected double ydest** – destinacija y pozicije za iduću iteraciju

**protected double xtemp** – privremena x pozicija

**protected double ytemp** – privremena y pozicija

**protected boolean topLeft** – vrijednost koja govori jeli gornji lijevi kut u koliziji

**protected boolean topRight** – vrijednost koja govori jeli gornji desni kut u koliziji

**protected boolean bottomLeft** – vrijednost koja govori jeli donji lijevi kut u koliziji

**protected boolean bottomRight** – vrijednost koja govori jeli donji desni kut u koliziji

**protected Animation animation** – animacija objekta

**protected int currentAction** – trenutna radnja objekta

**protected int previousAction** – prethodna radnja objekta

**protected boolean facingRight** – vrijednost koja govori jeli objekt trenutno gleda desno

**protected boolean left** – vrijednost govori jeli se objekt kreće lijevo

**protected boolean right** – vrijednost govori jeli se objekt kreće desno

**protected boolean up** – vrijednost govori jeli se objekt reće gore

**protected boolean down** – vrijednost govori jeli se objekt kreće dole

**protected boolean jumping** – vrijednost true ako trenutno skače  
**protected boolean falling** – vrijednost true ako trenutno pada  
**protected double moveSpeed** – brzina ubrzavanja  
**protected double maxSpeed** – maksimalna brzina kretanja  
**protected double stopSpeed** – brzina zaustavljanja  
**protected double fallSpeed** – brzina padanja  
**protected double maxFallSpeed** – maksimalna brzina padanja  
**protected double jumpStart** – početna brzina skoka  
**protected double stopJumpSpeed** – brzina usporavanja skoka

#### **2.2.11.2. Metode klase 11**

---

**public boolean intersects(MapObject o)** – provjera jeli se preklapa ovaj objekt sa proslijeđenim  
**public Rectangle getRectangle()** – dobiva okvirni pravokutnik objekta  
**public void calculateCorners(double x, double y)** – računa pozicije kuteva objekta  
**public void checkTileMapCollision()** – provjerava koliziju sa mapom na osnovu kuteva  
**public int getX()** – getter za x  
**public int getY()** – getter za y  
**public int getWidth()** – getter za x širinu  
**public int getHeight()** – getter za x visinu  
**public int getCWidth()** – getter za x kolizijsku širinu  
**public int getCHeight()** – getter za x kolizijsku visinu  
**public void setPosition(double x, double y)** – setter za poziciju  
**public void setVector(double dx, double dy)** – setter za vektor kretanja  
**public void setMapPosition()** – setter za lokalnu poziciju mape  
**public void setLeft(boolean b)** – setter za left  
**public void setRight(boolean b)** – setter za right  
**public void setUp(boolean b)** – setter za up  
**public void setDown(boolean b)** – setter za down  
**public void setJumping(boolean b)** – setter za jumping  
**public boolean notOnScreen()** – provjera jeli objekt van ekrana, vraća true ako jest  
**public void draw(java.awt.Graphics2D g)** – prikazuje objekt na ekran

#### **2.2.12 Klasa 12 - Player**

---

*Klasa za igrača, sadrži sve potrebne attribute i funkcije da bi se igrač mogao kretati, napadati, primati udarce i slično*

##### **2.2.12.1. Atributi klase 12**

---

**private int health** – broj života  
**private int maxHealth** – maksimalan broj života  
**private int shurikenCount** – broj shurikena  
**private int maxShurikens** – maksimalan broj shurikena  
**private boolean dead** – provjera jeli igrač mrtav  
**private boolean flinching** – provjera jeli igrač nedavno udaren  
**private long flinchTimer** – vrijeme dok ne može opet primiti udarac  
**private boolean throwing** – provjera jeli trenutno baca shuriken  
**private int throwCost** – koštanje jednog bacanja  
**private int shurikenDamage** – šteta od jednog projektila  
**private ArrayList<Shuriken> shurikens** – niz šurikena  
**private boolean slicing** – provjera jeli trenutno siječe mačem  
**private int sliceDamage** – šteta od strane mača



**private int sliceRange** – domet mača  
**private boolean gliding** – provjera jeli se igrač polako spušta  
**private float glideSpeed** – brzina polakog spuštanja  
**private boolean diving** – provjera jeli se igrač brzo spušta  
**private float diveSpeed** – brzina brzog spuštanja  
**private float maxDiveSpeed** – maksimalna brzina brzog spuštanja  
**private ArrayList<BufferedImage> sprites** – slike za animaciju  
**private final int[] numFrames** – broj frameova za određenu animaciju  
**private static final int IDLE** – indeks za stojeću animaciju  
**private static final int WALKING** – indeks za hodajuću animaciju  
**private static final int JUMPING** – indeks za skačuću animaciju  
**private static final int FALLING** – indeks za padajuću animaciju  
**private static final int GLIDING** – indeks za sporo padajuću animaciju  
**private static final int THROWING** – indeks za bacajuću animaciju  
**private static final int SLICING** – indeks za mašuću animaciju  
**private static final int DIVING** – indeks za brzo padajuću animaciju  
**private Random r = new Random()** – pseudorandom generator za zvukove  
**private HashMap<String, ClipPlayer> sfx** – niz zvučnih efekata

---

#### 2.2.12.2. Metode klase 12

**public int getHealth()** – getter za živote  
**public int getMaxHealth()** – getter za maksimalne živote  
**public int getShurikenCount()** – getter za broj shurikena  
**public int getMaxShurikens()** – getter za maksimalan broj shurikena  
**public boolean isDead()** – getter za dead  
**public void setThrowing()** – setter za throwing  
**public void setSlicing()** – setter za slicing  
**public void setGliding(boolean b)** – setter za gliding  
**public void setDiving(boolean b)** – setter za diving  
**public void checkAttack(ArrayList<Enemy> enemies)** provjera za napad sa svim neprijateljima igrača  
**public void hit(int damage)** funkcija za oduzimanje života  
**private void getNextPosition()** – getter za iduću poziciju  
**public void update()** – ažuriranje vrijednosti  
**public void draw(Graphics2D g)** – crtanje na ekran

#### 2.2.13 Klasa 13 - Shuriken

*Klasa za igračeve projekte – shurikene, koje maksimalno može imati 3 u bilo kojem trenutku, a veoma su korisne za napadanje neprijatelja na velikoj udaljenosti*

---

##### 2.2.13.1. Atributi klase 13

**private boolean hit** – provjera jeli shuriken udario  
**private boolean remove** – provjera treba li shuriken ukloniti  
**private BufferedImage[] sprites** – slike za animaciju  
**private BufferedImage[] hitSprites** – slike za udrnu animaciju  
**private ClipPlayer hitSteel = new ClipPlayer("/resources/SFX/hitSteel.mp3")** – zvuk udara od običan blok a ne od drugi objekt

---

##### 2.2.13.2. Metode klase 13

**public void setHit()** – setter za hit

**public boolean shouldRemove()** – provjera treba li ukoniti shuriken  
**public void update()** – ažuriranje vrijednost  
**public void draw(Graphics2D g)** – crtanje na ekran

#### 2.2.14 Klasa 14 - GamePanel

*Glavni prozor 2D platformer igrice, zapravo radi na Swing nitima za razliku od početnog dijela aplikacije koji je u javaFX nitima, ova klasa upravlja općim postavkama ekrana za igranje i izmjene prozora*

##### 2.2.14.1. Atributi klase 14

**public static final int WIDTH** = 420 – širina prozora  
**public static final int HEIGHT** = 240 – visina prozora  
**public static float SCALEW** = 2 – skaliranje širine  
**public static float SCALEH** = 2 – skaliranje visine  
**public static Dimension screenSize** – veličinan ekrana  
**private Thread thread** – nit za pokretanje  
**private boolean running** – provjera jeli igrica radi  
**private int FPS** = 60 – broj sličica u sekundi  
**private long targetTime** = 1000 / FPS – ciljano vrijeme između sličica  
**private BufferedImage image** – učitana slika  
**private Graphics2D g** – 2D grafika koja će se koristiti za crtanje  
**private GameStateManager gsm** – upravljač stanja igre

##### 2.2.14.2. Metode klase 14

**public void addNotify()** – funkcija za postavljanje listenera tipkovnice i miša na trenutnu nit  
**private void init()** – inicijalizacija tributa  
**public void run()** – glavna funkcija za izvršavanje  
**private void update()** – ažuriranje vrijednosti  
**private void draw()** – crtanje grafike  
**private void drawToScreen()** – prikazivanje grafike na ekran  
**public void keyTyped(KeyEvent key)** – detektira koja tipka otipkana  
**public void keyPressed(KeyEvent key)** – detektira koja je tipka stisnuta  
**public void keyReleased(KeyEvent key)** – detektira koje tipka otpuštena  
**public void mouseClicked(MouseEvent mouseEvent)** – detektira koja je tipka miša kliknuta  
**public void mousePressed(MouseEvent mouseEvent)** – detektira koja je tipka mišra stisnuta  
**public void mouseReleased(MouseEvent mouseEvent)** – detektira koja je tipka miša puštena  
**public void mouseEntered(MouseEvent mouseEvent)** – detektira kada miš uđe  
**public void mouseExited(MouseEvent mouseEvent)** – detektira kada miš izađe

#### 2.2.15 Klasa 15 - GameState

*Apstraktna klasa koja samo služi kao uputa što svako stanje igre mora implementirati*

##### 2.2.15.1. Atributi klase 15

**Protected GameStateManager gsm** – referenca na zajednički upravljač stanjima igre

#### **2.2.15.2. Metode klase 15**

**public abstract void init()** – inicijalizacija atributa  
**public abstract void update()** – ažuriranje vrijednosti  
**public abstract void draw(java.awt.Graphics2D g)** – crtanje grafike  
**public abstract void resetPlayerKeys()** – resetiranje igračevih inputa  
**public abstract void keyPressed(int k)** – provjera koja je tipka stisnuta  
**public abstract void keyReleased(int k)** – provjera koja je puštena  
**public abstract void mousePressed(int m)** – provjera koja je tipka miša stisnuta

#### **2.2.16 Klasa 16 - GameStateManager**

*Klasa za upravljanje stanjima igre, od kojih su sve izvedene iz zajedničke klase GameState*

##### **2.2.16.1. Atributi klase 16**

**private GameState[] gameStates** – popis stanja igre  
**private int currentState** – indeks trenutnog aktivnog stanja  
**private int previousState** – indeks prethodnog stanja  
**private static ClipPlayer menuHover** = new ClipPlayer("/resources/SFX/menu-hover.mp3") – zvuk za biranje na meniju  
**public static final int NUMGAMESTATES** = 7 – broj stanja  
**public static final int MENUSTATE** = 0 – stanje menija  
**public static final int PAUSESTATE** = 1 – stanje pauze  
**public static final int LEVEL1STATE** = 2 – stanje za prvi nivo  
**public static final int LEVEL2STATE** = 3 – stanje za prvi nivo  
**public static final int LEVEL3STATE** = 4 – stanje za prvi nivo  
**public static final int GAMEOVERSTATE** = 5 – stanje za izgubljen nivo  
**public static final int WINSTATE** = 6 – stanje za pobijeden nivo  
**public static int XP** – broj skupljenih experience poena tijekom trenutnog stanja  
**public static int score** – skupljeni rezultat tijekom trenutnog stanja

##### **2.2.16.2. Metode klase 16**

**private void loadState(int state)** – učitaj novo stanje na indeks state  
**private void unloadState(int state)** – izbriši stanje na indeksu state  
**public void setState(int state)** – postavi indeks stanja  
**public int getCurrentState()** – getter za currentState  
**public int getPreviousState()** – getter za previousState  
**public void update()** – ažuriranje vrijednosti  
**public void draw(java.awt.Graphics2D g)** – crtanje grafike  
**public void pauseState()** – pauziranje stanja  
**public void unPauseState()** – odpauziranje stanja  
**public void keyPressed(int k)** – provjera stisnute tipke  
**public void keyReleased(int k)** – provjera otpuštene tipke  
**public void mousePressed(int m)** – provjera pritisnute tipke miša

#### **2.2.17 Klasa 17 - MenuState**

*Menu state je stanje kada je igrač započeo igru ali nije još odabrao level, pa mu se ponude 3 moguće razine da izabere ili da se vrati na glavni izbornik*

#### **2.2.17.1. Atributi klase 17**

**private Background bg** – pozadina menija  
**private static ClipPlayer menuHover** = new ClipPlayer("/resources/SFX/menu-hover.mp3") – zvuk mijenjanja odabira u meniju  
**private int currentChoice** – indeks trenutnog izbora  
**private String[] options** – moguće opcije u meniju (nivoi i izlaz)  
**private Color titleColor** – boja naslova  
**private Font titleFont** – font naslova  
**private Font font** – font opcija

#### **2.2.17.2. Metode klase 17**

Sve metode osim select su već prethodno objašnjene u GameState

**public void init()**  
**public void update()**  
**public void draw(Graphics2D g)**  
**private void select()** – odabir trenutne opcije  
**public void resetPlayerKeys()**  
**public void keyPressed(int k)**  
**public void keyReleased(int k)**  
**public void mousePressed(int m)**

#### **2.2.18 Klasa 18 - PauseState**

*Ova klasa je veoma slična po funkciji sa menu stanjem, samo što se ova javlja kad je već pokrenut nivo i onda nudi izlaz ili odabir drugog nivoa*

##### **2.2.18.1. Atributi klase 18**

Atributi isti kao i za MenuState

##### **2.2.18.2. Metode klase 18**

Metode iste kao i za MenuState

#### **2.2.19 Klasa 19 - GameOverState**

*Ovo stanje se javlja kada igrač izgubi sve živote u određenom levelu, u bazu se sprema pokušaj, ali rezultat i experience poeni se ne razmatraju za njegov profil već su odbačeni. Nudi mu se opcija ponovnog pokušaja, odabir drugog nivoa ili izlaz*

##### **2.2.19.1. Atributi klase 19**

Atributi isti kao i za MenuState, jedini novi je  
**private static ClipPlayer gameoverSound** – zvuk koji se pusti pri inicijalizaciji objekta, koji naglašava gubitak nivoa

##### **2.2.19.2. Metode klase 19**

Metode iste kao za MenuState

#### **2.2.20 Klasa 20 - WinState**

*Stanje koje se javlja kada igrač pređe nivo, odnosno dođe do zadnje checkpointa u tom nivou, prilikom čega se svi skupljeni experience poeni i rezultat spremaju na njegov profil kao nagrada, u ovom stanju mu se ponudi da ponovo igra ako želi, da odabere drugi nivo ili izađe*

### **2.2.20.1. Atributi klase 20**

Atributi isti kao za GameState, jedini novi je private static ClipPlayer winSound – zvuk koji se pusti pri inicijalizaciji objekta, koji naglašava pobjedu nivoa

### **2.2.20.2. Metode klase 20**

Metode iste kao za GameState

## **2.2.21 Klasa 21 – Level1State**

*Stanje prvog nivoa igrice, ova klasa instancira sve unaprijed određene potrebne objekte, mapu, igrača itd. Također dohvati iz baze kolika je nagrada za prijeđeni nivo i za svakog poraženog neprijatelja. Ovo je jedno od stanja koje se može pauzirati*

### **2.2.21.1. Atributi klase 21**

private static TileMap tileMap – mapa nivoa  
private Background bg – pozadina nivoa  
private Player player - igrač  
private ArrayList<Enemy> enemies – popis neprijatelja  
private ArrayList<Explosion> explosions – popis eksplozija  
private Point[] checkpoints – popis cjelina, nakon što igrač prođe određenu cjelinu u nivou i umre, stvoriti će se na zadnjoj prođenoj cjelini umjesto od početka nivoa  
private int lastCheckpoint – indeks posljednje cjeline  
private HUD hud – grafičko sučelje  
public static AudioPlayer bgMusic – pozadinska glazba za prvi nivo  
private LevelModel levelInfo – informacije o nivou, naziv, broj poena  
private Random r = new Random() – nasumični generator za zvučne efekte  
private HashMap<String, ClipPlayer> sfx – popis zvučnih efekata

### **2.2.21.2. Metode klase 21**

Metode iste kao za GameState, nove su  
private void populateEnemies() – popuni popis neprijatelja  
private void getCheckpoints() – popuni popis cjelina  
private void updateCheckpoints() – ažurira cjelina da vidi ako je igrač prešao na novu  
public void checkPlayerAbyss() – provjera jeli igrač upao u provaliju, pri čemu ga treba ponovno stvoriti na zadnjoj prođenoj cjelini

## **2.2.22 Klasa 22 – Level2State**

*Stanje drugog nivoa igrice, funkcionalnosti su identične kao sa prvim nivoem, uz drugačije inicijalne vrijednosti, veće nagrade nego prethodni nivo*

### **2.2.22.1. Atributi klase 22**

Atributi isti kao za Level1State

#### **2.2.22.2. Metode klase 22**

Metode iste kao za Level1State

#### **2.2.23 Klasa 23 – Level3State**

*Stanje trećeg nivoa igrice, funkcionalnosti su identične kao sa prvim nivoem, uz drugačije inicijalne vrijednosti, veće nagrade nego prethodni nivo*

##### **2.2.23.1. Atributi klase 23**

Atributi isti kao za Level1State

##### **2.2.23.2. Metode klase 23**

Metode iste kao za Level1State

#### **2.2.24 Klasa 24 - AdministracijaModel**

*Ova klasa upravlja podacima iz baze vezane za administrativne svrhe, to uključuje popis korisnika i mogućnost mijenjanja istog kao najbitniju funkcionalnost, obuhvaća u svojim atributima podatke koji se fobivaju iz tablice za jednog korisnika*

##### **2.2.24.1. Atributi klase 24**

**SimpleIntegerProperty id** – id korisnika

**SimpleStringProperty username** – korisnicko ime

**SimpleStringProperty role** - uloga

**SimpleIntegerProperty level1Score** – najveći rezultat postignut na prvom nivou

**SimpleIntegerProperty level2Score** – najveći rezultat postignut na drugom nivou

**SimpleIntegerProperty level3Score** – najveći rezultat postignut na trećem nivou

**Button resetBtn** – tipka za resetiranje vrijednosti retka

**Button promoteBtn** – tipka za promoviranje korisnika u tom retku na admina, adminima je isključena ova tipka za korisnike koji su također admini

**Button deleteBtn** – tipka za brisanje korisnika, također isključeno ako je korisnik admin

##### **2.2.24.2. Metode klase 24**

**public Integer getId()** – getter za id

**public String getUsername()** – getter za korisnicko ime

**public String getRole()** – getter za ulogu

**public Integer getLevel1Score()** – getter za najveći postignuti rezultat na prvom nivou

**public Integer getLevel2Score()** – getter za najveći postignuti rezultat na drugom nivou

**public Integer getLevel3Score()** – getter za najveći postignuti rezultat na trećem nivou

**public Button getResetBtn()** – getter za tipku resetiranja

**public Button getPromoteBtn()** – getter za tipku promoviranja

**public Button getDeleteBtn()** – getter za tipku brisanja

**public static ObservableList<AdministracijaModel>**

**listaKorisnika(AdministracijaKontroler controller)** – funkcija koja dobavlja sve korisnike iz baze podataka i sprema u podatak tipa observable list koji je popunjen objektima ove klase

### 2.2.25 Klasa 25 - LogiraniKorisnikModel

*Klasa za upravljanje podacima u bazi trenutnog korisnika koji se logira, sadrži sve potrebne metode da se provjeri korisnikov unos i jeli se poklapa s podacima u bazi*

#### 2.2.25.1. Atributi klase 25

**SimpleIntegerProperty id** – id korisnika  
**SimpleStringProperty username** – korisnicko ime  
**SimpleStringProperty password** – lozinka

**SimpleStringProperty role** - uloga  
**SimpleIntegerProperty level1Score** – najveći rezultat postignut na prvom nivou  
**SimpleIntegerProperty level2Score** – najveći rezultat postignut na drugom nivou  
**SimpleIntegerProperty level3Score** – najveći rezultat postignut na trećem nivou  
**SimpleIntegerProperty experience** – ukupno postignuti experience poeni

#### 2.2.25.2. Metode klase 25

**public int getId()** – getter za id  
**public String getUsername()** – getter za korisnicko ime  
**public String getRole()** – getter za ulogu  
**public int getLevel1Score()** - getter za najveći postignuti rezultat na trećem nivou  
**public int getLevel2Score()** – getter za najveći postignuti rezultat na trećem nivou  
**public int getLevel3Score()** – getter za najveći postignuti rezultat na trećem nivou  
**public int getExperience()** – getter za experience poene  
**public static ObservableList<LogiraniKorisnikModel> userList()** – funkcija za dohvaćanje svih korisnika u bazi podataka, vraća rezultat u obliku observable list popunjen s ovom klasom  
**public static LogiraniKorisnikModel currentUser(String username)** – vraća jedan objekt ove klase koji sadrži podatke o trenutnom korisniku, kako bi ti podaci bili dostupni drugim komponentama koje trebaju  
**public static boolean login(String username, String password)** – funkcija koja pokušava logirati korisnika sa upisanim korisničkim imenom i šifrom  
**public String toString()** – ispisuje podatke trenutnog korisnika

### 2.2.26 Klasa 26 - RegistriraniKorisnikModel

*Ova klasa sadrži potrebnu funkcionalnost da bi se korisnik pokušao registrirati u bazu, pri čemu vraća drugu vrijednost od nule ukoliko je došlo do greške*

#### 2.2.26.1. Atributi klase 26

Ova klasa nema atributa

#### 2.2.26.2. Metode klase 26

**public static int register(String username, String password)** – ova metoda za proslijeđenu vrijednost korisničkog imena i šifre prvo pokušava utvrditi ako je to ime zauzeto, ukoliko nije pokušava ga spremiti u bazu, ako je ime zauzeto vraća 1, ako dođe do SQL iznimke vraća 2, inače vraća 0 ako je uspješno upisan u bazu

### 2.2.20 Klasa 27 - HighscoreListModel

*Ova klasa služi da dobavlja podatke o najvećim postignutim rezultatima, grupirano po nivoima, kako bi se moglo prikazati u NinjaMenuHSList klasi*

#### 2.2.27.1. Atributi klase 27

**SimpleIntegerProperty position** – pozicija na ljestvici rezultata

**SimpleStringProperty username** – korisnicko ime

**SimpleIntegerProperty highscore** - rezultat

#### 2.2.27.2. Metode klase 27

**public int getPosition()** – getter za poziciju

**public String getUsername()** getter za ime

**public Integer getHighscore()** getter za rezultat

**public static ObservableList<HighscoreListModel> highscoreList(Integer level)** –

funkcija koja dobavlja prvih 10 ljudi iz baze s najvećim rezultatom na proslijeđenom nivou level, i vraća u obliku observable liste popunjene s ovom klasom

**public String toString()** – ispisuje podatke o poziciji i rezultatu korisnika

### 2.2.28 Klasa 28 - NinjaProfileModel

*Ova klasa ima sve ugrađene funkcije da se za trenutnog korisnika mogu dobavljati podaci o njegovom profilu, te provjere prilikom pokušaja ažuriranja vrijednosti korisničkog imena i lozinke jesu li ispravni*

#### 2.2.28.1. Atributi klase 28

Ova klasa nema atributa

#### 2.2.28.2. Metode klase 28

**public static int updateUsername(String oldname, String newname)** – pokušaj ažurirati korisnikovo ime na temelju proslijeđenih vrijednosti, ukoliko je novo ime zauzeto vraća 1, ako dođe do iznimke vraća 2 a inače 0

**public static int updatePassword(String username, String oldpass, String newpass)** – pokušaj ažurirati korisnikovu lozinku na temelju proslijeđenih vrijednosti, ukoliko se stare lozinke ne poklapaju vraća 1, ako dođe do iznimke vraća 2 a inače 0

**public static void levelFinish(int level, int XP, int score)** – funkcija za kraj određenog nivoa, i nagrađivanje korisnika sa experience poenima i rezultatom koji je postigao. Poeni se upisuju u bazi za nivo koji je prosijeden

**public static void levelAttempt(int level,int status, int score)** – funkcija za zapisivanje pokušaja prelaska nekog nivoa, ovi podaci se uvijek upisuju dok se podaci za levelFinish funkciju upisuju samo ako je uspješno završen nivo

### 2.2.29 Klasa 29 - LevelModel

*Ova klasa služi da bi klase nivoa Level1State, Level2State, i Level3State mogle dobiti iz baze informacije o količini poena za nagrađivanje igrača.*

#### 2.2.29.1. Atributi klase 29

**SimpleIntegerProperty id** – id nivoa

**SimpleStringProperty name** – naziv nivoa (ne nužno samo „Level 1“ i slično)

**SimpleIntegerProperty baseXP** – najmanja moguća experience nagrada za završen nivo



**SimpleIntegerProperty killXP** – experience nagrada za svakog poraženog neprijatelja

**SimpleIntegerProperty baseScore** – najmanji mogući rezultat za završen nivo

**SimpleIntegerProperty killScore** – dodatni rezultat za svakog poraženog neprijatelja

#### **2.2.29.2. Metode klase 29**

---

**public int getId()** . getter za id nivoa

**public String getName()** – getter za naziv nivoa

**public int getBaseXP()** – getter za baseXP

**public int getKillXP()** – getter za killXP

**public int getBaseScore()** – getter za baseScore

**public int getKillScore()** – getter za killScore

**public static LevelModel getLevelInfo(int level\_id)** – funkcija koja dohvaća podatke o nivou s proslijeđenim id i vraća u obliku klase LevelModel

### **2.2.30 Klasa 30 - Konekcija**

---

*Klasa koja sadrži sve potrebne podatke za spajanje na odgovarajuću bazu podataka za ovu aplikaciju, također sadrži funkcije koje koriste biblioteku mysql java connector kako bi odradio funkciju spajanja*

#### **2.2.30.1. Atributi klase 30**

---

**private String host** – poslužitelj baze

**private String korisnik** – korisnik baze

**private String lozinka** – šifra za spajanje

**private String baza** – naziv baze podataka

**protected Connection konekcija** – veza sa određenom bazom podataka, konkretnije s bazom koju smo naveli u prethodnim atributima

#### **2.2.30.2. Metode klase 30**

---

**public void spoji ()** – funkcija za spajanje na predefiniranu bazu

**public void odspoji ()** – funkcija za odspajanje sa spojene baze

### **2.2.31 Klasa 31 - Baza**

---

*Ova klasa će nasljeđivati konekciju tako da već ima ugrađene sve potrebne funkcionalnosti spajanja, dodatno će tu biti neke funkcije za lakše generiranje pripremljenih upita*

#### **2.2.31.1. Atributi klase 31**

---

**private Statement upit** – upit koji želimo izvršiti nad bazom da prikazemo podatke

**private PreparedStatement execUpit** – pripremljeni upit koji se može izvršiti da mijenjamo podatke u bazi

#### **2.2.31.2. Metode klase 31**

---

**public ResultSet select (String sql)** – funkcija za dohvaćanje podataka select upitom, koji se generira na osnovu proslijeđenog stringa, vraća rezultat u obliku ResultSet

**public PreparedStatement prepare(String sql)** – funkcija za izvršavanje upita koji mijenja podatke u bazi na osnovu proslijeđenog stringa, rezultat koji vraća je generirani pripremljeni upit koji se onda može izvršiti

### 2.2.32 Klasa 32 - NinjaMenuApp

*Klasa za upravljanje glavnim izbornikom igre, tu su sadržani svi podelementi izbornika kao i funkcije za izmjenu iz jednog prizora na drugi, mijenjanje postavki, i pokretanje 2D platformer igrice*

#### 2.2.32.1. Atributi klase 32

**public static final int WIDTH** – širina prozora izbornika  
**public static final int HEIGHT** – visina prozora izbornika  
**public static Pane root** – glavni element na kojem se dodaju djeca elementi ovisno o tome koji prizor je trenutno aktivan  
**public static GamePanel gamePanel** – referenca na GamePanel objekt koji će poslužiti za pokretanje igrice  
**public static JFrame gameWindow** – prozor izbornika  
**public static VBox menuBox** – element koji sadrži sve Node elemente za izbornik  
**public static VBox settingsBox** – element koji sadrži sve Node elemente za settings opciju  
**public static VBox tutorialBox** – element koji sadrži sve Node elemente za tutorial opciju  
**public static VBox highscoresBox** – element koji sadrži sve Node elemente za highscores opciju  
**public static VBox profileBox** – element koji sadrži sve Node elemente za profil  
**public static NinjaMenuHSList leaderboard** – tablica najvećih rezultata  
**public static NinjaMenuProfile profileInfo** – informacije o trenutnom profilu  
**public static Line line** – linija koja će biti animirana prilikom pokretanja izbornika  
**static double lineX** – x pozicija linije  
**static double lineY** – y pozicija linije  
**public static boolean isFullScreen** – vrijednost koja označava jeli puni zaslon  
**public static boolean isMusicMute** – vrijednost koja naznačava jeli glazba isključena  
**public static boolean isSFXMute** – vrijednost koja provjerava jesu li zvucni efekti isključeni  
**public static float masterVolume** – glavna glasnoća, sa ovom vrijednošću glasnoće se množe sve ostale (vrijednost od 0 do 1)  
**public static NinjaMenuSlider volumeSlider** – slider element za kontroliranje glavne glasnoće  
**public static ImageView menuBg** – pozadina glavnog izbornika  
**public static ImageView tutorialBg** – pozadina tutorials opcije  
**public static ImageView highscoresBg** – pozadina highscores opcije  
**public static ImageView profileBg** – pozadina profil stranice  
**public static WindowListener exitListener** – listener (prislušivač) za izlaz iz trenutnog prozora tako da možemo reći koje je ponašanje pri takvom gašenju ekrana  
**public static List<Pair<String, Runnable>> menuData** – popis opcija za glavni izbornik  
**public static List<Pair<String, Runnable>> settingsData** – popis opcija za settings

**public static List<Pair<String, Runnable>> tutorialData** – popis opcija za tutorials

**public static List<Pair<String, Runnable>> highscoresData** – popis opcija za highscores

**public static List<Pair<String, Runnable>> profileData** – popis opcija za profil stranicu

#### **2.2.32.2. Metode klase 32**

---

**public Pane createContent()** – funkcija koja kreira potrebne komponente

**public static void addBackground()** – funkcija koja dodaje pozadinu glavnog izbornika

**public static void addTitle()** – dodavanje glavnog naslova

**public static void addLine(double x, double y)** – dodavanje linije

**public static void startAnimation(double lineDelay, double itemSpeed, double diff)** – animacija povlačenje linije

**public static void addMenu(double x, double y)** – funkcija za dodavanje opcija glavnog izbornika

**public static void addSettings(double x, double y)** – funkcija za dodavanje settings opcija

**public static void addTutorial(double x, double y)** – funkcija za dodavanje tutorials opcija

**public static void addHighscores(double x, double y)** – funkcija za dodavanje highscores opcija

**public static void addProfile(double x, double y)** – funkcija za dodavanje profiles opcija

**public static void toggleSettingFullscreen()** – mijenjanje između punog i smanjenog zaslona, primjenjuje se i na 2D platformer igricu

**public static void toggleSettingMusic()** – gasi/pali glazbu, primjenjuje se i na 2D platformer

**public static void toggleSettingSFX()** gasi/pali zvučne efekte, primjenjuje se i na 2D platformer

**public static void removeSubmenus()** – uklanja sve postojeće elemente sa glavnog root elementa, kako bi se mogla ispravni element učitati

### **2.2.33 Klasa 33 - NinjaMenuTitle**

---

*Klasa koja generira glavni naslov izbornika*

#### **2.2.33.1. Atributi klase 33**

---

**Private Text text** – tekst naslova generiran na temelju prosljeđenog stringa

#### **2.2.33.2. Metode klase 33**

---

**Public double getTitleWidth()** – vraća stvarnu širinu naslova

**Public double getTitleHeight()** – vraća stvarnu visinu naslova

### **2.2.34 Klasa 34 - NinjaSubtitle**

---

*Slična klasa klasi NinjaMenuTitle, samo što generira manji tekst ispod glavnog naslova*

#### **2.2.34.1. Atributi klase 34**

---

Atributi isti kao za NinjaMenuTitle

#### **2.2.34.2. Metode klase 34**

Metode iste kao za NinjaMenuTitle

#### **2.2.35 Klasa 35 - NinjaMenuitem**

*Jedan blok koji se može kliknuti da se pokrene određena akcija u glavnom izborniku, sastoji se od teksta i poligona, ima ugrađene funkcije za detektiranje klika, hovera i slično*

##### **2.2.35.1. Atributi klase 35**

**private Text text** – tekst upisan u blok  
**private Effect shadow** – predefinirana sjena bloka i teksta  
**private Effect blur** – predefinirani efekt zamagljenja teksta  
**private Effect bgblur** – predefinitani efekt zamagljenja bloka  
**private static ClipPlayer menuHover** – zvuk prijelaza mšem preko bloka  
**private static List<ClipPlayer> menuClick** – niz zvukova klika na bloka  
**private static Random rand** – nasumični generator za zvuk klikanja

##### **2.2.35.2. Metode klase 35**

**Public void setOnAction (Runnable action)** – postavlja radnju koja će se izvršiti kada je ovaj blok kliknut

#### **2.2.36 Klasa 36 - NinjaMenuHSList**

*Klasa koja učitava popis 10 najboljih rezultata za svaki nivo, koristi ugrađene funkcije klase HighscoreListModel klase*

##### **2.2.36.1. Atributi klase 36**

**private Effect glow** – predefinirani efekt svijetljenja teksta  
**private Effect shadow** – predefinirani efekt sjene  
**private Effect blur** predefinirani efekt zamagljenja  
**private Text mainTitle** – glavni naslov Highscore elementa  
**private Text[] titles** – naslovi za pojedini nivo  
**private ArrayList<ObservableList<HighscoreListModel>> level\_highscore\_list** – izvorni popis rezultata po nivoima  
**private Text[][] text\_highscore\_list** – tekst koji sačinjavaju pozicija i ime korisnika  
**private Text[][] number\_highscore\_list** – tekst koji sačinjava postignuti rezultat korisnika na toj poziciji

##### **2.2.36.2. Metode klase 36**

Ova klasa nema metoda

#### **2.2.37 Klasa 37 - NinjaMenuProfile**

*Klasa koja učitava profil trenutnog logiranog korisnika, pri tome koristi NinjaProfileModel i LoginControler klase da dobiva potrebne podatke*

##### **2.2.37.1. Atributi klase 37**

**private Effect glow** – predefinirani efekt sjaja

**private Effect shadow** – predefinirani efekt sjene  
**private Effect blur** – predefinirani efekt zamagljenje  
**private static Text highscoreTitle** – glavni highscore naslov, nalazi se na gornjem desnom dijelu profilne stranice  
**private static Text[] levelTitles** – naslovi za nivoe, nalaze se ispod highscore naslova  
**private static Text[] scoreText** – tekstovi za rezultate postignuti u nivoima  
**private static Text ninjaName** – naziv igrača  
**private static Text ninjaLevel** – nivo igrača, ovo se računa kao  $\text{Math.floor}(1 + \text{Math.sqrt}(\text{XP})/5)$ , tako da se s vremenom sve teže i teže dolazi do većeg Ninja levela  
**private static Text xp** – ukupni experience poeni igrača  
**private static RingProgressIndicator xpProgress** – postotak do idućeg Ninja levela prikazan kao kružni indikator progres  
**private static Circle xpbg** – pozadina xp progres  
**private static Text usernameText** – statični tekst „Username“  
**private static Text passwordText** – statični tekst „Password“  
**private static Text oldNameText** – tekst koji naznačava trenutno ime korisnika  
**private static Text newNameText** – statični tekst „New“  
**private static Text oldPassText** – statični tekst „Old“  
**private static Text newPassText** – statični tekst „New“  
**private static TextField usernameInput** – input za novo korisničko ime  
**private static PasswordField oldpasswordInput** – input za trenutnu lozinku  
**private static PasswordField newpasswordInput** – input za novu lozinku  
**private static Text usernameError** – dinamični tekst za prikaz greške kod unosa imena  
**private static Text passwordError** – dinamični tekst za prikaz greške unosa šifre  
**private static Button usernameBtn** – tipka za potvrdu ažuriranja imena  
**private static Button passwordBtn** – tipka za potvrdu ažuriranja šifre  
**private static ClipPlayer buttonClick** – zvučni efekt klika tipke  
**public static ImageView ninjaprofile** – ikona nindže

#### **2.2.37.2. Metode klase 37**

---

**public void updateView()** – ažurira trenutne podatke korisničkog profila  
**public void setUsername()** – pokušava promijeniti korisničko ime na upisano u input  
**public void updatePassword()** – pokušava promijeniti lozinku na onu upisanu u input

#### **2.2.38 Klasa 38 - NinjaMenuSlider**

---

*Slider element koji se koristi za podešavanje glavne glasnoće u settings izborniku, ima raspon od 0 do 100*

##### **2.2.38.1. Atributi klase 38**

---

**private Text text** – tekst za ispis ispod slidera  
**private Effect shadow** – predefinirani efekt sjene  
**private Effect blur** – predefinirani efekt zamagljenja teksta  
**private Effect bgblur** – predefinirani efekt zamagljenja pozadine  
**private static ClipPlayer menuHover** – zvučni efekt prijelaza mišom preko slidera  
**public static Slider slider** – JavaFX slider element

### 2.2.38.2. Metode klase 38

**Public void setOnAction(Runnable action)** – postavlja akciju koja se izvršava kad je ovaj element kliknut

## 2.2.39 Klasa 39 - ProgressCircleIndicator

*Element koji učitava i sadrži logiku kružnog učitavača postotka, kojim ćemo prikazivati postotak do idućeg nivoa ninja profila*

### 2.2.39.1. Atributi klase 39

**private static final int INDETERMINATE\_PROGRESS** – konstanta koja naznačava da je neispravna vrijednost postotka unesena

**private ReadOnlyIntegerWrapper progress** – vrijednost postotka

**private ReadOnlyBooleanWrapper indeterminate** – true ili false vrijednost na temelju koje utvrđuje ako je postotak neka neodrediva vrijednost

**private DoubleProperty innerCircleRadius** – unutarnji kružni polumjer

**private static class StyleableProperties** – klasa koja predstavlja css attribute koje možemo programski sređivati

### 2.2.39.2. Metode klase 39

**public int getProgress()** – getter za postotak

**public void setProgress(int progressValue)** – setter za postotak

**public ReadOnlyIntegerProperty progressProperty()** – getter za postotak ali kao vrijednost koja se samo može čitati

**public boolean isIndeterminate()** – provjera jeli postotak neodrediv

**public void makeIndeterminate()** – funkcija koja utvrđuje da je neodrediv postotak

**public ReadOnlyBooleanProperty indeterminateProperty()** – vraća svojstvo neodredivosti samo za čitanje

**private int defaultToHundred(int value)** – pretpostavljena vrijednost na 100

**public final void setInnerCircleRadius(int value)** – postavi unutarnji polumjer

**public final DoubleProperty innerCircleRadiusProperty()** – unutarnji polumjer

**public final double getInnerCircleRadius()** – getter za unutarnji polumjer

**public static List<CssMetaData<? extends Styleable, ?>> getClassCssMetaData()** - popis css metadata za ovu klasu

**public List<CssMetaData<? extends Styleable, ?>> getControlCssMetaData()** – p css metadata za kontrolnu klasu

## 2.2.40 Klasa 40 - RingProgressIndicator

*Klasa koja nasljeđuje kružni indikator tako da bi mogao ga učinit prstenom, uklanjajući unutarni krug izvorne klase*

### 2.2.40.1. Atributi klase 40

Ova klasa nema atributa

### 2.2.40.2. Metode klase 40

**protected Skin<?> createDefaultSkin()** – stvara defaultni stil

**public final void setRingWidth(int value)** – postavlja širinu prstena

**public final DoubleProperty ringWidthProperty()** getter za svojstvo širine prstena

**public final double getRingWidth()** getter za vrijednost širine prstena

**private DoubleProperty ringWidth** – getter za atribut širina prstena  
**public List<CssMetaData<? extends Styleable, ?>> getControlCssMetaData()** – getter za popis css atributa koji se mogu stilizirati

#### **2.2.41 Klasa 41 - RingProgressIndicatorSkin**

*Klasa za definiranje dodatnih efekata i slojeva na prstenasti indikator postotka, dodaje mu određene tekstove i boje na prstene koji olakšavaju vidljivost i poboljšavaju estetiku*

##### **2.2.41.1. Atributi klase 41**

**private final RingProgressIndicator indicator** – referenca na idikator  
**private final Label percentLabel** – tekst za postotak  
**private final Circle innerCircle** – unutarnji krug  
**private final Circle outerCircle** – vanjski krug  
**private final StackPane container** – container za ukupan element  
**private final Arc fillerArc** – luk za popunjavanje unutrašnjosti  
**private final RotateTransition transition** – tranzicija za postepeno mijenjanje izgleda

##### **2.2.41.2. Metode klase 41**

**private void setProgressLabel(int value)** – setter za tekst postotka  
**private void initTransition()** – inicijalizacija tranzicije  
**private void initFillerArc()** – inicijalizacija luka  
**private void initContainer(final RingProgressIndicator indicator)** – inicijalizacija containera  
**private void updateRadii()** – ažuriranje radiusa  
**private void initLabel(int value)** – inicijalizacija tekstualne vrijednosti  
**private void initIndeterminate(boolean newVal)** – inicijalizacije neodređene vrijednosti  
**public RingProgressIndicator getSkinnable()** – vraća RingProgressIndicator na koji se mogu primijeniti stilovi  
**public Node getNode()** – dobavlja trenutni čvor  
**public void dispose()** – odbacuje trenutni čvor

#### **2.2.42 Klasa 42 - Background**

*Pozadina 2D platformera, s ugrađenim funkcijom kretanja pri određenoj fiksnoj brzini i na osnovu igračeve brzine, također se mapira na način da se čini kontinuirana priliko prijelaza rubove slike i prozora, jer se generiraju višestruke slike dok se ne popuni prostor igranja*

##### **2.2.42.1. Atributi klase 42**

**private BufferedImage image** – slika pozadine  
**private int diff** – razlika igračeve i pozadinske pozicije  
**private double x** – x pozicija  
**private double y** – y pozicija  
**private double dx** – x brzina  
**private double dy** – y brzina  
**private double moveScale** – skala za odnos igračevog kretanja naspram pozadinskog (obično vrijednosti između 0 i 1)

#### 2.2.42.2. Metode klase 42

**public void setPosition(double x, double y)** – postavlja poziciju  
**public void setVector(double dx, double dy)** – postavlja brzinu  
**public void update()** – ažurira vrijednosti  
**public void draw(Graphics2D g)** – crta grafiku

#### 2.2.43 Klasa 43 - Tile

*Osnovni blok za izgradnju mape nekog nivoa, u ovom projektu su blokovi veličine 30 sa 30 pixela, i postoje dva tipa, pozadinski i fizikalni (blokovski), sa blokovskima se može događati kolizija*

##### 2.2.43.1. Atributi klase 43

**private BufferedImage image** – slika bloka  
**private int type** – tip bloka  
**public static final int NORMAL** – predefinirani tip pozadinskog bloka, vrijednost 0  
**public static final int BLOCKED** – predefinirani tip fizikalnog bloka, vrijednost 1

##### 2.2.43.2. Metode klase 43

**public BufferedImage getImage()** – getter slike  
**public int getType()** – getter tipa

#### 2.2.44 Klasa 44 - TileMap

*Tilemap predstavlja mapu blokova nekog nivoa, ova klasa učitava set temeljnih blokova i na osnovu tekstualne datoteke mape isčitava vrijednosti blokova koje treba postaviti, a nakoju poziciju određuje po rasporedu brojeva u tekstualnoj datoteci, broj u datoteci govori koji je blok iz skupa u pitanju*

##### 2.2.44.1. Atributi klase 44

**private double x** – x pozicija  
**private double y** – y pozicija  
**private int xmin** – minimalna x pozicija  
**private int ymin** – minimalna y pozicija  
**private int xmax** – maksimalna x pozicija  
**private int ymax** – maksimalna y pozicija  
**private double tween** – omjer „ljepljivosti“ kamere za igračeve pokrete, 1 predstavlja savršeno ljepljenje, a sve između 0 i 1 će glatko pratiti igračeve pokrete  
**private int[][] map** – popis brojeva iščitanih iz mape  
**private int tileSize** – veličina bloka  
**private int numRows** – broj redova  
**private int numCols** – broj stupaca  
**private int width** - širina  
**private int height** - visina  
**private BufferedImage tileset** – slika bloka  
**private int numTilesAcross** – broj blokova poprijeko  
**private Tile[][] tiles** - blokovi  
**private int rowOffset** – pomak retka  
**private int colOffset** – pomak stupca  
**private int numRowsToDraw** – broj redova za nacrtati



**private int numColsToDraw** – broj stupaca za nacrtati

---

#### **2.2.44.2. Metode klase 44**

**public void loadTiles(String s)** – učitava blokove

**public void loadMap(String s)** – učitava mapu

**public int getTileSize()** – getter za veličinu bloka

**public double getX()** – getter za x

**public double getY()** – getter za y

**public int getWidth()** – getter za širinu

**public int getHeight()** – getter za visinu

**public int getType(int row, int col)** – getter za tip bloka na određenom retku i stupcu

**public void setTween(double d)** – setter za tween

**public void setPosition(double x, double y)** – setter za poziciju

**private void fixBounds()** – popravljiva rubne uvijete pozicije mape kada prelazi granice

**public void draw(Graphics2D g)** – crtanje grafike

---

#### **2.2.45 Klasa 45 - GameStart**

*Main klasa, pokreće se prilikom pokretanja projekta, prvi prozor koji otvara je login screen*

---

##### **2.2.45.1. Atributi klase 45**

Ova klasa nema atributa

---

##### **2.2.45.2. Metode klase 45**

**Public static void main(String[] args)** – main program