PANEVROPSKI UNIVERZITET APEIRON, BANJA LUKA
FAKULTET INFORMACIONIH TEHNOLOGIJA

Student: Ismar Ezić 138-18/RITP

Objektno-Orijentisano Programiranje

Brick Breaker

Mentor: Prof. dr Saša Salapura

Banja Luka, januar.2021

Sadržaj

UVOD

Tema ovog seminarskog rada je program zabavnog karaktera, odnosno igrica. Konkretno se radi o tzv. „brick breaker“-u. Ovakva igrica predstavlja slobodno možemo reći veterana gaming svijeta budući da se radi o 2D igrici koja je i predstavnik početka razvoja igrica. Za ovaj program sam koristio javu, a za IDE sam izabrao intellij. Kroz cijeli seminarski rad ću objasniti šta je java, šta je inetllij, zašto je intellij bio moj izbor te sam programski kod igrice.

1. Java

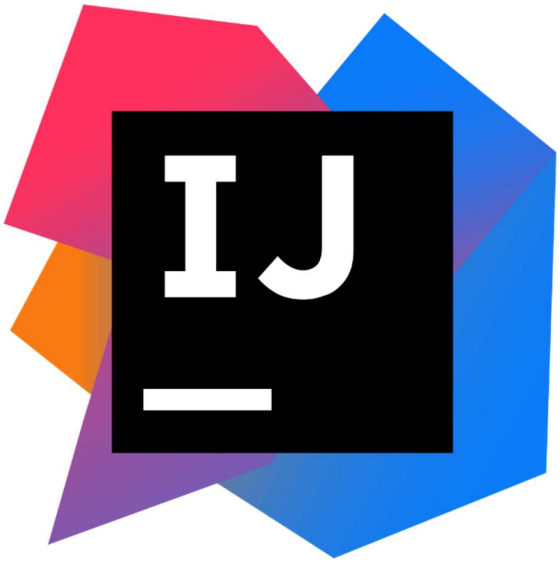
Java predstavlja programski jezik koji je baziran na objektima, zato i spada u grupu objektno-orijentisanih jezika. Naime objeknto-orijentisano programiranje u centar pažnje stavlja objekte. Objekat bi najlakše objašnjeno bila neka osoba, predmet ili nešto drugo što ima svoje karakteristike. Takve karakteristike nazivamo atributi. Svi se objekti smještaju u određene klase odakle sarađuju i komuniciraju. OOP (objektno-orijentisano programiranje) na ovaj način jednu aplikaciju tako reći rastavlja na više malih faktora koji zajedno sarađuju. Ovaj stil programiranja obuhvata primjenu različitih tehnika poput nasljeđivanja, enkapsulacije, polimorfizma i dr. Java je najrasprostranjeniji programski jezik ovoga tipa.

Razvoj programskog jezika Java je započeo 1991. godine, i to kao projekat namijenjen za programiranje kućanskih elektronskih uređaja. Za njegov početak su zaslužni inženjeri iz kompanije Sun Microsystems, među kojima su se istakli James Gosling i Patrick Naughton. Prvobitni naziv mu je bio Ouk (engl. Oak – hrast), te je kasnije preimenovan u Java po istomeinom brendu kafe Java Coffee. Java je „svjetlo dana ugledala“ 1995 godine.

Slika 1 - Java logo

2. IntelliJ IDE

IntelliJ IDEA je integrisano razvojno okruženje (IDE) napisano u javi, koje se koristi za pisanje programskog koda. U januaru 2001. godine je predstavljena prva verzija IntelliJ a iza istog stoji JetBrains. JetBrains je Češka kompanija koja izrađuje alate koji služe za izradu softwera. U 2010 godini InfoWorld pravi izvještaj o testiranju IDE-ova i tu uključuje četiri vodeća Java IDE-a : Eclipse, IntelliJ, NetBeans i JDeveloper. Među četiri najbolja IntelliJ zauzima prvo mjesto, te nakon toga Google 2014. godine najavljuje verziju 1.0 za Android studio koji je open source. Zanimljivo je da se ovaj Android IDE zasniva na IntelliJ IDEA.



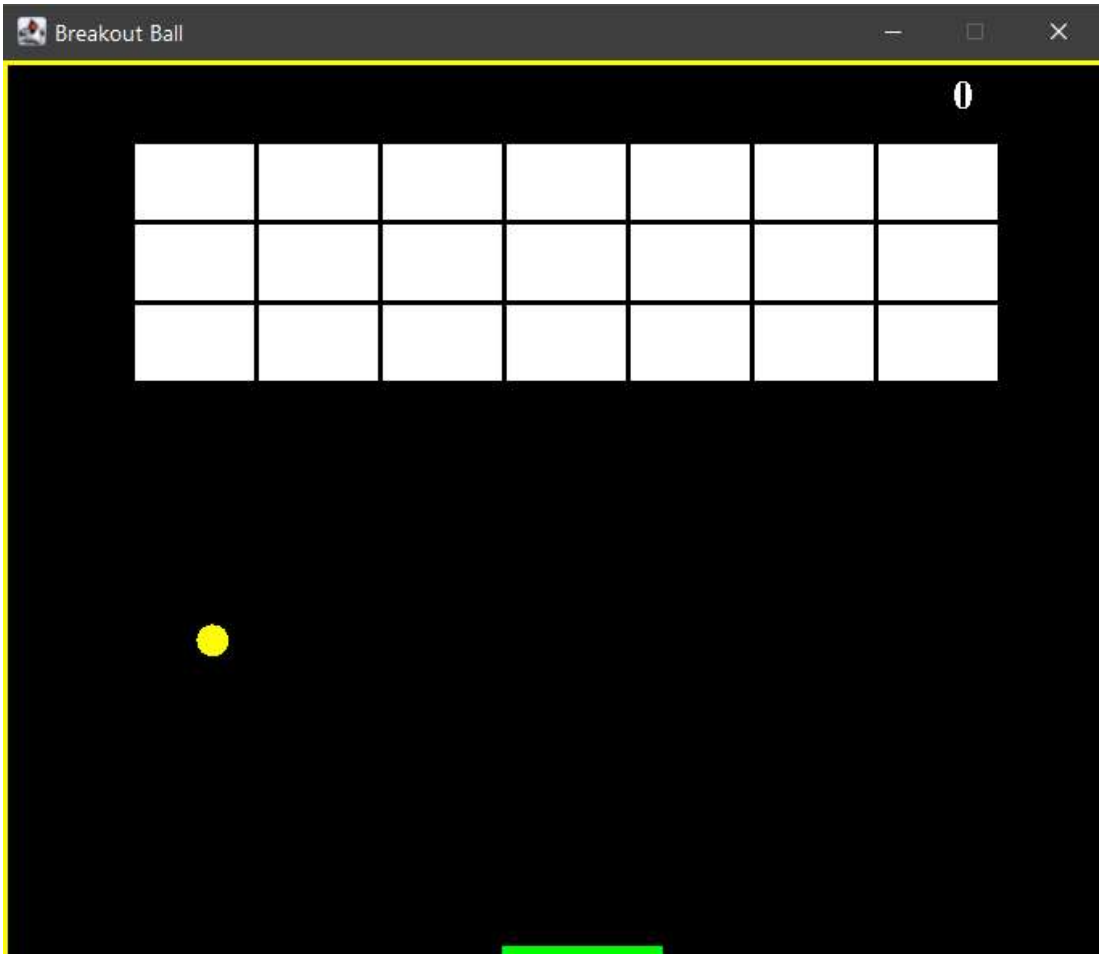
Slika 2 - IntelliJ logo

2.1. Zašto IntelliJ?

S javom sam se susreo prvi put u srednjoj školi i tu smo radili u NetBeans (IDE), međutim NetBeans-ov GUI daje dojam da NetBeans nije „moderan“. To jest u usporedbi sa svime što Eclipse i IntelliJ nude po pitanju pregleda samih komandi, NetBeans pada u drugi plan. Što se tiče Eclipsa dosta je sličan, međutim IntelliJ ima dosta više plugina, bolji debugging, i na kraju sama funkcija autocomplete je dosta bolja. Jedina mana je što IntelliJ je besplatan 30 dana ili 1 godinu ukoliko ste prijavljeni kao student.

3. Brick Breaker

Ideja igre je da postoji zelena podloga koja se kreće horizontalno, igrač bi tu podlogu pomjerao pomoću strelica. Njegov zadatak bi bio da predvidi kretanje loptice, namjesti podlogu, loptica bi se zatim odbila i udarala u blokove. Svaki put kada loptica pogodi blok, blok se briše iz igre a igrač dobija 5 bodova. Igrač gubi ako loptica prođe pored podloge, odnosno igrač će pobijediti ukoliko uništi sve blokove.



Slika 3 - Brick Breaker

3.1. Kod programa

U ovome dijelu seminarskog rada ću predstaviti i objasniti kod same igrice. Ukupan kod sam podijelio na svega tri klase:

1. Main – glavna klasa gdje sam postavio sve vezano za prozor igre
2. Gameplay – klasa gdje se odvija sve vezano za samo igranje igrice
3. Map Generator – klasa koja generiše mapu

3.1.1. Kod Programa – Main

U prvom dijelu smo dodali biblioteku JFrame, a u ostatku smo postavljali određene opcije:

- Postavili veličinu i dužinu (u pikselima)
- Postavili smo naslov prozora
- Postavili smo da je prozor vidljiv
- Postavili smo da se prozoru ne može mijenjati veličina

```
1   package com.company;
2
3   import javax.swing.JFrame;
4
5   public class Main {
6
7       public static void main(String[] args) {
8
9           JFrame obj = new JFrame();
10          obj.setBounds( x: 10,  y: 10,  width: 700,  height: 600);
11          Gameplay gamePlay = new Gameplay();
12          obj.setTitle("Breakout Ball");
13          obj.setResizable(false);
14          obj.setVisible(true);
15          obj.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16          obj.add(gamePlay);
17      }
18  }
19
```

Slika 4 – Main

3.1.2. Kod Programa – Gameplay

U ovome dijelu ću objasniti kod za klasu gameplay. Međutim koda u ovoj klasi ima mnogo, tako da ću objasniti neke dijelove koda uz slike, dok ću neke manje bitne izostaviti.

Ovo je početak klase gdje definišemo najbitnije varijable, postavljamo da je početni score (rezultat) jednak nuli, da je igra false, odnosno da igra nije krenula, postavljamo broj blokova, delay loptice i timer.

Što se tiče pozicija, igrač ima početnu poziciju x, to jest horizontalnu 310.

S druge strane loptica ima pos i dir, odnosno poziciju i smjer kretanja, pozicija x i y je po osi x i y odakle loptica kreće, a smjer je kuda će se kretati.

***Napomena: bitno je napomenuti da je ova igra linearna odnosno loptica se kreće uvijek po istoj „liniji“ budući da su varijable pozicije i smjera određene, što znači da loptica uvijek kreće sa iste pozicije i istim se smjerom kreće. Ovo sam mogao popraviti na način sam stavio random te odredio granice randoma odnosno slučajnog broja. Odredio bih granice samog prozora tako da se loptica pojavi unutar prozora međutim slučajan broj bi označio da se loptica unutar prozora uvijek pojavi na drugoj poziciji.**

```
public class Gameplay extends JPanel implements KeyListener, ActionListener {

    private boolean play= false;
    private int score = 0;

    private int totalBricks = 21;

    private Timer timer;
    private int delay = 8;

    private int playerX=310;

    private int ballposX = 120;
    private int ballposY = 350;
    private int ballXdir = -1;
    private int ballYdir = -2;
```

Slika 5 - Gampelay klasa 1. dio

U ovom dijelu klase gameplay imamo metodu paint koja nam omogućuje editovanje elemenata na garfičkom nivou. Odnosno podešavamo boju pozadinu, zatim imamo žute granice da igrač vidi gdje se loptica odbija, zatim editujemo bodove, podlogu igrača i samu lopticu. Elementima kao što su loptica i igrač postavljamo karakterističan oblik.

Također ova metoda iscrtava mapu.

```
public void paint(Graphics g){
//pozadina
    g.setColor(Color.black);
    g.fillRect( x: 1, y: 1,  width: 692,  height: 592);

    //mapa
    map.draw((Graphics2D) g);

    //bodovi
    g.setColor(Color.white);
    g.setFont(new Font( name: "serif", Font.BOLD,  size: 25));
    g.drawString( str: "" + score,  x: 590,  y: 30);


    //ivice
    g.setColor(Color.yellow);
    g.fillRect( x: 0, y: 0,  width: 3,  height: 592);
    g.fillRect( x: 0, y: 0, width: 692, height: 3);
    g.fillRect( x: 691, y: 0, width: 3, height: 592);

    //igrac
    g.setColor(Color.green);
    g.fillRect(playerX,  y: 550,  width: 100,  height: 8);

    //loptica
    g.setColor(Color.yellow);
    g.fillOval(ballposX, ballposY,  width: 20,  height: 20);
```

Slika 6 - Paint metoda

Sljedeći dio koda se odnosi na „pobijedili ste“ ili „izgubili ste“. To smo uradili pomoću dvije if komande. Ukoliko vrijednost pozicije Y loptice, odnosno visina loptice predje 570, znači da je loptica izašla iz okvira, odnosno loptica je prošla pored platforme igrača, igrač nije odbio loptu uslovno rečeno i izgubio je igru. U tom slučaju igra prestaje, i izlazi crveni tekst „Izgubili ste ...“. S druge strane igrač će pobijediti ukoliko ukupan broj blokova bude 0, tada će igra opet stati ali ovaj se pojavljuje zeleni tekst koji daje poruku igraču da je pobijedio, te mu daje ukupan rezultat.

***Napomena: Postoji niz koji prati koliko je blokova ostalo u igri, na ovaj način program zna kad je broj blokova „pao“ na 0, to jest kad je igrač pobijedio. O nizu koji prati broj blokova u nastavku.**

```
    if(ballposY>570){
        play=false;
        ballXdir=0;
        ballYdir=0;
        g.setColor(Color.red);

        g.setFont(new Font( name: "serif", Font.BOLD,  size: 20));
        g.drawString ( str: "Izgubili ste, vas rezultat: je " + score + ", pritisnite enter za ponovnu igru ",  x: 75,  y: 350);


    }

    if(totalBricks <= 0){
        play=false;
        ballXdir=0;
        ballYdir=0;
        g.setColor(Color.green);

        g.setFont(new Font( name: "serif", Font.BOLD,  size: 30));
        g.drawString ( str: "Pobijediliii ste!! Pritisnite enter za ponovnu igru ",  x: 75,  y: 200);
    }
}
```

Slika 7 - "Pobijedili ste" ili "Izgubili ste"

***Napomena: Što se tiče kretanja loptice svaki put se mapa na novo iscrtava, odnosno u kodu postoji metoda repaint. Ova metoda svaki put iscrtava novu mapu sa novom pozicijom loptice, ovakva metoda nije popularna danas, budući da mnoge igre danas uslovno rečeno ciljaju na „brzinu i precinost“. No, kao što smo rekli radi se o 2D igrici koja ima dosta primitivniju grafiku i fiziku od današnjih igrica.**

Nakon što smo objasnili grafički kako smo napravili elemente, kako program prati tok igre, sada je vrijeme da objasnimo kako smo napravili blokove. Pored toga bitan dio programa koji moramo objasniti je interakcija između loptice i bloka, te šta se dešava sa programom tokom interakcije.

Dio koda gdje definišemo same blokove i njihovu veličinu:

```
Rectangle rect = new Rectangle(brickX, brickY, brickWidth, brickHeight);
Rectangle ballRect = new Rectangle(ballposX, ballposY,  width: 20,  height: 20);
Rectangle brickRect = rect;
```

Slika 8 - Kod blokova

Što se tiče interakcije loptica-blok. Koristimo intersect. Pomoću intersecta stavljamo šta će se desiti ukoliko loptica „dotakne“ blok.

```
if(ballRect.intersects(brickRect)){
    map.setBrickValue( value: 0, i, j);
    totalBricks--;
    score+=5;

    if(ballposX + 19 <= brickRect.x || ballposX + 1 >= brickRect.x + brickRect.width){
        ballXdir = -ballXdir;
    }else{
        ballYdir = -ballYdir;
    }
```

Slika 9 - Intersect loptica-blok

Kao što se sa slike vidi ukoliko loptica intersect-a sa blokom, blok se briše iz niza, ukupan broj blokova se umanjuje za 1, dok se rezultat povećava za 5, jer svaki blok nosi 5 bodova. Također postavljamo da ukoliko loptica udari u blok njen smjer postaje suportan, odnosno loptica se odbija a blok nestaje.

Ovo isto vrijedi i za interakciju loptica-igrač s tim da igrač nije u nikakvom nizu, ne povećava rezultat, već se loptica samo odbija od isti.

U ovoj klasi nam je ostalo još samo da objasnimo kako se igrač kreće.

Za ovo smo koristili keyPressed, na koji način?

Postavljamo da ukoliko igrač pritisne desnu strelicu, pomjeri se desno, a ukoliko se radi o lijevoj strelici onda korisnik ide lijevo.

Konkretno se poziva funkcija moveRight() za desno i moveLeft() za lijevo.

```
public void moveRight(){
    play = true;
    playerX+=20;
}

public void moveLeft(){
    play = true;
    playerX-=20;
}
```

Slika 10 – Move

Ova funkcija konkretno ovisno o „strelici“ pomjera igrača u stranu za 20 pixela.

```
@Override
public void keyPressed(KeyEvent e) {
if(e.getKeyCode() == KeyEvent.VK_RIGHT){
    if(playerX>=600){
        playerX = 600;
    }
    else {
        moveRight();
    }
}
    if(e.getKeyCode() == KeyEvent.VK_LEFT){
        if(playerX < 10){
            playerX = 10;
        }
        else {
            moveLeft();
        }
    }
```

Slika 11 – KeyPressed

Također smo dodali dvije if komande. Razlog toga je što ako igrač ode skroz desno ne želimo da se igra sruši ili da igrač izadje iz okvira, zato postavljamo ako igrač pređe 600 program ga vraća na vrijednost 600 to je granica desne ivice. Što se tiče lijeve ivice isto je s tim što se radi o vrijednosti 10

3.1.3. Kod Programa – Map Generator

Ovo je zadnja klasa i ona se odnosi mapu. Kao što iz priloženog vidimo mapa predstavlja niz, također postvaljamo visinu i širinu blokova. Mapa ima svoju dužinu, kolone i redove. Svaki blok ima svoje mjesto na mapi, također mapa se nalazi u okviru (JFrame). Potrebno je bilo i postaviti odnos za veličine blokova

```
public class MapGenerator {

    public int map [][];
    public int brickWidth;
    public int brickHeight;
    public MapGenerator(int row, int col){
        map = new int [row] [col];
        for(int i=0;i< map.length;i++){
            for(int j=0;j<map[0].length;j++){
                map[i][j] = 1;
            }
        }
        brickWidth = 540/col;
        brickHeight = 150/row;
    }
```

Slika 12 - Klasa MapGenerator

Posljednji korak nam je upravo to iscrtavanje mape, te pravimo niz, u kojem završavamo izradu blokova, postavljamo tačnu veličinu blokova, postavljamo im bijelu boju, te pravimo razmak između istih. Također dodajemo „stroke“ koji nam omogućuje da dodatno dekorišemo izgled (outline).

```
public void draw(Graphics2D g) {
    for (int i = 0; i < map.length; i++) {
        for (int j = 0; j < map[0].length; j++) {
            if (map[i][j] > 0) {
                g.setColor(Color.white);
                g.fillRect( x: j * brickWidth + 80,  y: i * brickHeight + 50, brickWidth, brickHeight);
                g.setStroke(new BasicStroke( width: 3));
                g.setColor(Color.black);
                g.drawRect( x: j * brickWidth + 80,  y: i * brickHeight + 50, brickWidth, brickHeight);
            }
        }
    }
}
```

Slika 13 - Iscrtavanje mape

ZAKLJUČAK

Pored klasičnih programa, danas veliku pažnju dobijaju i programi zabavnog tipa kao što su: socijalne mreže, programi za medije, pa i same igrice. Mnoge multimilionerske firme su svoj fokus upravo stavile na ovu vrstu zabave. Iako se ne radi o takvoj igrici ovdje, ovakve su igrice bile na tom niovu prije par decenija i postavile su standard za ovo što se danas naziva igricom. Kroz cijeli seminarski rad moj cilj je bio objasniti izradu programa, te objasniti na logičkom i programerskom niovu zašto je nešto tako. Nisam išao u detalje šta je if, kako funkcionišu nizovi i sl. Cijela igrica je spoj grafike, fizike i samog programiranja.

Popis slika

Literatura

Java. (n.d.). Dohvaćeno iz Wikipedia: https://hr.wikipedia.org/wiki/Java_(programski_jezik)

Java Overview. (n.d.). Dohvaćeno iz Oracle: https://docs.oracle.com/javase/7/docs/api/overview-summary.html