**Univerzitet u Novom Sadu**

**Tehnički fakultet „Mihajlo Pupin“**

**Zrenjanin**

**SEMINARSKI RAD**

**Predmet: Projektovanje Softvera**

**Tema:**

***Library menagement application***

**Profesor: Student:**

[**prof. dr Eleonora**](http://www.tfzr.uns.ac.rs/Kadar/NastavnoOsoblje/20) **Brtka Milan Burić SI 39/20**

**Asistent: mr Siniša Mihajlović Smer: Softversko inženjerstvo**

**Zrenjanin, 2022.**

**Sadržaj:**

[**UVOD** 3](#_Toc112686415)

[**IDEJA** 4](#_Toc112686416)

[**POČETAK RADA** 5](#_Toc112686417)

[**Main\_Menu** 6](#_Toc112686418)

[**Game\_Menu** 10](#_Toc112686419)

[**Librarians** 13](#_Toc112686420)

[**List\_Of\_Books** 19](#_Toc112686421)

[**Readers** 25](#_Toc112686422)

[**IGRICE** 26](#_Toc112686423)

[**Pacman** 27](#_Toc112686424)

[**Space Invaders** 41](#_Toc112686425)

[**Snake** 54](#_Toc112686426)

[**ZAKLJUČAK** 60](#_Toc112686427)

[**LITERATURA** 61](#_Toc112686428)

# **UVOD**

U ovom seminarskom radu videće se kompletan process kreiranja menadžerske aplikacije namenjene za Biblioteke, knjižare, ali kao I druge radnje uz manju promenu koda.

Kod je dizajniran da bude univerzaljno korišten uz manje izmene detalja u tabelama, dok najbitnije funkcije I logika ostaju isti.

# **IDEJA**

Na ideju ovog projekta sam došao prilikom jedne od redovnih sesija predavanja koja su bila održavana iz ovog predmeta.

Naime nije bilo teštko zamisliti kako bi sam program funkcionisao I koja bi njegova svrha bila.

Sami program je bio manje više jednostavan za povezati I napraviti u Java programskom jeziku, ali na žalost nisam imao opštu ideju kako bih mogao da implementujem dve igrice koje sam odabrao kao dodatak ovom projektu.

Tokom rad na aplikaciji odlučio sam se da I te dve igre idu preko jednog menija koji bi sadržao sve igre koje sam planirao da ubacim.

Ideja je bila da se aplikacija izvršava preko generičkih menija, prozorčića koji bi mogli da obavljaju svoje funkije, a da nisu previše kompleksni za izradu u kodu.

Naakon što sam osmislio generalnu ideju za funkcionalnost programa I raspoređenost svakog prozora, krenuo sam sa radom.

# **POČETAK RADA**

Generalni plan I generalizacija izvršavanja plana

Prilikom početka rada na aplikaciji nije bilo puno problema, nisam jako stručan u programskom jeziku koji je odabran za rad, ali uspeo sam da iskoristim pun asortiman pomagala, uključujući predavanja I konsultacije.

Na žalost zbog moje starije konfiguracije računara nisam bio u mogućnosti da implementujem sve delove koje sam želeo, tako da u suštini ono što će biti prikazano ovde je samo 25% do 30% neračunajući igrice naravno.

Plan aplikacije:

* Prozori za rad Bibliotekara/radnika, korisnika, knjiga/proizvoda
* Dva menija:

1. Glavni meni
2. Meni za igrice

* Mogućnost pretrage
* Baza podataka
* Funkcija rezervacije
* Grafički prikaz rezervacija
* Dve igrice + jedna bonus igra

Naravno velika količina aplikacije je morala biti izbačena zbog starije konfiguracije I drugih problema.

Bez obzira na sve prethodno navedene činjenice I probleme uspešno sam napravio prezentativnu verziju aplikacije koju sam želeo.

Moj rad je počeo tako što sam kreirao glavni meni/Main\_Menu Jframe u mom projektu.

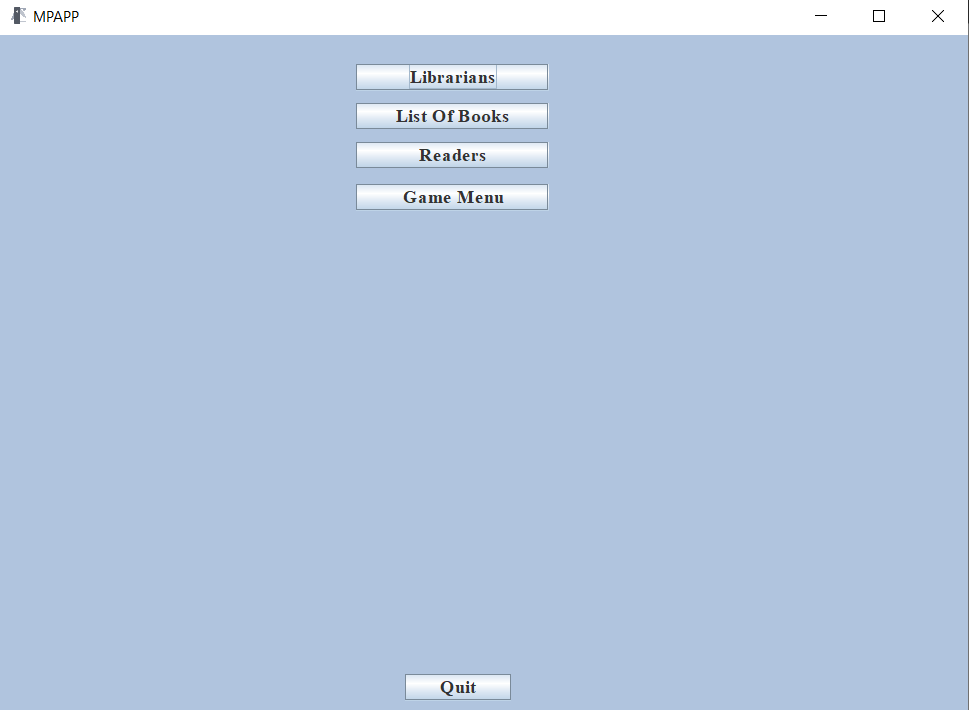
Zatim sam počeo sa kreacijama drugih prozora kao što su: Game\_Menu, Librarians, List\_Of\_Books I Readers.

**NAPOMENA:** Igrice Iako su u paketu pod nazivom pogledi nisam napravio kao JFrame nego kao obične klase zbog lakšeg kodiranja I jednostavnosti pristupa.

Znam da je ovo sve generalizujući pristup, ali preko detalja će se preći u posebno namenjenim segmentima za aplikaciju.

## **Main\_Menu**

Main\_Menu je prva I glavna JFrame klasa u mom projektu, preko nje se vrši pristup svim ostalim prozorima.



***Slika 1. Izgled glavnog menija aplikacije***

Dok glavni meni Iygleda generično I nezanimljivo, odlično obavlja svoje funkcije, naravno u narednim iteracijama aplikacije biće velikih promena za glavni meni kao I za sve ostale prozore I bazu podataka.

Kao što se na slici može videti meni poseduje pet Jbutton dugmadi koja vezuju glavni men isa svim ostalim segmentima aplikacije, ali nam i omogućavaju pristup svim ostalim segmentima aplikacije, kao što sam već rekao aplikacija je namenjena da bude jednostavna I prilagođena svim korisnicima, ali je dizajnirana I da poseduje malo stariji izgled I podseća na malo starije aplikacije sa prethodnih Windows operativnih Sistema.

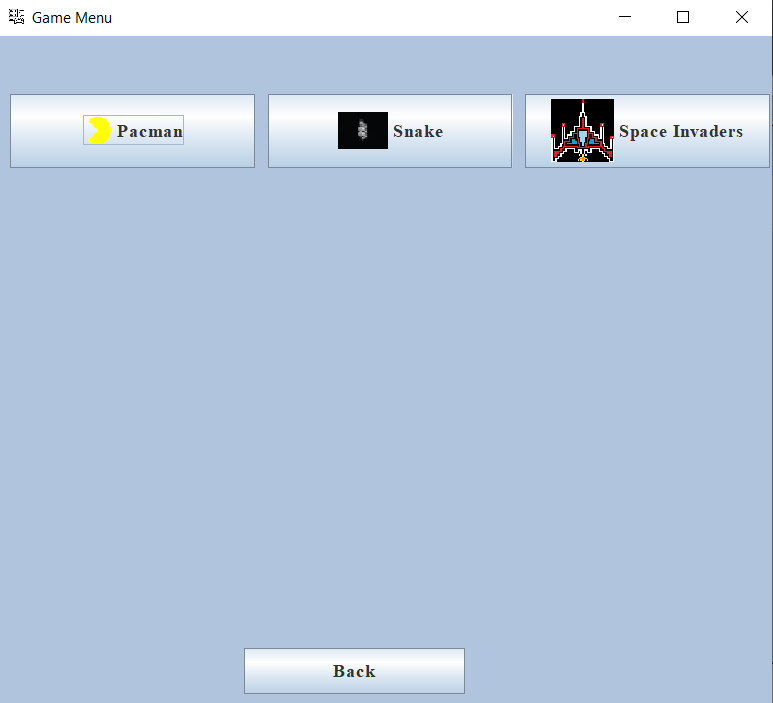
Kod za kreaciju glavnog menija:

|  |
| --- |
| package Pogledi;  import java.awt.BorderLayout;  import java.awt.EventQueue;  import javax.swing.JFrame;  import javax.swing.JPanel;  import javax.swing.border.EmptyBorder;  import javax.swing.JButton;  import java.awt.event.ActionListener;  import java.awt.event.ActionEvent;  import java.awt.Font;  import java.awt.Color;  import java.awt.Toolkit;  import javax.swing.ImageIcon;  public class Main\_Menu extends JFrame {  private JPanel contentPane;  /\*\*  \* Launch the application.  \*/  public static void main(String[] args) {  EventQueue.*invokeLater*(new Runnable() {  public void run() {  try {  Main\_Menu frame = new Main\_Menu();  frame.setVisible(true);  } catch (Exception e) {  e.printStackTrace();  }  }  });  }  /\*\*  \* Create the frame.  \*/  public Main\_Menu() {  setIconImage(Toolkit.*getDefaultToolkit*().getImage("C:\\Users\\Legion\\Desktop\\Milan Buric fakultet\\MPAPP.png"));  setFont(new Font("Times New Roman", Font.***BOLD***, 14));  setTitle("MPAPP");  setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);  setBounds(100, 100, 790, 580);  contentPane = new JPanel();  contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));  setContentPane(contentPane);  contentPane.setLayout(null);    JPanel panel = new JPanel();  panel.setBackground(new Color(176, 196, 222));  panel.setBounds(0, 0, 821, 543);  contentPane.add(panel);  panel.setLayout(null);    JButton btnLibrarians = new JButton("Librarians");  btnLibrarians.setFont(new Font("Times New Roman", Font.***BOLD***, 15));  btnLibrarians.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  dispose();  Librarians LB = new Librarians();  LB.setVisible(true);  }  });  btnLibrarians.setBounds(286, 24, 154, 21);  panel.add(btnLibrarians);    JButton btnBooks = new JButton("List Of Books");  btnBooks.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  dispose();  List\_Of\_Books LOB = new List\_Of\_Books();  LOB.setVisible(true);  }  });  btnBooks.setFont(new Font("Times New Roman", Font.***BOLD***, 15));  btnBooks.setBounds(286, 55, 154, 21);  panel.add(btnBooks);    JButton btnReaders = new JButton("Readers");  btnReaders.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  dispose();  Readers RD = new Readers();  RD.setVisible(true);  }  });  btnReaders.setFont(new Font("Times New Roman", Font.***BOLD***, 15));  btnReaders.setBounds(286, 86, 154, 21);  panel.add(btnReaders);    JButton btnGameMenu = new JButton("Game Menu");  btnGameMenu.setIcon(null);  btnGameMenu.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  dispose();  Game\_Menu GM = new Game\_Menu();  GM.setVisible(true);  }  });  btnGameMenu.setFont(new Font("Times New Roman", Font.***BOLD***, 15));  btnGameMenu.setBounds(286, 120, 154, 21);  panel.add(btnGameMenu);    JButton btnQuit = new JButton("Quit");  btnQuit.setFont(new Font("Times New Roman", Font.***BOLD***, 15));  btnQuit.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  dispose();  }  });  btnQuit.setBounds(325, 512, 85, 21);  panel.add(btnQuit);  }  } |

Sa ovim kodom bih završio priču o glavnom meniju.

## **Game\_Menu**

Game\_Menu je sledeći I poslednji meni koji imamo u aplikaciji on ima četiri JButton dugmeta koja nam omogućavaju otvaranje igrica ili povratak nazad na glavni meni, kao I glavni meni namenjen je da bude jako jednostavan I lak za upotrebu.



***Slika 2.izgled Menija za igrice.***

Kao što se može vidjeti sa slike iznad Game\_Menu je još jednostavniji od Glavnog menija I uprkos tome obavlja svoje funkcije odlično.

Kod za kreaciju Gamr\_Menu:

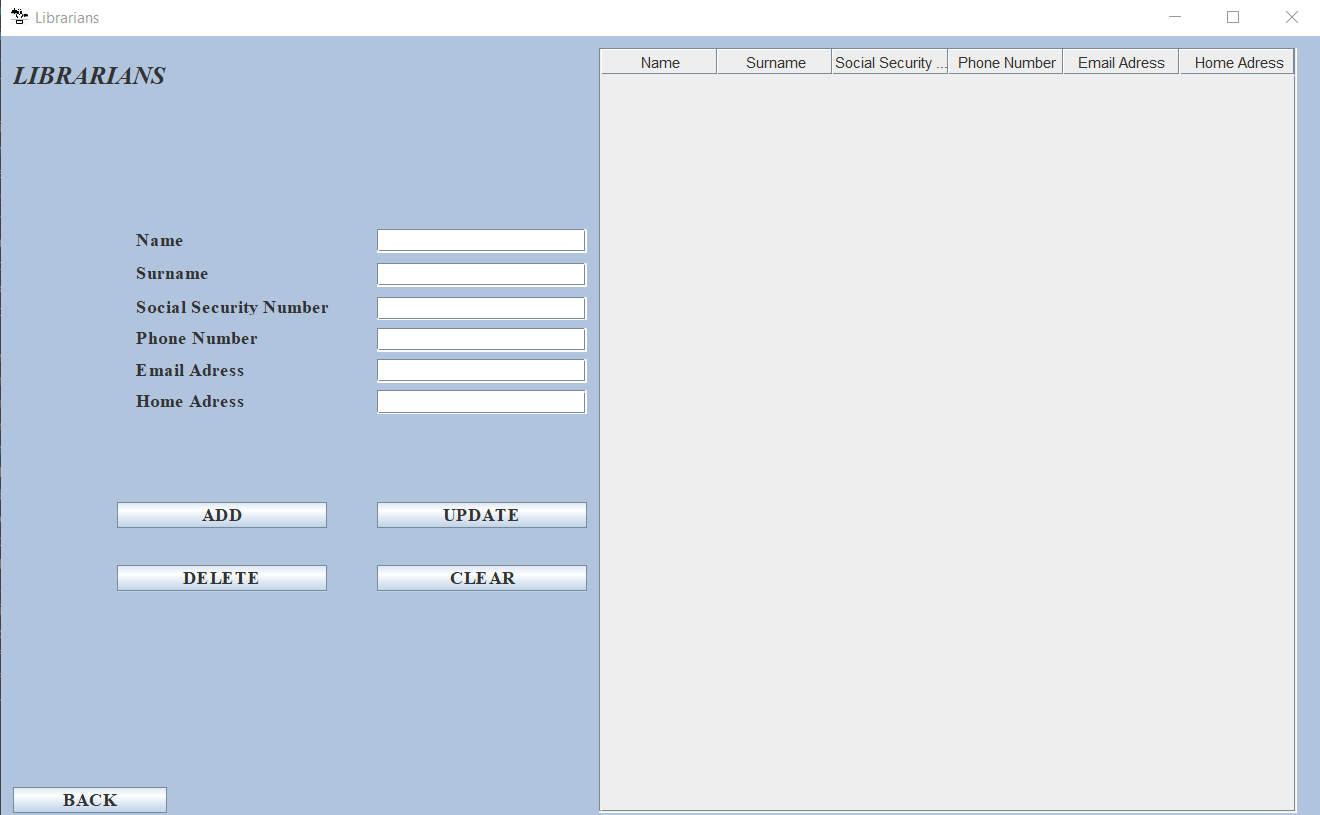
|  |
| --- |
| package Pogledi;  import java.awt.EventQueue;  import java.awt.Font;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  import javax.swing.ImageIcon;  import javax.swing.JButton;  import javax.swing.JFrame;  import javax.swing.JPanel;  import javax.swing.border.EmptyBorder;  import java.awt.Toolkit;  import java.awt.Color;  public class Game\_Menu extends JFrame {  private JPanel contentPane;  /\*\*  \* Launch the application.  \*/  public static void main(String[] args) {  EventQueue.*invokeLater*(new Runnable() {  public void run() {  try {  Game\_Menu frame = new Game\_Menu();  frame.setVisible(true);  } catch (Exception e) {  e.printStackTrace();  }  }  });  }  /\*\*  \* Create the frame.  \*/  public Game\_Menu() {  setIconImage(Toolkit.*getDefaultToolkit*().getImage("C:\\Users\\Legion\\Desktop\\Milan Buric fakultet\\Game menu.png"));  setTitle("Game Menu");  setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);  setBounds(100, 100, 634, 574);  contentPane = new JPanel();  contentPane.setBackground(new Color(176, 196, 222));  contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));  setContentPane(contentPane);  contentPane.setLayout(null);    JButton btnPacman = new JButton("Pacman");  btnPacman.setIcon(new ImageIcon("C:\\Users\\Legion\\eclipse-workspace\\Seminarski rad Projektovanje softvera\\src\\Images\\left.gif"));  btnPacman.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  dispose();  Pacman PM = new Pacman();  PM.setVisible(true);      }  });  btnPacman.setFont(new Font("Times New Roman", Font.***BOLD***, 15));  btnPacman.setBounds(10, 47, 196, 59);  contentPane.add(btnPacman);    JButton btnSpace = new JButton("Space Invaders");  btnSpace.setIcon(new ImageIcon("C:\\Users\\Legion\\eclipse-workspace\\Seminarski rad Projektovanje softvera\\src\\Images\\shipSkin.gif"));  btnSpace.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  dispose();  SpaceInvaders SI = new SpaceInvaders();  SI.setVisible(true);  }  });  btnSpace.setFont(new Font("Times New Roman", Font.***BOLD***, 15));  btnSpace.setBounds(422, 47, 196, 59);  contentPane.add(btnSpace);    JButton btnBack = new JButton("Back");  btnBack.setFont(new Font("Times New Roman", Font.***BOLD***, 15));  btnBack.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  dispose();  Main\_Menu MM = new Main\_Menu();  MM.setVisible(true);  }  });  btnBack.setBounds(197, 490, 177, 37);  contentPane.add(btnBack);    JButton btnNewButton = new JButton("Snake");  btnNewButton.setIcon(new ImageIcon("C:\\Users\\Legion\\eclipse-workspace\\Seminarski rad Projektovanje softvera\\Images\\Snake icon resized2.jpg"));  btnNewButton.setFont(new Font("Times New Roman", Font.***BOLD***, 15));  btnNewButton.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  dispose();  Snake SN = new Snake();  SN.setVisible(true);  }  });  btnNewButton.setBounds(216, 47, 196, 59);  contentPane.add(btnNewButton);  }  } |

## **Librarians**

Librarians je jedna od radnih klasa I podrazumevam je random osnovom za svaki naredni prozor koji je korišten u izradi ovog projekta.

Kao I čitava aplikacija ovaj prozor je namenjen da bude poprilično jednostavan za koristiti I bio je jedan od tri prozora za upis podataka u bazu podataka koja nažalost nikada nije izrađena sbog starije konfiguracije računara.

Uglavnom ovaj prozor se satstoji iz JTable, koji služi za čuvanje podataka, pet JButton dugmadi koja omogućavaju upis podataka, brisanje, izmenu/update, čišćenje TxtBox kutija I povratak na glavni meni I šest TxtBox kutija za upis podataka u JTable.



***Slika 3. Izgled prozora za menadžment bibliotekara-Librarians***

Kao što se sa slike može vidjeti izgled je jako jednostavan, ali preformanse su odlične.

Neću ulaziti u detalje kako je kreiran prozor, ali ću ostaviti kod pomoću kog je kreiran, tako da nema potrebe za objašnjavanjem, jako je jednostavna kreacija.

Kod za kreaciju prozora Librarians:

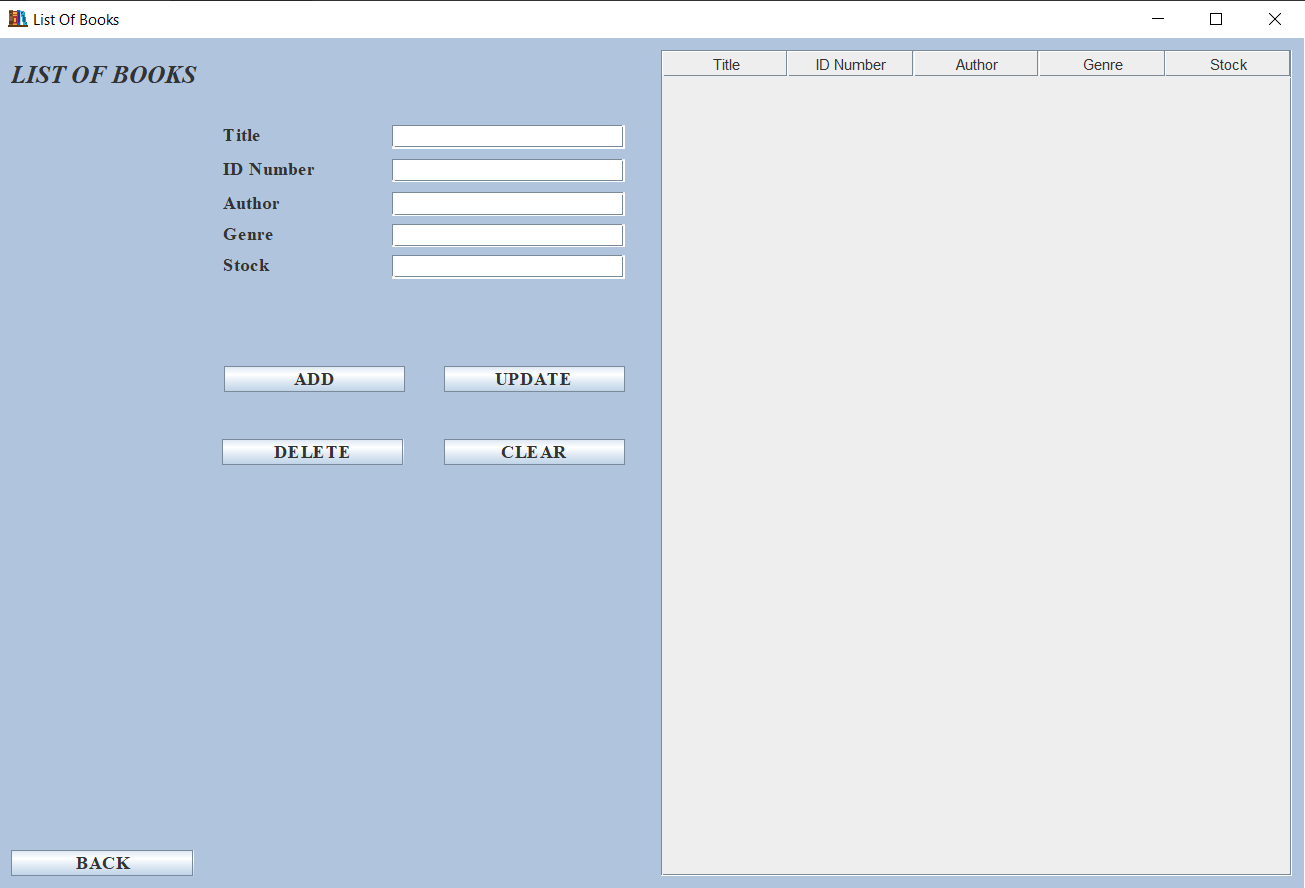
|  |
| --- |
| import java.awt.Color;  import java.awt.EventQueue;  import java.awt.Font;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  import javax.swing.JButton;  import javax.swing.JFrame;  import javax.swing.JLabel;  import javax.swing.JOptionPane;  import javax.swing.JPanel;  import javax.swing.JScrollPane;  import javax.swing.JTable;  import javax.swing.JTextField;  import javax.swing.border.EmptyBorder;  import javax.swing.table.DefaultTableModel;  import java.awt.event.MouseAdapter;  import java.awt.event.MouseEvent;  import javax.swing.ImageIcon;  import java.awt.Toolkit;  public class Librarians extends JFrame {  private JPanel contentPane;  private JTable Librarians;  private JTextField txtName;  private JTextField txtSur;  private JTextField txtSSN;  private JTextField txtPN;  private JTextField txtEmail;  private JTextField txtHome;  DefaultTableModel model;    /\*\*  \* Launch the application.  \*/  public static void main(String[] args) {  EventQueue.invokeLater(new Runnable() {  public void run() {  try {  Librarians frame = new Librarians();  frame.setVisible(true);  } catch (Exception e) {  e.printStackTrace();  }  }  });  }  /\*\*  \* Create the frame.  \*/  public Librarians() {  setIconImage(Toolkit.getDefaultToolkit().getImage("C:\\Users\\Legion\\Desktop\\Milan Buric fakultet\\Librarians.jpg"));  setTitle("Librarians");  setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  setBounds(100, 100, 1071, 669);  contentPane = new JPanel();  contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));  setContentPane(contentPane);  contentPane.setLayout(null);    JPanel panel = new JPanel();  panel.setBackground(new Color(176, 196, 222));  panel.setBounds(0, 0, 1057, 632);  contentPane.add(panel);  panel.setLayout(null);    JLabel lblNewLabel = new JLabel("Name");  lblNewLabel.setFont(new Font("Times New Roman", Font.BOLD, 15));  lblNewLabel.setBounds(108, 157, 168, 13);  panel.add(lblNewLabel);    JLabel lblNewLabel\_1 = new JLabel("Surname");  lblNewLabel\_1.setFont(new Font("Times New Roman", Font.BOLD, 15));  lblNewLabel\_1.setBounds(108, 184, 168, 13);  panel.add(lblNewLabel\_1);    JLabel lblNewLabel\_2 = new JLabel("Social Security Number");  lblNewLabel\_2.setFont(new Font("Times New Roman", Font.BOLD, 15));  lblNewLabel\_2.setBounds(108, 211, 168, 13);  panel.add(lblNewLabel\_2);    JLabel lblNewLabel\_3 = new JLabel("Phone Number");  lblNewLabel\_3.setFont(new Font("Times New Roman", Font.BOLD, 15));  lblNewLabel\_3.setBounds(108, 236, 168, 13);  panel.add(lblNewLabel\_3);    JLabel lblNewLabel\_4 = new JLabel("Email Adress");  lblNewLabel\_4.setFont(new Font("Times New Roman", Font.BOLD, 15));  lblNewLabel\_4.setBounds(108, 261, 168, 13);  panel.add(lblNewLabel\_4);    JLabel lblNewLabel\_5 = new JLabel("Home Adress");  lblNewLabel\_5.setFont(new Font("Times New Roman", Font.BOLD, 15));  lblNewLabel\_5.setBounds(108, 286, 168, 13);  panel.add(lblNewLabel\_5);    JScrollPane scrollPane = new JScrollPane();  scrollPane.setBounds(479, 10, 558, 612);  panel.add(scrollPane);    Librarians = new JTable();  Librarians.addMouseListener(new MouseAdapter() {  @Override  public void mouseClicked(MouseEvent e) {    int i = Librarians.getSelectedRow();    txtName.setText(model.getValueAt(i, 0).toString());  txtSur.setText(model.getValueAt(i, 1).toString());  txtSSN.setText(model.getValueAt(i, 2).toString());  txtPN.setText(model.getValueAt(i, 3).toString());  txtEmail.setText(model.getValueAt(i, 4).toString());  txtHome.setText(model.getValueAt(i, 0).toString());      }  });  Librarians.setBackground(new Color(240, 255, 255));  Librarians.setFont(new Font("Times New Roman", Font.PLAIN, 15));    model = new DefaultTableModel();  Object[] column = {"Name", "Surname", "Social Security Number", "Phone Number", "Email Adress", "Home Adress"};  Object[] row = new Object[6];  model.setColumnIdentifiers(column);  Librarians.setModel(model);    scrollPane.setViewportView(Librarians);    txtName = new JTextField();  txtName.setBounds(301, 155, 168, 19);  panel.add(txtName);  txtName.setColumns(10);    txtSur = new JTextField();  txtSur.setBounds(301, 182, 168, 19);  panel.add(txtSur);  txtSur.setColumns(10);    txtSSN = new JTextField();  txtSSN.setBounds(301, 209, 168, 19);  panel.add(txtSSN);  txtSSN.setColumns(10);    txtPN = new JTextField();  txtPN.setBounds(301, 234, 168, 19);  panel.add(txtPN);  txtPN.setColumns(10);    txtEmail = new JTextField();  txtEmail.setBounds(301, 259, 168, 19);  panel.add(txtEmail);  txtEmail.setColumns(10);    txtHome = new JTextField();  txtHome.setBounds(301, 284, 168, 19);  panel.add(txtHome);  txtHome.setColumns(10);    JButton btnNewButton = new JButton("ADD");  btnNewButton.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  if(txtName.getText().equals("") || txtSur.getText().equals("") || txtSSN.getText().equals("") || txtEmail.getText().equals("") ||txtHome.getText().equals("")) {    JOptionPane.showMessageDialog(null, "Please Enter the necessary Information!");  }else {  row[0] = txtName.getText();  row[1] = txtSur.getText();  row[2] = txtSSN.getText();  row[3] = txtPN.getText();  row[4] = txtEmail.getText();  row[5] = txtHome.getText();  model.addRow(row);    txtName.setText("");  txtSur.setText("");  txtSSN.setText("");  txtPN.setText("");  txtEmail.setText("");  txtHome.setText("");  JOptionPane.showMessageDialog(null, "Information Successfully Added!");  }    }  });  btnNewButton.setFont(new Font("Times New Roman", Font.BOLD, 15));  btnNewButton.setBounds(93, 373, 168, 21);  panel.add(btnNewButton);    JButton btnNewButton\_1 = new JButton("UPDATE");  btnNewButton\_1.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  int i = Librarians.getSelectedRow();  if(i >= 0) {  model.setValueAt(txtName.getText(), i, 0);  model.setValueAt(txtSur.getText(), i, 1);  model.setValueAt(txtSSN.getText(), i, 2);  model.setValueAt(txtPN.getText(), i, 3);  model.setValueAt(txtEmail.getText(), i, 4);  model.setValueAt(txtHome.getText(), i, 5);  JOptionPane.showMessageDialog(null, "Row Updated Successfully!");  }else {  JOptionPane.showMessageDialog(null, "Please Select The Row You Wish To Upadate!");  }    }  });  btnNewButton\_1.setFont(new Font("Times New Roman", Font.BOLD, 15));  btnNewButton\_1.setBounds(301, 373, 168, 21);  panel.add(btnNewButton\_1);    JButton btnNewButton\_2 = new JButton("DELETE");  btnNewButton\_2.setFont(new Font("Times New Roman", Font.BOLD, 15));  btnNewButton\_2.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  int i = Librarians.getSelectedRow();  if(i >= 0) {    model.removeRow(i);  JOptionPane.showMessageDialog(null, "Information Successfully Deleted!");    }else {  JOptionPane.showMessageDialog(null, "Please Select The Row in the Table You Wish To Delete!");  }  }  });  btnNewButton\_2.setBounds(93, 424, 168, 21);  panel.add(btnNewButton\_2);    JButton btnNewButton\_3 = new JButton("CLEAR");  btnNewButton\_3.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {    txtName.setText("");  txtSur.setText("");  txtSSN.setText("");  txtPN.setText("");  txtEmail.setText("");  txtHome.setText("");    }  });  btnNewButton\_3.setFont(new Font("Times New Roman", Font.BOLD, 15));  btnNewButton\_3.setBounds(301, 424, 168, 21);  panel.add(btnNewButton\_3);    JButton btnExit = new JButton("BACK");  btnExit.setFont(new Font("Times New Roman", Font.BOLD, 15));  btnExit.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  dispose();  Main\_Menu MM = new Main\_Menu();  MM.setVisible(true);  }  });  btnExit.setBounds(10, 601, 123, 21);  panel.add(btnExit);    JLabel lblNewLabel\_6 = new JLabel("LIBRARIANS");  lblNewLabel\_6.setIcon(null);  lblNewLabel\_6.setFont(new Font("Times New Roman", Font.BOLD | Font.ITALIC, 20));  lblNewLabel\_6.setBounds(10, 13, 168, 41);  panel.add(lblNewLabel\_6);  }  } |

Sa ovim kodm bih završio komplentu priču o prozoru Librarians.

## **List\_Of\_Books**

List\_Of\_Books je sledeća JF klasa koja izgleda identično klasi Librarians I klasi Readers, sa manjmo količinom neophodnih resursa, ali funkcioniše po istom principu kao I prethodne dve klase.

Sa time rečenim, ova klasa sadrži JTable koji služi za unost podataka, pet JB dugmadi koja omogućavaju unos, brisanje, izmenu/update-ovanje podataka I čišćenje TxtBox kutija koje služe za unos podataka u tabelu.



***Slika 4. Izgled prozora za menadžment listom knjiga/List\_Of\_Books***

Isto jako jednostavan dizajn I dobre preformanse.

Kod za kreaciju List\_Of\_Books prozora:

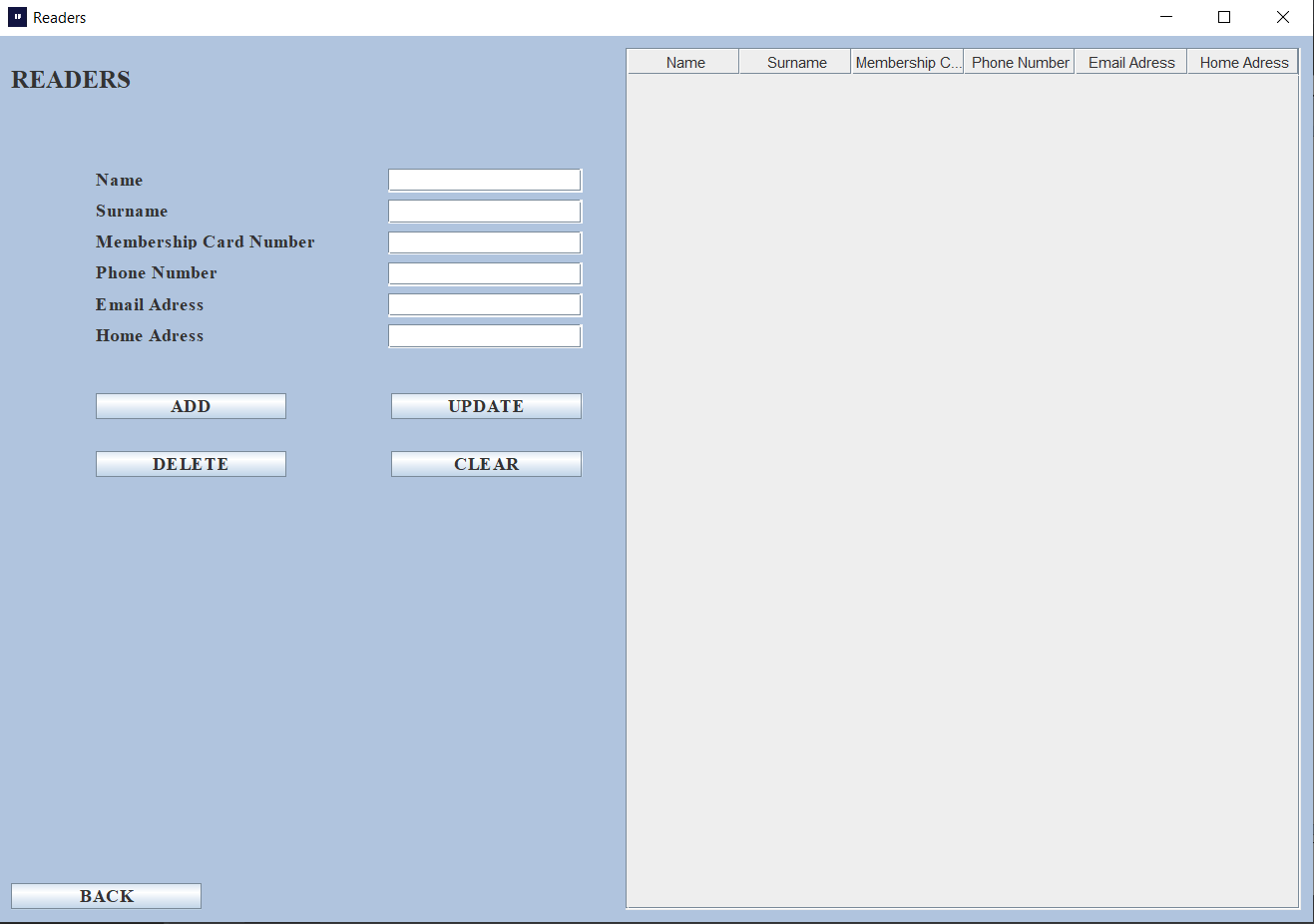
|  |
| --- |
| package Pogledi;  import java.awt.Color;  import java.awt.EventQueue;  import java.awt.Font;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  import java.awt.event.MouseAdapter;  import java.awt.event.MouseEvent;  import javax.swing.JButton;  import javax.swing.JFrame;  import javax.swing.JLabel;  import javax.swing.JOptionPane;  import javax.swing.JPanel;  import javax.swing.JScrollPane;  import javax.swing.JTable;  import javax.swing.JTextField;  import javax.swing.border.EmptyBorder;  import javax.swing.table.DefaultTableModel;  import java.awt.Toolkit;  import java.awt.Window.Type;  public class List\_Of\_Books extends JFrame {  private JPanel contentPane;  private JTextField txtTitle;  private JTextField txtID;  private JTextField txtAuthor;  private JTextField txtGenre;  private JTextField txtStock;  private JTable List\_Of\_Books;  DefaultTableModel model;  /\*\*  \* Launch the application.  \*/  public static void main(String[] args) {  EventQueue.invokeLater(new Runnable() {  public void run() {  try {  List\_Of\_Books frame = new List\_Of\_Books();  frame.setVisible(true);  } catch (Exception e) {  e.printStackTrace();  }  }  });  }  /\*\*  \* Create the frame.  \*/  public List\_Of\_Books() {  setIconImage(Toolkit.getDefaultToolkit().getImage("C:\\Users\\Legion\\Desktop\\Milan Buric fakultet\\LOB.png"));  setTitle("List Of Books");  setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  setBounds(100, 100, 1059, 718);  contentPane = new JPanel();  contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));  setContentPane(contentPane);  contentPane.setLayout(null);    JPanel panel = new JPanel();  panel.setBackground(new Color(176, 196, 222));  panel.setBounds(0, 0, 1045, 681);  contentPane.add(panel);  panel.setLayout(null);    JLabel lblNewLabel\_1 = new JLabel("ID Number");  lblNewLabel\_1.setFont(new Font("Times New Roman", Font.BOLD, 15));  lblNewLabel\_1.setBounds(179, 99, 126, 13);  panel.add(lblNewLabel\_1);    JLabel lblNewLabel\_2 = new JLabel("Author");  lblNewLabel\_2.setFont(new Font("Times New Roman", Font.BOLD, 15));  lblNewLabel\_2.setBounds(179, 126, 126, 13);  panel.add(lblNewLabel\_2);    JLabel lblNewLabel = new JLabel("Genre");  lblNewLabel.setFont(new Font("Times New Roman", Font.BOLD, 15));  lblNewLabel.setBounds(179, 151, 126, 13);  panel.add(lblNewLabel);    JLabel lblNewLabel\_3 = new JLabel("Stock");  lblNewLabel\_3.setFont(new Font("Times New Roman", Font.BOLD, 15));  lblNewLabel\_3.setBounds(179, 176, 126, 13);  panel.add(lblNewLabel\_3);    JLabel lblNewLabel\_4 = new JLabel("Title");  lblNewLabel\_4.setFont(new Font("Times New Roman", Font.BOLD, 15));  lblNewLabel\_4.setBounds(179, 72, 126, 13);  panel.add(lblNewLabel\_4);    txtTitle = new JTextField();  txtTitle.setBounds(315, 70, 186, 19);  panel.add(txtTitle);  txtTitle.setColumns(10);    txtID = new JTextField();  txtID.setBounds(315, 97, 186, 19);  panel.add(txtID);  txtID.setColumns(10);    txtAuthor = new JTextField();  txtAuthor.setBounds(315, 124, 186, 19);  panel.add(txtAuthor);  txtAuthor.setColumns(10);    txtGenre = new JTextField();  txtGenre.setBounds(315, 149, 186, 19);  panel.add(txtGenre);  txtGenre.setColumns(10);    txtStock = new JTextField();  txtStock.setBounds(315, 174, 186, 19);  panel.add(txtStock);  txtStock.setColumns(10);    JScrollPane scrollPane = new JScrollPane();  scrollPane.setBounds(530, 10, 505, 661);  panel.add(scrollPane);    List\_Of\_Books = new JTable();  List\_Of\_Books.addMouseListener(new MouseAdapter() {  @Override  public void mouseClicked(MouseEvent e) {  int i = List\_Of\_Books.getSelectedRow();    txtTitle.setText(model.getValueAt(i, 0).toString());  txtID.setText(model.getValueAt(i, 1).toString());  txtAuthor.setText(model.getValueAt(i, 2).toString());  txtGenre.setText(model.getValueAt(i, 3).toString());  txtStock.setText(model.getValueAt(i, 4).toString());      }  });  List\_Of\_Books.setFont(new Font("Times New Roman", Font.BOLD, 15));  scrollPane.setViewportView(List\_Of\_Books);  model = new DefaultTableModel();  Object[] column = {"Title", "ID Number", "Author", "Genre", "Stock"};  Object[] row = new Object[5];  model.setColumnIdentifiers(column);  List\_Of\_Books.setModel(model);    scrollPane.setViewportView(List\_Of\_Books);    JButton btnNewButton = new JButton("BACK");  btnNewButton.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  dispose();  Main\_Menu MM = new Main\_Menu();  MM.setVisible(true);  }  });  btnNewButton.setFont(new Font("Times New Roman", Font.BOLD, 15));  btnNewButton.setBounds(10, 650, 146, 21);  panel.add(btnNewButton);    JButton btnNewButton\_1 = new JButton("ADD");  btnNewButton\_1.setFont(new Font("Times New Roman", Font.BOLD, 15));  btnNewButton\_1.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  if(txtTitle.getText().equals("") || txtID.getText().equals("") || txtAuthor.getText().equals("") || txtGenre.getText().equals("") ||txtStock.getText().equals("")) {    JOptionPane.showMessageDialog(null, "Please Enter the necessary Information!");  }else {  row[0] = txtTitle.getText();  row[1] = txtID.getText();  row[2] = txtAuthor.getText();  row[3] = txtGenre.getText();  row[4] = txtStock.getText();  model.addRow(row);    txtTitle.setText("");  txtID.setText("");  txtAuthor.setText("");  txtGenre.setText("");  txtStock.setText("");  JOptionPane.showMessageDialog(null, "Information Successfully Added!");  }  }  });  btnNewButton\_1.setBounds(180, 263, 145, 21);  panel.add(btnNewButton\_1);    JButton btnNewButton\_2 = new JButton("UPDATE");  btnNewButton\_2.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  int i = List\_Of\_Books.getSelectedRow();  if(i >= 0) {  model.setValueAt(txtTitle.getText(), i, 0);  model.setValueAt(txtID.getText(), i, 1);  model.setValueAt(txtAuthor.getText(), i, 2);  model.setValueAt(txtGenre.getText(), i, 3);  model.setValueAt(txtStock.getText(), i, 4);  JOptionPane.showMessageDialog(null, "Row Updated Successfully!");  }else {  JOptionPane.showMessageDialog(null, "Please Select The Row You Wish To Upadate!");  }  }  });  btnNewButton\_2.setFont(new Font("Times New Roman", Font.BOLD, 15));  btnNewButton\_2.setBounds(356, 263, 145, 21);  panel.add(btnNewButton\_2);    JButton btnNewButton\_3 = new JButton("DELETE");  btnNewButton\_3.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {    int i = List\_Of\_Books.getSelectedRow();  if(i >= 0) {    model.removeRow(i);  JOptionPane.showMessageDialog(null, "Information Successfully Deleted!");    }else {  JOptionPane.showMessageDialog(null, "Please Select The Row in the Table You Wish To Delete!");  }  }  });  btnNewButton\_3.setFont(new Font("Times New Roman", Font.BOLD, 15));  btnNewButton\_3.setBounds(179, 321, 145, 21);  panel.add(btnNewButton\_3);    JButton btnNewButton\_4 = new JButton("CLEAR");  btnNewButton\_4.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {    txtTitle.setText("");  txtID.setText("");  txtAuthor.setText("");  txtGenre.setText("");  txtStock.setText("");  JOptionPane.showMessageDialog(null, "Information Successfully Added!");  }  });  btnNewButton\_4.setFont(new Font("Times New Roman", Font.BOLD, 15));  btnNewButton\_4.setBounds(356, 321, 145, 21);  panel.add(btnNewButton\_4);    JLabel lblNewLabel\_5 = new JLabel("LIST OF BOOKS");  lblNewLabel\_5.setFont(new Font("Times New Roman", Font.BOLD | Font.ITALIC, 20));  lblNewLabel\_5.setBounds(10, 0, 266, 60);  panel.add(lblNewLabel\_5);  }  } |

Svi kodovi su veoma jednostavni I nebi trebalo puno vremena da se pređu I razumeju u potpunosti, baš iz tog razloga ne ulazim u dubinu kodova.

## **Readers**

Readers readers JF klasa je istog izgleda kao prethodne dve navdene sa time da ona I kalsa Librarians dele par argumenata.

Uglavnom ovaj prozor se satstoji iz JTable, koji služi za čuvanje podataka, pet JButton dugmadi koja omogućavaju upis podataka, brisanje, izmenu/update, čišćenje TxtBox kutija I povratak na glavni meni I šest TxtBox kutija za upis podataka u JTable.



***Slika 5. Izgled prozora Čitalaca/Readers.***

# **IGRICE**

Sada prelazimo na segment igrica, ove igrice su vrlo jednostavne.

Igrice koje sam odlučio inkorporirati u moj projekat su:

1. Pacman
2. Space Invaders
3. Snake

Sada, znam šta mislite: “ove igrice su veoma stare”, slažem se, ali su jako jednostavne I jako male što ih čini perfektni kandidatima za inkorporaciju u ovakve aplikacije, njihova kompaktnost im omogućava da budu napravljene sa par linija koda I nekoliko slika.

Pogotovo u današnjem vremenu gde program imaju pre-set-ovane parameter za tastatur I miš, tako da pravljenje igrica ovog tipa nije neka preterano velika obaveza.

Jedini problem koji sam imao u ovom segment bio je apsolutna putanja do slika koje su mi bile neophodne za igre.

Ukoliko jedna od ovih slika nije lepo inkorporirana, ili putanja do nje nije tačna čitava igraa više ne može da radi.

Sada ću ubaciti kodove vezane za igre.

**NAPOMENA:** Za uspešnu izradu ovih igara potrebno je znati apsolutni put do slika koje ove igre koriste, ne samo ubaciti slike u projekat.

## **Pacman**

Pacman se u ovom projektu sastoji iz dve klase jedna klasa je table odnosno BoardPac koja sadrži svu logiku I način funkcionisanja I klasa Pacman koja sadrži načine pokretanja.

Kod klase BoardPac

|  |
| --- |
| package Resursi;  import java.awt.BasicStroke;  import java.awt.Color;  import java.awt.Dimension;  import java.awt.Event;  import java.awt.Font;  import java.awt.FontMetrics;  import java.awt.Graphics;  import java.awt.Graphics2D;  import java.awt.Image;  import java.awt.Toolkit;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  import java.awt.event.KeyAdapter;  import java.awt.event.KeyEvent;  import javax.swing.ImageIcon;  import javax.swing.JPanel;  import javax.swing.Timer;  public class Boardpac extends JPanel implements ActionListener {  private Dimension d;  private final Font smallFont = new Font("Helvetica", Font.BOLD, 14);  private Image ii;  private final Color dotColor = new Color(192, 192, 0);  private Color mazeColor;  private boolean inGame = false;  private boolean dying = false;  private final int BLOCK\_SIZE = 24;  private final int N\_BLOCKS = 15;  private final int SCREEN\_SIZE = N\_BLOCKS \* BLOCK\_SIZE;  private final int PAC\_ANIM\_DELAY = 2;  private final int PACMAN\_ANIM\_COUNT = 4;  private final int MAX\_GHOSTS = 12;  private final int PACMAN\_SPEED = 6;  private int pacAnimCount = PAC\_ANIM\_DELAY;  private int pacAnimDir = 1;  private int pacmanAnimPos = 0;  private int N\_GHOSTS = 6;  private int pacsLeft, score;  private int[] dx, dy;  private int[] ghost\_x, ghost\_y, ghost\_dx, ghost\_dy, ghostSpeed;  private Image ghost;  private Image pacman1, pacman2up, pacman2left, pacman2right, pacman2down;  private Image pacman3up, pacman3down, pacman3left, pacman3right;  private Image pacman4up, pacman4down, pacman4left, pacman4right;  private int pacman\_x, pacman\_y, pacmand\_x, pacmand\_y;  private int req\_dx, req\_dy, view\_dx, view\_dy;  private final short levelData[] = {  19, 26, 26, 26, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18, 22,  21, 0, 0, 0, 17, 16, 16, 16, 16, 16, 16, 16, 16, 16, 20,  21, 0, 0, 0, 17, 16, 16, 16, 16, 16, 16, 16, 16, 16, 20,  21, 0, 0, 0, 17, 16, 16, 24, 16, 16, 16, 16, 16, 16, 20,  17, 18, 18, 18, 16, 16, 20, 0, 17, 16, 16, 16, 16, 16, 20,  17, 16, 16, 16, 16, 16, 20, 0, 17, 16, 16, 16, 16, 24, 20,  25, 16, 16, 16, 24, 24, 28, 0, 25, 24, 24, 16, 20, 0, 21,  1, 17, 16, 20, 0, 0, 0, 0, 0, 0, 0, 17, 20, 0, 21,  1, 17, 16, 16, 18, 18, 22, 0, 19, 18, 18, 16, 20, 0, 21,  1, 17, 16, 16, 16, 16, 20, 0, 17, 16, 16, 16, 20, 0, 21,  1, 17, 16, 16, 16, 16, 20, 0, 17, 16, 16, 16, 20, 0, 21,  1, 17, 16, 16, 16, 16, 16, 18, 16, 16, 16, 16, 20, 0, 21,  1, 17, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 20, 0, 21,  1, 25, 24, 24, 24, 24, 24, 24, 24, 24, 16, 16, 16, 18, 20,  9, 8, 8, 8, 8, 8, 8, 8, 8, 8, 25, 24, 24, 24, 28  };  private final int validSpeeds[] = {1, 2, 3, 4, 6, 8};  private final int maxSpeed = 6;  private int currentSpeed = 3;  private short[] screenData;  private Timer timer;  public Boardpac() {  loadImages();  initVariables();  initBoard();  }    private void initBoard() {    addKeyListener(new TAdapter());  setFocusable(true);  setBackground(Color.black);  }  private void initVariables() {  screenData = new short[N\_BLOCKS \* N\_BLOCKS];  mazeColor = new Color(5, 100, 5);  d = new Dimension(400, 400);  ghost\_x = new int[MAX\_GHOSTS];  ghost\_dx = new int[MAX\_GHOSTS];  ghost\_y = new int[MAX\_GHOSTS];  ghost\_dy = new int[MAX\_GHOSTS];  ghostSpeed = new int[MAX\_GHOSTS];  dx = new int[4];  dy = new int[4];    timer = new Timer(40, this);  timer.start();  }  @Override  public void addNotify() {  super.addNotify();  initGame();  }  private void doAnim() {  pacAnimCount--;  if (pacAnimCount <= 0) {  pacAnimCount = PAC\_ANIM\_DELAY;  pacmanAnimPos = pacmanAnimPos + pacAnimDir;  if (pacmanAnimPos == (PACMAN\_ANIM\_COUNT - 1) || pacmanAnimPos == 0) {  pacAnimDir = -pacAnimDir;  }  }  }  private void playGame(Graphics2D g2d) {  if (dying) {  death();  } else {  movePacman();  drawPacman(g2d);  moveGhosts(g2d);  checkMaze();  }  }  private void showIntroScreen(Graphics2D g2d) {  g2d.setColor(new Color(0, 32, 48));  g2d.fillRect(50, SCREEN\_SIZE / 2 - 30, SCREEN\_SIZE - 100, 50);  g2d.setColor(Color.white);  g2d.drawRect(50, SCREEN\_SIZE / 2 - 30, SCREEN\_SIZE - 100, 50);  String s = "Press s to start.";  Font small = new Font("Helvetica", Font.BOLD, 14);  FontMetrics metr = this.getFontMetrics(small);  g2d.setColor(Color.white);  g2d.setFont(small);  g2d.drawString(s, (SCREEN\_SIZE - metr.stringWidth(s)) / 2, SCREEN\_SIZE / 2);  }  private void drawScore(Graphics2D g) {  int i;  String s;  g.setFont(smallFont);  g.setColor(new Color(96, 128, 255));  s = "Score: " + score;  g.drawString(s, SCREEN\_SIZE / 2 + 96, SCREEN\_SIZE + 16);  for (i = 0; i < pacsLeft; i++) {  g.drawImage(pacman3left, i \* 28 + 8, SCREEN\_SIZE + 1, this);  }  }  private void checkMaze() {  short i = 0;  boolean finished = true;  while (i < N\_BLOCKS \* N\_BLOCKS && finished) {  if ((screenData[i] & 48) != 0) {  finished = false;  }  i++;  }  if (finished) {  score += 50;  if (N\_GHOSTS < MAX\_GHOSTS) {  N\_GHOSTS++;  }  if (currentSpeed < maxSpeed) {  currentSpeed++;  }  initLevel();  }  }  private void death() {  pacsLeft--;  if (pacsLeft == 0) {  inGame = false;  }  continueLevel();  }  private void moveGhosts(Graphics2D g2d) {  short i;  int pos;  int count;  for (i = 0; i < N\_GHOSTS; i++) {  if (ghost\_x[i] % BLOCK\_SIZE == 0 && ghost\_y[i] % BLOCK\_SIZE == 0) {  pos = ghost\_x[i] / BLOCK\_SIZE + N\_BLOCKS \* (int) (ghost\_y[i] / BLOCK\_SIZE);  count = 0;  if ((screenData[pos] & 1) == 0 && ghost\_dx[i] != 1) {  dx[count] = -1;  dy[count] = 0;  count++;  }  if ((screenData[pos] & 2) == 0 && ghost\_dy[i] != 1) {  dx[count] = 0;  dy[count] = -1;  count++;  }  if ((screenData[pos] & 4) == 0 && ghost\_dx[i] != -1) {  dx[count] = 1;  dy[count] = 0;  count++;  }  if ((screenData[pos] & 8) == 0 && ghost\_dy[i] != -1) {  dx[count] = 0;  dy[count] = 1;  count++;  }  if (count == 0) {  if ((screenData[pos] & 15) == 15) {  ghost\_dx[i] = 0;  ghost\_dy[i] = 0;  } else {  ghost\_dx[i] = -ghost\_dx[i];  ghost\_dy[i] = -ghost\_dy[i];  }  } else {  count = (int) (Math.random() \* count);  if (count > 3) {  count = 3;  }  ghost\_dx[i] = dx[count];  ghost\_dy[i] = dy[count];  }  }  ghost\_x[i] = ghost\_x[i] + (ghost\_dx[i] \* ghostSpeed[i]);  ghost\_y[i] = ghost\_y[i] + (ghost\_dy[i] \* ghostSpeed[i]);  drawGhost(g2d, ghost\_x[i] + 1, ghost\_y[i] + 1);  if (pacman\_x > (ghost\_x[i] - 12) && pacman\_x < (ghost\_x[i] + 12)  && pacman\_y > (ghost\_y[i] - 12) && pacman\_y < (ghost\_y[i] + 12)  && inGame) {  dying = true;  }  }  }  private void drawGhost(Graphics2D g2d, int x, int y) {  g2d.drawImage(ghost, x, y, this);  }  private void movePacman() {  int pos;  short ch;  if (req\_dx == -pacmand\_x && req\_dy == -pacmand\_y) {  pacmand\_x = req\_dx;  pacmand\_y = req\_dy;  view\_dx = pacmand\_x;  view\_dy = pacmand\_y;  }  if (pacman\_x % BLOCK\_SIZE == 0 && pacman\_y % BLOCK\_SIZE == 0) {  pos = pacman\_x / BLOCK\_SIZE + N\_BLOCKS \* (int) (pacman\_y / BLOCK\_SIZE);  ch = screenData[pos];  if ((ch & 16) != 0) {  screenData[pos] = (short) (ch & 15);  score++;  }  if (req\_dx != 0 || req\_dy != 0) {  if (!((req\_dx == -1 && req\_dy == 0 && (ch & 1) != 0)  || (req\_dx == 1 && req\_dy == 0 && (ch & 4) != 0)  || (req\_dx == 0 && req\_dy == -1 && (ch & 2) != 0)  || (req\_dx == 0 && req\_dy == 1 && (ch & 8) != 0))) {  pacmand\_x = req\_dx;  pacmand\_y = req\_dy;  view\_dx = pacmand\_x;  view\_dy = pacmand\_y;  }  }  // Check for standstill  if ((pacmand\_x == -1 && pacmand\_y == 0 && (ch & 1) != 0)  || (pacmand\_x == 1 && pacmand\_y == 0 && (ch & 4) != 0)  || (pacmand\_x == 0 && pacmand\_y == -1 && (ch & 2) != 0)  || (pacmand\_x == 0 && pacmand\_y == 1 && (ch & 8) != 0)) {  pacmand\_x = 0;  pacmand\_y = 0;  }  }  pacman\_x = pacman\_x + PACMAN\_SPEED \* pacmand\_x;  pacman\_y = pacman\_y + PACMAN\_SPEED \* pacmand\_y;  }  private void drawPacman(Graphics2D g2d) {  if (view\_dx == -1) {  drawPacnanLeft(g2d);  } else if (view\_dx == 1) {  drawPacmanRight(g2d);  } else if (view\_dy == -1) {  drawPacmanUp(g2d);  } else {  drawPacmanDown(g2d);  }  }  private void drawPacmanUp(Graphics2D g2d) {  switch (pacmanAnimPos) {  case 1:  g2d.drawImage(pacman2up, pacman\_x + 1, pacman\_y + 1, this);  break;  case 2:  g2d.drawImage(pacman3up, pacman\_x + 1, pacman\_y + 1, this);  break;  case 3:  g2d.drawImage(pacman4up, pacman\_x + 1, pacman\_y + 1, this);  break;  default:  g2d.drawImage(pacman1, pacman\_x + 1, pacman\_y + 1, this);  break;  }  }  private void drawPacmanDown(Graphics2D g2d) {  switch (pacmanAnimPos) {  case 1:  g2d.drawImage(pacman2down, pacman\_x + 1, pacman\_y + 1, this);  break;  case 2:  g2d.drawImage(pacman3down, pacman\_x + 1, pacman\_y + 1, this);  break;  case 3:  g2d.drawImage(pacman4down, pacman\_x + 1, pacman\_y + 1, this);  break;  default:  g2d.drawImage(pacman1, pacman\_x + 1, pacman\_y + 1, this);  break;  }  }  private void drawPacnanLeft(Graphics2D g2d) {  switch (pacmanAnimPos) {  case 1:  g2d.drawImage(pacman2left, pacman\_x + 1, pacman\_y + 1, this);  break;  case 2:  g2d.drawImage(pacman3left, pacman\_x + 1, pacman\_y + 1, this);  break;  case 3:  g2d.drawImage(pacman4left, pacman\_x + 1, pacman\_y + 1, this);  break;  default:  g2d.drawImage(pacman1, pacman\_x + 1, pacman\_y + 1, this);  break;  }  }  private void drawPacmanRight(Graphics2D g2d) {  switch (pacmanAnimPos) {  case 1:  g2d.drawImage(pacman2right, pacman\_x + 1, pacman\_y + 1, this);  break;  case 2:  g2d.drawImage(pacman3right, pacman\_x + 1, pacman\_y + 1, this);  break;  case 3:  g2d.drawImage(pacman4right, pacman\_x + 1, pacman\_y + 1, this);  break;  default:  g2d.drawImage(pacman1, pacman\_x + 1, pacman\_y + 1, this);  break;  }  }  private void drawMaze(Graphics2D g2d) {  short i = 0;  int x, y;  for (y = 0; y < SCREEN\_SIZE; y += BLOCK\_SIZE) {  for (x = 0; x < SCREEN\_SIZE; x += BLOCK\_SIZE) {  g2d.setColor(mazeColor);  g2d.setStroke(new BasicStroke(2));  if ((screenData[i] & 1) != 0) {  g2d.drawLine(x, y, x, y + BLOCK\_SIZE - 1);  }  if ((screenData[i] & 2) != 0) {  g2d.drawLine(x, y, x + BLOCK\_SIZE - 1, y);  }  if ((screenData[i] & 4) != 0) {  g2d.drawLine(x + BLOCK\_SIZE - 1, y, x + BLOCK\_SIZE - 1,  y + BLOCK\_SIZE - 1);  }  if ((screenData[i] & 8) != 0) {  g2d.drawLine(x, y + BLOCK\_SIZE - 1, x + BLOCK\_SIZE - 1,  y + BLOCK\_SIZE - 1);  }  if ((screenData[i] & 16) != 0) {  g2d.setColor(dotColor);  g2d.fillRect(x + 11, y + 11, 2, 2);  }  i++;  }  }  }  private void initGame() {  pacsLeft = 3;  score = 0;  initLevel();  N\_GHOSTS = 6;  currentSpeed = 3;  }  private void initLevel() {  int i;  for (i = 0; i < N\_BLOCKS \* N\_BLOCKS; i++) {  screenData[i] = levelData[i];  }  continueLevel();  }  private void continueLevel() {  short i;  int dx = 1;  int random;  for (i = 0; i < N\_GHOSTS; i++) {  ghost\_y[i] = 4 \* BLOCK\_SIZE;  ghost\_x[i] = 4 \* BLOCK\_SIZE;  ghost\_dy[i] = 0;  ghost\_dx[i] = dx;  dx = -dx;  random = (int) (Math.random() \* (currentSpeed + 1));  if (random > currentSpeed) {  random = currentSpeed;  }  ghostSpeed[i] = validSpeeds[random];  }  pacman\_x = 7 \* BLOCK\_SIZE;  pacman\_y = 11 \* BLOCK\_SIZE;  pacmand\_x = 0;  pacmand\_y = 0;  req\_dx = 0;  req\_dy = 0;  view\_dx = -1;  view\_dy = 0;  dying = false;  }  private void loadImages() {  ghost = new ImageIcon("images/ghost.gif").getImage();  pacman1 = new ImageIcon("images/pacman.png").getImage();  pacman2up = new ImageIcon("images/up1.png").getImage();  pacman3up = new ImageIcon("images/up2.png").getImage();  pacman4up = new ImageIcon("images/up3.png").getImage();  pacman2down = new ImageIcon("images/down1.png").getImage();  pacman3down = new ImageIcon("images/down2.png").getImage();  pacman4down = new ImageIcon("images/down3.png").getImage();  pacman2left = new ImageIcon("images/left1.png").getImage();  pacman3left = new ImageIcon("images/left2.png").getImage();  pacman4left = new ImageIcon("images/left3.png").getImage();  pacman2right = new ImageIcon("images/right1.png").getImage();  pacman3right = new ImageIcon("images/right2.png").getImage();  pacman4right = new ImageIcon("images/right3.png").getImage();  }  @Override  public void paintComponent(Graphics g) {  super.paintComponent(g);  doDrawing(g);  }  private void doDrawing(Graphics g) {  Graphics2D g2d = (Graphics2D) g;  g2d.setColor(Color.black);  g2d.fillRect(0, 0, d.width, d.height);  drawMaze(g2d);  drawScore(g2d);  doAnim();  if (inGame) {  playGame(g2d);  } else {  showIntroScreen(g2d);  }  g2d.drawImage(ii, 5, 5, this);  Toolkit.getDefaultToolkit().sync();  g2d.dispose();  }  class TAdapter extends KeyAdapter {  @Override  public void keyPressed(KeyEvent e) {  int key = e.getKeyCode();  if (inGame) {  if (key == KeyEvent.VK\_LEFT) {  req\_dx = -1;  req\_dy = 0;  } else if (key == KeyEvent.VK\_RIGHT) {  req\_dx = 1;  req\_dy = 0;  } else if (key == KeyEvent.VK\_UP) {  req\_dx = 0;  req\_dy = -1;  } else if (key == KeyEvent.VK\_DOWN) {  req\_dx = 0;  req\_dy = 1;  } else if (key == KeyEvent.VK\_ESCAPE && timer.isRunning()) {  inGame = false;  } else if (key == KeyEvent.VK\_PAUSE) {  if (timer.isRunning()) {  timer.stop();  } else {  timer.start();  }  }  } else {  if (key == 's' || key == 'S') {  inGame = true;  initGame();  }  }  }  @Override  public void keyReleased(KeyEvent e) {  int key = e.getKeyCode();  if (key == Event.LEFT || key == Event.RIGHT  || key == Event.UP || key == Event.DOWN) {  req\_dx = 0;  req\_dy = 0;  }  }  }  @Override  public void actionPerformed(ActionEvent e) {  repaint();  }  } |

Kao što se iz koda već može videti sve funkcije neophodne za funkcionalnost igre kao što su tabla, logika kretanja Pacmana i duhova je uradjena ovde, kao i sami level dizajn.

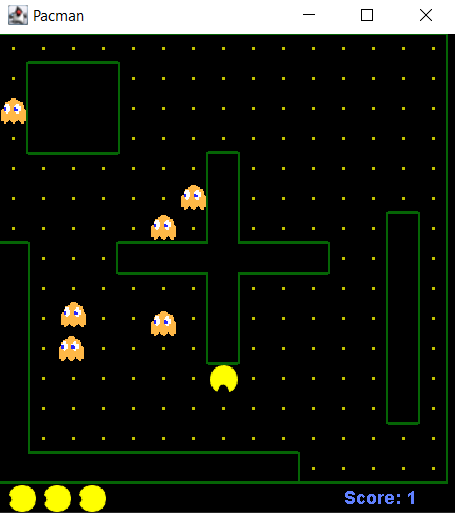
Dizajn nivoa dok je jako jednostavan je samo za prezentaciju i za testirne svrhe novije verzije će imati malo bolji level dizajn.

Kod klase Pacman:

|  |
| --- |
| package Pogledi;  import java.awt.EventQueue;  import javax.swing.JFrame;  import Resursi.Boardpac;  public class Pacman extends JFrame {  public Pacman() {    initUI();  }    private void initUI() {    add(new Boardpac());    setTitle("Pacman");  setDefaultCloseOperation(EXIT\_ON\_CLOSE);  setSize(380, 420);  setLocationRelativeTo(null);  }  public static void main(String[] args) {  EventQueue.invokeLater(() -> {  var ex = new Pacman();  ex.setVisible(true);  });  }  } |

Kao što se iz koda već može videti, ova klasa je main klasa za igricu pacman.

U suštini ona je tu samo kako bismo uspešno pokrenuli igricu.



***Slika 6. izgled Pacman igrice.***

Kao što se sa slike može videti, nivo je jako jednostavan.

Nažalost nisam uspeo da nađem način kako da ubacim zvug za bilo koju od ovih igara.

**NAPOMENA:** Za ubacivanje zvuka je neophodno koristiti JAVA FX iz mog iskustva, u verziji koja bi zapravo funkcionisala u Eclipse.

## **Space Invaders**

Space Invaders je bila jedna od jednostavnijih igara za uraditi u ovom projektu, pogotovo kada su slike u pitanju.

Kao i Pacman ova igra se sastoji iz više klasa koje su Board, SpaceInvaders, Alien, Plazer, Shot i Sprite.

Jedina razlika izmežu Pacmana i Space Invaders-a u komandama je to da u Space Invaders-u igrač ne može ići na gore ili na dole dok u Pacman-u može.

Board klase su im skoro potpuno iste sa razlikom u prethodno navedenim nivoima i dizajnu nivoa kao i da Space Invaders nema sistem za projanje poena ili više života od jednog.

Kod Klase Board:

|  |
| --- |
| package Resursi;  import javax.swing.ImageIcon;  import javax.swing.JPanel;  import javax.swing.Timer;  import java.awt.Color;  import java.awt.Dimension;  import java.awt.Font;  import java.awt.Graphics;  import java.awt.Toolkit;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  import java.awt.event.KeyAdapter;  import java.awt.event.KeyEvent;  import java.util.ArrayList;  import java.util.Iterator;  import java.util.List;  import java.util.Random;  public class Board extends JPanel {  private Dimension d;  private List<Alien> aliens;  private Player player;  private Shot shot;  private int direction = -1;  private int deaths = 0;  private boolean inGame = true;  private String explImg = "src/images/explosion.png";  private String message = "Game Over";  private Timer timer;  public Board() {  initBoard();  gameInit();  }  private void initBoard() {  addKeyListener(new TAdapter());  setFocusable(true);  d = new Dimension(Commons.BOARD\_WIDTH, Commons.BOARD\_HEIGHT);  setBackground(Color.black);  timer = new Timer(Commons.DELAY, new GameCycle());  timer.start();  gameInit();  }  private void gameInit() {  aliens = new ArrayList<>();  for (int i = 0; i < 4; i++) {  for (int j = 0; j < 6; j++) {  var alien = new Alien(Commons.ALIEN\_INIT\_X + 18 \* j,  Commons.ALIEN\_INIT\_Y + 18 \* i);  aliens.add(alien);  }  }  player = new Player();  shot = new Shot();  }  private void drawAliens(Graphics g) {  for (Alien alien : aliens) {  if (alien.isVisible()) {  g.drawImage(alien.getImage(), alien.getX(), alien.getY(), this);  }  if (alien.isDying()) {  alien.die();  }  }  }  private void drawPlayer(Graphics g) {  if (player.isVisible()) {  g.drawImage(player.getImage(), player.getX(), player.getY(), this);  }  if (player.isDying()) {  player.die();  inGame = false;  }  }  private void drawShot(Graphics g) {  if (shot.isVisible()) {  g.drawImage(shot.getImage(), shot.getX(), shot.getY(), this);  }  }  private void drawBombing(Graphics g) {  for (Alien a : aliens) {  Alien.Bomb b = a.getBomb();  if (!b.isDestroyed()) {  g.drawImage(b.getImage(), b.getX(), b.getY(), this);  }  }  }  @Override  public void paintComponent(Graphics g) {  super.paintComponent(g);  doDrawing(g);  }  private void doDrawing(Graphics g) {  g.setColor(Color.black);  g.fillRect(0, 0, d.width, d.height);  g.setColor(Color.green);  if (inGame) {  g.drawLine(0, Commons.GROUND,  Commons.BOARD\_WIDTH, Commons.GROUND);  drawAliens(g);  drawPlayer(g);  drawShot(g);  drawBombing(g);  } else {  if (timer.isRunning()) {  timer.stop();  }  gameOver(g);  }  Toolkit.getDefaultToolkit().sync();  }  private void gameOver(Graphics g) {  g.setColor(Color.black);  g.fillRect(0, 0, Commons.BOARD\_WIDTH, Commons.BOARD\_HEIGHT);  g.setColor(new Color(0, 32, 48));  g.fillRect(50, Commons.BOARD\_WIDTH / 2 - 30, Commons.BOARD\_WIDTH - 100, 50);  g.setColor(Color.white);  g.drawRect(50, Commons.BOARD\_WIDTH / 2 - 30, Commons.BOARD\_WIDTH - 100, 50);  var small = new Font("Helvetica", Font.BOLD, 14);  var fontMetrics = this.getFontMetrics(small);  g.setColor(Color.white);  g.setFont(small);  g.drawString(message, (Commons.BOARD\_WIDTH - fontMetrics.stringWidth(message)) / 2,  Commons.BOARD\_WIDTH / 2);  }  private void update() {  if (deaths == Commons.NUMBER\_OF\_ALIENS\_TO\_DESTROY) {  inGame = false;  timer.stop();  message = "Game won!";  }  // player  player.act();  // shot  if (shot.isVisible()) {  int shotX = shot.getX();  int shotY = shot.getY();  for (Alien alien : aliens) {  int alienX = alien.getX();  int alienY = alien.getY();  if (alien.isVisible() && shot.isVisible()) {  if (shotX >= (alienX)  && shotX <= (alienX + Commons.ALIEN\_WIDTH)  && shotY >= (alienY)  && shotY <= (alienY + Commons.ALIEN\_HEIGHT)) {  var ii = new ImageIcon(explImg);  alien.setImage(ii.getImage());  alien.setDying(true);  deaths++;  shot.die();  }  }  }  int y = shot.getY();  y -= 4;  if (y < 0) {  shot.die();  } else {  shot.setY(y);  }  }  // aliens  for (Alien alien : aliens) {  int x = alien.getX();  if (x >= Commons.BOARD\_WIDTH - Commons.BORDER\_RIGHT && direction != -1) {  direction = -1;  Iterator<Alien> i1 = aliens.iterator();  while (i1.hasNext()) {  Alien a2 = i1.next();  a2.setY(a2.getY() + Commons.GO\_DOWN);  }  }  if (x <= Commons.BORDER\_LEFT && direction != 1) {  direction = 1;  Iterator<Alien> i2 = aliens.iterator();  while (i2.hasNext()) {  Alien a = i2.next();  a.setY(a.getY() + Commons.GO\_DOWN);  }  }  }  Iterator<Alien> it = aliens.iterator();  while (it.hasNext()) {  Alien alien = it.next();  if (alien.isVisible()) {  int y = alien.getY();  if (y > Commons.GROUND - Commons.ALIEN\_HEIGHT) {  inGame = false;  message = "Invasion!";  }  alien.act(direction);  }  }  // bombs  var generator = new Random();  for (Alien alien : aliens) {  int shot = generator.nextInt(15);  Alien.Bomb bomb = alien.getBomb();  if (shot == Commons.CHANCE && alien.isVisible() && bomb.isDestroyed()) {  bomb.setDestroyed(false);  bomb.setX(alien.getX());  bomb.setY(alien.getY());  }  int bombX = bomb.getX();  int bombY = bomb.getY();  int playerX = player.getX();  int playerY = player.getY();  if (player.isVisible() && !bomb.isDestroyed()) {  if (bombX >= (playerX)  && bombX <= (playerX + Commons.PLAYER\_WIDTH)  && bombY >= (playerY)  && bombY <= (playerY + Commons.PLAYER\_HEIGHT)) {  var ii = new ImageIcon(explImg);  player.setImage(ii.getImage());  player.setDying(true);  bomb.setDestroyed(true);  }  }  if (!bomb.isDestroyed()) {  bomb.setY(bomb.getY() + 1);  if (bomb.getY() >= Commons.GROUND - Commons.BOMB\_HEIGHT) {  bomb.setDestroyed(true);  }  }  }  }  private void doGameCycle() {  update();  repaint();  }  private class GameCycle implements ActionListener {  @Override  public void actionPerformed(ActionEvent e) {  doGameCycle();  }  }  private class TAdapter extends KeyAdapter {  @Override  public void keyReleased(KeyEvent e) {  player.keyReleased(e);  }  @Override  public void keyPressed(KeyEvent e) {  player.keyPressed(e);  int x = player.getX();  int y = player.getY();  int key = e.getKeyCode();  if (key == KeyEvent.VK\_SPACE) {  if (inGame) {  if (!shot.isVisible()) {  shot = new Shot(x, y);  }  }  }  }  }  } |

Kod klase Alien:

|  |
| --- |
| package Resursi;  import javax.swing.ImageIcon;  public class Alien extends Sprite {  private Bomb bomb;  public Alien(int x, int y) {  initAlien(x, y);  }  private void initAlien(int x, int y) {  this.x = x;  this.y = y;  bomb = new Bomb(x, y);  var alienImg = "src/images/alien.png";  var ii = new ImageIcon(alienImg);  setImage(ii.getImage());  }  public void act(int direction) {  this.x += direction;  }  public Bomb getBomb() {  return bomb;  }  public class Bomb extends Sprite {  private boolean destroyed;  public Bomb(int x, int y) {  initBomb(x, y);  }  private void initBomb(int x, int y) {  setDestroyed(true);  this.x = x;  this.y = y;  var bombImg = "src/images/bomb.png";  var ii = new ImageIcon(bombImg);  setImage(ii.getImage());  }  public void setDestroyed(boolean destroyed) {  this.destroyed = destroyed;  }  public boolean isDestroyed() {  return destroyed;  }  }  } |

Kod klase Player:

|  |
| --- |
| package Resursi;  import java.awt.event.KeyEvent;  import javax.swing.ImageIcon;  public class Player extends Sprite {  private int width;  public Player() {  initPlayer();  }  private void initPlayer() {  var playerImg = "src/images/player.png";  var ii = new ImageIcon(playerImg);  width = ii.getImage().getWidth(null);  setImage(ii.getImage());  int START\_X = 270;  setX(START\_X);  int START\_Y = 280;  setY(START\_Y);  }  public void act() {  x += dx;  if (x <= 2) {  x = 2;  }  if (x >= Commons.BOARD\_WIDTH - 2 \* width) {  x = Commons.BOARD\_WIDTH - 2 \* width;  }  }  public void keyPressed(KeyEvent e) {  int key = e.getKeyCode();  if (key == KeyEvent.VK\_LEFT) {  dx = -2;  }  if (key == KeyEvent.VK\_RIGHT) {  dx = 2;  }  }  public void keyReleased(KeyEvent e) {  int key = e.getKeyCode();  if (key == KeyEvent.VK\_LEFT) {  dx = 0;  }  if (key == KeyEvent.VK\_RIGHT) {  dx = 0;  }  }  } |

Kod klase Sprite:

|  |
| --- |
| package Resursi;  import java.awt.Image;  public class Sprite {  private boolean visible;  private Image image;  private boolean dying;  int x;  int y;  int dx;  public Sprite() {  visible = true;  }  public void die() {  visible = false;  }  public boolean isVisible() {  return visible;  }  protected void setVisible(boolean visible) {  this.visible = visible;  }  public void setImage(Image image) {  this.image = image;  }  public Image getImage() {  return image;  }  public void setX(int x) {  this.x = x;  }  public void setY(int y) {  this.y = y;  }  public int getY() {  return y;  }  public int getX() {  return x;  }  public void setDying(boolean dying) {  this.dying = dying;  }  public boolean isDying() {  return this.dying;  }  } |

Kod klase Shot:

|  |
| --- |
| package Resursi;  import javax.swing.ImageIcon;  public class Shot extends Sprite {  public Shot() {  }  public Shot(int x, int y) {  initShot(x, y);  }  private void initShot(int x, int y) {  var shotImg = "src/images/shot.png";  var ii = new ImageIcon(shotImg);  setImage(ii.getImage());  int H\_SPACE = 6;  setX(x + H\_SPACE);  int V\_SPACE = 1;  setY(y - V\_SPACE);  }  } |

Kod klase SpaceInvaders:

|  |
| --- |
| package Pogledi;  import java.awt.EventQueue;  import javax.swing.JFrame;  import Resursi.Board;  import Resursi.Commons;  public class SpaceInvaders extends JFrame {  public SpaceInvaders() {  initUI();  }  private void initUI() {  add(new Board());  setTitle("Space Invaders");  setSize(Commons.BOARD\_WIDTH, Commons.BOARD\_HEIGHT);  setDefaultCloseOperation(EXIT\_ON\_CLOSE);  setResizable(false);  setLocationRelativeTo(null);  }  public static void main(String[] args) {  EventQueue.invokeLater(() -> {  var ex = new SpaceInvaders();  ex.setVisible(true);  });  }  } |

Klasa Board i klasa Sprite su dve klase koje su bazirane na funkcionalnosti, mogle bi se zamisliti kao motor igre u suštini one rade sve, kreiraju nivo postavljaju neprijatelje, određuju kako se vanzemanljci kreću itd.

Klase Shot, Player i Alien se mogu gledati kao objekti u ovom projektu zato što od njih klase Board i Sprite nasležuju veliku količinu atributa u smislu kretanja brzine, itd.

Klasa SpaceInvaders je main klasa pomoću koje se pokreće igra.



***Slika 7. Izgled igre Space Invaders***

## **Snake**

Snake je isto jedna jako mala igra i sadrži se iz dve klase bač kao i Pacman.

Igra je bila jako jednostavna za kreirati s obzirom da sam već imao sve neophodne funkcionalnosti iz prethodne dve igre.

Igra se sastoji iz dve klase pod nazivom BoarSn i Snake.

Kod klase BoardSn:

|  |
| --- |
| package Resursi;  import java.awt.Color;  import java.awt.Dimension;  import java.awt.Font;  import java.awt.FontMetrics;  import java.awt.Graphics;  import java.awt.Image;  import java.awt.Toolkit;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  import java.awt.event.KeyAdapter;  import java.awt.event.KeyEvent;  import javax.swing.ImageIcon;  import javax.swing.JPanel;  import javax.swing.Timer;  public class BoardSn extends JPanel implements ActionListener {  private final int B\_WIDTH = 300;  private final int B\_HEIGHT = 300;  private final int DOT\_SIZE = 10;  private final int ALL\_DOTS = 900;  private final int RAND\_POS = 29;  private final int DELAY = 140;  private final int x[] = new int[ALL\_DOTS];  private final int y[] = new int[ALL\_DOTS];  private int dots;  private int apple\_x;  private int apple\_y;  private boolean leftDirection = false;  private boolean rightDirection = true;  private boolean upDirection = false;  private boolean downDirection = false;  private boolean inGame = true;  private Timer timer;  private Image ball;  private Image apple;  private Image head;  public BoardSn() {    initBoard();  }    private void initBoard() {  addKeyListener(new TAdapter());  setBackground(Color.black);  setFocusable(true);  setPreferredSize(new Dimension(B\_WIDTH, B\_HEIGHT));  loadImages();  initGame();  }  private void loadImages() {  ImageIcon iid = new ImageIcon("images/dot.png");  ball = iid.getImage();  ImageIcon iia = new ImageIcon("images/apple.png");  apple = iia.getImage();  ImageIcon iih = new ImageIcon("images/head.png");  head = iih.getImage();  }  private void initGame() {  dots = 3;  for (int z = 0; z < dots; z++) {  x[z] = 50 - z \* 10;  y[z] = 50;  }    locateApple();  timer = new Timer(DELAY, this);  timer.start();  }  @Override  public void paintComponent(Graphics g) {  super.paintComponent(g);  doDrawing(g);  }    private void doDrawing(Graphics g) {    if (inGame) {  g.drawImage(apple, apple\_x, apple\_y, this);  for (int z = 0; z < dots; z++) {  if (z == 0) {  g.drawImage(head, x[z], y[z], this);  } else {  g.drawImage(ball, x[z], y[z], this);  }  }  Toolkit.getDefaultToolkit().sync();  } else {  gameOver(g);  }  }  private void gameOver(Graphics g) {    String msg = "Game Over";  Font small = new Font("Helvetica", Font.BOLD, 14);  FontMetrics metr = getFontMetrics(small);  g.setColor(Color.white);  g.setFont(small);  g.drawString(msg, (B\_WIDTH - metr.stringWidth(msg)) / 2, B\_HEIGHT / 2);  }  private void checkApple() {  if ((x[0] == apple\_x) && (y[0] == apple\_y)) {  dots++;  locateApple();  }  }  private void move() {  for (int z = dots; z > 0; z--) {  x[z] = x[(z - 1)];  y[z] = y[(z - 1)];  }  if (leftDirection) {  x[0] -= DOT\_SIZE;  }  if (rightDirection) {  x[0] += DOT\_SIZE;  }  if (upDirection) {  y[0] -= DOT\_SIZE;  }  if (downDirection) {  y[0] += DOT\_SIZE;  }  }  private void checkCollision() {  for (int z = dots; z > 0; z--) {  if ((z > 4) && (x[0] == x[z]) && (y[0] == y[z])) {  inGame = false;  }  }  if (y[0] >= B\_HEIGHT) {  inGame = false;  }  if (y[0] < 0) {  inGame = false;  }  if (x[0] >= B\_WIDTH) {  inGame = false;  }  if (x[0] < 0) {  inGame = false;  }    if (!inGame) {  timer.stop();  }  }  private void locateApple() {  int r = (int) (Math.random() \* RAND\_POS);  apple\_x = ((r \* DOT\_SIZE));  r = (int) (Math.random() \* RAND\_POS);  apple\_y = ((r \* DOT\_SIZE));  }  @Override  public void actionPerformed(ActionEvent e) {  if (inGame) {  checkApple();  checkCollision();  move();  }  repaint();  }  private class TAdapter extends KeyAdapter {  @Override  public void keyPressed(KeyEvent e) {  int key = e.getKeyCode();  if ((key == KeyEvent.VK\_LEFT) && (!rightDirection)) {  leftDirection = true;  upDirection = false;  downDirection = false;  }  if ((key == KeyEvent.VK\_RIGHT) && (!leftDirection)) {  rightDirection = true;  upDirection = false;  downDirection = false;  }  if ((key == KeyEvent.VK\_UP) && (!downDirection)) {  upDirection = true;  rightDirection = false;  leftDirection = false;  }  if ((key == KeyEvent.VK\_DOWN) && (!upDirection)) {  downDirection = true;  rightDirection = false;  leftDirection = false;  }  }  }  } |

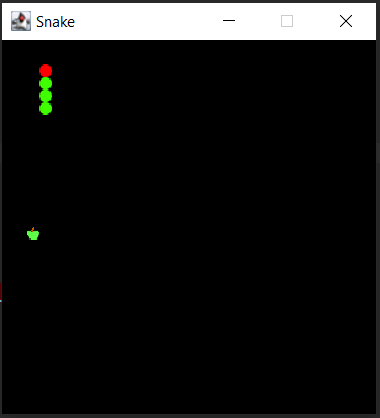
Klasa Board sadrži sve neophodne logičke funkcije igrice kao što su kretanje zmije, stvaranje jabuka, rast zmije, itd...

U suštini ovo je motor igre.

Kod klase Snake:

|  |
| --- |
| package Pogledi;  import java.awt.EventQueue;  import javax.swing.JFrame;  import Resursi.BoardSn;  public class Snake extends JFrame {  public Snake() {    initUI();  }    private void initUI() {    add(new BoardSn());    setResizable(false);  pack();    setTitle("Snake");  setLocationRelativeTo(null);  setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  }    public static void main(String[] args) {    EventQueue.invokeLater(() -> {  JFrame ex = new Snake();  ex.setVisible(true);  });  }  } |

Klasa Sanke jeste main klasa koja služi za pokretanje igre.



***Slika 8. Izgled igre Snake.***

Razlog zbog koga sam odabrao tačkice kao zmiju jeste lakše renderovanje i lakša animacija rasta.

# **ZAKLJUČAK**

Sa svime što sam rekao iznad u ovoj dokumentaciji ovaj projekat je bio previše ambiciozan za količinu vremena i za resurse koje sam imao na raspolaganju.

Naravno nisam očekivao da ću morati izbaciti preko 70% kompletnog projekta zbog limitacija konfiguracije.

Sve u svemu nisam zadovoljan sa količinom urađenog projekta i njegovim kvalitetom.

# **LITERATURA**

Većina literature koja je korištena u ovom projektu se može naći na sajtovima za koje će biti postavljen link ispod, ali i većina urađenog rada proizilazi iz održanih konsultacija sa predmetnim nastavnikom: [**prof. dr Eleonora**](http://www.tfzr.uns.ac.rs/Kadar/NastavnoOsoblje/20) **Brtka.**

Sajtovi:

1. [**https://zetcode.com/javagames/pacman/**](https://zetcode.com/javagames/pacman/)
2. [**https://www.youtube.com/watch?v=bI6e6qjJ8JQ&ab\_channel=BroCode**](https://www.youtube.com/watch?v=bI6e6qjJ8JQ&ab_channel=BroCode)
3. [**https://zetcode.com/javagames/snake/**](https://zetcode.com/javagames/snake/)
4. [**https://sourcecodehero.com/snake-game-in-java-with-source-code/**](https://sourcecodehero.com/snake-game-in-java-with-source-code/)
5. [**https://zetcode.com/javagames/spaceinvaders/**](https://zetcode.com/javagames/spaceinvaders/)
6. [**https://code-projects.org/space-invader-game-in-java-with-source-code/**](https://code-projects.org/space-invader-game-in-java-with-source-code/)