

Tiranë më, 10/02/2021

Detyrë Kursi Nr. 1

Lojë me Zara

Lenda: Programim i Orientuar nga Objekti
Dega: Inxhinieri Informatike

Punoi: Anxhela Halili
Armela Bakiasi
Joana Graci
Megi Rrena
Paskal Loli

Qëllimi i lojes:

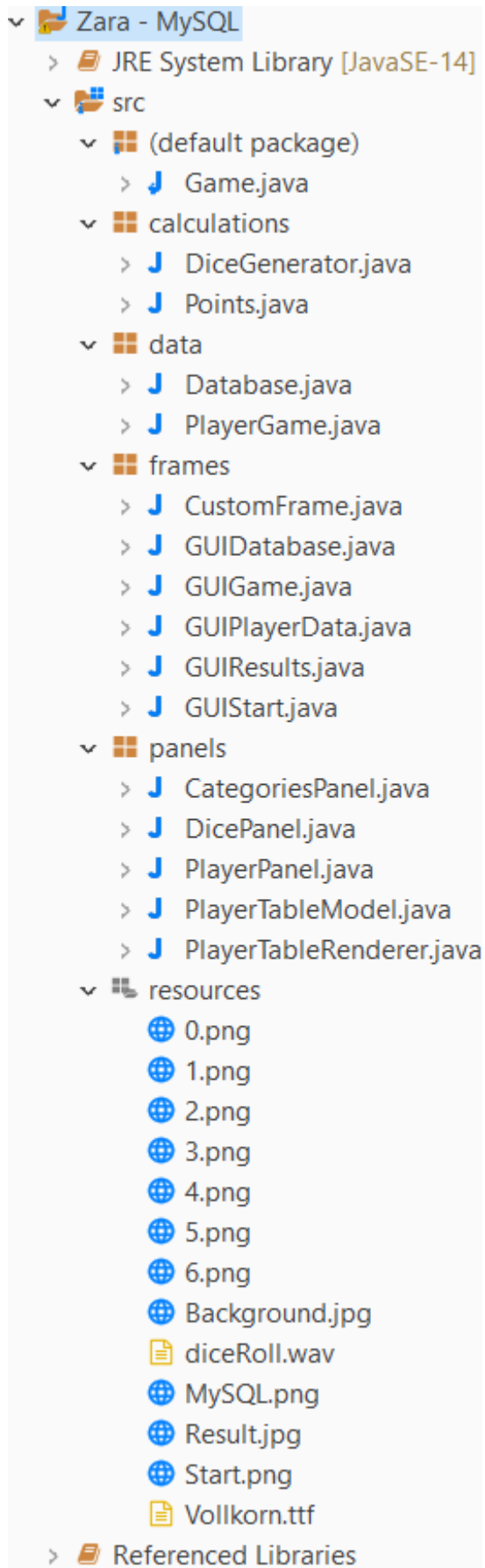
Në këtë lojë ka 5 zara dhe 1 deri në 4 lojtare (ky opsion duhet të zgjidhet nga përdoruesi që në momentin që hapet loja). Sa herë që një lojtar ka rradhën, ai duke shtypur një buton hedh 5 zarat me shpresë që konfigurimi që ka dalë të bëjë pjesë në një nga 13 kategoritë e lojës (kategoritë shpjegohen më poshtë). Nëse hedhja e parë nuk shkon sic duhet, lojtari mund të zgjedhë të hedhë sërish disa ose të gjithë zarat. Nëse dhe hedhja e dytë nuk është e suksesshme, prapë lojtari mund të zgjedhë të hedhë sërish disa ose të gjithë zarat. Në fund të hedhjes së tretë përdoruesi **duhet** të vendosi konfigurimin përfundimtar në një prej 13 kategorive që ndodhen në fushën e lojës. Nëse konfigurimi përputhet me kriteret për atë kategori, atëherë lojtari merr pikët përkatëse të kategorisë, ndryshe pikët për këtë kategori janë 0. Një cikël loje është kur secili prej lojtarëve (1 - 4) ka nga një herë rradhën. Në cdo rradhë një lojtar mund ti hedhë zarat 1 deri në 3 herë. Duke qenë se ka 13 kategori dhe cdo kategori mund të përzgjidhet e shumta një herë, loja ka 13 cikle gjithsej. Pas ciklit të 13-të, gjithë lojtarët do të marrin shumën e pikëve për secilën kategori. Lojtari që ka shumën e pikëve më të lartë është fituesi.

13-të kategoritë për konfigurimin e zarave dhe pikët përkatëse janë:

1. **Njësia:** cdo konfigurim i zarave është i përshtatshëm për këtë kategori. Pikët e fituara janë sa shuma e gjithë 1-ve në pesë zarat, nëse nuk ka 1-sha pikët e fituara janë 0.
- 2-6. **Dysha, Tresha, Katra, Pesa, Gjashta:** (njëlloj si më sipër po për vlera të ndryshme). Cdo konfigurim i zarave është i përshtatshëm për këto kategori. Pikët e fituara janë përkatësisht të njëjta me shumën e 2-ve, 3-ve, 4-ve, 5-ve dhe 6-ve, në pesë zarat.
7. **Tre me një vlerë:** të paktën tre nga zarat duhet të tregojnë të njëjtën vlerë. Pikët janë të njëjta me shumën e gjithë vlerave që kanë 5 zarat.
8. **Katër me një vlerë:** të paktën katër nga zarat duhet të tregojnë të njëjtën vlerë. Pikët janë të njëjta me shumën e gjithë vlerave që kanë 5 zarat.
9. **Tre dhe Dy:** Zarat duhet të tregojnë tre të njëjtën vlerë dhe dy të njëjtën vlerë; pikët që fitohen janë 25.
10. **Katër të njëpasnjëshme:** Zarat duhet të tregojnë të paktën katër vlera të njëpasnjëshme, p.sh. 2-3-4-5. Pikët që fitohen janë 30.
11. **Pesë të njëpasnjëshme:** Zarat duhet të tregojnë pesë vlera të njëpasnjëshme, p.sh. 1-2-3-4-5. Pikët që fitohen janë 40.
12. **E njëjta vlerë:** Të gjithë zarat duhet të tregojnë të njëjtën vlerë. Pikët e fituara janë 50.
13. **Cdo rast:** Gjithë konfigurimet janë të vlefshme për këtë kategori. Pikët e fituara janë të barabarta me shumën e vlerave që tregojnë të pesë zarat.

Vlera të “Piket e siperme” llogaritet në fund të lojës, duke bërë shumën e pikeve të kategorive me siper dhe nëse i tejkalon 63 piket, vendoset të rreshti “Bonus”, vlera 35, në të kundërt vendoset vlera 0. Vlera të “Piket e poshtme” llogaritet duke bërë shumën e pikeve të fituara në kategoritë e poshtme (pas rreshtit “Bonus”). Totali llogaritet si shuma e pikeve të siperme, bonusit dhe pikeve të poshtme. Cdo lojtar duhet të jetë regjistruar para se të luajë. Të dhënat për gjithë lojtarët që kanë luajtur me parë duhet të ruhen në një **baze të dhënash**. Nëse një lojtar luan për herë të parë, para fillimit të lojës duhet

te plotesoje regjistrimin. Te dhenat qe do te ruhen ne databaze per secilin lojtar jane: Emri, Mbiemri, mosha dhe piket qe ka fituar cdo here qe ka luajtur. Secili perdorues i aplikacionit duhet te kete mundesi te shohe rezultatin me te larte dhe rezultatin me te ulet qe eshte arritur deri ne ate moment (edhe emrin e lojtarit qe e ka arritur ate rezultat).



Projekti **Zara – MySQL**

Projekti përdor si librari të referencuar **Connector/J 8.0.23** meqënëse MySQL është përdorur si sistemi për të menaxhuar bazat e të dhënave që do të ruajnë të dhënat e lojtarëve.

Projekti përbëhet nga disa **paketa**.

(default package) përbëhet vetem nga klasa **Game.java (f.6)** që përmban metodën **main()**.

calculations përbëhet nga **DiceGenerator.java (f.6)** që përmban metoda për të gjeneruar zara të rinj gjatë lojës dhe **Points.java (f.7)** që përmban metoda për të llogaritur pikët për kombinime të ndryshme zarash për kategoritë.

data përbëhet nga **Database.java (f.8)** që përmban metoda për tu lidhur me MySQL, krijuar bazën e nevojshme të të dhënave, tabelat, aksesimin e të dhënave dhe ruajtjen e tyre, dhe **PlayerGame.java (f.9)** që përmban një ArrayList me të dhënat e lojtarëve (të organizuar në klasë PlayerGame) që po luajnë për momentin dhe metoda të ndryshme për të aksesuar ose modifikuar këto të dhëna.

frames përbëhet nga:

- **CustomFrame.java (f.10)** extends JFrame (ka një emër specifik, nuk është resizable dhe ka DefaultCloseOperation EXIT_ON_CLOSE). Në klasë ka një metodë për të shtuar një imazh në sfond të frame-it dhe një metodë që përcakton LookAndFeel të gjithë GUI-ve.

- **GUIDatabase.java (f. 11)** extends CustomFrame. Përdoret për të marrë të dhënat (userName, password, serverName, portNumber) për tu lidhur me MySQL (Lojëtari mund të zgjedhë të mos përdor MySQL).

- **GUIStart.java (f. 12)** extends CustomFrame. Përdoret për të caktuar numrin e lojtarëve që do luajnë.

- **GUIPlayerData.java (f. 13)** extends CustomFrame. Përdoret për të marrë të dhënat e lojtarëve (emër, mbiemër, moshë).

- **GUIGame.java (f. 14)** extends CustomFrame. Është frame-i kryesor i lojës.

- **GUIResults.java (f. 15)** extends CustomFrame. Përdoret për të shfaqur pikët e secilit lojtar në fund të lojës, fituesin dhe lojtarin me pikët më të larta/ulëta nga baza e të dhënave në MySQL.

panels përbëhet nga:

- **CategoriesPanel.java (f. 17)** extends JPanel. Përdoret për të shfaqur në GUIGame një JTable me kolonë header “Kategoritë” dhe rreshta kategoritë e ndryshme.
- **DicePanel.java (f. 18)** extends JPanel. Përdoret për të shfaqur zaret e gjeneruar në GUIGame.
- **PlayerPanel.java (f.18)** extends JPanel. Përdoret për të shfaqur një JTable me kolonë header emrin e lojtarit dhe rreshta pikët përkatëse për kategoritë në GUIGame.
- **PlayerTableModel.java (f.19)** extends DefaultTableModel (font, numra kolonash, rreshtash, ngjyra të përcaktuara). Përdoret si model i JTable në PlayerPanel.
- **PlayerTableRenderer.java (f.20)** extends DefaultTableCellRenderer. Përdoret për të gjeneruar qelizat e kolonës së tabelave të lojtarëve (gjeneron vlerat dhe ngjyra të përshtatshme për qelizat përkatëse të zgjedhura ose jo nga lojtarët).

resources është një folder që përmban imazhe të zarave, sfondeve. fontin që përdoret (Vollkorn) dhe një wav audio (efekt zanor) që përdoret kur hidhen zare.

Game.java

```
1 |
2● import frames.CustomFrame;
3 import frames.GUIDatabase;
4
5 public class Game {
6●     public static void main(String[] args) {
7         CustomFrame.setLookAndFeel(); //percakton LookAndFeel te GUI-t
8         new GUIDatabase(); //krijon GUI-n
9     }
10 }
```

Klasa përmban metodën **main()** përgjegjëse për ekzekutimin e programit. Metoda **main()** vendos **LookAndFeel** `com.sun.java.swing.plaf.windows.WindowsLookAndFeel` që do të përdoret në çdo GUI dhe percakton fontin `Vollkorn.ttf` si font default për komponente të ndryshme që janë përdorur në ndërtimin e GUI-t duke thërritur metodën **setLookAndFeel()** nga klasa **CustomFrame.java**. Gjithashtu, **main()** krijon **GUIDatabase** që shfaq frame-in për përdoruesin.

DiceGenerator.java

```
7 public class DiceGenerator {
8     public static List<Integer> diceNumbers = new ArrayList<>(List.of(0,0,0,0,0));
9     private static Random rand = new Random();
10
11● public static void generateDiceAll(){//gjeneron zaret te rinj per te gjithë zaret
12     for(int i=0;i<diceNumbers.size();i++) {
13         DiceGenerator.diceNumbers.set(i, rand.nextInt(6)+1);
14     }
15 }
16
17● public static void generateDiceSelect() { //gjeneron zaret te rinj vetem per zaret qe jane zgjedhur per tu hequr (vlere -1)
18     for(int i=0;i<diceNumbers.size();i++) {
19         if(diceNumbers.get(i)==-1)
20             DiceGenerator.diceNumbers.set(i, rand.nextInt(6)+1);
21     }
22 }
23 }
```

Klasa përmban **ArrayList<Integer> diceNumbers** që ruan vlerat e zareve për lojëtarët gjatë lojës dhe **Random rand** që përdoret për të gjeneruar vlera të reja për zaret. Metoda **generateDiceAll()** gjeneron vlera të reja për të gjithë zaret në **diceNumbers**, ndërsa **generateDiceSelect()** gjeneron vlera të reja vetëm për zaret e zgjedhur për tu hequr nga përdoruesi.

Points.java

Përbëhet nga metoda të ndryshme që bejnë llogaritjen e pikëve për vlera të ndryshme zarash sipas udhëzimeve në kërkesë të detyrës.

```
6 public class Points {
7
8     public static int addValueDice(List<Integer> n, int valueToAdd) { //kthen vlerën e zareve me valueToAdd pikë
15
16     public static int threeofakind(List<Integer> n) { //kthen vlerën e zareve kur ka 3 të njëjta
29
30     public static int fourofakind(List<Integer> n) { //kthen vlerën e zareve kur ka 4 të njëjta
40
41     public static int fiveofakind(List<Integer> n) { //kthen 50 pikë kur ka 5 të njëjta
46
47     public static int threeandtwo(List<Integer> n) { //kthen 25 pikë kur ka 3 të njëjta dhe 2 të njëjta
53
54     public static int fourSequence(List<Integer> n) { //kthen 30 pikë kur një sekuenca prej 4 zarësh me vlera të njëpasnjëshme
63
64     public static int fiveSequence(List<Integer> n) { //kthen 40 pikë kur një sekuenca prej 5 zarësh me vlera të njëpasnjëshme
72
73     public static int addAllDice(List<Integer> n) { //kthen vlerën e gjithë zareve
79 }
```

addValueDice() kthen vlerën e zareve me valueToAdd pikë, si për shembull addValueDice(n, 1) do të llogarisë shumën e gjithë 1-shave në n. **[Kategoria 1-6]**

threeofakind() kthen vlerën e gjithë zareve kur ka 3 zara të njëjtë. **[Kategoria 7]**

fourofakind() kthen vlerën e gjithë zareve kur ka 4 zara të njëjtë. **[Kategoria 8]**

fiveofakind() kthen 50 pikë kur ka 5 zara të njëjtë. **[Kategoria 12]**

threeandtwo() kthen 25 pikë kur ka një kombinim prej 3 dhe 2 zarësh të njëjtë respektivisht, si 2-2-2-3-3. **[Kategoria 9]**

fourSequence() kthen 30 pikë kur ka një kombinim zarësh ku 4 zara janë me vlera të njëpasnjëshme si 2-3-4-5. **[Kategoria 10]**

fiveSequence() kthen 40 pikë kur ka një kombinim zarësh ku 5 zara janë me vlera të njëpasnjëshme si 1-2-3-4-5. **[Kategoria 11]**

addAllDice() kthen vlerën e gjithë zareve. **[Kategoria 13]**

Database.java

```
1 package data;
2 import java.sql.Connection;
10
11 public class Database {
12
13     private static Connection conn = null; //nje lidhje me nje database
14     private static Statement stmt = null; //objekt per te ekzekutuar statements ne MySQL
15     private static ResultSet rs; //nje tabelë te dhenash qe merren si rezultat nga nje query ne MySQL
16
17     public static boolean getConnection(String userName, String password, String serverName, int portNumber) {}
34
35     public static void createPlayer(String fName, String lName, String age) { //krijon nje lojtar te ri ne database
40
41     public static int getCurrentPlayCount(String fName, String lName) { //kthen numrin e lojes qe ndodhet lojetari qe po luan ne moment
50
51     public static void highestScore() { //percakton ne rs rreshtin ne scoreTable me piket me te larta
57
58     public static void lowestScore() { //percakton ne rs rreshtin ne scoreTable me piket me te uleta
64
65     public static Object[] calculateHighestLowest() { //kthen nje array me emrin, mbiemrin, numrin e lojes dhe piket qe ka marre lojetari me pike me te larta/uleta
84
85     public static void writeDatabaseData() { //kalon te dhenat e lojtareve qe po luajne ne moment ne database ne fund te lojes
96 }
```

Connection conn është një objekt që mban lidhjen me MySQL.

Statment stmt është një objekt që përdoret për të ekzekutuar statements në MySQL.

ResultSet rs është një tabelë të dhënash që merren si rezultat nga një query në MySQL.

getConnection() vendos lidhjen me MySQL, përditëson stmt, krijon bazen e të dhënave **playerDatabase**, tabelat **playerTable** (që përmban emrin, mbiemrin dhe moshën e lojtarëve [emri dhe mbiemeri janë primary key]) dhe **scoreTable** (që përmban emrin, mbiemrin, numrin e lojës dhe pikët që kanë marrë). Baza dhe tabelat krijohen vetëm nëse nuk ekzistojnë.

createPlayer() krijon një rekord të ri në playerTable të një lojëtari. Në rast se lojëtari ekziston (emër dhe mbiemër i njëjtë), përditësohet mosha.

getCurrentPlayCount() kthen numrin e lojës që ndodhet lojëtari që po luan në moment (numri i lojës është numri më i madh që mund të gjendet në scoreTable + 1 ose 1 nëse nuk ekziston rekorde të mëparshme për këtë lojtar.

highestScore()/lowestScore() hedh në rs rreshtin e lojtarit me pikë më të larta/ulëta në scoreTable.

calculateHighestLowest() kthen një Object[] që përmban {Emrin e Lojetarit me Pike me të Larta, Mbiemrin, Numrin e Lojes, Pikët, Emrin e Lojetarit me Pike me te Uleta, Mbiemrin, Numrin e Lojës, Pikët}.

writeDatabaseData() hedh të dhënat e lojtarëve që po luajnë në moment në scoreTable në fund të lojës.

PlayerGame.java

```
1 package data;
2
3 import java.util.ArrayList;
4
5 public class PlayerGame{
6
7     public static ArrayList<PlayerGame> CurrentPlayers = new ArrayList<PlayerGame>(); //mban te dhenat e lojtareve qe po luajne per momentin
8
9     private String fName; //emri i lojtarit
10    private String lName; //mbiemeri
11    private int playCount; //numri i lojes
12    private int rollsMade; //hedhjet qe kane bere kete cikël
13    private boolean selectedThisTurn; //nese kane zgjedhur nje kategori kete cikël
14
15    public PlayerGame(String fName, String lName) { //konstruktori
16        //aksessore
17    }
18    public String getFirstName() {
19    }
20    public String getLastName() {
21    }
22    public int getPlayCount() {
23    }
24    public int getRollsMade() {
25    }
26    public boolean getSelectedThisTurn() {
27    }
28    //mutators
29    public void setSelectedThisTurn(boolean selectedThisTurn) {
30    }
31    public void setRollsMade(int rollsMade) {
32    }
33    public void incRollsMade() { //rrrit numrin e hedhjeve deri ne 3 dhe me pas ne 0
34    }
35 }
```

ArrayList<PlayerGame> CurrentPlayers është një ArrayList e klasës PlayerGame që përmban të dhëna e lojtarëve që po luajnë në moment (**fName** – emër, **lName** – mbiemër, **playCount** – numri i lojës të momentit, **rollsMade** – hedhje të zareve që kanë realizuar në një cikël loje dhe **selectedThisTurn** – nëse kanë zgjedhur një kategori në këtë cikël apo jo).

getFirstName(), **getLastName()**, **getPlayCount()**, **getRollsMade()**, **getSelectedThisTurn()** janë aksesore (kthejnë vlerën e të dhënës respektive) të të dhënave.

setSelectedThisTurn() dhe **setRollsMade()** janë mutator (ndryshojnë vlerën e të dhënave respektive).

incRollsMade() është një metodë që rrit numrin e hedhjeve me 1 deri në 3 dhe më pas prapë në 0 nga 3.

CustomFrame.java

```
20● CustomFrame(String name) {
21    super(name);
22    setResizable(false);
23    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
24    //krijon nje frame me emrin name, jo resizable dhe EXIT_ON_CLOSE si DefaultCloseOperation
25
26● public void addBackground(String name) {
27    try {
28        JLabel background;
29        Image image;
30        image = ImageIO.read(new File("src/resources/"+name));
31        background = new JLabel(new ImageIcon(image));
32        setContentPane(background);
33
34    } catch (IOException e) {}
35    } //shton nje JLabel me imazh si ContentPane te frame-it
36
37● public static void setLookAndFeel() { //percakton paraqitjen e GUIt, LookAndFeel te pergjithshem dhe font-et per elementet
38    try {
39        UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");
40
41        GraphicsEnvironment ge = GraphicsEnvironment.getLocalGraphicsEnvironment();
42        ge.registerFont(Font.createFont(Font.TRUETYPE_FONT, new File("src/resources/Vollkorn.ttf")));
43        Font myFont = new Font("Vollkorn", Font.PLAIN, 16);
44
45        UIManager.put("CheckBoxMenuItem.acceleratorFont", myFont);
46        UIManager.put("Button.font", myFont);
47        UIManager.put("CheckBox.font", myFont);
48        UIManager.put("ComboBox.font", myFont);
49        UIManager.put("Label.font", myFont);
50        UIManager.put("OptionPane.buttonFont", myFont);
51        UIManager.put("OptionPane.messageFont", myFont);
52        UIManager.put("Menu.font", myFont);
53        UIManager.put("PopupMenu.font", myFont);
54        UIManager.put("OptionPane.font", myFont);
55        UIManager.put("Panel.font", myFont);
56        UIManager.put("ProgressBar.font", myFont);
57        UIManager.put("ScrollPane.font", myFont);
58        UIManager.put("Viewport.font", myFont);
59        UIManager.put("TabbedPane.font", myFont);
60        UIManager.put("Slider.font", myFont);
61        UIManager.put("Table.font", myFont);
62        UIManager.put("TableHeader.font", new Font("Vollkorn", Font.BOLD, 16));
63        UIManager.put("TextField.font", myFont);
```

CustomFrame() krijon një JFrame të modifikuar (me emër `name`, jo resizable dhe DefaultCloseOperation EXIT_ON_CLOSE).

addBackground() vendos si ContentPane për CustomFrame një JLabel që përmban një imazh me emër `name` në folderin src/resources/.

setLookAndFeel() vendos LookAndFeel `com.sun.java.swing.plaf.windows.WindowsLookAndFeel` që do të përdoret në çdo GUI dhe percakton fontin `Vollkorn.ttf` si font default për komponente të ndryshme që janë përdorur në ndërtimin e GUI.

GUIDatabase.java

```
14 public class GUIDatabase extends CustomFrame implements ActionListener { //implementon ActionListener per te detektuar shtypjen e butonave
15     private static final long serialVersionUID = 1L;
16
17     private static JTextField userName = new JTextField("root", 10);
18     private static JTextField password = new JTextField("root", 10);
19     private static JTextField serverName = new JTextField("localhost", 10);
20     private static JTextField portNumber = new JTextField("3306", 10);
21     private static JButton ok = new JButton("Next");
22     private static JButton noData = new JButton("Luaj Pa Database");
23
24     public GUIDatabase() { //konstruktor i frame-it qe merr te dhenat per tu lidhur me MySQL
25         super("MySQL");
26
27         addBackground("MySQL.png");
28         setLayout(new GridLayout(5, 2, 0, 5)); //vendos GridLayout() si menaxherin default te frame-it, 5 rreshta 2 kolona 0 hapesire horizontale
29
30         add(new JLabel("Username:", JLabel.CENTER)); add(userName); //shtojme komponentet ne frame
31         add(new JLabel("Password:", JLabel.CENTER)); add(password);
32         add(new JLabel("Server Name:", JLabel.CENTER)); add(serverName);
33         add(new JLabel("Port Number:", JLabel.CENTER)); add(portNumber);
34         add(noData); noData.addActionListener(this);
35         add(ok); ok.addActionListener(this);
36
37         pack(); //ben qe frame-i te pershtatet ne gjatesi dhe gjeresi me komponentet e veta
38         setLocationRelativeTo(null); //vendos GUI ne qender te ekranit
39         setVisible(true); //ben frame-in te dukshem
40     }
41
42     @Override
43     public void actionPerformed(ActionEvent e) {
44         if (e.getSource() == ok) {
45             if (Database.getConnection(userName.getText(), password.getText(), serverName.getText(),
46                 Integer.parseInt(portNumber.getText())) == false) { //teston nese u krijua lidhje me MySQL
47                 JOptionPane.showMessageDialog(null, "Ju lutem kontrolloni te dhenat!",
48                     "Nuk u krijua dot nje lidhje me MySQL.", JOptionPane.ERROR_MESSAGE); //nese jo, shfaq error
49             } else {
50                 dispose();
51                 new GUIStart(); //nese po, heq frame-in ekzistent dhe krijon nje framr GUIStart()
52             }
53         }
54         if (e.getSource() == noData) {
55             dispose();
56             new GUIStart(); //krijon GUIStart(), pa u munduar te krijojte lidhje me MySQL
57         }
58     }
59 }
```

GUIDatabase() krijon një CustomFrame të modifikuar (GridLayout [5 rreshta, 2 kolona, 0 hapësirë horizontale, 5 hapësirë vertikale], sfond MySQL.png), që përmban 4 **JLabel**, 4 **JTextField** ku përdoruesi vendos të dhënat për tu lidhur me databazën MySQL, dhe 2 **JButton**.

GUIDatabase implementon **ActionListener** që detekton shtypjen e JButton-ave. Nëse shtypet butoni **noData**, programi vazhdon ekzekutimin duke injoruar nëse u lidh me MySQL ose jo. Nëse shtypet butoni **ok**, programi përpiqet të lidhet me MySQL me të dhënat e dhëna nga përdoruesi. Në rast se të dhënat janë të pasakta, ka një problem me instalimin e MySQL ose ajo nuk është e hapur, shfaqet një **MessageDialog** për të lajmëruar përdoruesin që ka një problem. Në rast se të dhënat janë të sakta, lidhja kryehet me sukses dhe programi vazhdon në **GUIStart()**.

GUIStart.java

```
12 public class GUIStart extends CustomFrame implements ActionListener {
13     private static final long serialVersionUID = 1L;
14
15     private JLabel playerNrAsk = new JLabel("Numri i lojtareve: ");
16     private JCheckBox nr1 = new JCheckBox("1 Lojtar", true);
17     private JCheckBox nr2 = new JCheckBox("2 Lojtar", false);
18     private JCheckBox nr3 = new JCheckBox("3 Lojtar", false);
19     private JCheckBox nr4 = new JCheckBox("4 Lojtar", false);
20     private JButton start = new JButton("Loje e re!");
21
22     public GUIStart(){//krijon nje frame qe merr sa lojtar do luajne
23         super("Zara");
24         addBackground("Start.png");
25         setLayout(new FlowLayout());
26
27         add(playerNrAsk);
28
29         ButtonGroup playerNr = new ButtonGroup();
30         playerNr.add(nr1);
31         playerNr.add(nr2);
32         playerNr.add(nr3);
33         playerNr.add(nr4);
34         add(nr1);
35         add(nr2);
36         add(nr3);
37         add(nr4);
38
39         start.addActionListener(this);
40         add(start);
41
42         pack();
43         setLocationRelativeTo(null);
44         setVisible(true);
45     }
46
47     @Override
48     public void actionPerformed(ActionEvent e) {//krijon nje GUIPlayerData() frame me aq rreshta sa lojtar te perzgjedhur
49         dispose();
50         if(nr1.isSelected()) {
51             new GUIPlayerData(1);
52         }else if(nr2.isSelected()) {
53             new GUIPlayerData(2);
54         }else if(nr3.isSelected()) {
55             new GUIPlayerData(3);
56         }else {
57             new GUIPlayerData(4);
58         }
59     }
```

GUIStart() krijon një CustomFrame të modifikuar (FlowLayout(), sfond Start.png), që përmban 1 **JLabel**, 4 **JCheckBox** në një **ButtonGroup** (në mënyrë që të zgjidhet vetëm një button dhe jo më shumë) dhe një **JButton** Start.

GUIStart implementon **ActionListener** që detekton shtypjen e butonit **Start**. Në varësi të **JCheckBox** të zgjedhur (1, 2, 3 ose 4) do të krijohet një **GUIPlayerData()** me argument numrin përkatës.

GUIPlayerData.java

```
19 public class GUIPlayerData extends CustomFrame implements ActionListener {
20     private static final long serialVersionUID = 1L;
21
22     private ArrayList<JPanel> playerRow = new ArrayList<JPanel>(); //rreshti qe kerkon te dhenat e lojtarit
23     private ArrayList<JTextField> firstNamesInput = new ArrayList<JTextField>(); //mban JTextField ku shkruhet emri
24     private ArrayList<JTextField> lastNamesInput = new ArrayList<JTextField>(); //mban JTextField ku shkruhet mbiemeri
25     private ArrayList<JComboBox<Integer>> ageInput = new ArrayList<JComboBox<Integer>>(); //mban JComboBox per moshen
26
27     public GUIPlayerData(int n){//krijon nje frame qe merr te dhenat e lojtareve
28         super("Lojtaret");
29         setLayout(new BorderLayout());
30
31         JPanel allRows = new JPanel(new GridLayout(n,1)); //n rreshta 1 kolone
32         for(int i=0;i<n;i++) {
33             playerRow.add(new JPanel());
34             playerRow.get(i).add(new JLabel("Emri i lojtarit nr. "+(i+1)+": ", JLabel.CENTER));
35             firstNamesInput.add(new JTextField(10));
36             playerRow.get(i).add(firstNamesInput.get(i));
37             playerRow.get(i).add(new JLabel("Mbiemri: ", JLabel.CENTER));
38             lastNamesInput.add(new JTextField(10));
39             playerRow.get(i).add(lastNamesInput.get(i));
40             playerRow.get(i).add(new JLabel("Moshë: ", JLabel.CENTER));
41             ageInput.add(new JComboBox<Integer>());
42             for(int l=0;l<100;l++)
43                 ageInput.get(i).addItem(l);
44             playerRow.get(i).add(ageInput.get(i));
45             allRows.add(playerRow.get(i));
46         }
47         add(allRows, BorderLayout.CENTER);
48
49         JButton next = new JButton("Vazhdo!");
50         next.addActionListener(this);
51         add(next, BorderLayout.SOUTH);
52
53         pack();
54         setLocationRelativeTo(null);
55         setVisible(true);
56     }
57
58     @Override
59     public void actionPerformed(ActionEvent e) {
```

ArrayList<JPanel> playerRow përmban rreshtat që kërkojnë të dhënat për një lojtar specifik. Madhësia e ArrayList përputhet me numrin e lojtarëve. Për çdo indeks në konstruktor vendoset në playerRow.get(i) 3 **JLabel**, 2 **JTextField** (për të futur emër, mbiemër) dhe 1 **JComboBox** (për të zgjedhur moshën).

ArrayList<JTextField> firstNamesInput/lastNamesInput përmban JTextField për të futur emër/mbiemër për një lojtar specifik. Teksti në JTextField përdoret më vonë për të shtuar të dhënat e lojtarëve në databazë dhe ArrayList<GamePlayer> CurrentPlayers.

ArrayList<JComboBox> ageInput përmban Items nga 0-99, që përdoruesi mund të zgjedh moshën.

GUIPlayerData() krijon një CustomFrame të modifikuar (GridLayout(n rreshta, 1 kolonë)) që përmban rreshta për të kërkuar të dhënat e çdo lojtari dhe një **JButton next** që teston të dhënat input dhe vendos vazhdimin apo jo të programit. GUIPlayerData() implementon **ActionListener()** që përdoret për të detektuar shtypjen e butonit next.

Në shtypje të butonit **next**, në rast se një prej **JTextField** është bosh ose dy lojtar kanë emër dhe mbiemër të njëjtë shfaqet një MessageDialog që kërkon lojtarëve të kontrollojnë inputin. Ndryshe, krijohet një rekord për lojtarin në playerTable në databazë dhe shtohet në CurrentPlayers.

GUIGame.java

```
21 public class GUIGame extends CustomFrame implements ActionListener {
22     private static final long serialVersionUID = 1L;
23
24     private static boolean gameStarted = false; //boolean-i perdoret ne fillim te lojes per te percaktuar nese duhet nje button "Fillo Lojen"
25     public static int playerIndex = -1; //mban indeksin e lojtarit qe ka rradhen
26     public static int cycleCount = 0; //mban numrin e cikleve qe kane kaluar ne loje
27
28     private static JButton startButton = new JButton("Fillo Lojen!");
29     private static JButton throwDiceButton = new JButton();
30     private static JLabel nameTurn = new JLabel("", SwingConstants.CENTER);
31     private static JLabel diceChances = new JLabel("", SwingConstants.CENTER);
32
33     public GUIGame() { //frame-i kryesor i lojes
34
35
36
37
38
39
40
41
42
43
44
45     private void addToFrame() {
46
47
48
49
50
51     private void addMainPanels() { // panels = zarat+kategorite+player panel
52
53
54
55
56
57
58
59
60
61
62     private void addStartPanel() { //shton ne frame nje panel me startButton
63
64
65
66
67
68
69
70
71     private void addInteractPanel() { //shton ne frame panel me throwDiceButton dhe JLabel qe shfaq lojtarin qe ka rradhen dhe sa mundesi per te hedhur zaret
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113     private void resetGUI() { //vizaton serish komponentet ne frame
114
115
116
117
118
119
120
121     public void actionPerformed(ActionEvent e) {
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139 }
```

boolean gameStarted përcakton në fillimin e lojës nëse butoni që do shfaqet është “Fillo Lojen” (false – loja nuk ka filluar) apo “Hidh Zaret” (true – loja ka filluar).

int playerIndex ruan indeksin e lojtarit që ka rradhen për momentin. Fillon me -1 përpara se të ketë “filluar” loja dhe vazhdon me 0, deri në lojtari me indeks më të lartë (1, 2 ose 3) dhe prapë në 0.

int cycleCount ruan numrin e cikleve të lojës që kanë kaluar (sa herë është kthyer indeksi në 0. Në 13 cikle loja përfundon.

GUIGame() krijon një CustomFrame të modifikuar (sfond Background.jpg, BorderLayout()). Mbi GUIGame, **addToFrame()** thërret **addMainPanels()** dhe **addStartPanel()** ose **addInteractPanel()** (nëse nuk ka filluar ose ka filluar loja). GUIGame implementon **ActionListener** për të përcaktuar kur shtypet JButton-at.

addMainPanels() shton panele për shfaqjen e zareve, një tabelë kategorishë dhe për tabelat e lojtarëve ku mund të përzgjedhin kategoritë.

addStartPanel() shton poshtë paneleve (BorderLayout.SOUTH) një JButton për të filluar lojën.

addInteractPanel() shton poshtë paneleve (BorderLayout.SOUTH) JButton throwDiceButton (për të hedhur zaret), JLabel nameTurn (që shfaq cili lojtar ka rradhën sipas indeksit) dhe JLabel diceChances (që shfaq sa mundësi ka një lojtar për të hedhur zare).

resetGUI() heq gjithçka nga ContentPane i frame-it, rivendos panelet e përditësuar, revalidate() layout-in dhe repaint() çdo komponent.

actionPerformed() ekzekutohet kur shtypet butoni “Fillo Lojen” ose “Hidh Zaret”. Në rast të startButton gjenerohen zare të rinj, gameStarted merr vlerën true, playerIndex shkon në 0 dhe thirret resetGUI(). Në rast të throwDiceButton, gjenerohen zare të rinj për zaret e zgjedhur për tu hequr dhe

rritet numri i mundësive të përdorura. Nëse një kategori është zgjedhur, gjenerohen zare krejtësisht të rinj dhe rritet indeksi i lojëtarit që ka rradhën. Nëse indeksi është kthyer në 0, ka kaluar një cikël. Nëse cycleCount arrin vlerën 13, GUIGame() hiqet [dispose()] dhe krijohet GUIResults().

GUIResults.java

```
24 public GUIResults() { //frame-i shfaq piket e lojtareve, fituesin dhe lojëtarin me piket me te larte/uleta nga MySQL
25     super("Zara");
26     addBackground("Result.jpg");
27     setLayout(new BorderLayout());
28
29     JPanel players = new JPanel();
30     players.setOpaque(false);
31     players.setBorder(new EmptyBorder(10, 10, 10, 10));
32
33     JPanel top = new JPanel(new GridLayout(2,1));
34     top.setOpaque(false);
35     top.setBorder(new EmptyBorder(10, 10, 10, 10));
36
37     ArrayList<JLabel> playerRows = new ArrayList<JLabel>();
38     int i=0;
39     for(; i<PlayerGame.CurrentPlayers.size();i++) {
40         playerRows.add(new JLabel("", SwingConstants.CENTER));
41         playerRows.get(i).setForeground(Color.WHITE);
42         playerRows.get(i).setText(PlayerGame.CurrentPlayers.get(i).getFirstName()+" : "
43             +PlayerTableRenderer.total.get(i)+" Pike");
44         playerRows.get(i).setForeground(Color.WHITE); playerRows.get(i).setFont(new Font("Vollkorn", Font.PLAIN, 18));
45         players.add(playerRows.get(i));
46     } //rradhit lojteret dhe sa pike kane marre
47
48     int max = -1;
49     for(int j=0; j<PlayerGame.CurrentPlayers.size();j++) {
50         if(max<PlayerTableRenderer.total.get(j))
51             max=PlayerTableRenderer.total.get(j);
52     }
53     int indexWinner = PlayerTableRenderer.total.indexOf(max); //percakton indeksin e lojterit me me shume pike totale
54
55     playerRows.add(new JLabel("Urime! "+PlayerGame.CurrentPlayers.get(indexWinner).getFirstName()
56         +" eshte fitues me "+PlayerTableRenderer.total.get(indexWinner)+" pike!", SwingConstants.CENTER));
57     playerRows.get(i).setForeground(Color.WHITE); playerRows.get(i).setFont(new Font("Vollkorn", Font.BOLD, 20));
58     players.add(playerRows.get(i)); //shfaq nje rresht tjeter per fituesin
59     i++;
60
61     players.setLayout(new GridLayout(playerRows.size(),1));
62     Database.writeDatabaseData(); //kalon lojteret e kesaj loje dhe piket qe kane marre ne MySQL
63
64     Object record[];
65     record = Database.calculateHighestLowest(); //percakton lojterin me piket me te larte/uleta nga databaza ne MySQL
66     if(record != null) {
67         playerRows.add(new JLabel("Rezultati me i larte: "+(String)record[0]+" "+(String)record[1]+" Loja nr. "+(Integer)record[2]+" "+(Integer)record[3]
68             +PlayerGame.get(i).setForeground(Color.WHITE); playerRows.get(i).setFont(new Font("Vollkorn", Font.PLAIN, 20));
```

GUIResults() krijon një CustomFrame të modifikuar (sfond Result.jpg, BorderLayout()). Në GUIResults krijohen rreshte (JLabel) për të shfaqur pikët që ka marr çdo lojëtar, një rresht që shfaq fituesin, një rresht që shfaq lojëtarin me pikët më të larta nga databaza dhe një rresht që shfaq lojëtarin me pikë më të ulëta nga databaza.

GUIResults implementon ActionListener dhe përmban një JButton exit që në shtypje System.exit(0) nga programi.

CategoriesPanel.java

```
14 public CategoriesPanel(){//JPanel i modifikuar qe shfaq nje JTable me kolone Kategorite dhe rreshta kategorite e ndryshme
15     String[] text = {"Njesha", "Dysha", "Tresha",
16         "Katra", "Pesa", "Gjashta", "Piket e Siperme",
17         "Bonus (35)", "Tre me nje vlere", "Kater me nje vlere",
18         "Tre dhe Dy (25)", "Kater te njepasnjeshme (30)", "Pese te njepasnjeshme (40)",
19         "E njejta vlere (50)", "Cdo rast", "Piket e Poshtme", "TOTAL"};
20
21     DefaultTableModel model = new DefaultTableModel();
22     JTable table = new JTable(model) {
23         private static final long serialVersionUID = 1L;
24         public boolean isCellEditable(int row, int column) {
25             return false;
26         } //ndalon ndryshimin e vleres se qelizes nga perdoruesi
27     };
28
29     model.addColumn("Kategorite");
30     for(int i=0;i<text.length;i++) {
31         model.addRow(new Object[]{text[i]});
32     }
33
34     table.getColumnModel().getColumn(0).setCellRenderer(new DefaultTableCellRenderer() {
35         private static final long serialVersionUID = 1L;
36
37         @Override
38         public Component getTableCellRendererComponent(JTable table, Object value, boolean isSelected, boolean hasFocus, int row, int column) {
39             table.getTableHeader().setBackground(Color.BLACK);//vendos ngjyren e background-it te headerit ("Kategorite") te zeze
40             table.getTableHeader().setOpaque(false);//e vendos header transperant
41             table.getTableHeader().setForeground(Color.WHITE);//vendos ngjyren e tekstit ne header te bardhe
42             setBackground(new Color(245,245,245));//vendos ngjyren e qelizave te si nje gri te lehte
43             setText((String)value);//vendos vleren cfaredo vlere ka pasur origjinalisht
44             return this;//kthen qelizen qe po render ne moment
45         } //ndryshon getTableCellRendererComponent te DefaultTableCellRenderer qe perdoret si CellRenderer per kolonen
46     });
47
48     table.getColumnModel().getColumn(0).setPreferredWidth(225);//percakton gjeresine e kolones
49     table.setRowHeight(25);//percakton gjatesine e rreshtit
50
51     table.getTableHeader().setReorderingAllowed(false);//ndalon levizjen e tabelës
52     table.setFocusable(false);//ndalon selektimin e tabelës nga perdoruesi
53     table.setRowSelectionAllowed(false);//ndalon selektimin e rreshtave nga perdoruesi
54
55     setLayout(new BorderLayout());
56     add(table, BorderLayout.CENTER);
57     add(table.getTableHeader(), BorderLayout.NORTH);
58 }
```

CategoriesPanel është një prej paneleve që vendoset mbi GUIGame. Paneli përmban një tabelë që ka si header “Kategoritë” dhe për rreshta kategoritë e ndryshme. JTabela është modifikuar që qelizat të mos jenë të ndryshueshme, gjerësi dhe gjatësi të përcaktuar dhe të mos lëvizet.

DicePanel.java

```
22 public class DicePanel extends JPanel implements MouseListener {
23     private static final long serialVersionUID = 1L;
24
25     private ArrayList<JLabel> imgLabels = new ArrayList<JLabel>();
26
27     public DicePanel() { //paneli i zareve
28         setLayout(new GridLayout(5, 1, 5, 5));
29         setOpaque(false);
30
31         try {
32             Image image;
33             for (int i = 0; i < DiceGenerator.diceNumbers.size(); i++) {
34                 image = ImageIO.read(new File("src/resources/" + DiceGenerator.diceNumbers.get(i) + ".png"));
35                 imgLabels.add(i, new JLabel(new ImageIcon(image)));
36                 imgLabels.get(i).setName(String.valueOf(DiceGenerator.diceNumbers.get(i)));
37                 imgLabels.get(i).addMouseListener(this);
38
39                 add(imgLabels.get(i));
40             } //shton ne panel imazhet e zareve
41         } catch (IOException e) {
42         }
43     }
44
45     @Override
46     public void mouseClicked(MouseEvent e) { //ne rast se nje nga imazhet e zareve shtypet, krijohet nje border ngjyre te bardhe rreth imazhit dhe vlera vendoset -
47         if (GUIGame.playerIndex != -1 //ndalon shtypjen e imazheve nese nuk ka filluar loja
48             && PlayerGame.CurrentPlayers.get(GUIGame.playerIndex).getRollsMade() < 3 //ndalon shtypjen e imazheve nese hedhjet e zareve jane 3
49             && PlayerGame.CurrentPlayers.get(GUIGame.playerIndex).getSelectedThisTurn() == false) { //ndalon shtypjen e imazheve nese eshte zgjedhur nje kategori
50             int i = imgLabels.indexOf(e.getSource());
51
52             if (DiceGenerator.diceNumbers.get(i) != -1) {
53                 imgLabels.get(i).setBorder(BorderFactory.createLineBorder(Color.WHITE, 2));
54                 DiceGenerator.diceNumbers.set(i, -1);
55
56             } else { //nese shtypet nje imazh qe eshte shtypur njehere, hiqen border-at dhe kthehet vlera origjinale
57                 imgLabels.get(i).setBorder(BorderFactory.createEmptyBorder());
58                 DiceGenerator.diceNumbers.set(i, Integer.parseInt(((JLabel) e.getSource()).getName()));
59             }
60         }
61     }
```

DicePanel është një prej paneleve që vendoset mbi GUIGame. Paneli përmban figurat e zareve të gjeneruar në një GridLayout(5 rreshta, 1 kolonë). DicePanel implementon MouseListener për të detektuar shtypjen e një prej zareve për tu hequr. Rreth zari-t të shtypur vendoset një Border ngjyrë të bardhë dhe vlera e tij në diceNumbers (që mban vlerat e zareve) bëhet -1 (nëse gjenerohen zare të rinj kjo do të testohet dhe do të ndryshohet). Kushtet ndalojnë zgjedhjen e zareve në rast kur nuk ka “filluar” loja, janë bërë 3 hedhje ose lojëtari ka zgjedhur një kategori këtë rradhë.

PlayerPanel.java

```
7 public class PlayerPanel extends JPanel {
8     private static final long serialVersionUID = 1L;
9
10    public PlayerPanel(String name, int i){//panel per lojeteret, tabel me header emrin e tyre dhe rreshta vlerat e kategorive
11
12        PlayerTableModel model = new PlayerTableModel();
13        model.addColumn(name);
14
15        JTable table = new JTable(model);
16        table.setName(""+i);//vendos emrin e tabelës, indeksin e lojtarit (perdoret nga PlayerTableRenderer)
17        PlayerTableRenderer tRenderer = new PlayerTableRenderer();
18        table.getTableHeader().setReorderingAllowed(false);
19        table.getColumnModel().getColumn(0).setCellRenderer(tRenderer);
20        table.getColumnModel().getColumn(0).setPreferredWidth(100);
21        table.setRowHeight(25);
22        setLayout(new BorderLayout());
23        add(table, BorderLayout.CENTER);
24        add(table.getTableHeader(), BorderLayout.NORTH);
25    }
```

PlayerPanel vendoset mbi **GUIGame**. Paneli përmban një tabelë që ka emër indeksin e lojtarit, si header emrin e lojtarit dhe rreshta pikët që mund të marrin ose kanë zgjedhur. Si model përdor **PlayerTableModel** që extends **DefaultTableModel**. Për render-in e qelizave në kolonë përdor **PlayerTableRenderer** që extends **DefaultTableCellRenderer**.

PlayerTableModel.java

```
3 import javax.swing.table.DefaultTableModel;
6
7 public class PlayerTableModel extends DefaultTableModel { //model tabele i modifikuar
8     private static final long serialVersionUID = 1L;
9
10    @Override
11    public boolean isCellEditable(int row, int column) {
12        return false;
13    } //qelizat nuk jane te modifikueshme
14
15    @Override
16    public int getRowCount() {
17        return 17;
18    } //tabelat duhet te kete 17 rreshta
19
20    @Override
21    public int getColumnCount() {
22        return 1;
23    } //tabelat duhet te kete 1 kolone
24
25    @Override
26    public Object getValueAt(int rowIndex, int columnIndex) {
27        switch(rowIndex) {
28            //1-6
29            case 0:
30            case 1:
31            case 2:
32            case 3:
33            case 4:
34            case 5: return Points.addValueDice(DiceGenerator.diceNumbers, rowIndex+1);
35
36            //case 6: above points
37            //PlayerTableRenderer
38            //case 7: bonus
39            //PlayerTableRenderer
40
41            //threeofakind
42            case 8: return Points.threeofakind(DiceGenerator.diceNumbers);
43            //fourofakind
44            case 9: return Points.fourofakind(DiceGenerator.diceNumbers);
45            //3and2
46            case 10: return Points.threeandtwo(DiceGenerator.diceNumbers);
47            //A sequence
```

PlayerTableModel nuk i ka qelizat e ndryshueshme në vlerë nga përdoruesi, ka 17 rreshta, 1 kolonë dhe vlera në indekset 0-5 (Kategoritë 1-6) dhe 8-14 (Kategoritë 7-13) janë të gjeneruara nga **Points.java**.

PlayerTableRenderer.java

```
13 public class PlayerTableRenderer extends DefaultTableCellRenderer { //renderer i modifikuar per kolonen e tabelave te lojtareve
14     private static final long serialVersionUID = 1L;
15
16     private static ArrayList<Map<Integer, Boolean>> selectedRows = new ArrayList<Map<Integer, Boolean>>(); //mban indeksin e rreshtave te zgjedhur
17     private static ArrayList<Map<Integer, Object>> selectedValues = new ArrayList<Map<Integer, Object>>(); //mban vlerat e rreshtave te zgjedhura
18     private static boolean initializedArrays = false; //percakton nese array-it jane inicializuar
19
20     private static ArrayList<Integer> abovePoints = new ArrayList<Integer>(); //mban vleren totale te rreshtave te siperm te zgjedhur (indeksi i lojtarit, bonus)
21     private static ArrayList<Integer> bonus = new ArrayList<Integer>(); //mban vleren bonus (indeksi i lojtarit, bonus)
22     private static ArrayList<Integer> belowPoints = new ArrayList<Integer>(); //mban vleren totale te rreshtave te poshtem te zgjedhur (indeksi i lojtarit, pike)
23     private static ArrayList<Integer> total = new ArrayList<Integer>(); //mban piket totale te rreshtave te zgjedhur (indeksi i lojtarit, pike)
24
25     @Override
26     public Component getTableCellRendererComponent(JTable table, Object value, boolean isSelected, boolean hasFocus, int row, int column) { //render
27
28         if(initializedArrays==false) {
29             for(int i=0; i<data.PlayerGame.CurrentPlayers.size(); i++) {
30                 selectedRows.add(new HashMap<Integer, Boolean>());
31                 selectedValues.add(new HashMap<Integer, Object>());
32                 abovePoints.add(0);
33                 bonus.add(0);
34                 belowPoints.add(0);
35                 total.add(0);
36             }
37             initializedArrays=true; //inicializon arraylist-at
38
39             int workingTable = Integer.parseInt(table.getName()); //emri i tabelës korrespondon me indeksin e lojtarit
40
41             if(GUIGame.playerIndex==workingTable) {
42                 table.getTableHeader().setOpaque(false);
43                 table.getTableHeader().setBackground(new Color(90, 80, 195));
44                 table.getTableHeader().setForeground(Color.WHITE);
45             } //vendos ngjyren e header-it te tabelës blu te erret nese eshte tabela e njejte me indeksin e lojtarit qe ka rradhen per momentin
46
47             if (selectedRows.get(workingTable).containsKey(row)) {
48                 value = selectedValues.get(workingTable).get(row);
49                 setText(value.toString()); //nese selectedRows permban kete rresht per kete tabelë, shfaq vleren e zgjedhur me perpara
50             } else {
51                 setText(value.toString()); //nese jo, shfaq nje vlere te re
52             }
53         }
54     }
55 }
```

ArrayList<Map<Integer, Boolean>> selectedRows është një ArrayList ku indeksi korrespondon me indeksin e lojtarit, Integer është indeksi i rreshtit dhe Boolean është vlerë true ose false në varësi nëse ky rresht është zgjedhur apo jo.

ArrayList<Map<Integer, Object>> selectedValues është një ArrayList ku indeksi korrespondon me indeksin e lojtarit, Integer është indeksi i rreshtit dhe Object është vlerë e zgjedhur më përpara në këtë rresht.

boolean initializedArrays përcakton nëse array-t janë inicializuar ose jo.

ArrayList<Integer> abovePoints është një ArrayList ku indeksi korrespondon me indeksin e lojtarit dhe Integer është vlera e pikëve të sipërme të zgjedhura.

ArrayList<Integer> bonusPoints është një ArrayList ku indeksi korrespondon me indeksin e lojtarit dhe Integer është vlera e pikëve bonus në varësi të pikëve të sipërme.

ArrayList<Integer> belowPoints është një ArrayList ku indeksi korrespondon me indeksin e lojtarit dhe Integer është vlera e pikëve të poshtme të zgjedhura.

ArrayList<Integer> total është një ArrayList ku indeksi korrespondon me indeksin e lojtarit dhe Integer është vlera totale e pikëve të zgjedhura.

getTableCellRendererComponent () thirret çdo herë kur një qelizë gjenerohet në tabelë. Në gjenerimin e parë inicializohen vlera default për të gjithë array-it në mënyrë që të ndryshohen më vonë.

int workingTable mban vlerën e tabelës që po ndërtohet në moment dhe korrespondon me indeksin e lojtarit.

Nëse tabela që therr **getTableCellRendererComponent** është e njëjtë me indeksin e lojtarit, ngjyra e header-it të tabelës vendoset blue e errët.

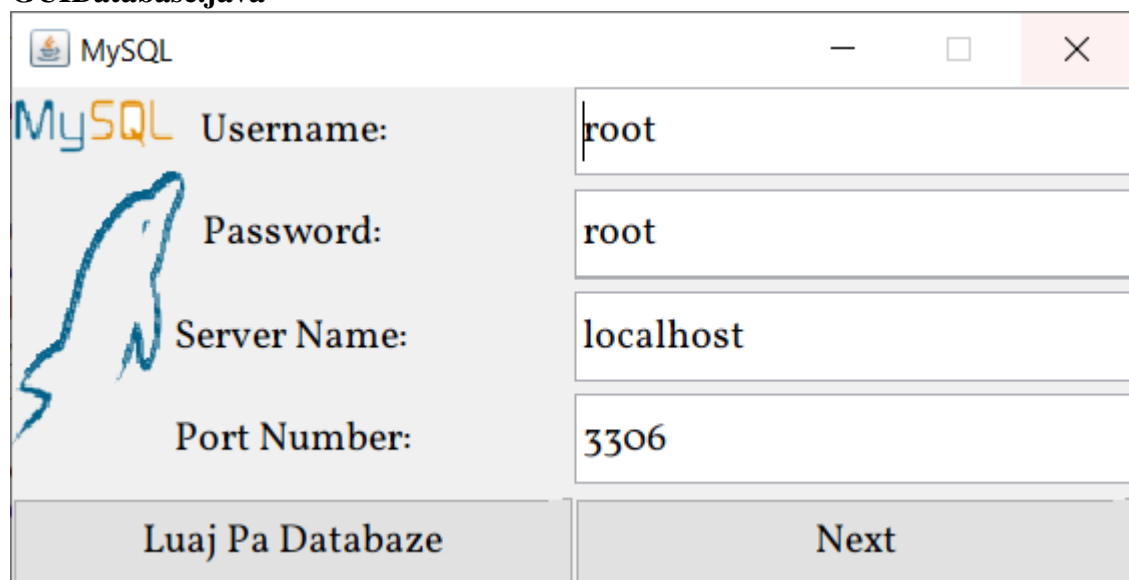
Nëse vlera e rreshtit për workingTable përmbahet në selectedValues, kjo vlerë vendoset në qelizën përkatëse, ndryshe shfaqet vlera e gjeneruar nga Points.java.

Në rreshtin me indeks 6, 7, 15, 16 përkatësisht shfaqen vlerat përkatëse për lojtarin për pikët e sipërme, pikët bonus, pikët e poshtme dhe pikët totale.

Rreshtet me kategori të zgjedhshme shfaqen në një ngjyrë gri të lehtë, ato që kanë pikë më të larta se 0 janë ngjyrë jeshile, të zgjedhurat janë ngjyrë portokalli.

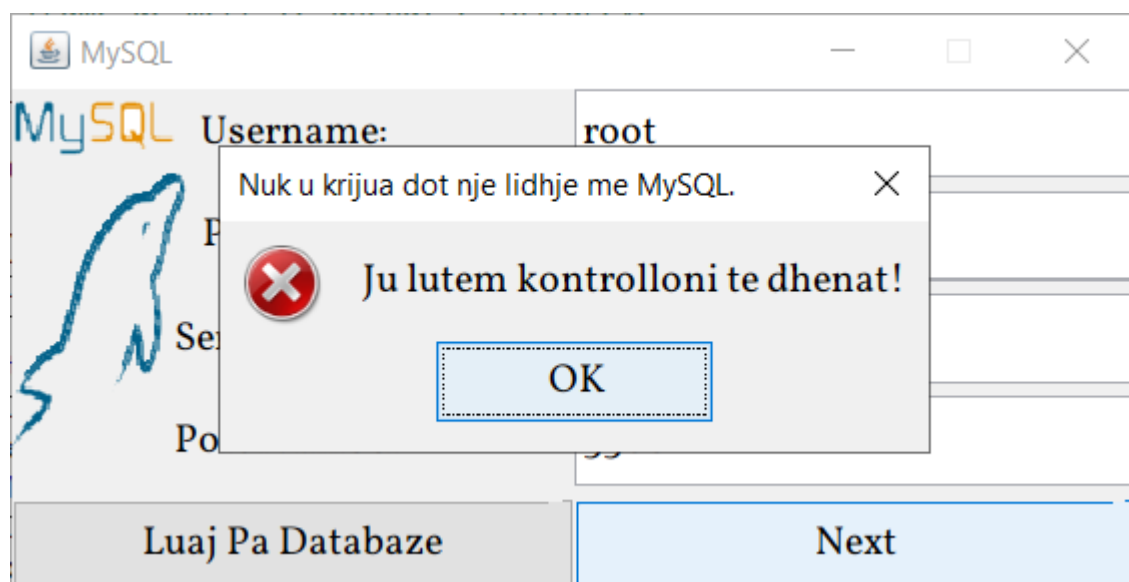
Nëse një prej kategorieve i vendoset fokusi (hasFocus), zgjidhet qeliza, ndryshohet ngjyra, ruhet rreshti dhe vlera.

GUIDatabase.java



A screenshot of a Java Swing window titled "MySQL". The window has a standard title bar with minimize, maximize, and close buttons. On the left side, there is a MySQL logo and a blue silhouette of a fish. The main area contains four labels with corresponding text input fields: "Username:" with "root", "Password:" with "root", "Server Name:" with "localhost", and "Port Number:" with "3306". At the bottom, there are two buttons: "Luaj Pa Databaze" and "Next".

MySQL	Username:	root
	Password:	root
	Server Name:	localhost
	Port Number:	3306
Luaj Pa Databaze		Next



A screenshot of the same MySQL connection form, but with an error dialog box displayed in the foreground. The dialog box has a title bar that says "Nuk u krijua dot nje lidhje me MySQL." and a close button. It contains a red "X" icon and the text "Ju lutem kontrolloni te dhenat!". There is an "OK" button at the bottom of the dialog. The background form is partially obscured by the dialog box.

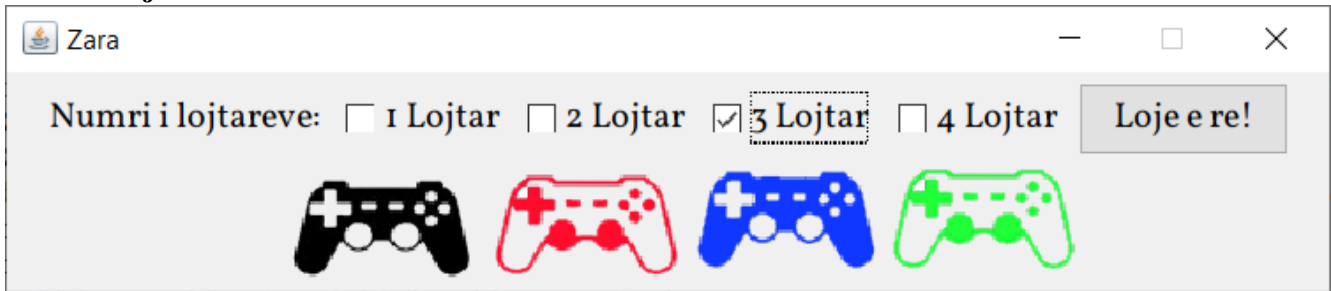
MySQL	Username:	root
	Password:	root
	Server Name:	localhost
	Port Number:	3306
Luaj Pa Databaze		Next

Nuk u krijua dot nje lidhje me MySQL.

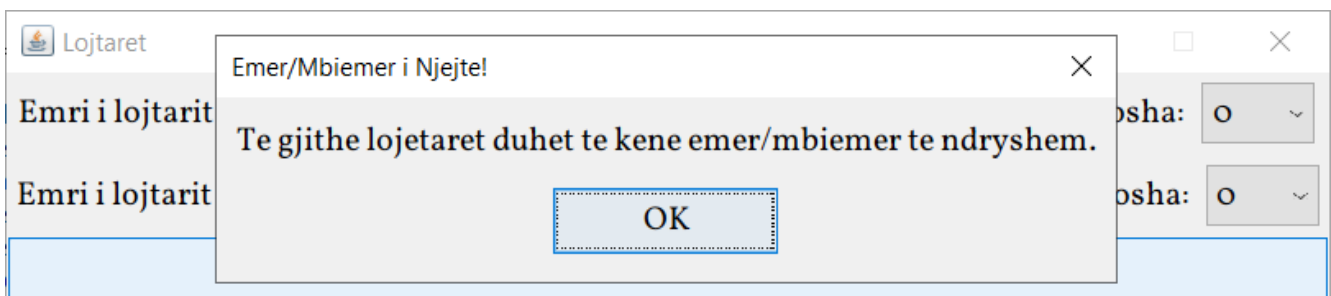
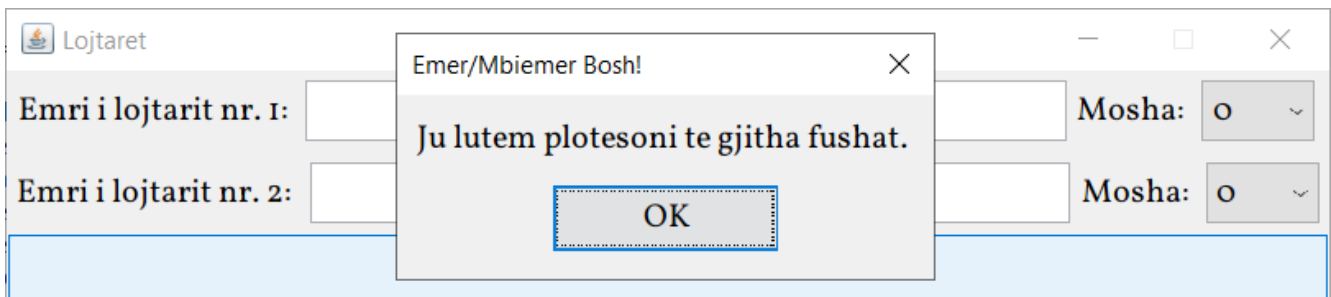
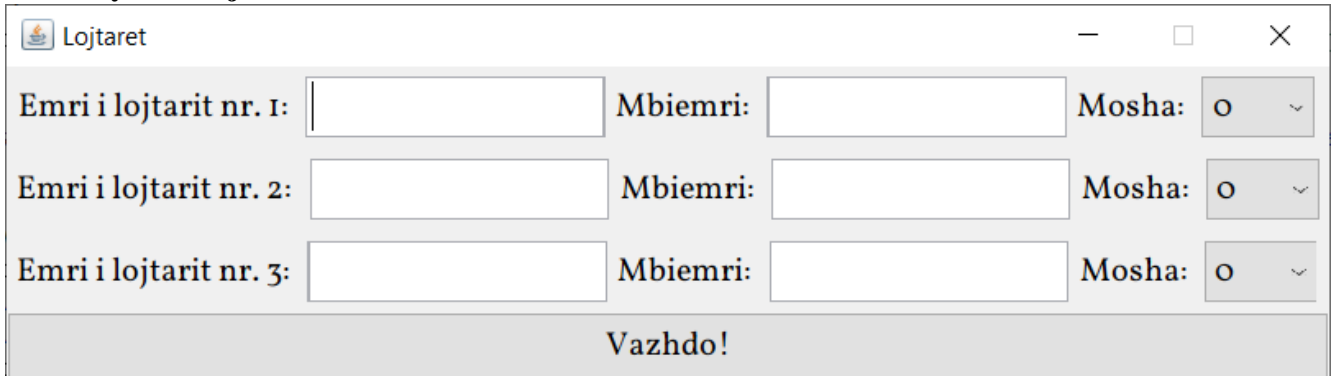
X Ju lutem kontrolloni te dhenat!

OK

GUIStart.java



GUIPlayerData.java



Fillo Lojen!

Kategorite	Ana	Miri	Genti
Njesha			
Dysha			
Tresha			
Katra			
Pesa			
Gjashta			
Piket e Siperme			
Bonus (35)			
Tre me nje vlere			
Kater me nje vlere			
Tre dhe Dy (25)			
Kater te njepasnjeshme (30)			
Pese te njepasnjeshme (40)			
E njehta vlere (50)			
Çdo rast			
Piket e Poshtme			
TOTAL			

Fillo Lojen!

Hidh Zaret!

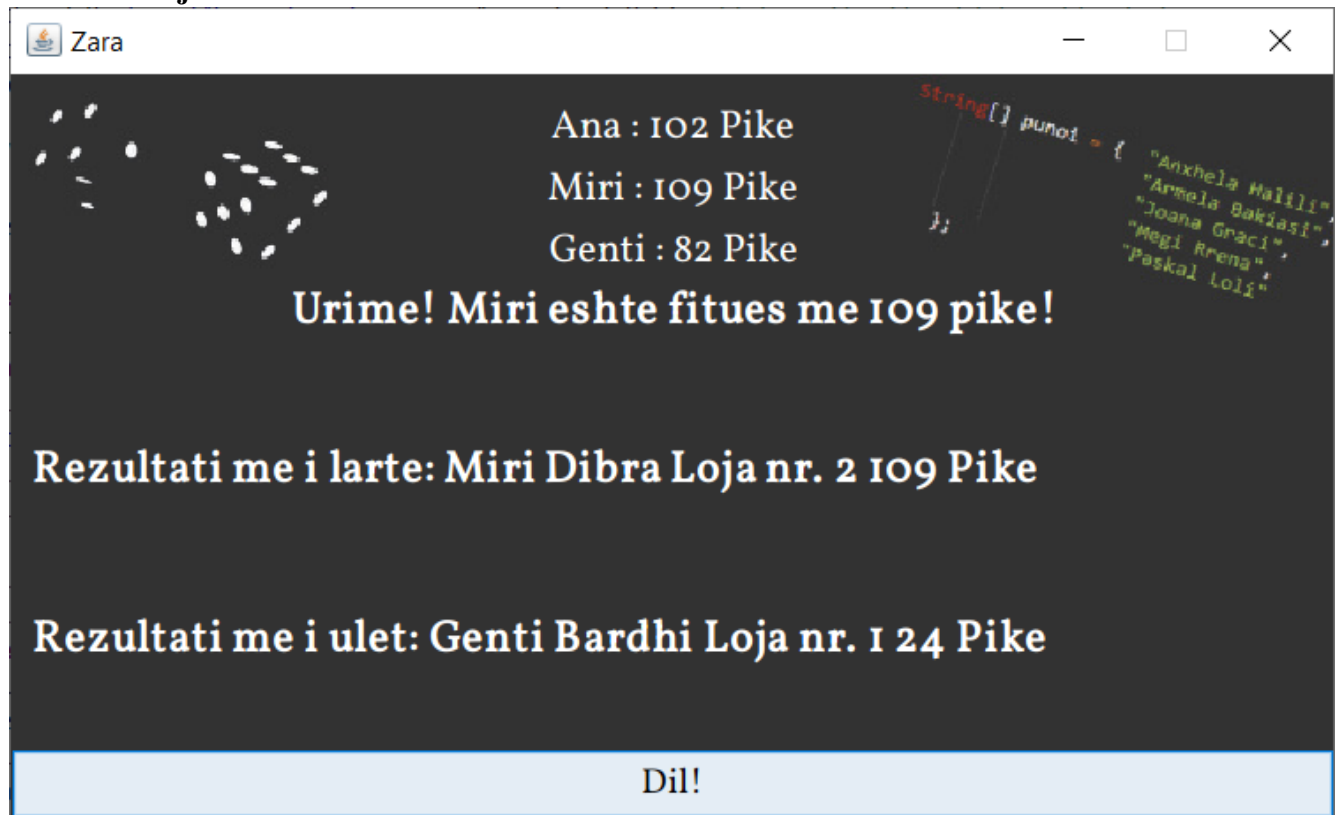
Rradha - Ana

Zgjidhni zarat qe doni te hidhni perseri dhe shtypni "Hidh Zarat" ose zgjidhni nje kategori. Mundesi: 0/3

Perfundo!

Shtyp "Perfundo".

GUIResults.java



MySQL

The screenshot shows the MySQL Workbench interface. On the left, the 'playerdatabase' is expanded, showing 'Tables' with 'playertable', 'scoretable', and 'world'. The 'scoretable' is selected, showing its 'Columns', 'Indexes', 'Foreign Keys', and 'Triggers'. On the right, the 'Result Grid' displays the data for 'scoretable'.

	firstName	lastName	age
▶	a	b	0
	Ana	Hoxha	21
	Ana	Luka	20
	Ana	Ras	17
	c	d	0
	Genti	Bardhi	22
	Miri	Dibra	22

The screenshot shows the MySQL Workbench interface. On the left, the 'playerdatabase' is expanded, showing 'Tables' with 'playertable', 'scoretable', and 'world'. The 'scoretable' is selected, showing its 'Columns', 'Indexes', 'Foreign Keys', and 'Triggers'. On the right, the 'Result Grid' displays the data for 'scoretable'.

	firstName	lastName	playCount	score
▶	Ana	Luka	1	38
	Miri	Dibra	1	43
	Genti	Bardhi	1	24
	Ana	Ras	1	62
	Ana	Ras	2	71
	a	b	1	42
	Ana	Hoxha	1	102
	Miri	Dibra	2	109
	Genti	Bardhi	2	82