```
 1 /*
 2  * Click
nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
change this license
 3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to
edit this template
 4  */
 5 package server;
 6
 7 import java.rmi.RemoteException;
 8 import java.rmi.server.UnicastRemoteObject;
 9 import service.ISlotMachineService;
10
11 import java.rmi.registry.Registry;
12 import java.rmi.registry.LocateRegistry;
13 import java.time.LocalDateTime;
14 import java.util.ArrayList;
15 import java.util.List;
16 import java.util.Random;
17 import model.Play;
18 import model.User;
19 import util.EncryptUtil;
20
21 /**
22  *
23  * @author rault
24  */
25 public class Server extends UnicastRemoteObject implements
ISlotMachineService {
26
27     private final int DIFFICULTY = 10;
28     private List<User> user = new ArrayList<>();
29     private int userSelect;
30
31     public Server() throws RemoteException {
32         super();
33
34         user.add(new User("Raul Tavares Danielli",
"raultavares27@outlook.com", 12.2f, "raulzito123"));
35         user.add(new User("Isabelle Massonetto", "isa26@outlook.com",
12.2f, "test132"));
36         user.add(new User("Fabiano Godine", "221fabin@outlook.com",
12.2f, "fabin213"));
37     }
38
39     @Override
```

```
40      public LocalDateTime getLocalDateTime() throws RemoteException {
41          return LocalDateTime.now();
42      }
43
44      @Override
45      public int[] getRandomNumbers() throws RemoteException {
46          int[] randomNumbers = new int[3];
47          Random rand = new Random();;
48
49          if (rand.nextInt(DIFFICULTY) == rand.nextInt(5)) {
50              int win = rand.nextInt(5);
51              for (int i = 0; i < 3; i++) randomNumbers[i] = win;
52          }
53          else {
54              for (int i = 0; i < 3; i++) randomNumbers[i] =
rand.nextInt(5);
55          }
56
57          return randomNumbers;
58      }
59
60      //
61      // EXECUÇÃO DO SERVER
62      //
63      public static void main(String[] args) {
64
65          try {
66              ISlotMachineService srv = new Server();
67
68              Registry rg = LocateRegistry.createRegistry(PORT);
69              rg.bind(NAME, srv);
70
71              System.out.println("Servidor " + NAME + " iniciado.");
72
73          } catch (Exception e) {
74              System.err.println("ERRO: " + e.getMessage());
75          }
76      }
77
78      @Override
79      public List<String> getHistoryPlay() throws RemoteException {
80          List<String> historyPlay = new ArrayList<>();
81
82          this.user.get(userSelect).getPlays().forEach(p -> {
83              historyPlay.add(p.toString());
84          });
```

```
 85
 86         return historyPlay;
 87     }
 88
 89     @Override
 90     public String getMoneyStorage() throws RemoteException {
 91         return Float.toString(this.user.get(userSelect).getMoney());
 92     }
 93
 94 //    @Override
 95     public void addUser(User u) throws RemoteException {
 96         this.user.add(u);
 97     }
 98
 99     @Override
100     public void addPlayToHistory(int[] plays, float moneyBet) throws
RemoteException {
101         Play play = new Play(plays, moneyBet);
102         this.user.get(userSelect).addPlay(play);
103         if (this.user.get(userSelect).listPlaySize() > 10)
this.user.get(userSelect).popPlay();
104     }
105
106     @Override
107     public void updateDecreaseMoneyStorage(float value) throws
RemoteException {
108         float updateMoney = this.user.get(userSelect).getMoney() -
value;
109         this.user.get(userSelect).setMoney(updateMoney);
110     }
111
112     @Override
113     public void updateIncreaseMoneyStorage(float value) throws
RemoteException {
114         float updateMoney = this.user.get(userSelect).getMoney() +
value;
115         this.user.get(userSelect).setMoney(updateMoney);
116     }
117
118     @Override
119     public String getUserName() throws RemoteException {
120         return this.user.get(userSelect).getName();
121     }
122
123     @Override
124     public void setUserSelect(int userSelect) throws RemoteException {
```

```
125         this.userSelect = userSelect;
126     }
127 
128     @Override
129     public void addUser(String name, String email, float money, String
password) throws RemoteException {
130         user.add(new User(name, email, money, password));
131     }
132 
133     @Override
134     public boolean authenticator(String email, String password) throws
RemoteException {
135         String passwordEncrypt = EncryptUtil.toMD5(password);
136         for (User u : this.user) {
137             if(u.getEmail().equals(email) &&
u.getPassword().equals(passwordEncrypt))return true;
138             System.out.println("");
139         }
140 
141         return false;
142     }
143 
144     @Override
145     public int getUserIndexByEmail(String email) throws RemoteException
{
146 
147         for (int i=0; i<user.size(); i++) {
148             if(user.get(i).getEmail().equals(email))return i;
149         }
150 
151         return -1;
152     }
153 }
```