```java
import javax.swing.*;

/* This class is the main System level class which creates all the objects
 * representing the game logic (model) and the panel for user interaction.
 * It also implements the main game loop
 */

public class Game extends JFrame {

    private int timeAllowed = 200;
    private int score = 0;
    private int difficulty = 1;
    private int gameDelay = 500;
    private int energy = 200;
    private final int produceTime = 20;
    private boolean checkGame = true;
    private boolean isPause = false;
    private JButton start = new JButton("Start");
////////////////// all button starts here
    private JButton restart = new JButton("Restart");
    private JButton pause = new JButton("Pause");
    private JButton login = new JButton("LogIn");
    private JButton register = new JButton("Register");
    private JButton rank = new JButton("Rank");
    private JButton setting = new JButton("Setting");
    private JLabel timeLabel = new JLabel("Time Remaining : " + timeAllowed);
    private JLabel scoreLabel = new JLabel("Score : " + score);
    private JLabel energyLabel;
    private UserData userData = new UserData();
    private static Game game;
    private Grid grid;
    private Player player;
    private ArrayList<Monster> monsters = new ArrayList<>();
    private BoardPanel boardPanel;

    public static void main(String args[]) throws Exception {
        game = new Game();
        game.gameStart();
    }
```

```
43
44      /*
45       * This constructor creates the main model objects and the panel used for UI. It
46       * throws an exception if an attempt is made to place the player or the monster
47       * in an invalid location.
48       */
49      public Game() throws Exception {
50          grid = new Grid(difficulty);
51          player = new Player(grid, 0, 0);
52          monsters.add(new Monster(grid, player, 5, 5));
53          boardPanel = new BoardPanel(grid, player, monsters);
54          energyLabel = new JLabel("Energy : " + player.getEnergy());
55
56          setTitle("RunLikeHell");
57          setSize((int) (640 * (1 + difficulty * 0.25)), (int) (480 * (1 + difficulty * 0.25)));
58          setLocationRelativeTo(null); // center the frame
59          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
60          setVisible(true);
61          // Create a separate panel and add all the buttons
62          JPanel controlPane = new JPanel();
63          controlPane.setBorder(new EmptyBorder(20, 20, 20, 20));
64          controlPane.setLayout(new BorderLayout(5, 5));
65          controlPane.setLayout(new GridLayout(10, 10, 10, 10));
66          controlPane.add(start);
67          controlPane.add(restart);
68          controlPane.add(pause);
69          controlPane.add(login);
70          controlPane.add(register);
71          controlPane.add(rank);
72          controlPane.add(setting);
73          controlPane.add(energyLabel);
74          controlPane.add(scoreLabel);
75          controlPane.add(timeLabel);
76
77          // add Action listeners to all button events
78          start.addActionListener(new MyActionListener());
79          restart.addActionListener(new MyActionListener());
80          pause.addActionListener(new MyActionListener());
```

```
81          login.addActionListener(new MyActionListener());
82          register.addActionListener(new MyActionListener());
83          rank.addActionListener(new MyActionListener());
84          setting.addActionListener(new MyActionListener());
85          start.addKeyListener(boardPanel);
86          // add panels to frame
87          this.add(boardPanel, BorderLayout.CENTER);
88          this.add(controlPane, BorderLayout.EAST);
89      }
90
91      // method to delay by specified time in ms
92      public void delay(int time) {
93          try {
94              Thread.sleep(time);
95          } catch (InterruptedException e) {
96              e.printStackTrace();
97          }
98      }
99
100     /*
101      * This method waits until play is ready (until start button
    is pressed) after
102      * which it updates the moves in turn until time runs out
    (player won) or player
103      * is eaten up (player lost).
104      */
105     public void gameStart() throws Exception {
106         do {
107             play();
108             player.setReady(false);
109         } while (checkGame);
110     }
111
112     private void reset() throws Exception {
113         grid = new Grid(difficulty);
114         player = new Player(grid, 0, 0);
115         monsters.clear();
116         monsters.add(new Monster(grid, player, 5, 5));
117         boardPanel.reset(grid, player, monsters);
118         player.setReady(false);
119         player.setEnergy(energy);
120         score = 0;
```

```
121         boardPanel.repaint();
122     }
123 
124     // Game Run
125     public String play() throws Exception {
126         int time = 0;
127         boolean check = true;
128         boolean checkEaten = false;
129         String message;
130         player.setDirection(' '); // set to no direction
131         while (!player.isReady())
132             delay(100);
133         do {
134             while (isPause)
135                 delay(100);
136             Cell newPlayerCell =
    player.move(player.getPresses());
137             ArrayList<Cell> MonstersCell = new ArrayList<>();
138 
139             for (int i = 0; i < monsters.size(); ++i)
140                 MonstersCell.add(monsters.get(i).move(1));
141             for (int i = 0; i < monsters.size(); ++i) {
142                 if (newPlayerCell == monsters.get(i).getCell()
    && MonstersCell.get(i) == player.getCell()) {
143                     checkEaten = true;
144                 }
145             }
146 
147             if (!checkEaten) {
148                 player.setDirection(' '); // reset to no
    direction
149                 // update time and repaint
150                 time++;
151                 score += (3 - difficulty) * (2 - (double)
    gameDelay / 1000);
152 
153                 if (time % produceTime == 0) {
154                     Monster baby = new Monster(grid, player,
    MonstersCell.get(0).row, MonstersCell.get(0).col);
155                     baby.isBaby = true;
156                     monsters.add(baby);
157                 }
```

```
158                 for (int i = 1; i < monsters.size(); ++i) {
159                     if (monsters.get(i).isBaby() &&
    monsters.get(i).getCell() == newPlayerCell) {
160                         monsters.remove(i);
161                     }
162                 }
163 
164                 energyLabel.setText("Energy : " +
    player.getEnergy());
165                 scoreLabel.setText("Score : " + score);
166                 timeLabel.setText("Time Remaining : " +
    (timeAllowed - time));
167                 delay(gameDelay);
168                 boardPanel.repaint();
169             } else
170                 check = false;
171 
172         } while (time < timeAllowed && check &&
    player.isReady());
173         message = time < timeAllowed ? "Player Lost" : "Player
    Won"; // players has been eaten up
174         userData.saveScore(score);
175         timeLabel.setText(message);
176         return message;
177     }
178 
179     class MyActionListener implements ActionListener {
180         public void actionPerformed(ActionEvent e) {
181             String label = e.getActionCommand();
182             if (label.equals("LogIn")) {
183                 new Login();
184             }
185             if (label.equals("Register")) {
186                 new Register();
187             }
188             if (label.equals("Rank")) {
189                 new Rank();
190             }
191             if (label.equals("Setting")) {
192                 new Setting();
193             }
194             if (label.equals("Start")) {
```

```
195                 isPause = false;
196                 if (userData.isLogin())
197                     player.setReady(true);
198                 else
199                     JOptionPane.showMessageDialog(null, "you
    need to login first");
200             }
201             if (label.equals("Restart")) {
202                 try {
203                     reset();
204                 } catch (Exception e1) {
205                     e1.printStackTrace();
206                 }
207             }
208             if (label.equals("Pause")) {
209                 isPause = true;
210             }
211         }
212     }
213
214     class Rank extends JFrame {
215         public Rank() {
216             setTitle("Rank");
217             Container container = getContentPane();
218             container.setLayout(null);
219             JLabel label1 = new JLabel("Name");
220             container.add(label1);
221             JLabel label2 = new JLabel("Score");
222             container.add(label2);
223
224             User[] userList = userData.getList();
225             JLabel[] nameList = new JLabel[userList.length];
226             JLabel[] scoreList = new JLabel[userList.length];
227             for (int i = 0; i < userList.length; ++i) {
228                 nameList[i] = new
    JLabel(userList[i].getUserName());
229                 scoreList[i] = new JLabel("" +
    userList[i].getScore());
230                 container.add(nameList[i]);
231                 container.add(scoreList[i]);
232             }
233             container.setLayout(new GridLayout(userList.length +
```

```
   2, 3, 40, 40));
234             JButton button = new JButton("Confirm");
235             button.setSize(20, 20);
236             button.addActionListener(new ActionListener() {
237                 public void actionPerformed(ActionEvent e) {
238                     dispose();
239                 }
240             });
241
242             container.add(button);
243             setBounds(0, 0, 300, 300);
244             setAlwaysOnTop(true);
245             setResizable(false);
246             setLocationRelativeTo(null);
247             setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
248             setVisible(true);
249         }
250     }
251
252     class Login extends JFrame {
253         public Login() {
254             setTitle("Login");
255             JLabel label1 = new JLabel("UserName");
256             label1.setBounds(10, 10, 200, 18);
257             JLabel label2 = new JLabel("Password");
258             label2.setBounds(10, 50, 200, 18);
259             final JTextField textField1 = new JTextField();
260             textField1.setBounds(90, 10, 150, 18);
261             JPasswordField passwordField = new JPasswordField();
262             passwordField.setBounds(90, 50, 150, 18);
263             final JButton button1 = new JButton("Confirm");
264             button1.addActionListener(new ActionListener() {
265                 public void actionPerformed(ActionEvent e) {
266                     boolean flag =
   userData.login(textField1.getText(),
   String.valueOf(passwordField.getPassword()));
267                     if (flag) {
268                         JOptionPane.showMessageDialog(button1,
   "Login Successful");
269                         dispose();
270                     } else {
271                         JOptionPane.showMessageDialog(button1,
```

```
   "Login Fail");
272                     }
273                 }
274             });
275             button1.setBounds(40, 80, 100, 18);
276             JButton button2 = new JButton("Cancel");
277             button2.addActionListener(new ActionListener() {
278                 public void actionPerformed(ActionEvent e) {
279                     dispose();
280                 }
281             });
282             button2.setBounds(150, 80, 100, 18);
283             Container container = getContentPane();
284             container.setLayout(null);
285             container.add(label1);
286             container.add(label2);
287             container.add(textField1);
288             container.add(passwordField);
289             container.add(button1);
290             container.add(button2);
291             setBounds(0, 0, 300, 150);
292             setAlwaysOnTop(true);
293             setResizable(false);
294             setLocationRelativeTo(null);
295             setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
296             setVisible(true);
297         }
298     }
299
300     class Register extends JFrame {
301         public Register() {
302             setTitle("Register");
303             JLabel label1 = new JLabel("UserName");
304             label1.setBounds(10, 10, 200, 18);
305             JLabel label2 = new JLabel("Password");
306             label2.setBounds(10, 50, 200, 18);
307             final JTextField textField1 = new JTextField();
308             textField1.setBounds(90, 10, 150, 18);
309             JPasswordField passwordField = new JPasswordField();
310             passwordField.setBounds(90, 50, 150, 18);
311             final JButton button1 = new JButton("Confirm");
312             button1.addActionListener(new ActionListener() {
```

```
313                 public void actionPerformed(ActionEvent e) {
314                     User user = new User(textField1.getText(),
    String.valueOf(passwordField.getPassword()));
315                     if (userData.register(user))
316                         JOptionPane.showMessageDialog(button1,
    "Register Successful");
317                     else
318                         JOptionPane.showMessageDialog(button1,
    "Register Fail");
319                     dispose();
320                 }
321             });
322             button1.setBounds(40, 80, 100, 18);
323             JButton button2 = new JButton("Cancel");
324             button2.addActionListener(new ActionListener() {
325                 public void actionPerformed(ActionEvent e) {
326                     dispose();
327                 }
328             });
329             button2.setBounds(150, 80, 100, 18);
330             Container container = getContentPane();
331             container.setLayout(null);
332             container.add(label1);
333             container.add(label2);
334             container.add(textField1);
335             container.add(passwordField);
336             container.add(button1);
337             container.add(button2);
338             setBounds(0, 0, 300, 150);
339             setAlwaysOnTop(true);
340             setResizable(false);
341             setLocationRelativeTo(null);
342             setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
343             setVisible(true);
344         }
345     }
346
347     class Setting extends JFrame {
348         public Setting() {
349             setTitle("Setting");
350
351             JLabel gameDifficulty = new JLabel("Game
```

```
   Difficulty");
352             gameDifficulty.setBounds(10, 10, 200, 18);
353             JRadioButton easy = new JRadioButton("Easy");
354             JRadioButton normal = new JRadioButton("Normal",
   true);
355             JRadioButton hard = new JRadioButton("Hard");
356             ButtonGroup group1 = new ButtonGroup();
357             group1.add(easy);
358             group1.add(normal);
359             group1.add(hard);
360
361             JLabel gameDuration = new JLabel("Game Duration");
362             gameDuration.setBounds(10, 50, 200, 18);
363             JRadioButton gd1 = new JRadioButton("100");
364             JRadioButton gd2 = new JRadioButton("200", true);
365             JRadioButton gd3 = new JRadioButton("300");
366             ButtonGroup group2 = new ButtonGroup();
367             group2.add(gd1);
368             group2.add(gd2);
369             group2.add(gd3);
370
371             JLabel gameFrequency = new JLabel("Game Frequency");
372             gameFrequency.setBounds(10, 100, 200, 18);
373             JRadioButton gf1 = new JRadioButton("0.2s/m");
374             JRadioButton gf2 = new JRadioButton("0.5s/m", true);
375             JRadioButton gf3 = new JRadioButton("1s/m");
376             ButtonGroup group3 = new ButtonGroup();
377             group3.add(gf1);
378             group3.add(gf2);
379             group3.add(gf3);
380
381             JLabel playerEnergy = new JLabel("Player Energy");
382             playerEnergy.setBounds(10, 100, 200, 18);
383             JRadioButton pe1 = new JRadioButton("40E");
384             JRadioButton pe2 = new JRadioButton("200E", true);
385             JRadioButton pe3 = new JRadioButton("1000E");
386             ButtonGroup group4 = new ButtonGroup();
387             group4.add(pe1);
388             group4.add(pe2);
389             group4.add(pe3);
390
391             final JButton button1 = new JButton("Confirm");
```

```
392             button1.addActionListener(new ActionListener() {
393                 public void actionPerformed(ActionEvent e) {
394                     if (easy.isSelected())
395                         difficulty = 2;
396                     else if (normal.isSelected())
397                         difficulty = 1;
398                     else if (hard.isSelected())
399                         difficulty = 0;
400                     if (gd1.isSelected())
401                         timeAllowed = 100;
402                     else if (gd2.isSelected())
403                         timeAllowed = 200;
404                     else if (gd3.isSelected())
405                         timeAllowed = 300;
406                     if (gf1.isSelected())
407                         gameDelay = 200;
408                     else if (gf2.isSelected())
409                         gameDelay = 500;
410                     else if (gf3.isSelected())
411                         gameDelay = 1000;
412                     if (pe1.isSelected())
413                         energy = 40;
414                     else if (pe2.isSelected())
415                         energy = 200;
416                     else if (pe3.isSelected())
417                         energy = 1000;
418                     game.setSize((int) (640 * (1 + difficulty *
    0.25)), (int) (480 * (1 + difficulty * 0.25)));
419                     try {
420                         reset();
421                     } catch (Exception e1) {
422                         e1.printStackTrace();
423                     }
424                     dispose();
425                 }
426             });
427             button1.setBounds(30, 100, 100, 18);
428 
429             JButton button2 = new JButton("Cancel");
430             button2.addActionListener(new ActionListener() {
431                 public void actionPerformed(ActionEvent e) {
432                     dispose();
```

```
433                 }
434             });
435             button2.setBounds(140, 100, 100, 18);
436             JPanel container = new JPanel();
437 
438             container.add(gameDifficulty);
439             container.add(easy);
440             container.add(normal);
441             container.add(hard);
442 
443             container.add(gameDuration);
444             container.add(gd1);
445             container.add(gd2);
446             container.add(gd3);
447 
448             container.add(gameFrequency);
449             container.add(gf1);
450             container.add(gf2);
451             container.add(gf3);
452 
453             container.add(playerEnergy);
454             container.add(pe1);
455             container.add(pe2);
456             container.add(pe3);
457 
458             container.add(button1);
459             container.add(button2);
460             this.add(container);
461             setBounds(0, 0, 350, 200);
462             setAlwaysOnTop(true);
463             setResizable(false);
464             setLocationRelativeTo(null);
465             setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
466             setVisible(true);
467         }
468     }
469 }
470 
```