```
import javax.swing.*;
import javax.swing.border.*;
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;

/* 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();
	}

	/*
	 * This constructor creates the main model objects and the panel used for UI. It
	 * throws an exception if an attempt is made to place the player or the monster
	 * in an invalid location.
	 */
	public Game() throws Exception {
		grid = new Grid(difficulty);
		player = new Player(grid, 0, 0);
		monsters.add(new Monster(grid, player, 5, 5));
		boardPanel = new BoardPanel(grid, player, monsters);
		energyLabel = new JLabel("Energy : " + player.getEnergy());

		setTitle("RunLikeHell");
		setSize((int) (640 * (1 + difficulty * 0.25)), (int) (480 * (1 + difficulty * 0.25)));
		setLocationRelativeTo(null); // center the frame
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setVisible(true);
		// Create a separate panel and add all the buttons
		JPanel controlPane = new JPanel();
		controlPane.setBorder(new EmptyBorder(20, 20, 20, 20));
		controlPane.setLayout(new BorderLayout(5, 5));
		controlPane.setLayout(new GridLayout(10, 10, 10, 10));
		controlPane.add(start);
		controlPane.add(restart);
		controlPane.add(pause);
		controlPane.add(login);
		controlPane.add(register);
		controlPane.add(rank);
		controlPane.add(setting);
		controlPane.add(energyLabel);
		controlPane.add(scoreLabel);
		controlPane.add(timeLabel);

		// add Action listeners to all button events
		start.addActionListener(new MyActionListener());
		restart.addActionListener(new MyActionListener());
		pause.addActionListener(new MyActionListener());
		login.addActionListener(new MyActionListener());
		register.addActionListener(new MyActionListener());
		rank.addActionListener(new MyActionListener());
		setting.addActionListener(new MyActionListener());
		start.addKeyListener(boardPanel);
		// add panels to frame
		this.add(boardPanel, BorderLayout.CENTER);
		this.add(controlPane, BorderLayout.EAST);
```

```
	}

	// method to delay by specified time in ms
	public void delay(int time) {
		try {
			Thread.sleep(time);
		} catch (InterruptedException e) {
			e.printStackTrace();
		}
	}

	/*
	 * This method waits until play is ready (until start button is pressed) after
	 * which it updates the moves in turn until time runs out (player won) or player
	 * is eaten up (player lost).
	 */
	public void gameStart() throws Exception {
		do {
			play();
			player.setReady(false);
		} while (checkGame);
	}

	private void reset() throws Exception {
		grid = new Grid(difficulty);
		player = new Player(grid, 0, 0);
		monsters.clear();
		monsters.add(new Monster(grid, player, 5, 5));
		boardPanel.reset(grid, player, monsters);
		player.setReady(false);
		player.setEnergy(energy);
		score = 0;
		boardPanel.repaint();
	}

	// Game Run
	public String play() throws Exception {
		int time = 0;
		boolean check = true;
		boolean checkEaten = false;
		String message;
		player.setDirection(' '); // set to no direction
		while (!player.isReady())
			delay(100);
		do {
			while (isPause)
				delay(100);
			Cell newPlayerCell = player.move(player.getPresses());
			ArrayList<Cell> MonstersCell = new ArrayList<>();

			for (int i = 0; i < monsters.size(); ++i)
				MonstersCell.add(monsters.get(i).move(1));
			for (int i = 0; i < monsters.size(); ++i) {
				if (newPlayerCell == monsters.get(i).getCell() && MonstersCell.get(i) ==
player.getCell()) {
					checkEaten = true;
				}
			}

			if (!checkEaten) {
				player.setDirection(' '); // reset to no direction
				// update time and repaint
				time++;
				score += (3 - difficulty) * (2 - (double) gameDelay / 1000);

				if (time % produceTime == 0) {
					Monster baby = new Monster(grid, player, MonstersCell.get(0).row,
MonstersCell.get(0).col);
					baby.isBaby = true;
					monsters.add(baby);
				}
				for (int i = 1; i < monsters.size(); ++i) {
					if (monsters.get(i).isBaby() && monsters.get(i).getCell() ==
newPlayerCell) {
						monsters.remove(i);
					}
				}

				energyLabel.setText("Energy : " + player.getEnergy());
				scoreLabel.setText("Score : " + score);
				timeLabel.setText("Time Remaining : " + (timeAllowed - time));
				delay(gameDelay);
				boardPanel.repaint();
			} else
				check = false;

		} while (time < timeAllowed && check && player.isReady());
		message = time < timeAllowed ? "Player Lost" : "Player Won"; // players has been eaten up
```

```
        userData.saveScore(score);
        timeLabel.setText(message);
        return message;
}

class MyActionListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
                String label = e.getActionCommand();
                if (label.equals("LogIn")) {
                        new Login();
                }
                if (label.equals("Register")) {
                        new Register();
                }
                if (label.equals("Rank")) {
                        new Rank();
                }
                if (label.equals("Setting")) {
                        new Setting();
                }
                if (label.equals("Start")) {
                        isPause = false;
                        if (userData.isLogin())
                                player.setReady(true);
                        else
                                JOptionPane.showMessageDialog(null, "you need to login first");
                }
                if (label.equals("Restart")) {
                        try {
                                reset();
                        } catch (Exception e1) {
                                e1.printStackTrace();
                        }
                }
                if (label.equals("Pause")) {
                        isPause = true;
                }
        }
}

class Rank extends JFrame {
        public Rank() {
                setTitle("Rank");
                Container container = getContentPane();
                container.setLayout(null);
                JLabel label1 = new JLabel("Name");
                container.add(label1);
                JLabel label2 = new JLabel("Score");
                container.add(label2);

                User[] userList = userData.getList();
                JLabel[] nameList = new JLabel[userList.length];
                JLabel[] scoreList = new JLabel[userList.length];
                for (int i = 0; i < userList.length; ++i) {
                        nameList[i] = new JLabel(userList[i].getUserName());
                        scoreList[i] = new JLabel("" + userList[i].getScore());
                        container.add(nameList[i]);
                        container.add(scoreList[i]);
                }
                container.setLayout(new GridLayout(userList.length + 2, 3, 40, 40));
                JButton button = new JButton("Confirm");
                button.setSize(20, 20);
                button.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                                dispose();
                        }
                });

                container.add(button);
                setBounds(0, 0, 300, 300);
                setAlwaysOnTop(true);
                setResizable(false);
                setLocationRelativeTo(null);
                setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
                setVisible(true);
        }
}

class Login extends JFrame {
        public Login() {
                setTitle("Login");
                JLabel label1 = new JLabel("UserName");
                label1.setBounds(10, 10, 200, 18);
                JLabel label2 = new JLabel("Password");
                label2.setBounds(10, 50, 200, 18);
                final JTextField textField1 = new JTextField();
                textField1.setBounds(90, 10, 150, 18);
                JPasswordField passwordField = new JPasswordField();
```

```
                        passwordField.setBounds(90, 50, 150, 18);
                        final JButton button1 = new JButton("Confirm");
                        button1.addActionListener(new ActionListener() {
                                public void actionPerformed(ActionEvent e) {
                                        boolean flag = userData.login(textField1.getText(),
String.valueOf(passwordField.getPassword()));
                                        if (flag) {
                                                JOptionPane.showMessageDialog(button1, "Login
Successful");
                                                dispose();
                                        } else {
                                                JOptionPane.showMessageDialog(button1, "Login Fail");
                                        }
                                }
                        });
                        button1.setBounds(40, 80, 100, 18);
                        JButton button2 = new JButton("Cancel");
                        button2.addActionListener(new ActionListener() {
                                public void actionPerformed(ActionEvent e) {
                                        dispose();
                                }
                        });
                        button2.setBounds(150, 80, 100, 18);
                        Container container = getContentPane();
                        container.setLayout(null);
                        container.add(label1);
                        container.add(label2);
                        container.add(textField1);
                        container.add(passwordField);
                        container.add(button1);
                        container.add(button2);
                        setBounds(0, 0, 300, 150);
                        setAlwaysOnTop(true);
                        setResizable(false);
                        setLocationRelativeTo(null);
                        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
                        setVisible(true);
                }
        }

        class Register extends JFrame {
                public Register() {
                        setTitle("Register");
                        JLabel label1 = new JLabel("UserName");
                        label1.setBounds(10, 10, 200, 18);
                        JLabel label2 = new JLabel("Password");
                        label2.setBounds(10, 50, 200, 18);
                        final JTextField textField1 = new JTextField();
                        textField1.setBounds(90, 10, 150, 18);
                        JPasswordField passwordField = new JPasswordField();
                        passwordField.setBounds(90, 50, 150, 18);
                        final JButton button1 = new JButton("Confirm");
                        button1.addActionListener(new ActionListener() {
                                public void actionPerformed(ActionEvent e) {
                                        User user = new User(textField1.getText(),
String.valueOf(passwordField.getPassword()));
                                        if (userData.register(user))
                                                JOptionPane.showMessageDialog(button1, "Register
Successful");
                                        else
                                                JOptionPane.showMessageDialog(button1, "Register Fail");
                                        dispose();
                                }
                        });
                        button1.setBounds(40, 80, 100, 18);
                        JButton button2 = new JButton("Cancel");
                        button2.addActionListener(new ActionListener() {
                                public void actionPerformed(ActionEvent e) {
                                        dispose();
                                }
                        });
                        button2.setBounds(150, 80, 100, 18);
                        Container container = getContentPane();
                        container.setLayout(null);
                        container.add(label1);
                        container.add(label2);
                        container.add(textField1);
                        container.add(passwordField);
                        container.add(button1);
                        container.add(button2);
                        setBounds(0, 0, 300, 150);
                        setAlwaysOnTop(true);
                        setResizable(false);
                        setLocationRelativeTo(null);
                        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
                        setVisible(true);
                }
        }
```

```
	class Setting extends JFrame {
		public Setting() {
			setTitle("Setting");

			JLabel gameDifficulty = new JLabel("Game Difficulty");
			gameDifficulty.setBounds(10, 10, 200, 18);
			JRadioButton easy = new JRadioButton("Easy");
			JRadioButton normal = new JRadioButton("Normal", true);
			JRadioButton hard = new JRadioButton("Hard");
			ButtonGroup group1 = new ButtonGroup();
			group1.add(easy);
			group1.add(normal);
			group1.add(hard);

			JLabel gameDuration = new JLabel("Game Duration");
			gameDuration.setBounds(10, 50, 200, 18);
			JRadioButton gd1 = new JRadioButton("100");
			JRadioButton gd2 = new JRadioButton("200", true);
			JRadioButton gd3 = new JRadioButton("300");
			ButtonGroup group2 = new ButtonGroup();
			group2.add(gd1);
			group2.add(gd2);
			group2.add(gd3);

			JLabel gameFrequency = new JLabel("Game Frequency");
			gameFrequency.setBounds(10, 100, 200, 18);
			JRadioButton gf1 = new JRadioButton("0.2s/m");
			JRadioButton gf2 = new JRadioButton("0.5s/m", true);
			JRadioButton gf3 = new JRadioButton("1s/m");
			ButtonGroup group3 = new ButtonGroup();
			group3.add(gf1);
			group3.add(gf2);
			group3.add(gf3);

			JLabel playerEnergy = new JLabel("Player Energy");
			playerEnergy.setBounds(10, 100, 200, 18);
			JRadioButton pe1 = new JRadioButton("40E");
			JRadioButton pe2 = new JRadioButton("200E", true);
			JRadioButton pe3 = new JRadioButton("1000E");
			ButtonGroup group4 = new ButtonGroup();
			group4.add(pe1);
			group4.add(pe2);
			group4.add(pe3);

			final JButton button1 = new JButton("Confirm");
			button1.addActionListener(new ActionListener() {
				public void actionPerformed(ActionEvent e) {
					if (easy.isSelected())
						difficulty = 2;
					else if (normal.isSelected())
						difficulty = 1;
					else if (hard.isSelected())
						difficulty = 0;
					if (gd1.isSelected())
						timeAllowed = 100;
					else if (gd2.isSelected())
						timeAllowed = 200;
					else if (gd3.isSelected())
						timeAllowed = 300;
					if (gf1.isSelected())
						gameDelay = 200;
					else if (gf2.isSelected())
						gameDelay = 500;
					else if (gf3.isSelected())
						gameDelay = 1000;
					if (pe1.isSelected())
						energy = 40;
					else if (pe2.isSelected())
						energy = 200;
					else if (pe3.isSelected())
						energy = 1000;
					game.setSize((int) (640 * (1 + difficulty * 0.25)), (int) (480 * (1
+ difficulty * 0.25)));
					try {
						reset();
					} catch (Exception e1) {
						e1.printStackTrace();
					}
					dispose();
				}
			});
			button1.setBounds(30, 100, 100, 18);

			JButton button2 = new JButton("Cancel");
			button2.addActionListener(new ActionListener() {
				public void actionPerformed(ActionEvent e) {
					dispose();
```

```
                }
            });
            button2.setBounds(140, 100, 100, 18);
            JPanel container = new JPanel();

            container.add(gameDifficulty);
            container.add(easy);
            container.add(normal);
            container.add(hard);

            container.add(gameDuration);
            container.add(gd1);
            container.add(gd2);
            container.add(gd3);

            container.add(gameFrequency);
            container.add(gf1);
            container.add(gf2);
            container.add(gf3);

            container.add(playerEnergy);
            container.add(pe1);
            container.add(pe2);
            container.add(pe3);

            container.add(button1);
            container.add(button2);
            this.add(container);
            setBounds(0, 0, 350, 200);
            setAlwaysOnTop(true);
            setResizable(false);
            setLocationRelativeTo(null);
            setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
            setVisible(true);
        }
    }
}
```