```java
/**
     * Returns the root pane of the game.
     *
     * @return The root pane of the game.
     */
    public Pane getRoot() {
        if (root == null) {
            start(new Stage());
            root = new Pane();
        }
        return root;
    }
```

```java
/**
     * The main method that launches the JavaFX application.
     *
     * @param args The command line arguments.
     */
    public static void main(String[] args) {
        launch(args);
    }
```

```java
/**
     * Initializes the game window and sets up the game elements.
     *
     * @param primaryStage The primary stage of the JavaFX application.
     */
    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("Continuous Jumping Game");
        root = new Pane();
        Scene scene = new Scene(root, WIDTH, HEIGHT);

        initPlayer();
        initGround();
        initCoin();
        initMissile();

        initScoreDisplay();
        initLevelDisplay();

        scene.setOnKeyPressed(event -> {
            if (event.getCode() == KeyCode.SPACE) {
                jump();
            }
        });

        root.setBackground(setBackground(BACKGROUND_IMAGES[0]));

        primaryStage.setScene(scene);
        primaryStage.show();

        AnimationTimer gameLoop = new AnimationTimer() {
            @Override
            public void handle(long now) {
                update();
                moveCoin();
                moveMissile();
                updateLevel();
            }
        };
        gameLoop.start();
    }
```

```java
// Player methods

    /**
     * Initializes the player character.
     */
    private void initPlayer() {
        player.setX(50);
        player.setY(HEIGHT - GROUND_HEIGHT - PLAYER_SIZE);
        player.setPreserveRatio(true);
        Group rootGroup = new Group(player);
        root.getChildren().add(rootGroup);
    }

    /**
     * Makes the player character jump.
     */
    private void jump() {
        velocityY = -JUMP_STRENGTH;
    }
```

```java
// Ground methods

    /**
     * Initializes the ground.
     */
    private void initGround() {
        Rectangle ground = new Rectangle(0, HEIGHT - GROUND_HEIGHT, WIDTH, GROUND_HEIGHT);
        ground.setFill(Color.GREEN);
        root.getChildren().add(ground);
    }

    // Coin methods

    /**
     * Initializes the coin.
     */
    private void initCoin() {
        coin = new Circle(HEIGHT, COINY, COIN_SIZE);
        coin.setFill(Color.YELLOW);
        root.getChildren().add(coin);
    }

    /**
     * Moves the coin horizontally.
     */
    private void moveCoin() {
        coin.setCenterX(coin.getCenterX() - FACTOR * SPEED);
        if (coin.getCenterX() + COIN_SIZE < 0) {
            coin.setCenterX(WIDTH + COIN_SIZE);
        }
    }
```

```java
// Missile methods

    /**
     * Initializes the missile.
     */
    private void initMissile() {
        missile.setX(100);
        missile.setY(100);
        missile.setPreserveRatio(true);
        Group rootGroup = new Group(missile);
        root.getChildren().add(rootGroup);
    }

    /**
     * Moves the missile horizontally.
     */
    private void moveMissile() {
        missile.setX(missile.getX() - FACTOR * SPEED);
        if (missile.getX() + COIN_SIZE < 0) {
            missile.setX(WIDTH + COIN_SIZE);
        }
    }
```

```java
// Score display methods

    /**
     * Initializes the score display.
     */
    private void initScoreDisplay() {
        scoreText = new Text("Score: " + score);
        scoreText.setFill(Color.BLACK);
        scoreText.setFont(Font.font("Arial", FontWeight.BOLD, 24));
        scoreText.setLayoutX(20);
        scoreText.setLayoutY(40);
        root.getChildren().add(scoreText);
    }

    /**
     * Updates the score display.
     */
    private void updateScoreDisplay() {
        scoreText.setText("Score: " + score);
    }
```

```java
// Level display methods

    /**
     * Initializes the level display.
     */
    private void initLevelDisplay() {
        levelText = new Text("Level: " + level);
        levelText.setFill(Color.BLACK);
        levelText.setFont(Font.font("Arial", FontWeight.BOLD, 24));
        levelText.setLayoutX(WIDTH - 120);
        levelText.setLayoutY(40);
        root.getChildren().add(levelText);
    }

    /**
     * Updates the level display.
     */
    private void updateLevelDisplay() {
        levelText.setText("Level: " + level);
    }

```

```java
// Game logic methods

/**
 * Updates the game logic.
 */
private void update() {
    velocityY += GRAVITY;
    player.setY(player.getY() + velocityY);

    if (player.getY() >= HEIGHT - GROUND_HEIGHT - PLAYER_SIZE) {
        player.setY(HEIGHT - GROUND_HEIGHT - PLAYER_SIZE);
    }

    if (checkCollision(player, missile)) {
        Random random = new Random();

        missile.setVisible(false);
        missile.setX(HEIGHT);
        missile.setY(random.nextInt(0, (int) (HEIGHT - COIN_SIZE)));
        missile.setVisible(true);
        score++;
        updateScoreDisplay();
        updateLevelDisplay();
    }

    if (checkCollision(player, coin)) {
        Random random = new Random();
        coin.setVisible(false);
        coin.setCenterX(WIDTH + COIN_SIZE);
        coin.setCenterY(random.nextInt(0, (int) (HEIGHT - COIN_SIZE)));
        coin.setVisible(true);
        score--;
        updateScoreDisplay();
        updateLevelDisplay();
    }
}
```

```java
/**
     * Updates the level of the game.
     */
    private void updateLevel() {
        if (score == 5) {
            level++;
            score = 0;
            increaseMissileSpeed();
        }
        if (level == MAX_LEVEL) {
            freezeAllMovement();
            updateScoreDisplay();
            updateLevelDisplay();
            displayVictory();
        }

        if (level == 4) {
            root.setBackground(updateBackground(BACKGROUND_IMAGES[1]));
        }

        if (level == 7) {
            root.setBackground(updateBackground(BACKGROUND_IMAGES[2]));
        }
    }
```

```java
/**
     * Increases the speed of the missiles.
     */
    private void increaseMissileSpeed() {
        FACTOR++;
    }

    /**
     * Displays the victory message.
     */
    private void displayVictory() {
        Text victoryText = new Text("You win!");
        victoryText.setFill(Color.BLACK);
        victoryText.setFont(Font.font("Arial", FontWeight.BOLD, 24));
        victoryText.setLayoutX(WIDTH / 2 - 50);
        victoryText.setLayoutY(HEIGHT / 2);
        root.getChildren().add(victoryText);
    }
```

```java
/**
     * Freezes all movement in the game.
     */
    private void freezeAllMovement() {
        velocityY = 0;
        FACTOR = 0;
    }
```

```java
/**
     * Checks for collision between the player and the missile.
     *
     * @param player The image view of the player.
     * @param missile The image view of the missile.
     * @return true if there is a collision, false otherwise.
     */
    private boolean checkCollision(ImageView player, ImageView missile) {
        return player.getBoundsInParent().intersects(missile.getBoundsInParent());
    }
```

```java
/**
     * Checks for collision between the player and the coin.
     *
     * @param player The image view of the player.
     * @param coin The circle representing the coin.
     * @return true if there is a collision, false otherwise.
     */
    private boolean checkCollision(ImageView player, Circle coin) {
        return player.getBoundsInParent().intersects(coin.getBoundsInParent());
    }
```

```java
// Background methods

    /**
     * Sets the background of the game.
     *
     * @param imageUrl The file path of the background image.
     * @return The background with the specified image.
     */
    private Background setBackground(String imageUrl) {
        Image image = new Image(imageUrl);

        BackgroundImage backgroundImage = new BackgroundImage(
                image,
                BackgroundRepeat.NO_REPEAT,
                BackgroundRepeat.NO_REPEAT,
                BackgroundPosition.CENTER,
                new BackgroundSize(BackgroundSize.AUTO, BackgroundSize.AUTO, false, false, true, false)
        );

        return new Background(backgroundImage);
    }
```

```java
/**
     * Updates the background of the game.
     *
     * @param imageUrl The file path of the updated background image.
     * @return The updated background with the specified image.
     */
    private Background updateBackground(String imageUrl) {
        Image image = new Image(imageUrl);

        BackgroundImage backgroundImage = new BackgroundImage(
                image,
                BackgroundRepeat.NO_REPEAT,
                BackgroundRepeat.NO_REPEAT,
                BackgroundPosition.CENTER,
                new BackgroundSize(BackgroundSize.AUTO, BackgroundSize.AUTO, false, false, true, false)
        );

        return new Background(backgroundImage);
    }
```