**OOP Game Report**

*A simple game written in Java applying OOP principles*

**Members**

Trần Nguyễn Thương Trường – ITITIU19228

Lê Quốc Phong – ITITIU19181

1. **Introduction:**

In this report, we introduce our remake version of the Snake game. We developed this game by JSwing framework of Java. We will give you a brief and concise overview of our game. The following sections are organized as follows:

* **Introduction**
* **Github repository**
* **Play and Rule**
* **Score and level**
* **Diagram**

1. **Github repository:**

Click [**here**](https://github.com/thuongtruong1009/Snake-Game-OOP) to have more details about the project through the source code.

1. **Game Rules :**

The player controls a dot, square, or object on a bordered plane. As it moves forward, it leaves a trail behind, resembling a moving snake. In some games, the end of the trail is in a fixed position, so the snake continually gets longer as it moves. In another common scheme, the snake has a specific length, so there is a moving tail a fixed number of units away from the head.

Eating food makes the snake grow. When the food is eaten it moves to another random position.

The snake will wrap around to the other side of the screen when it goes off the edge.

The game is over when the snake crashes into itself and the screen border.

1. **Score and level :**

* The player's score should be based on the number of apples eaten (or equivalently, the length of the snake). The exact scoring algorithm is up to you; most games award more points for each apple as the snake gets longer.
* Keep the ten highest scores, along with the names of the players who made those scores. The scores should be kept across all runs of the game. That is, they should not be lost when the program ends.
* Keeping (and displaying) high scores isn't as "glamorous" as writing the game itself, but it is harder than you might expect (and more generally useful).
* We have three level of snake game that the player can choose.
  + Easy
  + Medium
  + Hard

The levels will determine the speed of the snake on the screen.

1. **Diagram and Design Explanation:**
2. **Firstly, class LoginPage that apper first when run code:**
3. LoginPage(): constructor that contain JFrame to display

* Some method of swing.JFrame to create JFrame
* Create JmenuBar, JMenuItem(newGame)
* Create JMenuItem (level) to chose level want playing. Every level(easy, medium, difficult) have different speed.
* Performed action for every JmenuItem when click on it.

1. play(): start game when click on new game button.

* set speed value when player click chose level
* called start() function of class GamePanel to start game.
* Print announcement and speed chosen out screen.

1. main(): this is function that running first when run code.
2. **Secondly, GamePanel class that include all funtions to control snake. This class will be implements java.awt.event.ActionListener to peform Action from keyboard.**
3. GamePanel(int SPEED): constructor contain SPEED variable:
   * Random(): this function that help apple appear randomly position on board
   * music: import music from system by ImageLoader class and start sound when game started.
   * startGame(): this function to start game.
   * Some function to create and set the panel.
4. startGame():

* newApple(): create new apple after snake eaten before them.
* running==true: the snake will travel.
* play(): this function extends from LoginPage class. When start game will run with different speeds.

1. paintComponent(Graphics g):

* paintComponent(g): extends from awt.Graphics to draw components.
* draw(g): call function draw(g).

1. draw(Graphics g):

* if(running): draw rows, coloums and coordinates of panel. Fill color for apples and color for the extra length of bodyParts. Draw text score announcement
* else(): this function will draw game over screen when snake stop .

1. newApple(): this function that create next new apple according units size & coordinate XY.
2. move(): this function that help snake follow 4 directions(UP, DOWN, LEFT, RIGHT):

* switch case(): every case will have different direction.

1. checkApple(): this function have tasks are check, count the number of eaten apples and length of bodyParts

* newApple(): called from newApple() function. Make new apple after sum number of them to bodyParts.
* If coordinate XY of apple the same position of head snake: this apple will change to 1 unit of bodyParts, score will be plus +1 and new apple appear.

1. checkCollision(): this function will check when case that makes game stop

* use if() function to stop snake when head of snake touches 1 in 4 border of panel game.
* When coordinate XY position head & tail of snake the same, game will stop.

1. gameOver(Graphics g): this function will draw text to announcement score that player achieved.

* Some draw function of awt.Graphics.

1. @Override

actionPerformed(ActionEvent e): this is function that perform actions when snake eat apple, collision border…. This function have performed when implements awt.events.ActionListener**.**

1. Inner class myKeyAdapter that extends from awt.event.KeyAdapter to listen and active events is control direction of snake when we press the keyboard.

* keyPressed(KeyEvent e): this function catch the action after press keyboard.
* Switch case for 4 direction.

1. **Thirdly, class ImageLoader: is place to contain and translate sound, image of game**

* use static BufferedImage loaderImage(String path): to get image from system and open it in class.
* use static Clip LoadSound(String direction): to get sound from system and play it when called.

1. **Finally, class WelcomePage that create JFrame contain GamePanel class:**

* WelcomePage(): constructor control and perform JFrame.
* Some method of swing.JFrame and add Panel() from GamePanel class.