# Snake Game

Group 1 :

Raihan Farid
Erlangga Wahyu Utomo
Mohammed Fachry Dwi Handoko

# Explanation

- We make a platform game called ‘snake game’, the player will be the ‘snake’ that will eat the corresponding ‘target’
- We move the snake using ‘arrow’ button
- Everytime the snake eat the food, the score increased
- The edge of the map will loop back into the platform
- The snake died if it eat its own body
- We can respawn by pressing button ‘R’

# Implementation of Encapsulation

In our code, we implement ‘encapsulation’ almost in every aspect of our game

We ‘encapsulate’ the data and use it in another class to make it easier accessing the data in each class

Example:

We declare class “Enemy”

```
8  public class Enemy {
9
10     private int xposEnemy;
11     private int yposEnemy;
12     private Image enemyImage;
13
14     public Enemy() {
```

And call that class in another class called “Gameplay” to change one or another data in class “enemy”, can be seen in the image

```
52     private void init() {
53
54         s = new Snake();
55         e = new Enemy();
56         e.setXposEnemy(25 + (rand.nextInt(34) * 25));
57         e.setYposEnemy(75 + (rand.nextInt(23) * 25));
58         score = 0;
59
```

# Implementation of Inheritance and Polymorphism

- We inherit most of our public class in the gameplay superclass
- Example:
- We declare a class called snake

```
9  public class Snake {
10
11     private int[] snakeXLength = new int[750];
12     private int[] snakeYLength = new int[750];
13     private Image rightMouth, leftMouth, downMouth, upMouth, snakeImage;
14     private boolean left = false, right = false, up = false, down = false;
15     private int lengthOfSnake;
16     private boolean moves;
17     private boolean dead;
```

- And reuse it in the class called ‘gameplay’ to initialize new object

```
52     private void init() {
53
54         s = new Snake();
55         e = new Enemy();
56         e.setXposEnemy(25 + (rand.nextInt(34) * 25));
57         e.setYposEnemy(75 + (rand.nextInt(23) * 25));
58         score = 0;
59
```

# Implementation of Override

- We use override to checking the user input if the button is pressed or released
- Example:

```
162     @Override
163     public void keyReleased(KeyEvent e) {
164         if (!s.isDead()) {
165             if (e.getKeyCode() == KeyEvent.VK_SPACE) {
166                 delay = savedDelay;
167                 timer.stop();
168                 initTimer(delay);
169             }
170         }
171     }
```

# Implementation of Composition

- Our code mainly focus in composition because the class ‘gameplay’ cannot run itself without class from ‘snake’ ‘enemy’ or ‘gameImage’
- Example:
- The class ‘gameplay’ that used ‘snake’, ‘enemy’, and ‘gameImages’

```
22 public class Gameplay extends JPanel implements KeyListener, ActionListener {
23
24
25
26     private Random rand = new Random();
27     private Snake s = new Snake();
28     private Enemy e = new Enemy();
29     private gameImages gi = new gameImages();
30
31     private int score;
32     private Timer timer;
33     private int delay = 100;
34     private int savedDelay;
```

# Thank you For your attention