**Project Milestone of Gluttonous Snake game**

**COMPT\_220L shuhan**

**Abstract**

The game is simple and fun, so I want to try to use java as a simple snake, use the keyboard to control the direction, through the observer mode, the snake will respond accordingly when the player presses the button. Use the game to control the direction of the snake up and down, look for something to eat, every time you eat a bit you can get a certain score, and the snake body will eat longer and longer, the longer the body is playing, the more difficult it is to touch the wall. Can't bite into your body, you can't bite your tail, wait for a certain score, you can pass, then continue to play the next level

**Introduction**

Application scenario: The keyboard listener uses the observer mode. The observer is a snake. The observer is the keyboard button. The player presses the “up, down, left and right” direction keys on the keyboard. When the observer changes, the observer Being able to get the message in time and respond accordingly, the snake will respond to this button event in time, and then go in the corresponding direction. snake is a computer game that was produced in the mid to late 1970s. This type of game was re-popular in the 1990s dues to the introduction of some small screen devices, which can be installed on current mobile phones. The version is also different.

In the game, the player controls a slender straight line (commonly known as a snake or insect), which will keep moving forward. The player can only manipulate the head of the snake (up, down, left, and right), picking up the touched object (or called "Beans") and avoid touching yourself or other obstacles. Every time a snake eats a piece of food, its body grows. After eating some food, the speed of the snake will gradually increase, making the game more difficult. The game design is roughly divided into four walls (all are not traversable) and a certain part of the wall can be traversed, and the four walls can pass through the pattern

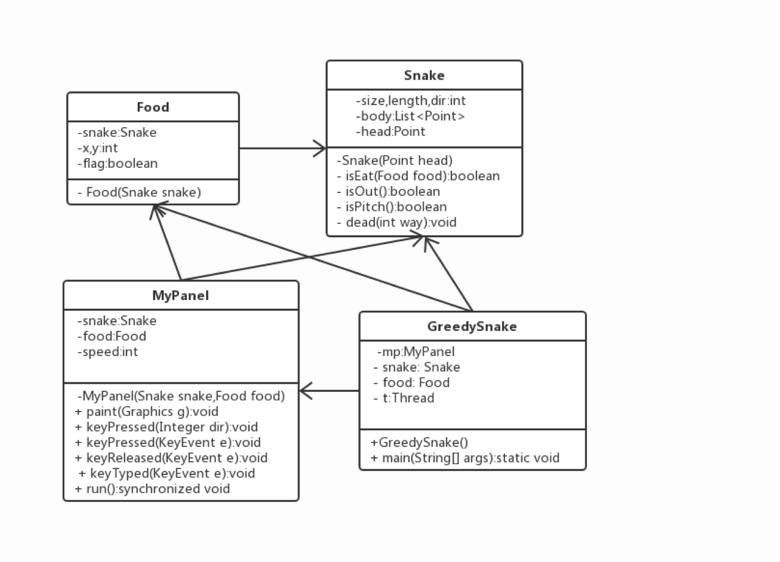
**Detailed System Description (process and thinking)**

1. Observer - Snake (jud **Detailed System Description** ge whether the snake has the isEat function of eating food, determine whether the snake exceeds the limit of the isOut function, determine whether the snake's head touches its body's isPitch function, and a snake exits the program after death. Dead function.)

2. The object associated with the snake - food (the constructor initializes the x, y coordinates and the flag flag. When the snake's head touches the food, that is, the snake's head or the coordinates of the body and the coordinates of the thing, it means that the food is eaten.)

3. Observer and map-window components and keyboard events (MyPanel class) (paint function is the snake's head, body, food fill color, call the fillRect and setColor functions in the Graphics class; keyPressed, used here two the secondary listener converts the button event into the change of the direction of the snake head for the first time, and transforms the change of the direction of the snake head into the change of the actual direction of the snake head for the second time.

4. Main function - application (GreedySnake class) (initialize the position of the snake head, initialize the position of the food, initialize the thread, and then add the keyboard listener whose parameters are mp, set some control panel related content



**Detailed System Description(Final)**

First, the most basic element of the game is the map. The classes used for drawing in java are swing and awt, and the swing class is mainly used here. The classes used for window display in swing are JFrame and its subclasses. JFrame can add components directly, but the essence is to add components to a default panel in JFrame. For code clarity, I will use the JPanel panel to draw all the animations, then add the panel to the JFrame form.

We will wonder what is the snake body that is greed snake? How to make it? We can understand the Snake as a collection, with a fixed starting element that represents the snake body at the beginning of the game. When a snake eats a point, the set adds an element and the length of the snake increases by one. So, what are the elements in the collection? To understand this problem, you must first pay attention to the environment in which the snake body moves. In the JFrame form, the position is distinguished by the X and Y axis coordinates. Snake can be seen as a close-knit point that is displayed on the axis. Whenever you move in a certain direction, the coordinates of the snake change according to a certain rule. For example, when we manipulate the Snake to move upwards, the Y axis of the snake's overall coordinates is decremented by one; if the snake's first coordinate coincides with a certain coordinate of the snake body, it means that the snake encounters itself; if the snake the first coordinate hits the border and the snake hits the wall. This is the essence of the snake. Let’s build the object that builds each point on the snake. The snake body is made up of one such object.

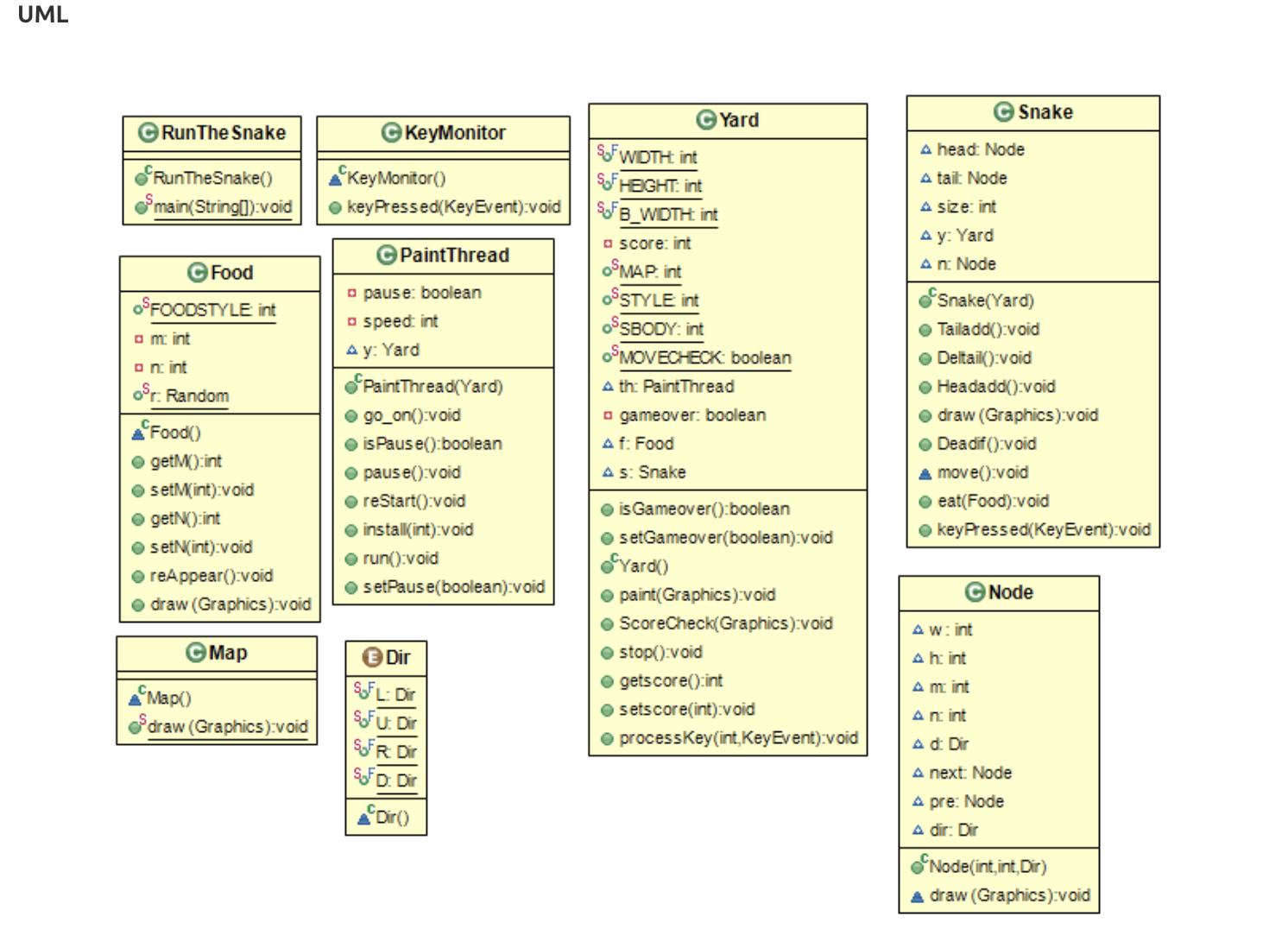
Next, we have to set the range for each snake's body, because the snake has a limit of the range of movement, beyond a certain distance or length, it will lead to the termination of the game.

Then the next, we need a collection to store the points on the snake. We need to define a variable to represent the randomly occurring point (the target of the snake) and define a variable Length to represent the length of the snake.

After defining the member variables of the class, we start to define the constructor so that the program will start running after the object of mainMap is constructed. We need to add some elements to the collection in the constructor, which represents the initial snake body. We also need to use a method to create random points. Random pints cannot appear on the snake, that is, the coordinates of random points cannot be arbitrary with the body of the snake. The coordinates are the same, otherwise there will be a BUG.

The next step in our game is to control the movement of the snake. We use "wsad" or "up, down, left and right" on the keyboard to control the movement of the snake body. Many of these original ideals can be immediately thought of: the listener. Here we have to set the listener object is no longer a button, a label, but the entire panel. We need to add a keyboard listener to the entire panel to monitor its actions on the keyboard. Here we unify and use "↑↓←→" to control the direction. When we use the keyboard to capture the corresponding action

We can find the KeyEvent, which has static constants to represent the corresponding touches on the keyboard. VK\_UP stands for the up arrow, VK\_DOWN stands for the down arrow, VK\_LEFT stands for the left arrow, and VK\_RIGHT stands for the right arrow. We can immediately think of it: get the keyboard event through the getKeyCode method, and compare it with the four constants. If it matches, you can call the method in the corresponding direction to move the snake body. We can define a Move () method and define a variable direction to represent the direction. By assigning a different value to the direction to Move (), we can have different movement effects on the snake body.

****

**User Manual**

1. The score is displayed in the upper left corner of the screen.

2, each plus one plus 5 points

3, the default collision boundary is not dead

4, the default 30 points to upgrade a speed grade

5, the default is to change the map after each speed is completed and the speed is restored to the default, the points are zero

Function key:

1. Press F1 to start over.

2. Press the space bar to pause/continue

3. Press the 1/2/3/4/5 number keys to set the snake's moving speed (gradually faster)

4. Press F2 to change the map (3 types)

5, press F3 ordinary snake / color snake switch

6. Press F4 to change the game mode (the snake hits the outermost periphery to die or returns from the other end

**Conclusion**

I want to write this game a long time ago. As a classic game on mobile phones, Snake and Tetris have brought us a lot of fun in our childhood.

**References/Bibliography**

**《Java Basics》 Tsinghua University Press**