Homework Assignment

7 Snake

Snake is a simple computer game designed in the 1970s and often found on cell phones. The object is to guide a moving a snake, i.e., a chain of blocks or other shapes around the screen such that it doesn't run into obstacles (walls, poison, itself) and grows by running over food. When the snake runs over food, it will grow, i.e., get longer.

This assignment is to write a basic version of the game.

7.1 Implementation

7.1.1 Grid

Use a rectangular grid of so-called cells of the same size (e.g., 10 by 10 pixels) as the playing field. A cell can be occupied by a piece of wall, by a segment of the snake, by food, etc. You may use a two-dimensional array of integers for this grid, where the integers represent the various occupations, for example:

```
final int EMPTY = 0;
final int WALL = −1;
final int SNAKE = −2;
// food is a positive value representing the amount of food
```

Alternatively, you can use objects of a self-designed class or use enumarations (`Enum`) (if you know how to make and use these).

7.1.2 Doubly Linked List

It is *required* that the snake is represented as a doubly linked list of nodes that each contain a `Point` (from the package `java.awt`). A Point is an object with two (publicly accessible) `int`s `x` and `y`. The nodes of the linked list represent the segments of the snake, each Point containing the coordinates of the corresponding cell in the grid.

The doubly linked list should have the following methods, with the behavior suggested by their name:

- `public boolean isEmpty()`
- `Point getFirst()`
- `Point getLast()`
- `public void addFirst( Point p )`
- `public void AddLast( Point p )`
- `public Point removeFirst()`
- `public Point removeLast()`
- `public String toString()`

Note that there has to be a class `Node` for the elements of the list.

7.1.3 Movement

The snake moves in steps, triggered by a Timer. Use an interval of about 100 ms. At each step, the head of the snake moves one cell in a certain direction, being up, down, left, or right (no diagonal movement) and the other segments follow.

Implement this movement by removing the tail segment and adding a new head segment at the cell where the snake is moving (unless it has eaten food recently, see below). The other segments can stay where they are.

7.1.4 Food

When the snake has moved over a piece of food, it will grow a number of segments equal to the amount of food. For a number of time steps equal to the amount of food, the tail segment will not be removed but stay where it is. A new head segment is added as usual, so the snake will effectively grow one segment during each of these time steps.

You can display food on the screen as a number between 1 and 9. At the start of the game there should be at least 5 portions of food on the screen. The game starts when the user presses an arrow key. Make sure there is at least one piece of food at the same row or column as where the snake starts, thus simplifying testing for the grader.

7.1.5 Control

As long as no key is pressed, the snake moves in the same direction. When the user presses an arrow key, the snake starts moving in that direction. There are two ways to implement this. One possibility is that the snake changes course on the next time event. Another possibility is to have the snake move immediately at the press of the key, and then again on the next time event. Both choices are allowed.

7.1.6 Death

In the classic version of the game, the snake dies (and the game is over) when the snake runs into wall, poison, or itself. To make testing easier, it is allowed in this game that the snake does not die, but the move into the wall or other obstacle does not take place and a message is printed. When the user presses an arrow key in a direction where there is no obstacle, the snake will move into that direction.

7.1.7 Starter files

Some starter files are provided on the wiki. These contain classes with some partially implemented methods, etc. Methods have to be completed, instance variables and other methods may have to be added. More classes may have to be added.

It is not required that you follow the design of the starter files.

7.2 Embellishments

The version of the game described above will earn you maximally 7.5 points. Implement embellishments to get more points. Suggestions:

1. Display head and tail differently from the other segments.
2. Add poison at random places that the snake dies from when crawling over it.

3. Create food during the game.
4. Increase the speed of the snake when it has gotten a certain length.
5. Add worm holes, which are hidden connections between two cells in the grid. When the snake enters one end of a worm hole, its head will appear at the other end and the other segments will follow.
6. Poison does not kill the worm, but will confuse it. For a certain amount of time steps the arrow keys will behave in a non-standard way. E.g., up is left, down is up, etc.