CMPE 160 Introduction to Object Oriented Programming,

SPRING 2018 – PROJECT 2

PROJECT REPORT

Problem Description:

The problem is writing a java program which plays Snakes game itself. The snakes move and hunt for food themselves without any input from the user. Snakes are able to move in four directions(up, down, left, right) and they cannot go past the world boundaries. At any time there should be a randomly placed food on the grid so that snakes can try to find and eat the food. Initially the snake's size is 4. Once a snake eats the food and grows by 1. When a snake reaches the size of 8(maximum size), it divides and produces 2 seperate snakes of size 4. So the game will follow this path and the snakes will multiply and eat for more food.

Class and Package Explanation:

My project contains 5 packages.

1) Game: This package contains the basic classes that create the game(such as GridGame.java). I do not make any changes in this package's classes.
   a) **Direction.java**
   b) **Drawable.java**
   c) **GridGame.java** (abstract class)

2) Main
   a) **Main.java** that can be used as a playground to test my project. It creates and starts the game, adds the food and snake, creates application window that contains the game panel. **I can change the game speed, size of each grid square in pixels, world's width and height.** Starts the game adds the first snake and adds the food.

3) NatureSimulator: The classes in this package provide the game to work properly.

a) **Action.java** representing the possible actions for Snakes game. I nearly copied this class from Project 1. I delete stay and attack action. I add the eat action.

b) **Localinformation.java** representing the information a node has about its surroundings. Automatically created and passed by the game to each node at each timer tick method.

c) **SnakesGameSimulator.java** that implements the game logic for Snakes. Extends GridGame.

4) Project: This package contains the essential classes that creates the game's characters like food and snake.

a) **Node.java** is the primary class of the game. Game board consists of nodes so each square on the board is a node. Implements drawable.

b) **Food.java** extends Node and implements drawable.

c) **Snake.java** implements drawable. This class for the snake which is a LinkedList consists of nodes.

5) Ui: I do not make any changes in this package except changing the colors in the GridPanel.java

a) **ApplicationWindow.java**

b) **GridPanel.java:**

Problem Solution:

I handle the assignment by using nodes and LinkedList. Each nodes in the game has an x and y coordinate. I create node class for nodes. Since food class extends node class, food is a node which is **YELLOW. In snake class,** I create a linkedlist called snakeParts. In the game, snakeParts is a snake that consists of nodes. "addNode" method adds node to the snakeParts. Move method adds a new node to the first part of the snakeParts according to the direction and removes the tail(last node). So I take the node from the tail and add to the head.

In eat method, I just add a new node to the head of the snakeParts. In reproduce method that returns a snake, I create a new snake which consists of the parent snake's last 4 nodes and the last node of the parent snake is the head of the newborn snake. **The color of the snake's head is "CYAN.brighter()" and the color of the snake's body is "GREEN.brighter()".** ChooseAction method in the Snake class takes a LocalInformation and returns the available action. If the snake reaches the size of 8, it chooses to reproduce. If one the x or y coordinates of the head minus one of the x or y coordinates of food is equal to 1, snake chooses eat action towards the available direction. I also create a new ArrayList called sensible directions and a freeDirection that gets the free directions from the information to use them in choosing move action. **In the AI**, I added the directions that helps to abbreviate the distance between the head of the snake and the food, to the sensible directions. **But** if the directions are not free I do not add them to the sensible directions. It is not sensible to add the directions which are not free to the sensibleDirections because snake can only move to the null nodes. In the end, if the sensibleDirection list is not empty, that means there are some sensible directions around head to shorten the road to the food. I get a random direction from sensibleDirection list. If the sensibleDirection is empty, I get a random direction from the freeDirections.

In SnakesGameSimulator class, I initialize a new food, a new list consists of snakes and a map for keeping the node. "addSnake method" adds the first snake of the game. This method adds this snake to the snakes list, calls addDrawable method to draw this snake and calls updateNodesMap method to update the map. "addFood" method creates a new food, calls the addDrawable method to draw the food and determines the coordinates in the map. "removeNode" method removes the node. "createLocalInformationForNode" that takes a node as a parameter, creates a local information for node and keeps the information of node. "isPositionFree" controls whether the position is in the grid or not. "isDirectionFree" controls whether the direction is free or not. **"timerTick"** method is called by the gridGame in each iteration. It creates a new ArrayList called snakesCopy for determining and execute actions for all snakes. I add a "for loop" to loop for all snakes. If the chosen action by snake is move, it implements the move

method, updates the map and add the snake to the drawable. So the main idea is updating the map by using updateNodesMap after each movement in the game and drawing the last situation. If the chosen action is eat it follows the similar order and adds the food into the null node. If the chosen action is reproduce, it creates a newSnake adds this newSnake to the snakes list and continue by following the same order.

SERTAY AKPINAR
2016400075