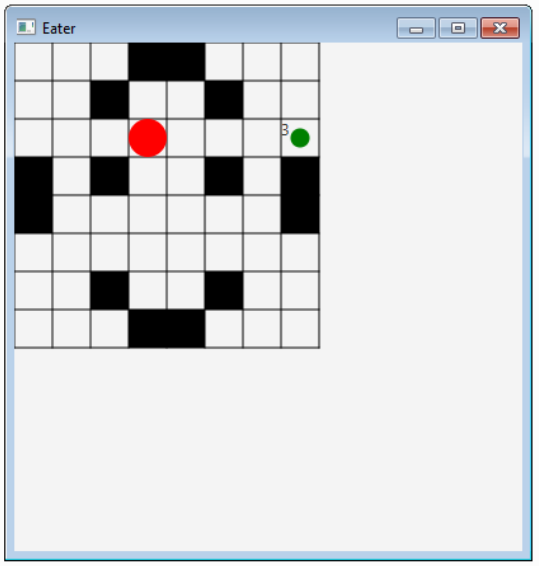
CSS-102: Programming Technologies (Java)
Final Project (Part-1 of 2)
Deadline: May 10 (Wednesday) 18:00
100 points (10% from Total)

Introduction

The purpose of this project is to provide you with the opportunity to review some of the ideas and techniques you should be familiar with from this course, including file I/O, manipulation of arrays, fundamental principles of object-oriented development, user-interface design and event-driven programming.

You should be able to complete the assignment within a week. Please start early so that you have time to get your questions answered and review any areas in which you feel weak. The second part is going to be the continuation of this assignment, the details of which will be posted soon, and you will have one more week to complete that one.

Project Requirements Overview



You will be implementing a simple game simulation (similar to "Packman") using JavaFX. The game is started with a command-line argument representing the file that describes a specific map for the game.
The basic options of the player are:
Move within the map (table) in four directions: right, left, up, down.
Collect the food that appears in some region of the given map.

The game finishes when there is no more food to eat, and displays the total points.

Program Design

Good object-oriented design is required for this assignment. For example, all mutable data should be encapsulated in private fields, and all the implementations must satisfy the requirements described below. If you need help or hints on how to design your program, you should ask in office, by email, on Edmodo, or on the preparation lessons that we will have next week. Reviewing and practicing good design is a primary goal of this assignment.

Documentation and Style

In this assignment you must meet the following requirements to receive documentation and style points:

1. All methods, classes, member variables, etc. must have comments to explain their purpose.
2. Methods may not be longer than 60 lines in order to receive full credit. If you are writing longer methods, consider how you can break the task into smaller pieces and write them separately.

Design Requirements

Certain specific design requirements that you must follow are described below.

1. *Position class*
   - Stores the position (x and y coordinates).

2. *Player class*
   - Interface for the Player, with basic commands.

3. *MyPlayer class*
   - Implements the Player interface;
   - Creates the player as a ball on the board;
   - Moves the ball;
   - Makes sure it does not go out of bounds or through the walls.

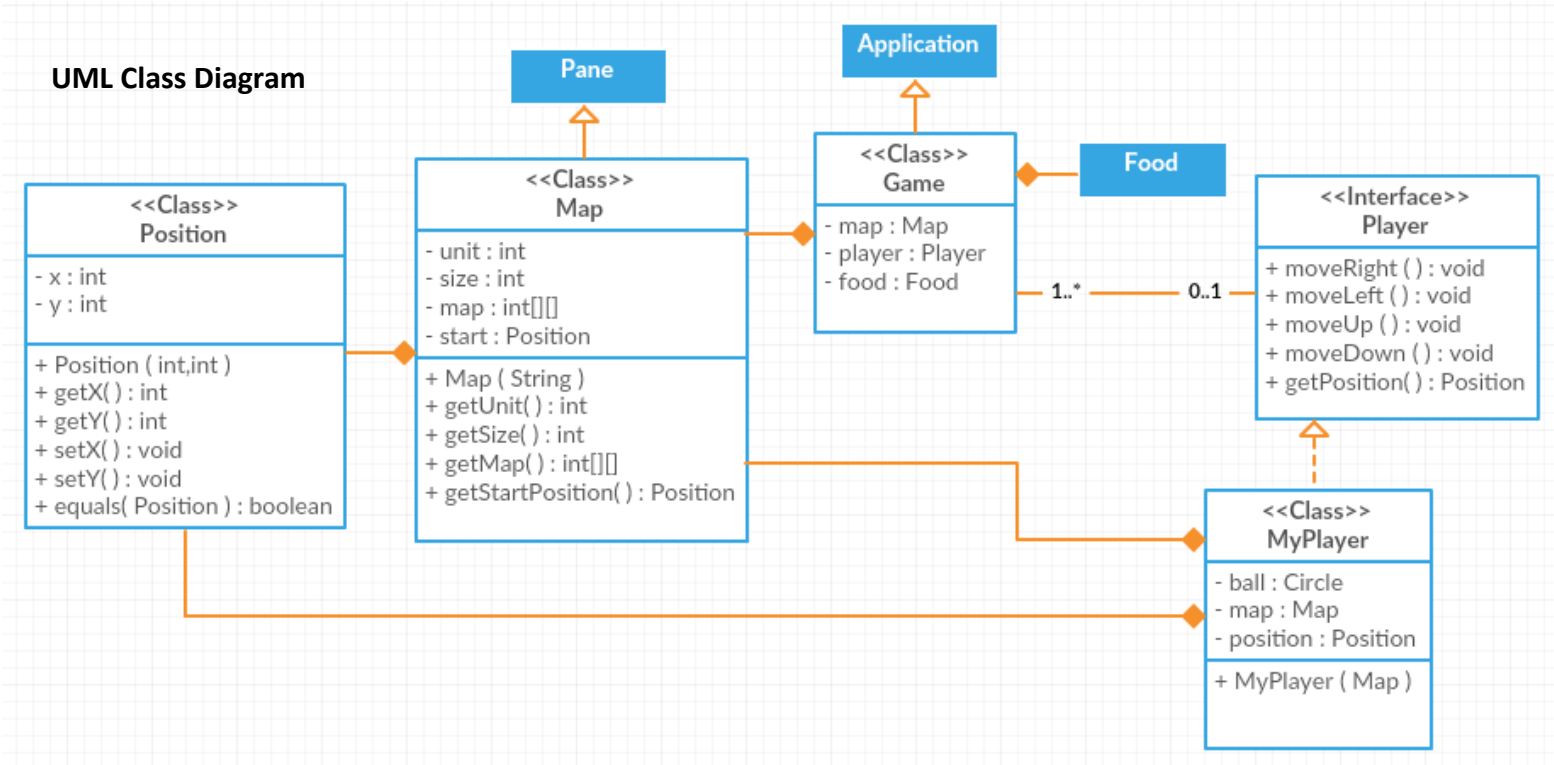
4. *Map class*
   - Extends the Pane class;
   - Constructs a map from a given text file;
   - **`unit`**: size of one cell (in pixels)
   - **`size`**: size of map (number of columns/rows)
   - Keeps the data in a two-dimensional array;
   - Fills the map;
   - Draws the border lines;
   - Draws the walls;
   - Provides the starting point for the player (ball).

5. *Game class*
   - JavaFX application;
   - Creates the map;
   - Creates the player;
   - Creates food instance;
   - Controls key events;
   - The whole game starts as follows:
     `> java Game map.txt`

6. *Food class*
   - The class file is already provided for you (you don't need to implement this);
   - Responsible for creating food on the board;
   - The constructor signature:
     `Food(Map,Player)`.

7. *Map text file*
   - The sample txt file is attached;
   - 0 - for empty cell, 1 - for wall, and 2 - for the starting point of the player;
   - You can try your own maps as well.

UML Class Diagram



You must meet all the design criteria described above. All the classes, variables, and methods must have exactly same names and types. You may add extra fields and helper methods as well.

Notes on Grading

Make sure that every class does exactly what is described in the project requirements. Do not attempt to show some ready Packman implementation downloaded from Internet. It shall be considered as plagiarism, and get ZERO points. A fully tested, compiling, and well-documented class will be worth some points even if you do not finish the whole assignment.

Any kind of extra improvements on the game mechanism and user interface design (Ex: adding animations) are welcome, and may be considered for extra bonus points.

Do NOT share your code with anyone. It would be considered academic dishonesty, and strictly penalized. All the works submitted shall be inspected by a special program and reviewed by the instructors. Any kind of similarity, or not being able to answer questions on the project gives the instructor full right to penalize the work, and even cancel the results that have already been graded because of cheating issues. In simple words, the fact that you did not cheat yourself, or that you showed your work and it has already been graded, does not help you. All the sides included in cheating (which is a crime) will be penalized. Again, do not share your code under any conditions.

Although you are strongly encouraged to ask questions and have discussions on Edmodo, make sure you do not share your code there as well, please.

Turning In

1) Collect all your work (7 files in total) into one **zip** file, and name it in the following format: *NameSurname.zip*;
2) Turn in via **Edmodo** (unless told otherwise by your lab instructor);
3) See your lab instructor to demonstrate your work, answer some questions, and get it graded.
   Your lab instructor may have a special schedule for project submission.

Works that are not turned in will not be accepted. Submitted but not demonstrated works will not be graded.