CS-420-1: Object-Oriented Design
Fall 2019
Northeastern Illinois University
Homework #5: Due Thursday, 10/17/19 at 5:00 p.m.
Enumerated Types and Nested Classes

Coding - Getting Started

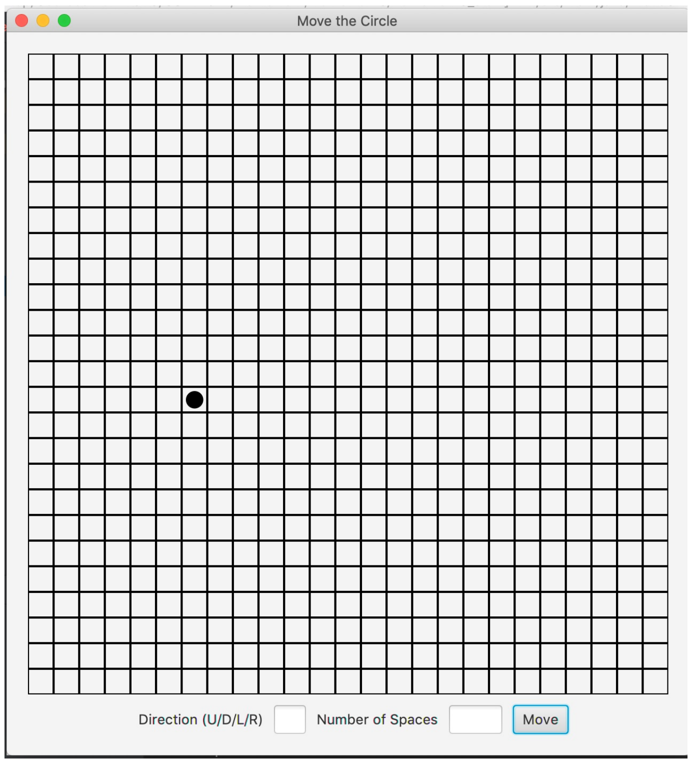
- Download the files provided for you from the course website and unzip them into a folder.
- Rename the folder to `homework5`.
- With IntelliJ (v2019.2.3RC or higher), choose `Import Project` and navigate to the `homework5` folder.
- Choose `Import model from external model` and select Gradle and click `Finish`.
- Go to `Preferences` → `Build, Execution and Deployment` → `Build Tools` → `Gradle`.
- Make sure that the checkbox for `Automatically import this project on changes in build script files` is checked and that both the `Build and run using:` and `Run tests using:` drop downs are set to Gradle. Change the `Use Gradle from 'gradle-wrapper.properties' file` to `'wrapper' task in Gradle build script`. Make sure that the Gradle JVM is set to Java 11!
- Also go to `File` → `Project Structure/Project Settings` → `Project` and make sure that the JDK is set to Java 11.
- Remember that to run the project for the first time, you will need to click on the Gradle tab on the right-hand side of the project and go to `homework5` → `Tasks` → `application`. Right-click on `run` and choose `homework5 [run]`. You can run the project at this point - it will only show the text fields and labels, and the button.

The Problem:

You will be guided through building a JavaFX application (with 5 classes, of which you will create 3!) that simulates moving a ball on a grid (think of this like a little robot, see image below). The grid is a 25 x 25 grid where the top-left-hand corner of the grid is at (0, 0) and the bottom left hand corner is (24, 24). The ball should be initialized to a random location in the grid, and if the user enters an invalid direction (other than U/D/L/R) or an invalid number of spaces (either an invalid value or a value that takes the ball off the grid), the ball will not move from its current location.

Provided Classes

- `MoveCircleDemo.java`: The main application class (also set in the Gradle build file). This contains the initial stage and scene set-up and also launches the application. While most of the code in this class has been created for you, you will be adding some material to this class.
- `InputHBox.java`: This is the horizontal box that includes the text fields and the `Move` button. While most of this class has been provided for you, some functionality will need to be added.



Grid.java:

Create a class named `Grid.java` that inherits from `GridPane` with the following:

1. A public member inner class named `Cell` that inherits from `StackPane` with the following:
   - A private final integer instance variable named `size`.
   - A constructor that takes an integer as a parameter and sets the `size` instance variable. It should also set the border color to be black and the width and height to be `size x size`. Note: Remember that this inherits from `StackPane`, so you have access to **all** the `StackPane` methods - look up how to do these things! And remember to look at the methods that `StackPane` inherits from its parent class!
2. A private 2D array instance variable named `cells` of type `Cell`.
3. A constructor that takes two integer parameters named `n` and `dimension`. It should create the `n x n` `cells` array. It should iterate over the `cells` array and for each element `(i, j)`, it should create a new `Cell` object of size `dimension` and add it to the `Grid` at location `(i, j)`. Remember that `Grid` inherits from `GridPane` and you have access to all the methods from `GridPane`.
4. Create a method named `getCell` that takes an integer representing the row and an integer representing the column an returns the `Cell` at the location in `cells`.
5. Create a method named `getSize` that returns the number of rows in the `cells` array.
6. In the `MoveCircleDemo` class, declare a private instance variable of type `Grid` named `grid`. In the `start` method (at the beginning), create the `grid` and pass in `N` and `DIM` for the parameters.

7. In the `setupBorderPane` method, uncomment both the commented lines and run the application. You should see a grid similar to the one shown above.

CircleMarker.java:

Create a class named `CircleMarker.java` that inherits from `Circle` with the following:

1. A private 1D array integer instance variable named `coordinates`.
2. A constructor that takes an integer parameter named `size`. It should set the radius to be $\sqrt{size^2}/\pi$ (take a look at the `Circle` class constructors - remember how subclasses access superclass constructors). It should also set the border to be black and the fill to be black. Finally, it should create the `coordinates` array to be of size 2 with values of zero.
3. Create a getter for `coordinates`.
4. In the `MoveCircleDemo` class, declare a private instance variable of type `CircleMarker` named `marker`.
5. In the `start` method (under the `grid` creation), create a new `CircleMarker` object and pass in `DIM` for the parameter. Run the application - you won't see a circle yet - it has not yet been added to the grid, but make sure your code does not blow up.
6. In the `CircleMarker` class, create a `void` method named `drawCircle` that takes a `Grid` and a 1D integer array (named `coords` as parameters. The method should get the `Cell` for the current coordinates (x is in element `0`, y is in element 1) and add the `CircleMarker` to that `Cell`. Remember that `this` refers to the current `CircleMarker` object that is calling the method. Then set the alignment of the `CircleMarker` so that it is centered in the `Cell`. Finally, update the current `coordinates` of the `CircleMarker` to be the values of `coords`.
7. In the `start` method of the `MoveCircleDemo` class, randomly generate two integer values between `0` and `25` (exclusive) and then call the `drawCircle` method and pass in `grid` and an array with the two randomly generated integers. Run the application and make sure that you see a black circle randomly added to the grid.
8. Refactor/extract the three lines for the creation of the marker, randomly generated the integers, and drawing the circle into a private method (use IntelliJ!).

Direction.java:

Create an enumerated type named `Direction.java` with the following:

1. Four constants named `UP, DOWN, LEFT` and `RIGHT`. They should each take a `char` as a parameter - this `char` should be the first uppercase character of their name.
2. A private final integer instance variable named `val`.
3. A constructor that sets the instance variable.
4. A abstract method named `updateCoordinates` that takes a 1D integer array named `coords` and an integer named `spaces` and returns a new 1D integer array.

5. Implement the abstract method for each constant such that it returns a new 1D integer array:

   - For `UP`, the x-value stays the same and changes the y-value by spaces (to move the marker up).
   - For `DOWN`, the x-value stays the same and changes the y-value by spaces (to move the marker down).
   - For `LEFT`, it changes the x-value by spaces (to move the marker left) and the y-value stays the same.
   - For `RIGHT`, it changes the x-value by spaces (to move the marker right) and the y-value stays the same.

6. Create a `static` method named `getNewCoordinatesForDirection` that takes a character representing the direction, a 1D integer array named `coords` and an integer named `spaces` and returns a new 1D integer array. The method should determine which enumerated type corresponds to the character parameter and call the resulting `updateCoordinates` method for that type. If the character does not match any of the enumerated types, then throw an `IllegalArgumentException` with an appropriate error message.

Getting the ball "moving":

1. In the `CircleMarker` class, create a method named `updateLocation` that takes a `Grid` type and a 1D integer array. The method should do the following:
   - It should do determine if the coordinates in the 1D array are valid coordinates for the grid.
   - If the coordinates are valid, first remove the marker from its current location in the cell in the grid (you can figure out which cell to remove it from by using the current coordinates - this is one of the many reasons we stored them). Then draw the circle using the new coordinates.
   - If the coordinates are not valid, then do nothing (i.e. the marker should not move).
2. In the `InputHBox` class, create a void method named `addMoveButtonAction` that takes a `Grid` and a `CircleMarker` as parameters. In the method do, the following:
   - Call the `setOnAction` method for the button and implement an anonymous class that implements `EventHandler<ActionEvent>`.
   - The anonymous class should do the following:
     - Get the first character from the `directionText` box.
     - Get the number of spaces from the `spacesText` box and convert to an integer.
     - Use the `getNewCoordinatesForDirection` to get the new coordinates
     - If the direction is invalid (not U/D/L/R), handle the error by clearing the text fields and do not move the marker.
     - Assuming the direction is valid, call the `updateLocation` method and pass in the appropriate values.
     - The text fields should be cleared after any move.

3. Call the `addMoveButtonAction` method on the `InputHBox` reference variable in the `start` method in the `MoveCircleDemo` class.

4. Run your code and test the application. **At no point should the application blow up, even with bad/invalid text field input.**

Submitting your homework to D2L

- In IntelliJ, under `File`, choose "`Export to Zip File`".
  Use the name of `homework5.zip`.

- Submit this file to D2L.