Programming on Purpose (CSC 120)
Winter 2024

Project 4 – Igel Ärgern (now in Java)

In this project, you will re-implement "Igel Ärgern" in Java.

1 Objectives

- Learn the Java syntax equivalents to the Python constructs you already know.
- Get more practice with OOP.
- Practice applying good software development practices such as iterative development, testing and debugging, turning pseudo-code algorithms into code.

2 Due Dates

Milestone 1: Wednesday, 2/14 Tests for the `Board` class

Milestone 2: Monday, 2/19 `Board` class implementation

Milestone 3: Wednesday, 2/21 Pseudo-code for the main-phase

Milestone 4: Friday, 2/23 Implementation of the forward move in the main phase

Milestone 5: Monday, 2/26 Implementation of the sidways move

Milestone 6: Thursday, 2/29 Full game

3 Your Mission

This implementation should have the same functionality as the version you implemented in Project 2. That is, it implements the full game including traps and it supports players by validating their input and giving them an opportunity to provide a new input in case their first input was invalid.

In addition, you should use this project to gain some experience with iterative development. To help you with this goal, the project is divided into several milestones. At each milestone, you will submit a deliverable and you will exchange a (code) review with a partner. (I will make more information about the logistics of the code reviews available around the time of each milestone.)

4 Milestone 1: Tests for the `Board` class

Find the starter code for Milestones 1 and 2 on Nexus and have a look at it. It provides stubs for a set of methods in the `Board` and `Cell` classes. Your implementation is required to use these two classes and the given methods. Do not change the signatures (i.e., name, return type, type and number of parameters) of these methods. (You will be allowed to add additional methods if you need them.)

Also note that all of the `Board` and `Cell` classes are in a package called `igel`. Please do not change that package structure.

The starter code also provides a class with testing utility methods, `Testing`. `CellTester.java` gives you an example of how to use these testing utility methods.

In this project, you will practice using **test driven development**, which means that you will write tests for the `Board` and `Cell` classes before you implement these classes.

Your deliverables for Milestone 1 are test suites for the `Board` and `Cell` classes. Complete the `CellTester` and also add a class called `BoardTester`. **Upload both classes to Gradescope by Wednesday, 2/14.**

A note on grading: Since you haven't implemented the `Board` and `Cell` classes, yet, you cannot actually test your tests all that well. But you can and should use the method stubs in the starter code to make sure that they do not produce exceptions. Grading will focus on the coverage of your test suites.

5 Milestone 2: `Board` class implementation

Your deliverables for Milestone 2 are thoroughly tested implementations of the `Board` and `Cell` classes as well as revised versions of your test suites for these classes.

Upload `Board.java`, `Cell.java`, `BoardTester.java`, and `CellTester.java` to Gradescope by Monday, 2/19.

6 Milestone 3: Pseudo-code for the main-phase

Now that you have a functioning implementation of an *Igel Ärgern* board, you have to figure out how to use it in an implementation of the game logic. That means, you need to write some (English) pseudo-code that describes the algorithms your implementation is going to use and how and when methods from the `Board` class are going to be used.

Your deliverable for Milestone 3 is pseudo-code for the game's main phase. Upload it as a pdf document to Gradescope by Wednesday, 2/21.

Note: the starter code for Milestone 4 will provide an implementation of the start-up phase. That's why you only have to deliver pseudo-code for the main-phase of the game.

7 Milestone 4: Implementation of the forward move in the main phase

There is additional starter code for this milestone:

- `IgelAergern` Represents the game logic.
- `IgelView` A Java implementation of a terminal based interface. Do not modify this class. As before, all interaction with the user has to be done through an `IgelView` object. Have a look at the Javadocs for the public methods. They are slightly different than in the Python implementation.

Your deliverable for Milestone 3 is a partial implementation of the game's main phase. You can ignore the sideways move for this implementation. So, in each player's turn, the die gets rolled and the player moves one token forward (in the row indicated by the die roll). When three tokens of one player reach the goal column, that player wins.

Use your pseudo code to help guide your implementation. You may want to introduce additional classes.

Compress your entire project into a .zip file and upload it to Gradescope by Friday, 2/23.

8 Milestone 5: Implementation of the sidways move

Add in an implementation of the sideways move. So, in each player's turn, the die gets rolled, the player gets to choose whether they want to move a token sideways, and then player moves one token forward (in the row indicated by the die roll). When three tokens of one player reach the goal column, that player wins.

Again, use your pseudo code to help guide your implementation.

Compress your entire project into a .zip file and upload it to Gradescope by Monday, 2/26.

9 Milestone 6: Full game

Test and debug (if necessary), and refactor your implementation.

Remember: *Just because it works doesn't mean it's good.*

- Make sure that all classes and public methods are documented.
- Re-read your code and check for ways in which readability could be improved (e.g., good variable/function names, no magic numbers, consistent formatting, code is organized into logical units, each method serves one main purpose).
- Don't forget to cite who you received help from and include the honor code affirmation at the top of each Java file you have written or edited:

```
I affirm that I have carried out my academic endeavors
with full academic honesty. [Your Name]
```

Compress your entire project into a .zip file and upload it to Gradescope by Thursday, 2/29.