CS62 Lab2: Silver Dollar

Basic Data Structures

3.10 Laboratory: The Silver Dollar Game

Objective. To implement a simple game using `Vectors` or arrays.

Discussion. The Silver Dollar Game is played between two players. An arbitrarily long strip of paper is marked off into squares:

The game begins by placing silver dollars in a few of the squares. Each square holds at most one coin. Interesting games begin with some pairs of coins separated by one or more empty squares.

The goal is to move all the n coins to the leftmost n squares of the paper. This is accomplished by players alternately moving a single coin, constrained by the following rules:

1. Coins move only to the left.
2. No coin may pass another.
3. No square may hold more than one coin.

https://www.cs.williams.edu/~bailey/JavaStructures/Book_files/JavaStructures.pdf

Lab 2 agenda

- Quiz
- Lab
- A bit about HW3

We’re using our first data structure!

- An ArrayList is like an Array, but more flexible. It can change sizes, so more like a Python list. To access it, you import `java.util.ArrayList`.
- It’s good to learn how to read Java documentation: https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html
  - Specifically, we use the .add(), .get(). and .size() methods. Figure out what parameters they take (if any) and what they return through the documentation.
- We’ll learn more about the implementation of an ArrayList in class tomorrow

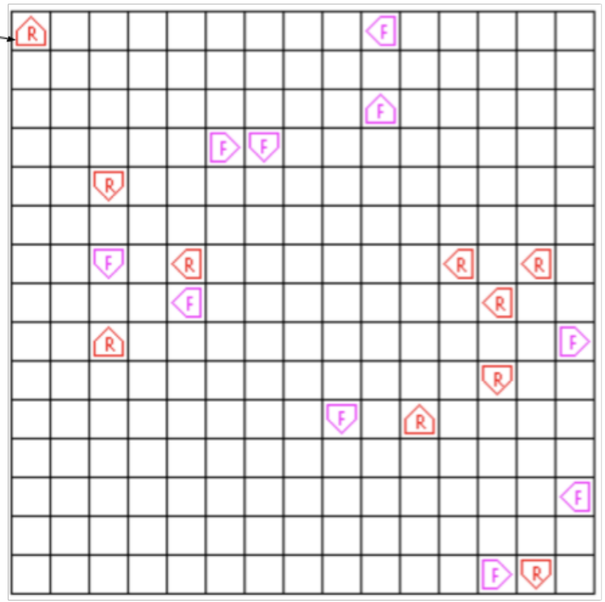
Lab structure

- TextCoinStrip.java - this is the only file that you're required to be concerned with, you have to fill in 4 methods to complete the lab
- Optionally, get practice using the Java GUI and creating graphics - implement code in 4 places in GraphicsCoinStrip.java
  - It's an ArrayList of CoinSquares, instead of an ArrayList of booleans. CoinSquares extend Rectangle2D, so you can use Rectangle2D built in methods (like .contains()).
- We'll be checking that you commit regularly for style points for this lab - try to commit after finishing each method

HW3: Darwin

Each of these is a **Creature** object which has a field of type **Species** which determines its color, letter and behavior

The **WorldMap** is this graphical representation of a **World** object. The world is based on a grid formed by the **Matrix** object, which is built by nested ArrayLists



The **Darwin** class object controls and combines all of these objects and runs the program through steps to simulate the game

- 2 week pair programming assignment
- The biggest you've seen yet - 8 different interacting Java Classes
- Simulating a self-running video game (a digital world of creatures who try to infect other creatures)
- Matrix is represented with nested ArrayLists
- Create your own creature in a class wide competition!

HW3: Darwin

- Notes from your TAs
  - Start early
  - Before you code, read through all the .java files and take notes on all the available methods, so you don’t end up writing redundant code
    - Portion out 1 hour just to read and understand the assignment
  - Just because your code runs doesn’t mean it’s correct (it should follow the rules of the simulation)
  - It might take mentors ~30 min to understand your code, so have patience

Reading data with a buffered reader

- `import java.io.FileReader;`
- `import java.io.BufferedReader;`

```
FileReader fr = new FileReader(“fileToRead.txt”);

BufferedReader br = new BufferedReader(fr);

String line = br.readLine();

while ((line!= null) {

    //do something

    line = br.readLine();

}
```

a BufferedReader object takes a FileReader object as input.

the .readLine() method will return null when the file has no more lines to read, so we can write a while loop

You’ll see this in Darwin