CSE 8B discussion: week 4

HW3, enum types, testing, HW4
Agenda

1. HW3 recap
2. Enum types
3. Testing
4. HW4 overview
Enum types

- Kind of like declaring your own variable type
- Useful for when you are describing a type that can only take values from a short *enumerated* list
  - e.g., Direction (NORTH, EAST, ...), Coin (QUARTER, DIME, ...), Planet (MERCURY, VENUS, ...)
- They can have fields and methods, just like classes
- Refer to them like Direction.NORTH, or just NORTH inside of switches

[https://docs.oracle.com/javase/tutorial/java/javaOO/enum.html](https://docs.oracle.com/javase/tutorial/java/javaOO/enum.html)
HW3 recap

- Seems to have been less challenging, based on Piazza activity
- How did you solve it?
- [my solution]
Testing, or how to write code you can believe in

- **Scenario:** You write some code, test it manually, make some more changes, test it manually... before long, you realize that manual testing is lame

- **Scenario:** You write some code, it seems to work. Write some more, but you’ve changed some stuff around and you could’ve broken something. Manual testing takes time that you don’t have.

- A complex program has a lot of possible situations, and you want to make sure that it handles them all correctly

- **Solution:** Automated testing for the correct behavior in your code

- Companies live and die by the technical debt they do/don’t accumulate, and by the agility of their development processes
Testing: JUnit

- The de facto tool used for Java **unit testing**
  - definition: testing the smallest pieces of your code (methods) individually; contrasts with **integration testing**
- "assert" == specify something that should be true; else consider the test to have failed
- Many test frameworks for other languages are similarly designed to JUnit
- How does one use JUnit on ieng6?

http://junit.org/
Agenda

1. HW3-recap
2. Enum-types
3. Testing
4. HW4 overview
Review: homework tips

- **Start early** -- you never know how long/short things like this will take
- Read the assignment **thoroughly**
- Complete the assignment in the order of the directions
  - Often concepts build on each other throughout the assignment
- Working with unfamiliar classes? Read their docs, know what methods are available to you
- Google often (StackOverflow is your friend). ex.: “java check if two arrays are equal”
- Discuss approaches to the problem with your classmates
- Read all the questions on Piazza
- **Start early**
HW4 overview

- Problem: Implement the game 2048.
  - [https://gabrielecirulli.github.io/2048/](https://gabrielecirulli.github.io/2048/)
  - [http://doge2048.com/](http://doge2048.com/)
  - [https://ov3y.github.io/2048-AI/](https://ov3y.github.io/2048-AI/)

- Gives you practice with breaking down a problem into its parts
- Should the board be a class? Should a tile?
  - What methods should classes have? (tell, don’t ask)
  - What will I be telling the objects to do?
- Write it down. Try and get a high-level picture of how your program will flow.
- Make the design, think it over, and then write the code
- The assignment contains a sample design (the UML diagram); modify it as you see fit
Agenda

1. HW3-recap
2. Enum-types
3. Testing
4. HW4-overview
Closing

- Questions?