CSSE 220

Console Input

Check out ConsoleAndUnitTestingPractice.

Outline

- Console Input
- Unit Testing
- Prep for Exam 1 paper part
- Maybe some time to work on Scene

Reading keyboard input from the console

CONSOLE INPUT WITH JAVA.UTIL.SCANNER

Console input with Scanner

- Creating a Scanner object
  - import java.util.Scanner;
  - Scanner inputScanner = new Scanner(System.in);
- Defines methods to read from keyboard
  - inputScanner.nextInt();
  - inputScanner.nextDouble();
  - inputScanner.nextLine();
  - inputScanner.next();
- Exercise: Look at UnitTesting/src/ConsoleWorker.java.
  Add missing methods to read from console

Q1

Unit Testing

- Idea: Test “small pieces” of larger program
  - Do the expected values match what you ACTUALLY get?
- How to test in this manner?
  - Could make a main method that calls all the methods
  - JUnit!
    - Creating a Tester JUnit class

Q2

Why Unit Testing?

- There are several goals of unit testing:
  - Make sure your code works (as specified!)
  - Keep it working
  - Confirm understanding of the specification
  - Confirm pieces of code in isolation
  - Provide Documentation

Q3

Unit Tests (as done in CSSE120)

1. Construct one or more objects of the class that is being tested
2. Invoke one or more methods
3. Print out one or more results
4. Print the expected results
5. Do 3 and 4 match?

(Pages 102-103 in book)

Why JUnit?

- Provides a Framework
  - Framework: Collection of classes to be used by another program
- Provides easy-to-read output in Eclipse
- Prints require you to analyze all lines
  - What if it scrolls off the page?
  - What if it's only 1 character different?

What are good unit tests?

- Unit tests should be small pieces that test:
  1. The most common cases
  2. The edge cases (minimum, maximum, switching from positive to negative, etc.)
  3. All specific/special cases (e.g., when 0 or null the behavior is different than for any other value)
  4. When you find and fix a bug, you should have a unit test for this so it doesn't ever happen again. Fix things once and for all!
  5. Any overly complex code that 1-4 above don't cover

Q4

Unit Testing

- Use “assert” to make sure results match
- Let’s look at BadFrac.java and BadFracTest.java
  - Let’s make some unit tests and figure out why this project has been yielding some strange results

Review for written portion of Exam 1

EXAM 1 REVIEW - WRITTEN