CSSE 220 Day 5

Console Input, equality
Unit Testing

Check out *InputAndUnitTests* from SVN

Questions?

Outline

- String Input and Output
- `==` vs. `equals()`
- Unit Testing

Multiple ways to read console input

- Using System.console()
  - Creates a Java console ***outside*** an IDE (e.g. Eclipse)
  - Used in a command line environment (e.g. LINUX)
  - Reading: readLine(), readPassword()
  - We will not use this approach in class

- InputStremReader wrapped in BufferedReader
  - InputStreaReader reads bytes from an inputStream and converts them to chars
  - BufferedReader reads text from a char-input stream
  - Reading: read(), readLine()
  - We will not use this approach in class

Read Console Input with `java.util.Scanner`

- Creating a Scanner object:

  -
    ```
    Scanner inputScanner =
                new Scanner(System.in);
    ```
- Defines methods to read from keyboard:
  - `inputScanner.nextInt()`
  - `inputScanner.nextDouble()`
  - `inputScanner.nextLine()`
  - `inputScanner.next()`
- Exercise: Look at ScannerExample.java
  - Complete the TODO items

Formatting with printf and format

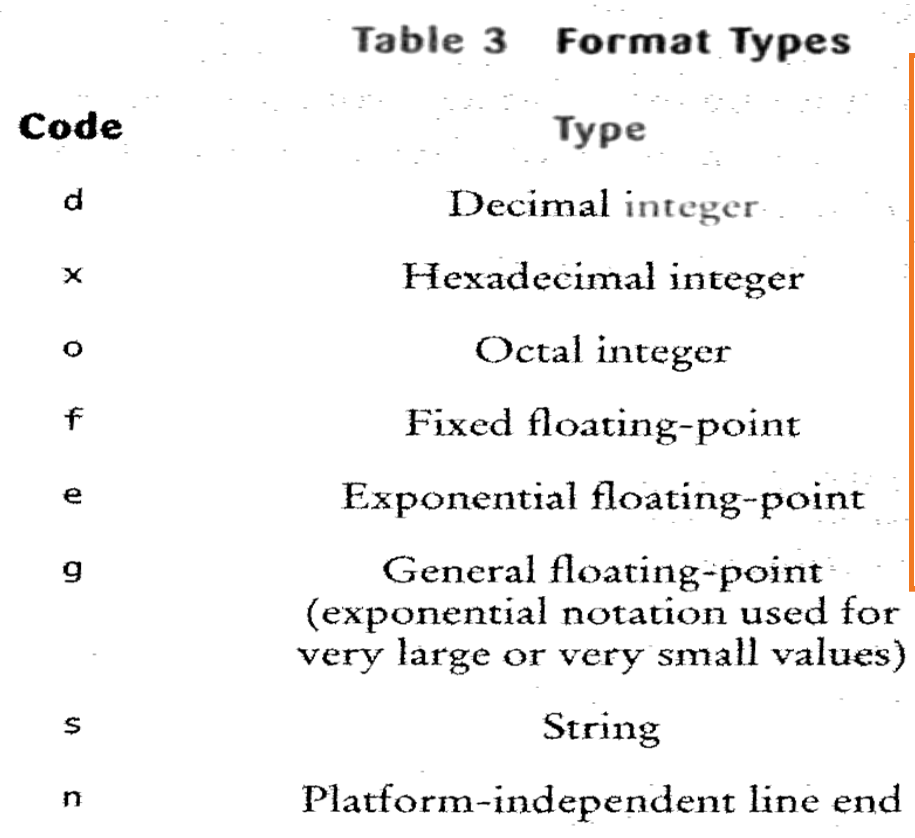


Table 3 Format Types

Code	Type
d	Decimal integer
x	Hexadecimal integer
o	Octal integer
f	Fixed floating-point
e	Exponential floating-point
g	General floating-point (exponential notation used for very large or very small values)
s	String
n	Platform-independent line end

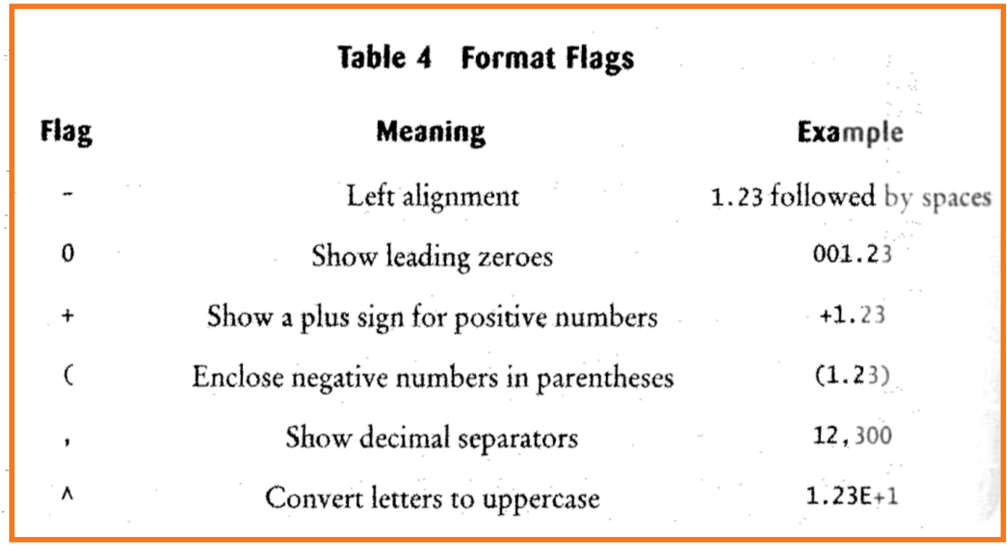


Table 4 Format Flags

Flag	Meaning	Example
-	Left alignment	1.23 followed by spaces
0	Show leading zeroes	001.23
+	Show a plus sign for positive numbers	+1.23
(	Enclose negative numbers in parentheses	(1.23)
,	Show decimal separators	12,300
^	Convert letters to uppercase	1.23E+1

We used a couple in recent examples.
Can you find them?

Tables from Horstmann, Big Java (3e), John Wiley & Sons, Copyright 2007

Q1 – Q2

Formatting with `printf` and `format`

- Printing:
  - `System.out.printf("%5.2f%n", Math.PI);`
- Formatting strings:

  -
    ```
    String message =
              String.format("%5.2f%n", Math.PI);
    ```
  - `System.out.println(message);`

Comparing Objects

- Exercise: EmailValidator
  - Use a **Scanner** object
  - Prompt for user's email address
  - Prompt for it again
  - Compare the two entries and report whether or not they match

- Notice anything strange?

Comparing Objects

The *equals* method is intended to dig inside objects and compare their data in a "sensible" way.

- In Java:
  - `o1 == o2` compares *values*
    - It evaluates to *true* only if their *bits* are the same
      - So for variables of class type, which store *references*, they are == only if they refer to the ***same object*** (same place in memory)
  - There is an `equals` method defined in the `Object` class, that all objects inherit.
    - It behaves the same as == does.
    - But subclasses can, and often do, override the equals method to give their own semantics to "equality", using their internal state (their fields). For example:
      - For Strings: s1.equals(s2) iff their characters are all ==.
      - `new Integer("0").equals(new Integer("-0"))`

How should you compare the email addresses in the exercise?

Work Time

- Correct the code in *EqualsPractice.java* and answer the associated quiz questions.

Q3 – Q4

Test Coverage

- *Black box testing*: testing without regard to internal structure of program
  - For example, user testing
- *White box testing*: writing tests based on knowledge of how code is implemented
  - For example, unit testing
- *Test coverage*: the percentage of the source code executed by all the tests taken together
  - Want high test coverage
  - Low test coverage can happen when we miss branches of switch or if statements

Q5

Unit Testing

- A type of white box testing

- Using code that you write to test whether the code that you care about is fit for use
  - Focused on testing individual pieces of code (units) in isolation
    - Individual methods
    - Individual classes

- Why would software engineers do unit testing?

Q6

Unit Testing With JUnit

- JUnit is a unit testing *framework*
  - A *framework* is a collection of classes to be used in another program.
  - Does much of the work for us!
- JUnit was written by
  - Erich Gamma
  - Kent Beck
- Open-source software
- Now used by **millions** of Java developers

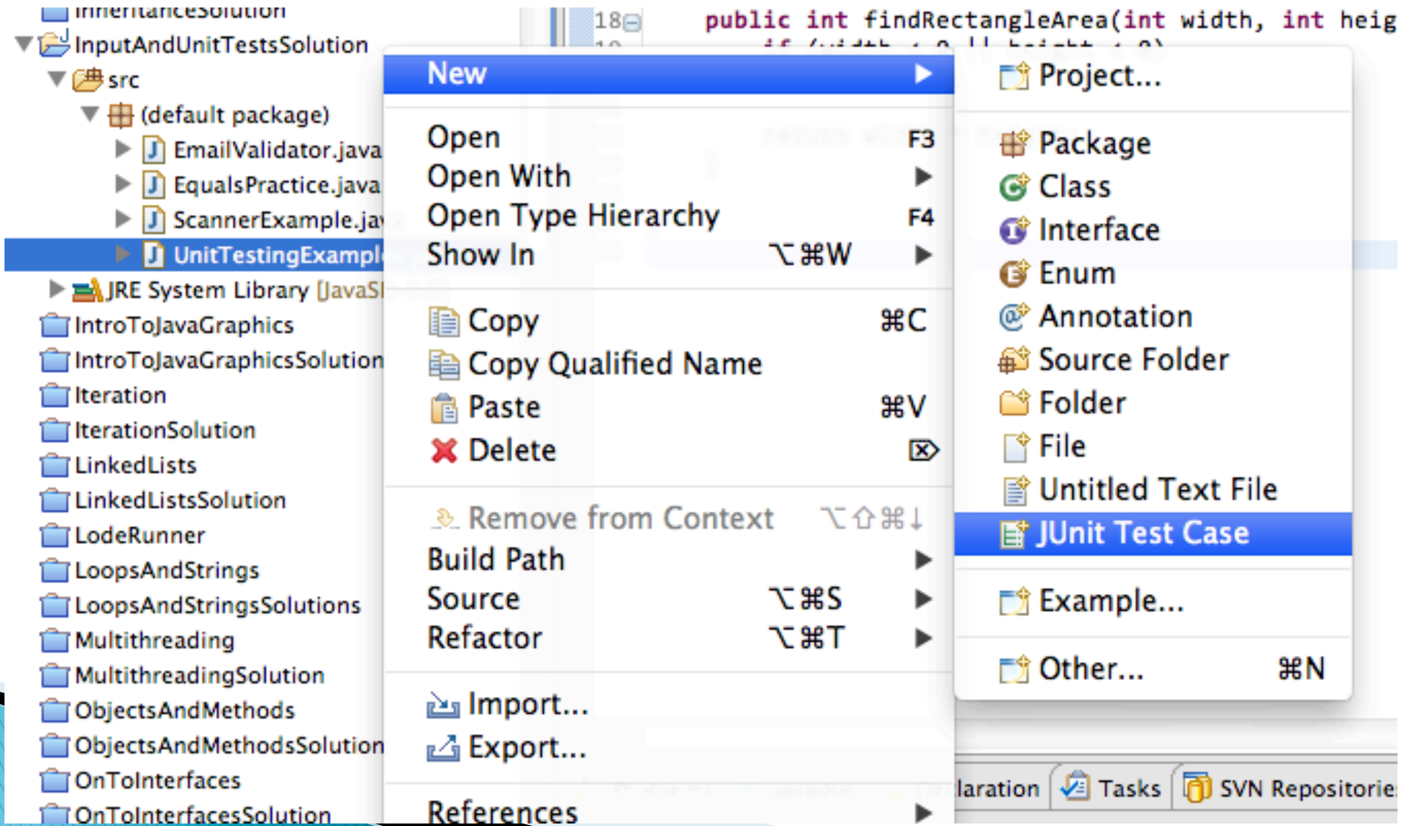
Q7

JUnit Example

- `BankAccountTester` in Big Java shows how to write tests in plain Java (pg. 103)
- Look at `UnitTestingExamples` in today's repository
  - We will write some test cases
  - Let's look at the comments and code together…

Using Junit in Eclipse

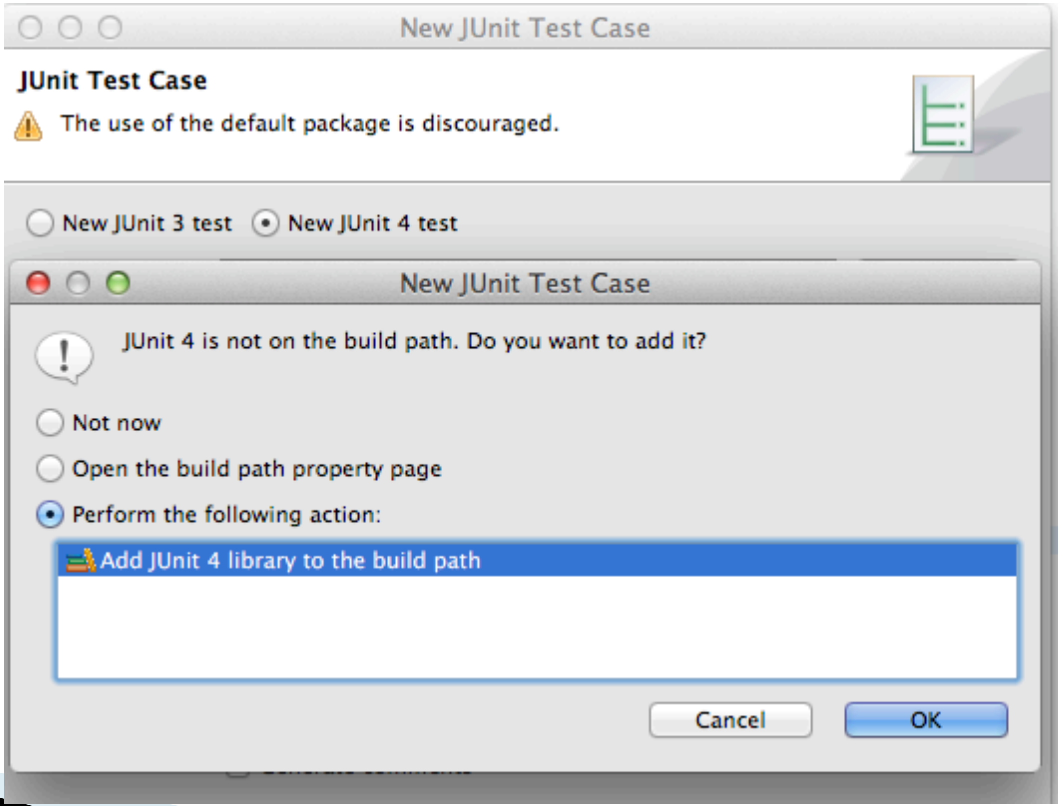
- Identify java file with code you wish to test
- Right-click on the file and create new JUnit Test Case



```
public int findRectangleArea(int width, int heig
```

Using Junit in Eclipse (2)

- Accept the suggested name for the JUnit file
- Select New JUnit 4 test andclick Finish


New JUnit Test Case

JUnit Test Case

The use of the default package is discouraged.

New JUnit 3 test [✓ New JUnit 4 test]

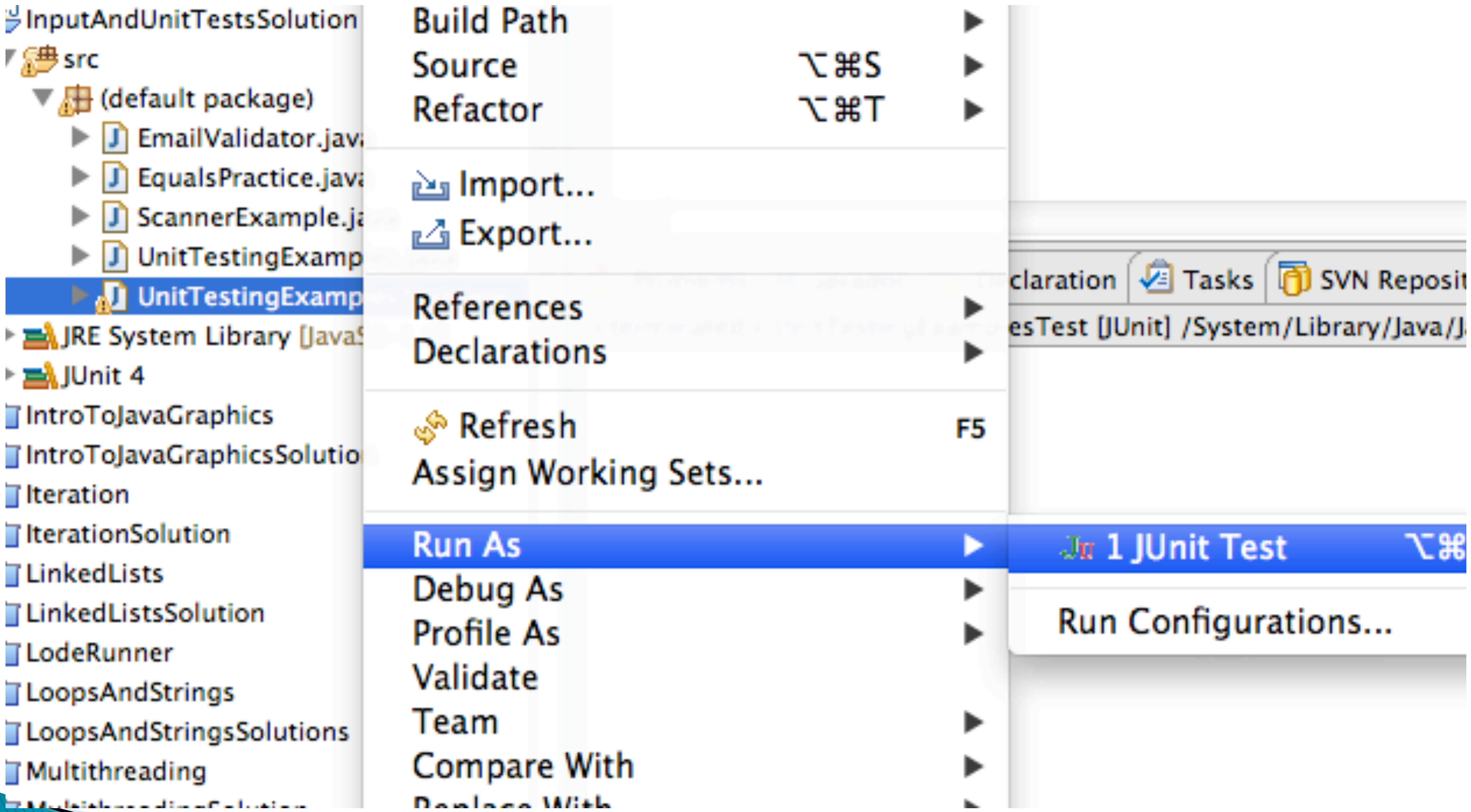
New JUnit Test Case

JUnit 4 is not on the build path. Do you want to add it?

- Not now
- Open the build path property page
- [✓ Perform the following action:]

Add JUnit 4 library to the build path

Running JUnit test cases



Q8

Tests passing or failing

- Green bar means the all the test pass
  - Tests that pass will have a green icon with a tick
  - May have to expand the items below the bar

- Maroon bar means at least one test fails
  - Tests that fail have a blue or red icon with an X.
  - The Failure Trace in the left hand pane describes the AssertionError.
  - Double clicking the AssertionError will identify in the code which test fails.
  - Study the error and fix code to eliminate the error.

Interesting Tests

Important Slide: Use this as a reference!

- Test “boundary conditions”
  - Intersection points: −40℃ == −40℉
  - Zero values: 0℃ == 32℉
  - Empty strings
- Test known values: 100℃ == 212℉
  - But not too many
- Tests things that might go wrong
  - Unexpected user input: “zero” when 0 is expected
- Vary things that are “important” to the code
  - String length if method depends on it
  - String case if method manipulates that

Work Time

- Hand in quiz.
  Complete the unit tests for
  UnitTestingExamples

Q9 – Q10