CSSE 220 Day 7

Fundamental Data Types, Constants, Console Input, More Text Formatting

Check out *FundamentalDataTypes* from SVN

Questions?

Today's class

- Quiz questions 1-3 review choosing fields for a class

- The rest of class is review of fundamental data types:
  - Work through the slides, quiz, and related exercises at your own pace
  - Please ask questions as needed!
  - Start the HW when you are done

Data Type Smorgasbord

- Basic Types and Casts
- Big Integers
- Constants
- Strings and Conversions
- Understanding Error Messages
- String Input and Output

Basic Types (again)

Table 1 Primitive Types

Type	Description	Size
`int`	The integer type, with range –2,147,483,648 . . . 2,147,483,647 (about 2 billion)	4 bytes
`byte`	The type describing a single byte, with range –128 . . . 127	1 byte
`short`	The short integer type, with range –32768 . . . 32767	2 bytes
`long`	The long integer type, with range –9,223,372,036,854,775,808 . . . 9,223,372,036,854,775,807	8 bytes
`double`	The double-precision floating-point type, with a range of about $\pm 10^{308}$ and about 15 significant decimal digits	8 bytes
`float`	The single-precision floating-point type, with a range of about $\pm 10^{38}$ and about 7 significant decimal digits	4 bytes
`char`	The character type, representing code units in the Unicode encoding scheme (see Advanced Topic 4.5)	2 bytes
`boolean`	The type with the two truth values `false` and `true` (see Chapter 5)	1 bit

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

Conversions and Casts

- Consider:

```
int i, j;
double d, e;
i = 10;
d = 20.1;
e = i; // OK
j = d; // ERROR!
```

- Why the difference?
  - Assigning a double to an int can result in information loss (the fractional part)
- Add a cast to tell Java that we understand there could be a problem here:

```
j = (int) d; // OK
```

- But what happens to the fractional part of d?
  - It is truncated (lost)

Q4,5

Example

- Look at RoundAndRound.java
  - What does it do?
- Run it and try some different numbers, like:
  - 1.004
  - 1.005
  - 1.006
  - -1.006
  - 4.35
- Zoinks! What's up with these, especially the last one?
  - Try changing the `%f` format specifier to `%24.20f`

When Nine Quintillion Isn't Enough

- `BigInteger` for arbitrary size integer data
- `BigDecimal` for arbitrary precision floating point data

- We plan to revisit BigInteger later in the course

Constants in Methods

- Constants let us avoid *Magic Numbers*
  - Hardcoded values within more complex expressions
- Why bother?
  - Code becomes more readable, easier to change, and less error-prone!
- Example:

```
final double relativeEyeOutset = 0.2;
final double relativeEyeSize = 0.28;
final double faceRadius = this.diameter / 2.0;
final double faceCenterX = this.x + faceRadius;
final double eyeDiameter = relativeEyeSize * this.diameter;
…
```

`final` tells Java to stop us from changing a value (and also gives a "hint" to the compiler that lets it generate more efficient code)

Q7,8

Constants in Classes

- We've also seen constant fields in classes:
  - `public static final int FRAME_WIDTH = 800;`

- Why put constants in the class instead of a method?
  1. So they can be used by other classes
  2. So they can be used by multiple methods
  3. So they are easier to find and change

Strings in Java

- Already looked at some String methods
- Can also use + for string concatenation
- Quiz question:
  - Look at StringFoo.java
  - Based on the four uses of + in `main()`, can you figure out how Java decides whether to do string concatenation or numeric addition?
  - Decide what the 3 commented-out uses of + in `main()` will print, then uncomment them and see if you were right.
    - Do you see why they work as they do?

Converting Strings to Numbers

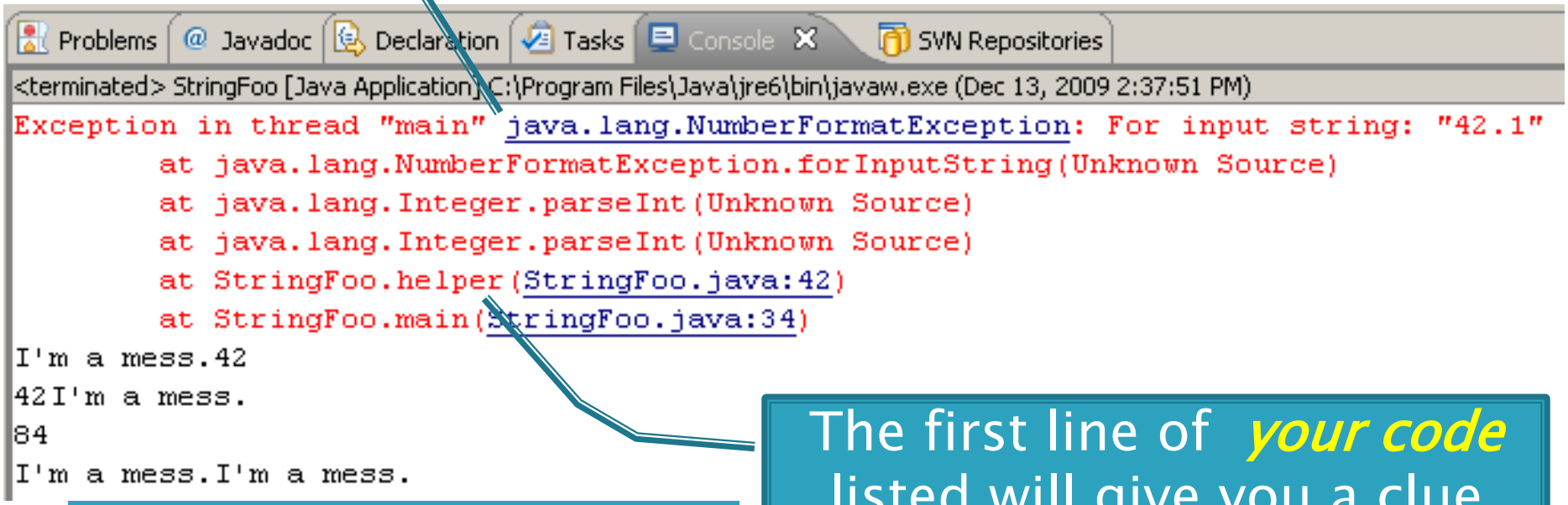
- You can convert strings to numbers:
  - `double Double.parseDouble(String n)`
  - `int Integer.parseInt (String n)`
- Can also convert numbers to strings:
  - `String Double.toString(double d)`
  - `String Integer.toString(int i)`
- Or maybe easier:
  - `"" + d`
  - `"" + i`

Conversions Gone Awry

- Go back to StringFoo.java
- Uncomment the last line of `main()`:
  - `StringFoo.helper();`
- Run it
- What happened?

Reading Exception Stack Traces

The first line will usually give you a hint about what went wrong.



The first line of *your code* listed will give you a clue where to look.

The error output appears at the *top* of the Console window (even though the error occurred *after* the output that is displayed).

char Type in Java is Like C's

- In Python:
  - "This is a string"
  - 'and so is this'
- In Java:
  - "This is a string"
  - This is a character: 'R'
  - 'This is an error'

Iterating Over Strings in Java

- Can use **`charAt(index)`**
- Example:

```
String message = "Rose-Hulman";
for (int i=0; i < message.length(); i++) {
    System.out.println(message.charAt(i));
}
```

- **`charAt()`** returns a 16-bit **`char`** value*
- Exercise: Work on TODO items in StringsAndChars.java

* Unfortunately there are more than 2^{16} (65536) symbols in the known written languages. See `Character` API docs for the sordid details.

Reading 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
  - Add **`println`**'s to the code to prompt the user for the values to be entered
    ```

Formatting with printf and format

Table 3 Format Types

Code	Type	Example
d	Decimal integer	123
x	Hexadecimal integer	7B
o	Octal integer	173
f	Fixed floating-point	12.30
e	Exponential floating-point	1.23e+1
g	General floating-point (exponential notation used for very large or very small values)	12.3
s	String	Tax:
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

More options than in C. I used a couple in today's examples. Can you find them?

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

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);`
- Display dialog box messages
  - `JOptionPane.showMessageDialog(null, message);`

Q13,14

Exercise

- Create a `CubicPlot` class as described in the HW