Data structures in Java

Session 2

Instructor: Bert Huang

http://www1.cs.columbia.edu/~bert/courses/3134

Announcements

- Nikhil's office hours: Friday 10 AM-12 PM
  Monday 2 PM-4PM
- Clarification for HW1: **Collection** test function should allow user manipulation; write a simple prompt
- Homework 1 is due on 9/22 by class time; that is in a little less than 12 days

Today’s Plan

- Java review
  - Some slides with general info
  - Live demo using CUNIX and emacs
- Math review

Java Syntax Basics

- You can write comments via C style

  `/* The compiler ignores this */`

  or double slashes `// this is ignored`

- ```
  System.out.print("Hello World");
  System.out.println("Hello World");
  ```

- Strings can be added, numbers automatically converted:

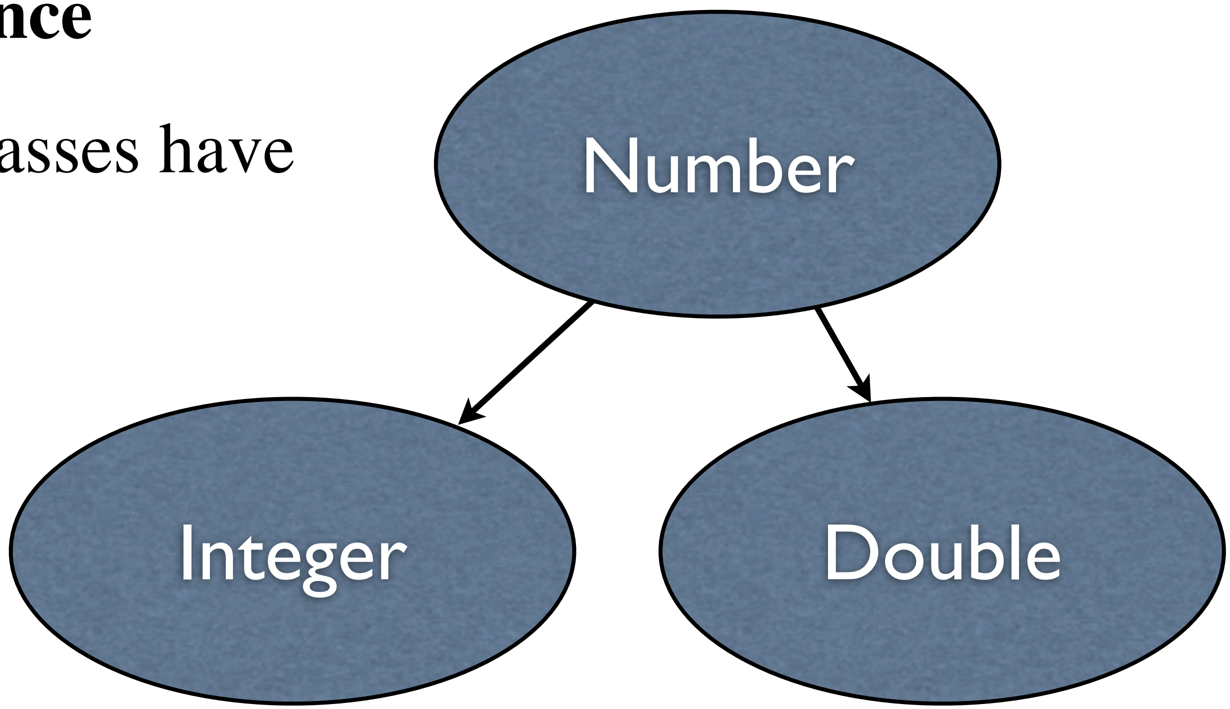
  ```
  System.out.println("Pi is "+Math.PI);
  ```
  ```

Objected Oriented Programming

- Java is an **object oriented** programming language
  - Even the programs themselves are objects that manipulate other smaller objects
  - Objects are classified into **class**es, which exist in a hierarchy of **inheritance**
  - Furthermore, similar classes have **polymorphism**

Java is an **object oriented** programming language

- Even the programs themselves are objects that manipulate other smaller objects
- Objects are classified into **class**es, which exist in a hierarchy of **inheritance**
- Furthermore, similar classes have **polymorphism**



Classes

- A **class** is a type of object. It has
  - **methods**, which are the functions available for objects of this class
  - **data members**, which contain the information used by the class
- The class and its components can be either **public** or **private**

Encapsulation

- Preserve abstraction in your code
- Anything that doesn't need to be public should be private
- Limit what a user of your class can do so those limited features are secure, robust, well-tested

Primitives vs. Objects

- Primitives: int, boolean, double, long...
  - primitives are passed by value
- Objects: Integer, Boolean, Double, String, Scanner, LinkedList, Collection, any class we write...
  - Objects are passed by reference.

Working with Objects

- After declaring a variable that represents an object, you must also instantiate the object
  `Integer myNumber = new Integer();`
  - Variables start out as NULL
- **new** creates an instance in memory
- The variable name **refers** to the instance

Exceptions

- Java has built in support for handling errors by using **exception** objects
- Exceptions are **thrown** and **catch**'d, (caught?) e.g.,

```
try {
    SomethingDangerous();
} catch (Exception error) {
    System.out.println("Something went wrong:
                        +error);
}
```

Common Modifiers

- static - value is the same for all objects of this class. Static methods and variables can be used without instantiating (e.g., main)
- final - value cannot be changed; useful for setting constants
- abstract - used on a class if some methods are unimplemented; means they must be implemented in a subclass

Generics

- We want our data structures to be very general, but Java typically wants all variables to have a type
- The old way to get around this is to **cast** the object as an **Object**
- Since Java 5, we can now use **generic**s
- public class Collection<MyType>

Generics continued

- `public class Collection<MyType>`
- ```
  Collection<Integer> foo = new
                Collection<Integer>();
  ```
- Now `foo` must always work with Integers, even though the class `Collection` is written without specifying a type.
  ```

Warning: Generics Arrays

- ```
  MyType[] A = new MyType[N];
                                  // doesn't work!
  ```
  Generic array declarations are not allowed exactly (because Java is stupid*)

- Instead, instantiate an array of **Object**s, and cast it as a generic. For example, an array **A** of **N** MyType objects is:

  ```
  MyType[] A = (MyType[]) new Object[N];
  ```

*Java is not stupid
  ```

CUNIX Demo

Math Background: Exponents

$$\begin{aligned}
X^A X^B &= X^{A+B} \\
\frac{X^A}{X^B} &= X^{A-B} \\
\left(X^A\right)^B &= X^{AB} \\
X^N + X^N &= 2X^N \neq X^{2N} \\
2^N + 2^N &= 2^{N+1}
\end{aligned}$$

Math Background: Logarithms

$$X^A = B \text{ iff } \log_X B = A$$

$$\begin{aligned}\log_A B &= \frac{\log_C B}{\log_C A};\ \boxed{A, B, C > 0, A \neq 1} \\ \log AB &= \log A + \log B;\ \boxed{A, B > 0}\end{aligned}$$

Math Background: Series

$$\sum_{i=0}^{N} 2^i = 2^{N+1} - 1$$

$$\sum_{i=0}^{N} A^i = \frac{A^{N+1} - 1}{A - 1}$$

$$\sum_{i=1}^{N} i = \frac{N(N+1)}{2} \approxeq \frac{N^2}{2}$$

$$\sum_{i=1}^{N} i^2 = \frac{N(N+1)(2N+1)}{6} \approx \frac{N^3}{3}$$

Math Background: Proofs

- Proof by Induction:
  - Prove base case,
  - Inductive hypothesis. Prove claim for current state assuming truth in previous state
- Proof by Contradiction: assume claim is false.
  - Show that assumption leads to contradiction

Reading

- We covered today material in Weiss Ch. 1 - 2.1
- For Tuesday, the rest of Weiss Ch. 2