CMSC 131
Object-Oriented Programming I

Libraries, Round Off Errors

Dept of Computer Science
University of Maryland College Park

This material is based on material provided by Ben Bederson, Bonnie Dorr, Fawzi Emad, David Mount, Jan Plane
Overview

- Constants
- Floating Point Calculations
- Libraries
- Java API
About Naming Constants

- If a constant is static then use uppercase letters
  
  ```java
  final static int MAX = 10;
  ```

- If a constant is not static then **do not** use uppercase letters (use camel case)
  
  ```java
  final int maxPressure = 50;
  ```

Example: ScienceExperiment.java
**Accessing Private Instance Variables**

- **IMPORTANT:** Remember that methods of a class can access private elements of parameters that belong to the same class.
- Assume we have a `Student` class with a private instance variable called `name` and a method called `checking`

```java
public boolean checking(Student p) {
    // We can access p.name here even though is private
    // We don’t need a get method. The method `checking`
    // can access its own private instance variables and
    // also those of objects that belong to the same class
}
```
Project Documentation

- A general comment at the top of every class
- A detailed comment above each method (describing the "contract" for the method... pre-conditions, post-conditions, etc.)
- Frequently comments should be embedded in the midst of complex code fragments to guide the reader about what is going on
- As already mentioned avoid code duplication
Floating Point Calculations

What will this print?

```java
public class SimpleMath {
    public static void main(String[] args) {
        if (3.9 - 3.8 == 0.1) {
            System.out.println("I am a very smart computer.");
        } else {
            System.out.println("I can't do simple arithmetic.");
        }
    }
}
```

- What is the output?
- Floating point numbers in Java are stored in binary representation, and frequently numbers that are easily represented in base 10 cannot be represented precisely in base 2
- What can we do?
Floating Point Calculations

Two important rules:

- You can never use `==` to compare floating point values. Instead, check if two numbers are within a certain tolerance of each other:
  
  ```
  Math.abs((3.9 - 3.8) - 0.1) < EPSILON
  ```

- Never use floating point values to represent money, e.g., 3.52 to represent $3.52. Instead, use integer 352 to represent 352 pennies.
**Libraries in Java**

- **Library** → implementation of useful routines that are shared by different programs
- Java mechanism for creating libraries: **packages**
  - Package: group of related classes
  - Example: `java.util` (contains `Scanner` class)
- To create a package in Eclipse use
  - File → New → Package
- To use a class from a package, you can use a **fully qualified name** (package name + class name):
  ```java
  java.util.Scanner s = new java.util.Scanner(System.in);
  ```
- You can also import the class in the beginning of the file
  ```java
  import java.util.Scanner;
  ```
- To import class in a package:
  ```java
  import java.util.*;
  ```
  (Imports `Scanner` as well as other classes in package)
- To create unique package name we use reverse domain names
  - Let’s see some packages in the Eclipse installation (plugins folder)
Package java.lang

- A special package containing widely used classes:
  - String
  - Math
  - etc.
- java.lang.* is automatically imported by every Java program
A class can be added to a package by including:

```java
package <name of package>;
```
in source file (usually very first line)

- The variables/methods provided by a class/package are often called its **API** (= Application Programmers Interface)
- APIs should be documented
- **java.lang** documentation:

  [documentation](http://download.oracle.com/javase/6/docs/api/java/lang/package-summary.html)