Lecture Set #7: Libraries, Encapsulaton, and "this"

1. Review of Parameter passing
2. Libraries
3. public vs. private Choices
4. this

Parameters and Methods

- Recall that methods / constructors can have parameters

```
public Student (String newName, int IDDesired) {
    name = newName;
    id = IDDesired;
    tokenLevel = 3;
}
```

- What is printed by the following?

```
String newName = "Joe";
Student s = new Student(newName + " Schmoe", 123456789);
System.out.println (s.name);
System.out.println (newName);
```

- `Joe Schmoe`
  `Joe`

How Does Java Evaluate Method / Constructor Calls?

```
int newName = "Joe";
Student s = new Student
   (newName + " Schmoe", 123456789);
```

1. Arguments are evaluated using stack in effect at call site (place where method called)
   - `newName + " Schmoe"`, evaluates to `Joe Schmoe`
   - `123456789` evaluates to `123456789`
2. Stack frame (temporary addition to stack) created to associate method parameters with values
3. Stack frame put into stack
4. Body of method executed in modified stack
5. Stack frame removed from stack

Libraries in Java

- Library: implementation of useful routines that are shared by different programs
- Java mechanism for creating libraries: packages
  - Package: group of related clases
  - Example: `java.util` (contains `Scanner` class)
- To use a class from a package, you can use a fully qualified name (package name + class name):

```
java.util.Scanner s = new java.util.Scanner(System.in);
```

- You can also import the class in the beginning of the file

```
import java.util.Scanner;
```

- To import class in a package:

```
import java.util.*;
```

(Imports `Scanner` as well as other classes in package)

Package java.lang

- A special package containing widely used classes:
  - `String`
  - `Math`
  - etc.
- `java.lang.*` is *automatically imported* by every Java program

Package Management

- A class can be added to a package by including:

```
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: http://java.sun.com/j2se/1.3/docs/api/java/lang/package-summary.html
- On the resources page of the class web site – javadoc generated descriptions.

String API & Math API

- `String` implements lots of string functions
  - `StringExample.java`
- `Math` implements lots of mathematical functions
  - `MathExample.java`

Public Declarations

- public variables/methods and classes
  - Keyword `public` used in declaration
  - Every user of an object can access any `public` element
- Sometimes access should be restricted!
  - To avoid giving object users unnecessary info (keep API small)
  - To enforce consistency on instance variables

Private Declarations

- private variables, methods and classes

```
private int tokenLevel = 3;
```

- Private variables / members cannot be accessed outside the class definition
- Declaring instance variables private means they can only be modified using public methods

What Should Be Public / Private?

- Class interface = API = public variables / methods
- Only make something public if there is a reason to
- Why? Encapsulation
  - As long as interface is preserved, class can change without breaking other code
  - The more limited the interface, the less there is to maintain
- Rule of thumb
  - Make instance variables private
  - Implement `set` / `get` methods
  - Make auxiliary methods private

Separate: API and the workings of the class

- Design so that
  - you can change how the class works without having to change the API
  - the only things in the API are things the user will absolutely need (make the interface as simple as possible)
- Demonstrations in Class
  - Significantly Modifying the Student class – without changing the API (or the driver)
  - The Cat class and its drivers
    - with adding a copy constructor
  - Project 3
    - API described – you are using those classes
    - documentation / comments needed

this

- a reference to the current object. (Only makes sense in a non-static method.)
- In an instance method, this is the object that is assumed
  - easy to refer to members (data or methods) using the assumed object
  - difficult to refer to the whole object without having a name to call it
- Only use when needed – using it all the time makes the code more difficult to read