Computer Science 22: Object Oriented Programming

Lecture #6: Java Programming

About this Lecture

- Java programming statements
  - Package declarations
  - Import statements
  - Class and Interface declarations

Package Declaration

- In Java, classes are grouped in packages
  - Grouped according to functionality or usage
  - Equivalent to ‘library’ or ‘namespace’
- Packages may have sub packages
- Package naming follows Java naming rules

Package Declaration

```
package a;     //single package

package a.b;   //sub-package b inside a

package org.uplb.systemone;
```

Package Declaration: Compiling

- When compiling source codes with package declaration:
- The –d (directory) tells us where the compiled produce will be outputted to
- e.g., javac –d . *.java
  - The "." means put the class files in the current directory and follow package specification
  - The package and sub-packages will become directories and subdirectories

Import Statements

- Import statements are equivalent to `#include` in C/C++
- When using classes in your source code, you may be required to use the fully-qualified name of the class or use an import statement.
- You may have more than one import statement in your source code.

Import Statements

- Fully-qualified class name uses the following format:
  - packagename.classname

```
java.util.Vector v = new java.util.Vector();
```

```
import java.util.Vector;
.
.
Vector v = new Vector();
```

Import Statements

- The statement below means “import ALL CLASSES found in subpackage b which is inside package a

```
import a.b.*;
```

- The statement below means “import the SPECIFIC CLASS named SomeClass inside package a”

```
import a.SomeClass;
```

Class Declaration

```
[public] [abstract|final] class ClassName
[extends SuperClass]
[implements Interface1, Interface2, ...]
```

- Note: *The terms inside* **[ ]** *are optional and used only when required and the* **|** *symbol means exclusive OR.*

Class Declaration

- `public` is a package access modifier. Only one public class (or interface) is allowed in a source code.
- **Abstract** classes have unimplemented methods.
- **Final** classes are classes that do not have subclasses

Class Declaration

- Example of an abstract class

```
public abstract class Shape {}
```

- Example of a subclass that extends a superclass

```
class Rectangle extends Shape {}
```

Example of a subclass that extends a superclass and implements an interface

```
class Rectangle extends Shape
implements Drawable {}
```

Class Declarations

```
public class Rectangle implements Drawable,
Transform { }

abstract class Animal { }
```

Inner Class Declarations

- Advanced topic – we will cover at the end of our course if we have time
- Class declared inside another class

Interface Declarations

- Interfaces are ‘cousins’ of classes
- They are used to specify behavior

```
[public] interface InterfaceName [extends
ParentInterface]
```

Assignment

- What are the different access modifiers for variables?