Lecture 11: Scope and Packages

Data Organization and Access Control
Access & Organization

• Your computer has thousands of files.
• The internet has millions of sites.
• Projects have thousands of classes.
• How do we find particular classes, files, or sites?
• How do we deal with duplicate or similar names?
• How do we restrict access to private data?
This Lecture

• Scope: What is accessible where.

• How to use Java packages.

• How to create Java packages.

• Reading: Eckel, “Chapter 5: Hiding the Implementation”, *Thinking in Java*
Scope

• A variable, field, method, or class’s scope consists of all code where it is accessible.
• Variables are accessible within the block that they are declared.
• Blocks are contained in curly braces.
• Local variable: A variable defined only in the current block.
• The accessibility of fields, methods, and classes is determined by their modifiers.
Anonymous Variables

- **Anonymous variables** are allocated and initialized, but never declared.
- In C/C++ anonymous variables are a bug; in Java they are a feature.
- They have no name, and cannot be referenced -- thus have no scope.
- Examples:
  - `System.out.println("Hello");`
  - `(new String("Hello"))`
  - `(new int[] {1,2,3,4})`
- (The last example is new notation we haven’t used; in fact we just learned.)
Variable Scope Example

0    void foo(int x) {
1            int y = 3;
2            if (y > 0) {
3                       int z = 4;
4                        if (z > 0) {
5                                       int w = 0;
6                                           {
7                                              int v = 1;
8                                               } // end block
9                                           } // end if (z>0)
10                                      } // end if (y>0)
11                          } // end Foo

Which lines are v, w, x, y, and z accessible?
class TestScope {
    int x = 0;
    void foo(int z) {
        int y = 20;
        x = 10;
        int z = 30; // Error
    } // end foo()
    void print() {
        System.out.println(x);
        foo(x);
        System.out.println(x);
        /* Next line error */
        System.out.println(y);
    } // end print()
} // end TestScope

• x is defined for the whole class block.
• y is defined inside the method f(int).
• z is already defined in f(int) by the argument.
class Scope {
    int x = 3;
    void foo(int y) {
        System.out.println(x);
        int x = 2;
        System.out.println(x);
        System.out.println(this.x);
        System.out.println(y);
    }
    public static void main(String[] args) {
        int x = 1;
        (new Scope()).foo(x); // Anonymous Object
    }
}

What is the output of this program?
int sigma(int n) {
    for (int i = 0; i < n; i++) {
        int sum += i;
    }
    return sum;
}

• The method sigma is supposed to return $\sum_{i=0}^{n} i$

• Why won't sigma compile?
Scope Quiz

What is the output of this program?

- 0
- 10
- 0

```java
class TestScope {
    int x = 0;
    void foo() {
        int y = 20;
        x = 10;
    } // end foo
    void print() {
        int y = 0;
        System.out.println(x);
        foo();
        System.out.println(x);
        System.out.println(y);
    } // end print()
} // end Class TestScope
```
Scope Quiz 2

```java
class TestScope {
    int x = 0;
    int y = 0;
    void foo() {
        int y;
        y = 20;
        x = 10;
    }
    void print() {
        System.out.println(x);
        foo();
        System.out.println(x);
        System.out.println(y);
    }
}
```

- Now, we declare a new field, y.

- What is the output of print()?
  - 0
  - 10
  - 0
Scope Quiz 3

class TestScope {
    int x = 0;
    int y = 0;
    void foo() {
        y = 20;
        x = 10;
    }
    void print() {
        System.out.println(x);
        foo();
        System.out.println(x);
        System.out.println(y);
    }
}

• Now, we change the method foo().

• What is the output of print()?
  – 0
  – 10
  – 20
Naming in Big Projects

- Large projects may have thousands of classes.
- You may collaborate and share classes with people all over the world.
- How do we make sure our class names don’t collide with someone else’s?
- Could use really long, unique names..
  - “UtilityClassForPreparingTaxReturnsByAnthonyGfromWestlandsNairobiKenya”
- This is difficult to remember and use.
Hierarchical Namespaces

- Names are organized into a hierarchy.
- `crypto.csail.mit.edu`: Crypto group, at CSAIL, at MIT, an educational institutions.
- 254-020-5555555: Kenya (254), Nairobi (020), Number 5555555
- `java.lang.String`: String class, in the lang `package`, in the java package.
Defining Packages

• Organize your classes into sets or units called packages.
• Reduces problems with name conflicts and identifies functionality, e.g. java.util.
• Can restrict access to within a package.
• How to specify the package for a class:

```java
package packageName;

class className {
    /* Class Body */
}
```
Package Caveats

• Package declaration must be first non-comment line in a file.
• The first class defined in a file must be the named the same as the file name.
• Only the first class may be public.
• javac and java will search in appropriately named subdirectories or JAR (Java Archive) for source and binaries:
  - \mypkg\util for mypkg.util (on Windows)
  - /mypkg/util/arrays for mypkg.util.arrays (on Unix)
Using Packages

• Can use fully-qualified names:
  - `java.util.Date d =
    new java.util.Date();`

• Or specify the classes you want to import:
  - `import java.util.Date;`
  - `import java.util.ArrayList;`

• Import all classes in a package:
  - `import java.util.*;`

• Packages can have sub-packages:
  - `import java.util.logging.*;`

• Default imported package:
  - `import java.lang.*;`
Access Modifiers

- Fields, Methods, Constructors, or Classes with **public access** are accessible to any other class in any package.
- Fields, Methods, or Constructors with **protected access** are accessible to any child class* (later lecture).
- Fields, Methods, Constructors, or Classes with **package access** are available to any class within the same package. This is the default.
- Fields, Methods, or Constructors with **private access** are only accessible within that class.
## Levels of Access Control

<table>
<thead>
<tr>
<th>Accessible..</th>
<th>private</th>
<th>package (default)</th>
<th>protected</th>
<th>public</th>
</tr>
</thead>
<tbody>
<tr>
<td>From same class</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>From same package</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>From child classes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>From anywhere</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Package Example: Person.java

package examples;

class Person {
    String name;

    // We will complete this class by:
    // Adding a field to store a birthday
    // Writing a method to set the birthday
    // Writing a method to get the birthday

}
package examples;

class Person {
    String name;
    java.util.Date birthDay;
    void setBirthday(java.util.Date d) {
        this.birthDay = d;
    }
    java.util.Date getBirthday() {
        return this.birthDay;
    }
}

Importing java.util.Date

package examples;
import java.util.Date;
class Person {
    String name;
    Date birthDay;
    void setBirthday(Date d) {
        this.birthDay = d;
    }
    Date getBirthday() {
        return this.birthDay;
    }
}
Importing java.util.ArrayList

```java
package examples;
import java.util.Date;
import java.util.ArrayList;

class Person {
    String name;
    ArrayList friends;
    Date birthDay;
    void setBirthday(Date d) {
        this.birthDay = d;
    }
    Date getBirthday() {
        return this.birthDay;
    }
}
```
import java.util.*;

package examples;

class Person {
    String name;
    ArrayList friends;
    Date birthDay;
    void setBirthday(Date d) {
        this.birthDay = d;
    }
    Date getBirthday() {
        return this.birthDay;
    }
}