Scope Static
Lecture 27
Reusing some slides by Margaret Dulat and Joe Luk

Readings
This Week: Ch 8.3-8.8 and into Ch 9.1-9.3
(Ch 9.3-9.8 and Ch 11.1-11.3 in old 2nd ed)
(Reminder: Readings are absolutely vital for learning this stuff!)

Labs and Tutorials
This Week: Lab #8 – A catch-up lab…

Survey #2
- Dr. Ben Yu’s second survey for you all is up on WebCT now.
- Completely optional…
- But, there’s a bit of extra credit for doing all three surveys by their respective deadlines!
- (I think the deadline for this one is March 22, but check on WebCT to be sure.)

Learning Goals
By the end of class today you will be able to…
- Figure out exactly which declarations (variables, methods) are visible from which parts of a program.
- Explain what “static” means in Java programs and why you might want to use static methods and fields.

Variable Scope
- Scope tells you which declarations you can see from which points in the program.
- The scope of a variable is the places in the program where the variable can be accessed.
- Or you can think of scope as the places where you can see some variable.
- (Think “scope” meaning “extent”, as in “the scope of my legal authority” or “the scope of the inquest”. Not “scope” as in “telescope” or “scoping out the beach”.  }
Variable Scope

- The world is broken up into packages.
- `import` lets you see public stuff from other packages.
- Or you can give full name, e.g., `java.util.Scanner`.
- Other than that, can’t see inside other packages.

Example:
```java
import java.util.Random;
Random foo = new Random();
```

Variable Scope

- Packages contain classes
  - Within a package, everyone can see all classes and non-private stuff within classes.

Example:
```java
TalkingDoll a = new TalkingDoll();
System.out.println(a.getName());
System.out.println(a.name);
```

The variable `name` is private in the TalkingDoll class!

Variable Scope

- Classes contain methods and fields
  - All methods can see other methods and fields in same class.

Example:
```java
public void setLabScore(int labNum, double score) {
    labScores[labNum-1] = score;
    computeTotalLabScore();
}
```

Variable Scope

- Methods contain statements, executed one-by-one
  - You can see declarations that happened already.
  - Parameters are like declarations that happen at start of method.
  - Curly braces define a compound statement, which limits scope.

Example:
```java
public static void main(String[] args) {
    int a = 0;
    System.out.println(args[0]);
    int b = 2*a;
    int a = 3; // Not OK!
    b = c+a; // Not OK! -- can’t see c
    int c;
}
```
Variable Scope

- Compound statement limits scope
- for loop is special: variables declared in header are treated as if declared inside loop body

Example:
```java
public static void main(String[] args) {
    int a = 0;
    int b = 0;
    for (int c=0; c < 3; c++)
    {
        // start of a block
        int a;  
        System.out.println(c);  // OK
    }
    System.out.println(c);  // Not OK!
}
```

Variable Scope: Shadowing

- Local variable declarations wins over instance variable declarations.

Example:
```java
private static int a = 0;
public static void main(String[] args) {
    int b = 0;
    { // start of a block
        int a = 2;  // OK!  Shadows other a
        int c = a;
        System.out.println(c);  // prints 2
    }
    System.out.println(a);  // prints 0
    System.out.println(c);  // Not OK!
    int c = 2;  // OK
}
```

Variable Scope -- Summary

- The world is broken up into packages.
- import lets you see public stuff from other packages.
- Or you can give full name, e.g., java.util.Scanner
- Packages contain classes
- Within a package, everyone can see all classes and non-private stuff within classes
- Classes contain methods and fields
- All methods can see other methods and fields in same class
- Methods contain statements, executed one-by-one
- You can see declarations that happened already
- Parameters are like declarations that happen at start of method.
- Curly braces define a compound statement, which limits scope
- Compound statement limits scope
- for loop is special: variables declared in header are visible only inside the loop

Questions

The Meaning of static

- From our very first Java program, we had:
  public static void main (String[] args)
- But what does static mean?

Review: Static Methods

A method that is tagged as static:

- Can be called on the class, rather than on an object of the class (e.g., Math.sin(1.0))
- Does not have an implicit parameter this
- Cannot directly access non-static methods or fields (because there is no this!)
The Meaning of static

- But what does static mean?
- In general, static means something that belongs to the class as a whole, and not to a specific object:
  - Static methods have no implicit parameter this
  - Static fields belong to the whole class, shared by all objects.

Java Classes

- When you write a class in Java, you describe what goes into each object of the class:
  ```java
  public class UBCStudent {
      String name;
      int studentID;
      public String getName() {
          ...
      }
  }
  ```
  That says that each object of type UBCStudent contains a field name, and a field studentID, and the object knows how to do the getName method (applied to itself).

Analogy/Intuition on static

- A class is a pattern/blueprint/factory that shows, by example, how to create specific types of objects:
  - Each object has these instance fields...
  - Each object has these methods, that do the following statements...
- Sometimes, you want to talk about the pattern/blueprint itself, not the objects. Those are when you want to use static

Real Life Analogy of static Fields

- A factory/blueprint for a car is like the class. The cars are the objects.
  - The blueprint shows an engine, battery, etc.
  - Each car gets its own engine, battery, etc. These are instance fields.
  - But to give each car a serial number, you need a field that lives in the factory. That should be a static field.

Example: UBCStudent with Automatically Assigned Student ID Numbers

- Each object should get a unique student ID number assigned to it.
  - We'll keep a variable nextID that keeps track of the next available ID number.
  - nextID is one variable for the whole class, so it should be static!
Recap: Static Fields
A field that is tagged as static:
- Is a single field of the class, rather than a separate field of each object of the class
- Is shared by all objects of the class
- Can be accessed without creating an object of the class
- Can be accessed from static methods

Why Use static?
- Static methods are almost never used, with two common exceptions:
  - Numbers aren’t objects in Java, so to have a class of methods that do computation, we’ll often put them in static methods (e.g., the Math class).
  - The main method starts running before there are any objects created, so it has to be static.

Why Use static?
- Static fields are not very common, but are needed when the class as a whole has to coordinate some information, e.g.:
  - Counting how many objects were created
  - Assigning serial numbers
  - Having all objects share or negotiate some information in common.

Recap: Variable Types
- Static variables
  - declared within class
  - associated with class, not instance, indefinite lifetime
- Instance variables
  - declared within class
  - associated with instance
  - accessible throughout object, lifetime of object
- Local variables
  - declared within method
  - accessible within method after declared, lifetime of method
  - (to be more precise, replace “method” by “block”)
- Parameters
  - declared in parameter list of method
  - accessible throughout method, lifetime of method

Questions

Midterm Info
- Out of 100: raw avg=56, low=1, high=97
  - Will be scaled (info later)!
- Long exam, but not unreasonable:
  - Everyone should be able to do it (with time).
  - Talk to friends, learning centre, WebCT, TAs, prof.
  - See Lecture 17 notes for tips from Beth Simon…
- Solutions will be posted.
- Protocol for re-mark requests:
  - read solutions first, carefully
  - no re-mark requests accepted until Thursday
  - re-mark requests must be in writing (paper attached to exam) and submitted to instructor
  - entire exam re-marked
Midterm #1

- Raw Score Stats:
  - Average: 56, Low: 1, High: 97 (Out of 100)
  - I've made ArrayList question extra credit (3pts).
- Scaling Formula:
  \[(\text{raw}/97)^{0.7} \times 100\]

<table>
<thead>
<tr>
<th>Raw Score</th>
<th>Scaled Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>27</td>
</tr>
<tr>
<td>30</td>
<td>44</td>
</tr>
<tr>
<td>45</td>
<td>58</td>
</tr>
<tr>
<td>60</td>
<td>71</td>
</tr>
<tr>
<td>75</td>
<td>84</td>
</tr>
<tr>
<td>90</td>
<td>95</td>
</tr>
</tbody>
</table>