Lecture 25:

Static Methods and Variables;
revisiting Visibility Modifiers

Mar 31 2015
Exam 2

Covers all material through Lecture 23 (Intro to objects and classes, including constructors)

• Methods
• Scope
• Arrays
• Objects and classes
**static** variables, constants, and methods

- Static variables are shared by all the instances (objects) of the class.
- Static methods are not tied to a specific object. A static method cannot access instance members of the class.
- Static constants are final variables shared by all the instances of the class.
Static variables

Static variables are shared by all the instances (objects) of the class.

UML Notation:
+: public variables or methods
underline: static variables or methods

If you want all objects of a class to share a property, use a static variable. Static variables store values for the variables into a common location => if one object changes the value of a static variable, all object of the class are affected.
Static vs. non-static methods

A static method

Can
- Invoke a static method
- Access a static data field

Can NOT
- Invoke an instance method
- Access an instance data field

A non-static method

Can
- Invoke a static method
- Access a static data field
- Invoke an instance method
- Access an instance data field
public class CircleStatic {
    double radius;
    static int numObjects = 0;
    CircleStatic() {
        radius = 1;
        numObjects++;
    }
    CircleStatic(double newRadius) {
        radius = newRadius;
        numObjects++;
    }
    static int getNumObj() {
        return numObjects;
    }
}

public class TestCircleStatic {
    // in main
    public static void main(String[] args) {
        System.out.println("Before creating objects:");
        System.out.println("No of Circle objects is: "+ CircleStatic.numObjects);
        CircleStatic c1 = new CircleStatic();
        System.out.println("After creating c1");
        System.out.println("c1 radius = " + c1.radius + " and no of Circle obj = " + c1.numObjects);
    }
}

program outputs...
Before creating objects:
No of Circle objects is: 0
After creating c1
  c1 radius = 1 and no of Circle obj = 1
Example continued

```java
class TestCircleStatic2 {
    public static void main(String[] args) {
        System.out.println("Before creating objects:");
        System.out.println("No of Circle objects is: " + CircleStatic.numObjects);

        CircleStatic c1 = new CircleStatic();
        System.out.println("After creating c1");
        System.out.println("c1 radius = " + c1.radius + " and no of Circle obj = " + c1.numObjects);

        CircleStatic c2 = new CircleStatic(5);
        c1.radius = 9;
        System.out.println("After creating c2");
        System.out.println("c1 radius = " + c1.radius + " and no of Circle obj = " + c1.numObjects + 
                         "\n                         c2 radius = " + c2.radius + " and no of Circle obj = " + c2.numObjects);
    }
}
```

Before creating objects:
No of Circle objects is: 0
After creating c1
After creating c2
After creating c1

c1 radius = 9
After creating c2
After creating c1

After creating c1

After creating c2
Visibility Modifiers

By default, the class, variable, or method can be accessed by any class in the same package.

**public** modifier: The class, data, or method is visible to any class in any package.

**private** modifier: The data or methods can be accessed only by the declaring class.

**default** modifier restricts access to within a package.
The **private** modifier restricts access to within a class, the **default** modifier restricts access to within a package, and the **public** modifier enables unrestricted access.
Getter and setter methods

When using private data fields, the

get (getter / accessor methods)

set (setter / mutator methods)

are used to read and modify private properties.

For use of private data fields, as well as getters and setters in action, see Lecture 24 slides
Improving the previous Example with the `private` modifier

```java
public class CircleStatic {
    double radius;
    
    private static int numObjects = 0;

    CircleStatic() {
        radius = 1;
        numObjects++;
    }

    CircleStatic(double newRadius) {
        radius = newRadius;
        numObjects++;
    }

    static int getNumObj() {
        return numObjects;
    }
}
```