Access modifiers

Argument passing

Method overloading

Constructor overloading

Java's Access Modifiers

- public
- private
- protected (will cover in chapter 8)

- default is assumed if no modifier present
  same as public -
  unless program is broken into packages

  (packages are a grouping of classes)

When a member of a class is labeled public it can be accessed by any other code in your program

including methods defined in other classes.

ex:

```
class myClass {
    public int varX;                          <-- data

    public checkBounds(int k) {               <-- method
        ....
    }
}
```

When a member of a class is labeled private it can only be accessed by methods within that class.

ex:

```
class myClass {
    private int varX;                          <-- data

    private checkBounds(int k) {               <-- method
        ....
    }
}
```

When a member of a class has no access specification the default is used, which is the same as public ...

Unless

the program is broken into packages

(packages are groupings of classes which help with program organization and also access control. Chapter 8 will cover this)

Argument Passing

- call-by-value (pass-by-value)
  copies the value of an argument

  the method to which a value is passed
  cannot change the original variable
  in the main

Argument Passing (continued)

- call-by-reference (pass-by-reference)
  a reference to an object is passed

  the location of the object is passed to the called method, not the object itself.

  When the called method changes the contents of a variable in the object, the original object is changed.

Method Overloading

- one way that Java implements polymorphism

- overloading means two or more methods within the same class can share the same name ….
  so long as their parameter definitions are different

- For example, a method that adds two private variables could have an integer version and
  a real number version (i.e. a double type)

- Constructors in a class can be overloaded as well as methods

```
// Demonstrate method overloading.
class Overload {

  void ovlDemo() {
    System.out.println("No parameters");
  }

                    // Overload for one integer parameter.
  void ovlDemo(int a) {
    System.out.println("One parameter: " + a);
  }

                   // Overload for two integer parameters.
  int ovlDemo(int a, int b) {
    System.out.println("Two parameters: " + a + " " + b);
    return a + b;
  }
}
```

```
class OverloadDemo {
   public static void main(String args[]) {
       Overload ob = new Overload();

       ob.ovlDemo();
       System.out.println();

       ob.ovlDemo(2);
       System.out.println();
  }
}
```

Constructor overloading works the same as for method overloading. The data type of and number of parameters being passed to a constructor will determine which constructor will be used if there are multiple constructors.