1.00 Lecture 6

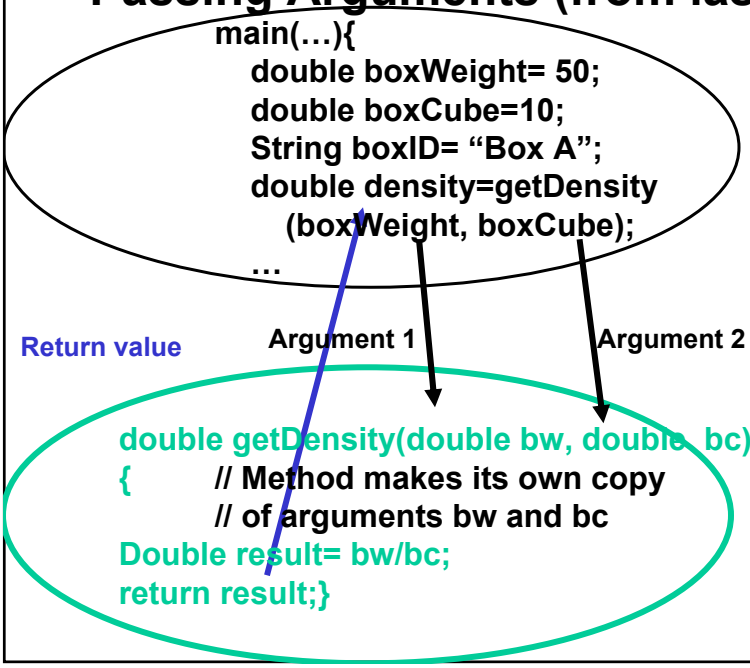
Methods and Scope

Reading for next time: Big Java: sections 2.1-2.5

Java Methods

- **Methods are the interface or communications between program components**
  - **They provide a way to invoke the same operation from many places in your program, avoiding code repetition**
  - **They hide implementation details from the component using the method**
  - **Variables defined within a method are not visible to users of the method; they have local scope within the method.**
  - **The method cannot see variables in the component that calls it either. There is logical separation between the two, which avoids variable name conflicts.**

Passing Arguments (from last time)



Communi-cation only via arg list, return value

Arguments matched by position

Assume method is written first: can't know main() vars

Method Exercise: step 1

- In Eclipse, create a new class AvgTest
- After its main method, write methods to:
  - Return the average of three doubles
  - Return the maximum of three doubles

Method Exercise: step 2

- In the main method:
  - Call your two methods with two sets of variables:
    - 10, 17, 55
    - 59, -3, 85
  - Output (System.out.println) the results
- For this exercise:
  - Use different variable names in your main method (e.g. r1, r2, r3) and in the argument list in your methods(e.g. x1, x2, x3)

Pass by copy

- In Java, arguments are passed from one method to another by copy (also called by value):
  - The called method makes a copy of the arguments. Even if it changes their values, they do not change in the calling method.
  - What is the output of the following program?

```
public class TripleTest {
  public static void main(String[] args) {
     double z=5.0;
     System.out.println("z main 1: "+z);
     triple(z);
     System.out.println("z main 2: "+z);
  }
  public static void triple(double z) {
     System.out.println("z 1: "+z);
     z *= 3;
     System.out.println("z 2: "+z);
  } }
```

Scope

- You've already seen that methods have different scope:
  - A variable of the same name in two methods is two separate variables
- Scope of local variables, the only kind we've seen so far, is defined by additional rules
- And, there are other kinds of variables, with their own scope rules
- We'll revisit all this later, but for now, we focus on local variable scope

Local Variable Scope

- Local variables (in a method or block)
  - Exist from point of definition to end of block
    - Blocks are defined by curly braces{ }
    - Blocks are most often used to define:
      - Method body
      - Multiple statements in if-else and loop operations
  - Local variables are very restricted:
    - Other methods cannot see local variables even in the same class.
    - Variables of the same name in different methods are different variables
    - More generally, variables of the same name in different blocks are different variables
  - Arguments to a method are local variables:
    - The method copies them upon receipt and they live until the ending curly brace of the method

Exercise

- Mark where variables d, e, i, j exist

```
public class ScopeTest0 {
   public static void main(String[] args) {
        int i= 1;
        double d= 0.0;
        for (int j= 0; j < 5; j++) {
                double e= j;
                d += i;
                e += j;
                System.out.println("d: "+d+" e: "+e);
        }
        if (d > 0) {
                int j= 2;
                double e= 4.0;
                System.out.println("If line d: "+d+" e: "+e);
        }
        double e= 0.0;
        e += d + i;
        System.out.println("Last line d: "+ d+" e: "+e);
   }
}
```

Scope exercise

- The following code doesn't work. Fix it.

```
public static double test1() {
    for (int i=0; i < 10; i++) {
           if (Math.sqrt(i) > 2.5)
                  break;
    }
    return i;
}
```

Scope exercise 2

- The following code doesn't work. Fix it.

```
public static double test2() {
    int i= 4;
    if (i*i > 6) {
        int i6= i;
    }
    int i7= i6 + 2;
}
```

Scope exercise 3

```
// What's wrong? Fix it. Find a general strategy to help.
public class ScopeTest {
   public static void main(String[] args) {
        test3();
   }
   public static void test3() {
        int i1;
        for (i1 = 0; i1 < 10; i1++)
            System.out.println("d: "+getDensity(i1));
        int i2;
        for (i2 = 0; i2 < 10; i2++)
            System.out.println("c: "+getCube(i1));
   }
   public static double getDensity(int i) {
        return i;
   }
   public static double getCube(int i) {
        return i * i;
   }
}
```