

Intro to Programming II

Scope and Parameters

Chris Brooks

Department of Computer Science
University of San Francisco

Department of Computer Science — University of San Francisco — p. 1/77

3-2: Scope

- Scope refers to the area of a program where a variable can be accessed.
- Java has three types of scope:
 - Local scope - the variable exists only within a method
 - Object scope - the variable can be accessed from any method belonging to an object.
 - Class scope - the variable can be accessed by all instances of a class.

Department of Computer Science — University of San Francisco — p. 2/77

3-3: Local scope

- Exists only when a method is executing
- The garbage collector reclaims a local variable when the method ends.
- Local variables are useful for temporary variables and counters

```
/* raise x to the yth power */
public int exponentiate(int x, int y) {
    int total;
    int i;
    for (i = 0; i < y; i++) {
        total = total * x;
    }
    return total;
}
```

Department of Computer Science — University of San Francisco — p. 3/77

3-4: Object scope

- Variables are available anywhere within an object.
- This is useful for data associated with an object that will be used by multiple methods.
- This is also called instance data.

```
public class circle {
    public int radius;
    public static final double pi = 3.14;

    public double getArea() {
        return pi * radius * radius;
    }
}
```

Department of Computer Science — University of San Francisco — p. 4/77

3-5: Class scope

- Class variables are available to all members of a class.
- These are declared as *static*
- This means that one copy of the variable is shared by all objects.
- Useful for defining constants.

Department of Computer Science — University of San Francisco — p. 5/77

3-6: Class scope

```
public class circle {
    public static final double pi = 3.14;
    public int radius;

    public double getArea() {
        return pi * radius * radius;
    }

    public static void main(String args[]) {
        circle c1 = new circle();
        circle c2 = new circle();
        c1.radius = 5;
        c2.radius = 6;
        System.out.println("c1's area is: " + c1.getArea());
        System.out.println("c2's area is: " + c2.getArea());
    }
}
```

Department of Computer Science — University of San Francisco — p. 6/77

3-7: Scope

- In the previous example, each circle had its own copy of radius.
- They all shared a copy of pi.
- radius has object scope, whereas pi has class scope.
 - Identify variables in Bank account program.

3-8: Parameters

- Parameters are the variables passed into a method.
- We can talk about:
 - Formal parameters - these are the variables named in the method definition.
 - Actual parameters - these are the variables in the method invocation.

3-9: Example

```
/** This is a method definition */
public double depositFunds(double amt) {
    balance = balance + amt;
    return balance;
}

...
Bankacct b = new Bankacct();
paycheck = 100.0
/* this is a method invocation */
b.depositFunds(paycheck);
```

3-10: Method signature

- Specification of all data types coming in and out of a method.
 - Type and order of input parameters
 - Type of return variable
- A method signature allows the compiler to uniquely identify a method.

3-11: Example

- Consider the following method declaration:
- `double calculate(double a1, double a2, double a3);`
- Which of the following are valid calls to this method?
 - `calculate(3, 52.0, -5.1);`
 - `double y = calculate(0, 1.1, 2.2);`
 - `calculate(1.1, 2.3);`
 - `calculate("Hello", 4.4, 2);`
 - `calculate();`
 - `calculate(3.3);`

3-12: Questions

- What happens in this case?

```
public class circle {
    public int radius = 5;

    public void printArea() {
        int radius = 4;
        System.out.println("Area is " + (radius * radius * 3.14));
    }
}
```

3-13: Questions

- What happens in this case?

```
public class bankacct {
    public double balance;
    public void updateBalance(double newAmount) {
        double newAmount = 12.0;
        balance += newAmount;
    }
}
```

3-14: Questions

- What happens in this case?

```
public class circle {
    public int radius = 5;

    public void printArea(int radius) {
        System.out.println("Area is " + (radius * radius * 3.14));
    }
    ...
    circle c = new circle();
    c.getArea(3);
}
```

3-15: Specifying scope

- In general, it's a bad idea to give local variables the same name as an instance or class variable.
 - Confusing, leads to bugs.
- If it can't be avoided, you can use "this" to indicate the instance variable should be used rather than the local variable.
- You can use the class name (e.g. circle.pi) to indicate that a class variable should be used.