Benazir Bhutto Shaheed University Lyari, Karachi

Department of Computing Science & Information Technology

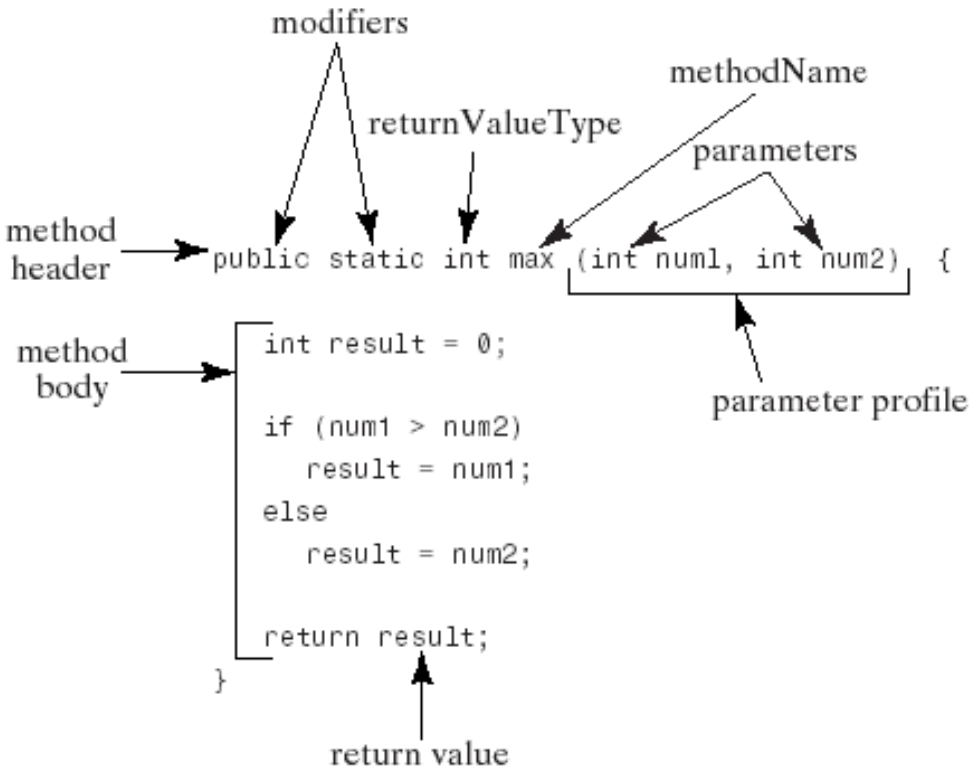
Objected Oriented Programming (OOP) IT 2nd Semester (CS IT-321)

OOP Method/Function

Introducing Methods

A method is a collection of statements that are grouped together to perform an operation.

Method Structure



Introducing Methods, cont.

- *parameter profile* refers to the type, order, and number of the parameters of a method.
- *method signature* is the combination of the method name and the parameter profiles.
- The parameters defined in the method header are known as *formal parameters*.
- When a method is invoked, its formal parameters are replaced by variables or data, which are referred to as *actual parameters*.

Declaring Methods

```
public static int max(int num1, int
  num2) {
    if (num1 > num2)
      return num1;
    else
      return num2;
}
```

Calling Methods

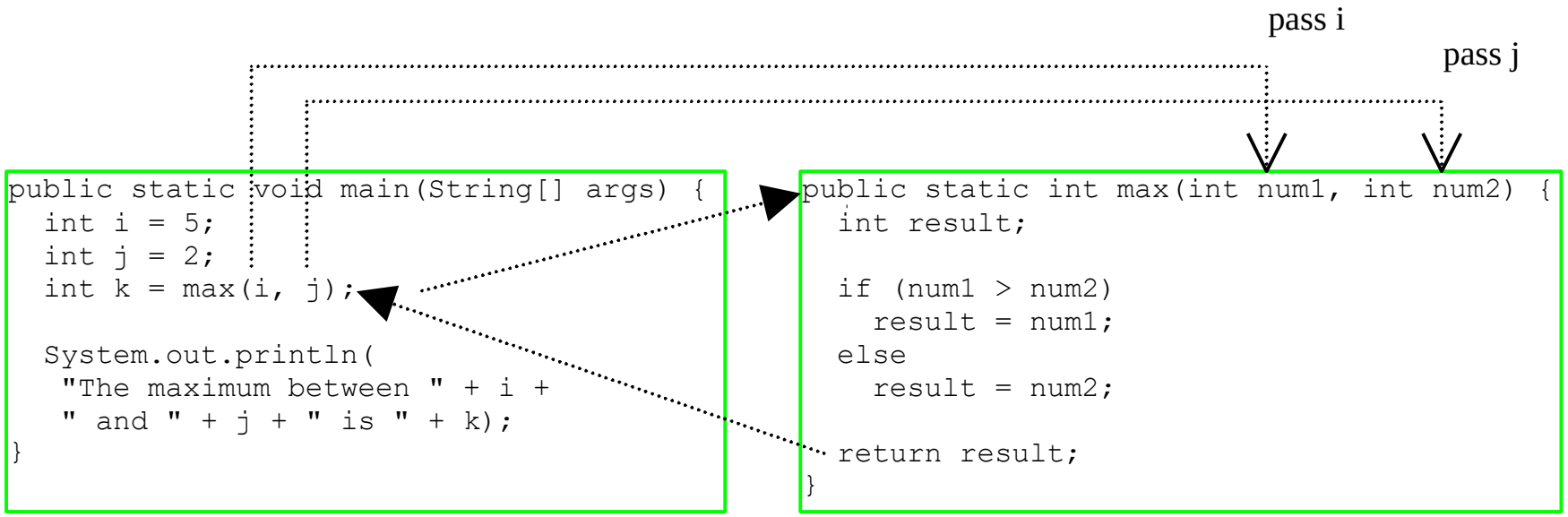
Example 4.1 Testing the `max` method

This program demonstrates calling a method max to return the largest of the `int` values

TestMax

Run

Calling Methods, cont.



CAUTION

A return statement is required for a nonvoid method. The following method is logically correct, but it has a compilation error, because the Java compiler thinks it possible that this method does not return any value.

```
public static int xMethod(int n) {
  if (n > 0) return 1;
  else if (n == 0) return 0;
  else if (n < 0) return -1;
}
```

To fix this problem, delete if (n<0) in the code.

Passing Parameters

```
public static void nPrintln(String message,
  int n) {
   for (int i = 0; i < n; i++)
     System.out.println(message);
}
```

Pass by Value

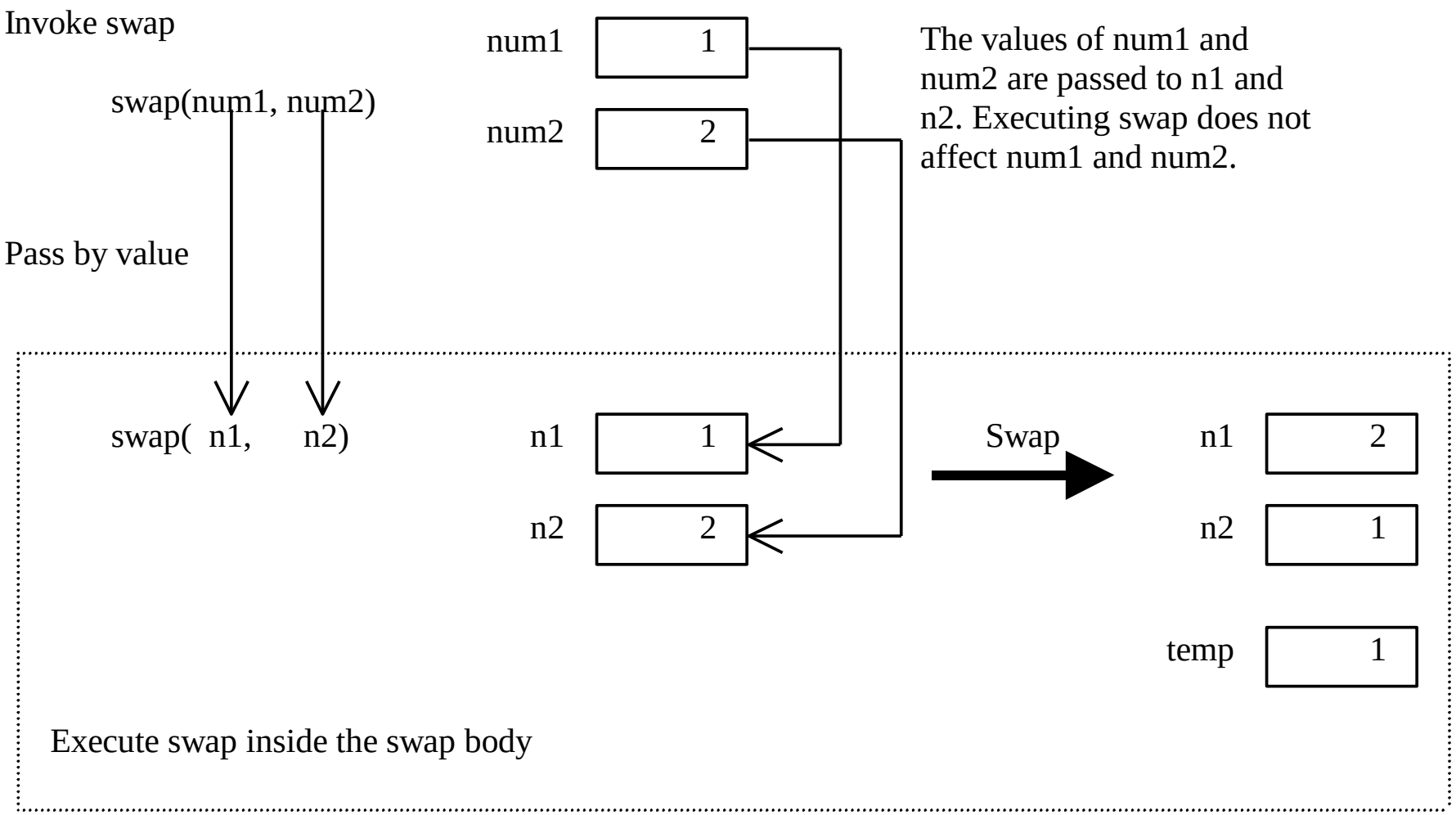
Example 4.2 Testing Pass by value

This program demonstrates passing values to the methods.

TestPassByValue

Run

Pass by Value, cont.



Ambiguous Invocation

Sometimes there may be two or more possible matches for an invocation of a method, but the compiler cannot determine the most specific match. This is referred to as *ambiguous invocation*. Ambiguous invocation is a compilation error.

Ambiguous Invocation

```
public class AmbiguousOverloading {
  public static void main(String[] args) {
    System.out.println(max(1, 2));
  }

  public static double max(int num1, double num2)
{
    if (num1 > num2)
      return num1;
    else
      return num2;
  }

  public static double max(double num1, int num2)
{
    if (num1 > num2)
      return num1;
    else
      return num2;
  }
}
```

Scope of Local Variables

A local variable: a variable defined inside a method.

Scope: the part of the program where the variable can be referenced.

The scope of a local variable starts from its declaration and continues to the end of the block that contains the variable. A local variable must be declared before it can be used.

Scope of Local Variables, cont.

You can declare a local variable with the same name multiple times in different non-nesting blocks in a method, but you cannot declare a local variable twice in nested blocks. Thus, the following code is correct.

Scope of Local Variables, cont.

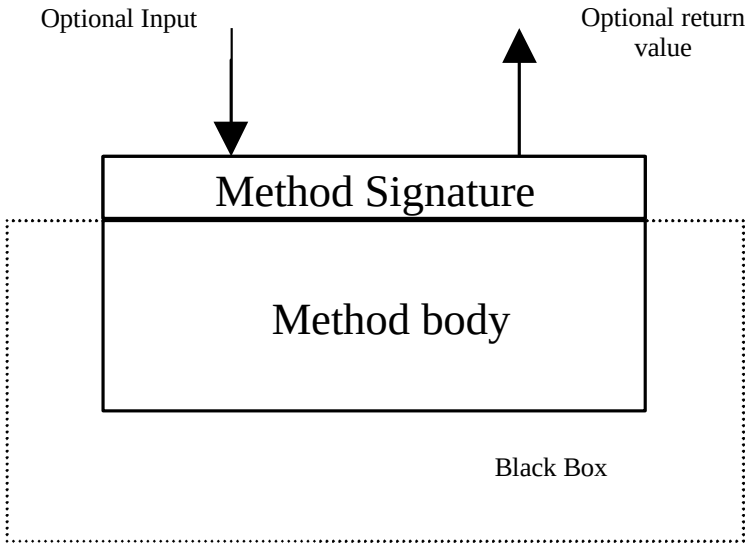
```
// Fine with no errors
public static void correctMethod()
  {
   int x = 1;
   int y = 1;
   // i is declared
   for (int i = 1; i < 10; i++) {
     x += i;
   }
   // i is declared again
   for (int i = 1; i < 10; i++) {
     y += i;
   }
}
```

Scope of Local Variables, cont.

```
// With no errors
public static void incorrectMethod() {
   int x = 1;
   int y = 1;
   for (int i = 1; i < 10; i++) {
     int x = 0;
     x += i;
   }
}
```

Method Abstraction

You can think of the method body as a black box that contains the detailed implementation for the method.



Benefits of Methods

- Write once and reuse it any times.
- Information hiding. Hide the implementation from the user.
- Reduce complexity.

The `Math` Class

- Class constants:
  - `PI`
  - `E`
- Class methods:
  - Trigonometric Methods
  - Exponent Methods
  - Rounding Methods
  - min, max, abs, and random Methods

Trigonometric Methods

- `sin(double a)`
- `cos(double a)`
- `tan(double a)`
- `acos(double a)`
- `asin(double a)`
- `atan(double a)`

Exponent Methods

- `exp(double a)`
  Returns `e` raised to the power of `a`.
- `log(double a)`
  Returns the natural logarithm of `a`.
- `pow(double a, double b)`
  Returns `a` raised to the power of `b`.
- `sqrt(double a)`
  Returns the square root of `a`.

Rounding Methods

- `double ceil(double x)`

  `x rounded up to its nearest integer. This integer is returned as a double value.`

- `double floor(double x)`

  `x is rounded down to its nearest integer. This integer is returned as a double value.`

- `double rint(double x)`

  `x is rounded to its nearest integer. If x is equally close to two integers, the even one is returned as a double.`

- `int round(float x)`

  `Return (int)Math.floor(x+0.5).`

- `long round(double x)`

  `Return (long)Math.floor(x+0.5).`

min, max, abs, and random

- `max(a, b)` and `min(a, b)`

  Returns the maximum or minimum of two parameters.

- `abs(a)`

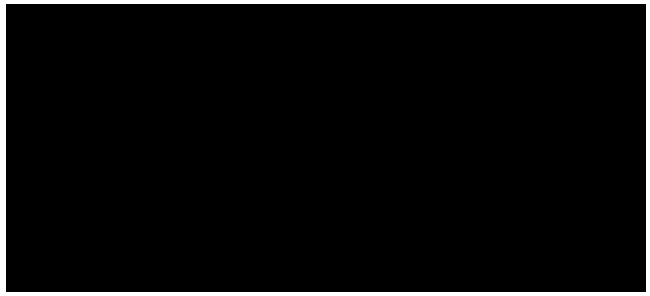
  Returns the absolute value of the parameter.

- `random()`

  Returns a random `double` value
  in the range [0.0, 1.0).

Example 4.4 Computing Mean and Standard Deviation

Generate 10 random numbers and compute the mean and standard deviation



ComputeMeanDeviation

Run

Example 4.5 Obtaining Random Characters

Write the methods for generating random characters. The program uses these methods to generate 175 random characters between ‘!' and ‘~' and displays 25 characters per line. To find out the characters between ‘!' and ‘~', see Appendix B, “The ASCII Character Set.”

RandomCharacter

Run

Fibonacci Numbers

Example 4.8 Computing Finonacci Numbers

0 1 1 2 3 5 8 13 21 34 55 89…

f0 f1

fib(2) = fib(0) + fib(1);

$$fib(0) = 0;$$

$$fib(1) = 1;$$

$$fib(n) = fib(n-2) + fib(n-1);\ n>=2$$

Towers of Hanoi

Example 4.9 Solving the Towers of Hanoi Problem

Solve the towers of Hanoi problem.