



COMPUTER SCIENCE DEPARTMENT FACULTY OF  
ENGINEERING AND TECHNOLOGY

## ADVANCED PROGRAMMING COMP231

Instructor :Murad Njoun  
Office : Masri322

### Chapter 6 Methods

liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoun



## Defining Methods

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

Define a method

```
public static int max(int num1, int num2) {
    int result;

    if (num1 > num2)
        result = num1;
    else
        result = num2;

    return result;
}
```

Invoke a method

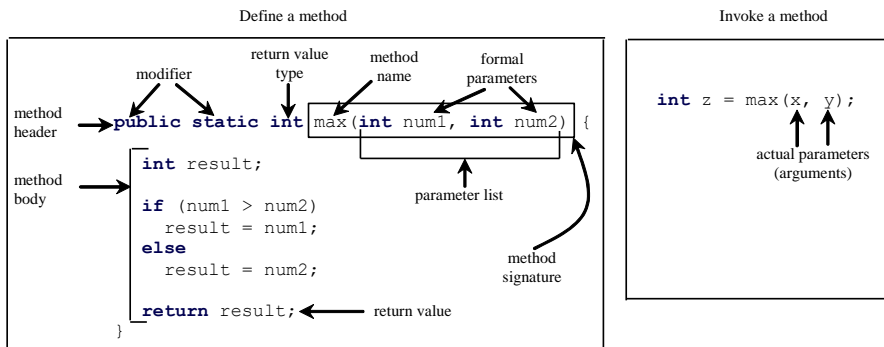
```
int z = max(x, y);
      ↑ ↑
      actual parameters
      (arguments)
```

liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoun



## Defining Methods

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

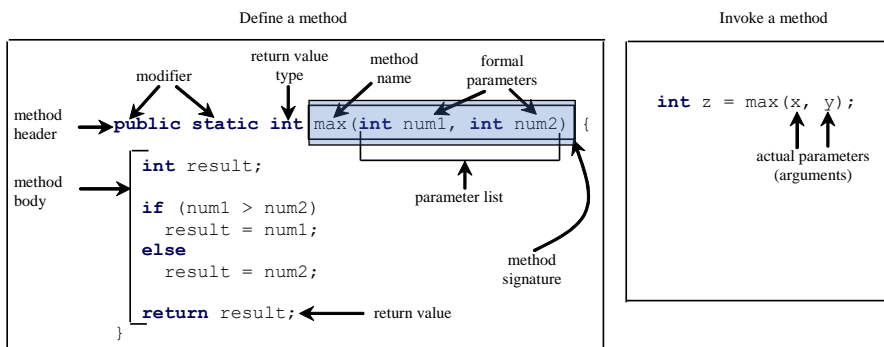


liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoum



## Method Signature

*Method signature* is the combination of the method name and the parameter list.

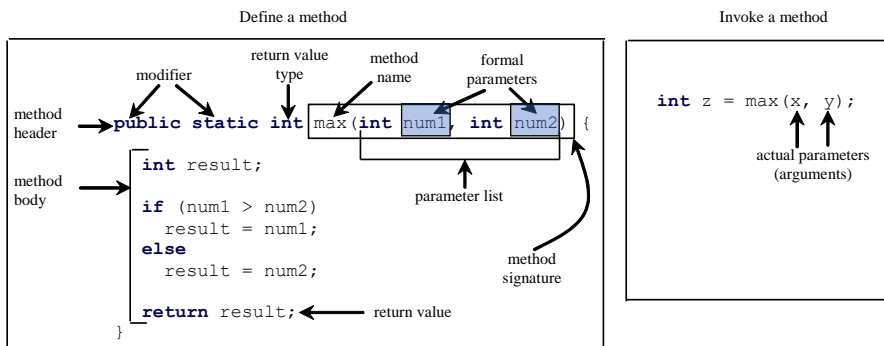


liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoum



## Formal Parameters

The variables defined in the method header are known as *formal parameters*.

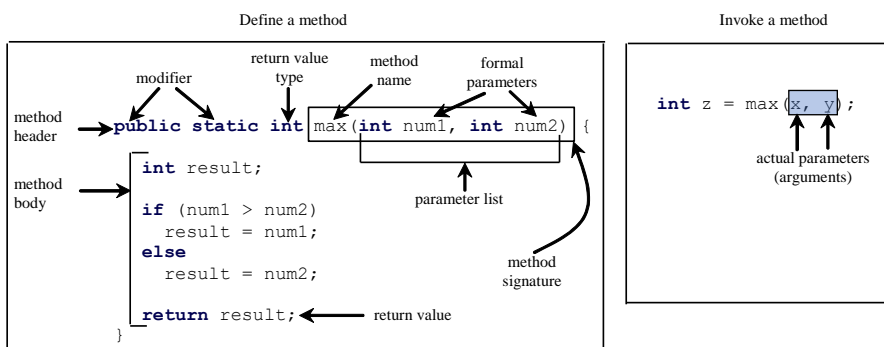


liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoum



## Actual Parameters

When a method is invoked, you pass a value to the parameter. This value is referred to as *actual parameter or argument*.

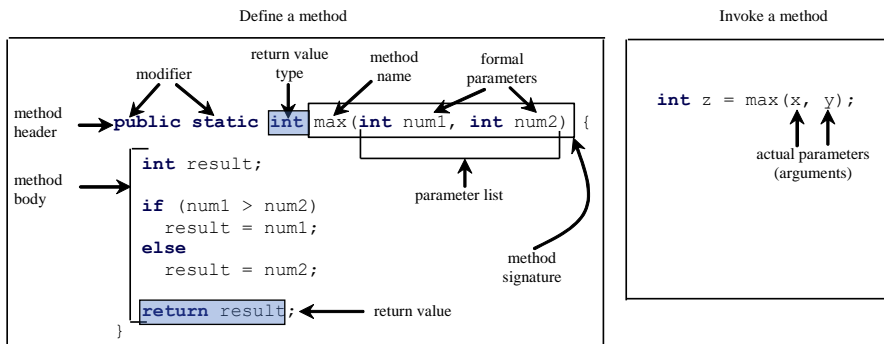


liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoum



## Return Value Type

A method may return a value. The returnValueType is the data type of the value the method returns. If the method does not return a value, the returnValueType is the keyword void. For example, the returnValueType in the main method is void.

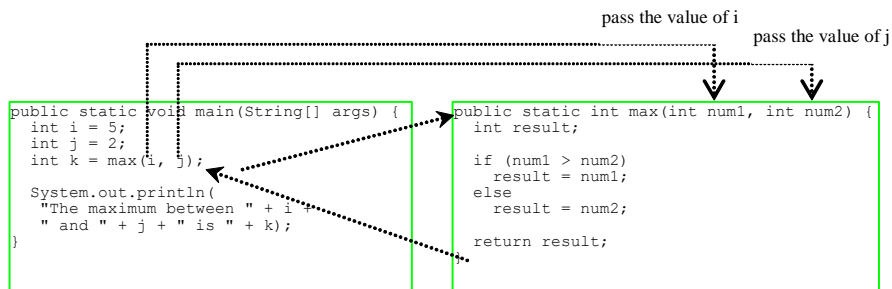


liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoum



animation

## Calling Methods, cont.



liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoum



animation

## Trace Method Invocation

declare variable result

```
public static void main(String[] args) {
    int i = 5;
    int j = 2;
    int k = max(i, j);

    System.out.println(
        "The maximum between " + i +
        " and " + j + " is " + k);
}
```

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



liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoum

animation

## Trace Method Invocation

return max(i, j) and assign the  
return value to k

```
public static void main(String[] args) {
    int i = 5;
    int j = 2;
    int k = max(i, j);

    System.out.println(
        "The maximum between " + i +
        " and " + j + " is " + k);
}
```

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



liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoum

animation

## Trace Method Invocation

Execute the print statement

```
public static void main(String args) {
    int i = 5;
    int j = 2;
    int k = max(i, j);
    System.out.println(
        "The maximum between " + i +
        " and " + j + " is " + k);
}
```

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



liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoum

## CAUTION

A return statement is required for a value-returning method. The method shown below in (a) 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 sign(int n) {
    if (n > 0)
        return 1;
    else if (n == 0)
        return 0;
    else if (n < 0)
        return -1;
}
```

(a)

Should be

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

(b)

To fix this problem, delete *if*( $n < 0$ ) in (a), so that the compiler will see a return statement to be reached regardless of how the if statement is evaluated.



liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoum

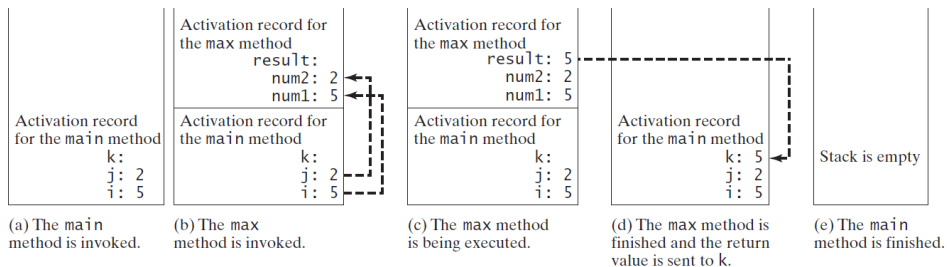
## Reuse Methods from Other Classes

NOTE: One of the benefits of methods is for reuse. The `max` method can be invoked from any class besides `TestMax`. If you create a new class `Test`, you can invoke the `max` method using `ClassName.methodName` (e.g., `TestMax.max`).



liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoum

## Call Stacks



liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoum

animation

## Trace Call Stack

i is declared and initialized

```

public static void main(String[] args) {
    int i = 5;
    int j = 2;
    int k = max(i, j);

    System.out.println(
        "The maximum between " + i +
        " and " + j + " is " + k);
}

```

```

public static int max(int num1, int num2) {
    int result;

    if (num1 > num2)
        result = num1;
    else
        result = num2;

    return result;
}

```

liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoum



animation

## Trace Call Stack

j is declared and initialized

```

public static void main(String[] args) {
    int i = 5;
    int j = 2;
    int k = max(i, j);

    System.out.println(
        "The maximum between " + i +
        " and " + j + " is " + k);
}

```

```

public static int max(int num1, int num2) {
    int result;

    if (num1 > num2)
        result = num1;
    else
        result = num2;

    return result;
}

```

liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoum





animation

## Trace Call Stack

Declare k

```

public static void main(String[] args) {
    int i = 5;
    int j = 2;
    int k = max(i, j);

    System.out.println(
        "The maximum between " + i +
        " and " + j + " is " + k);
}

```

```

public static int max(int num1, int num2) {
    int result;

    if (num1 > num2)
        result = num1;
    else
        result = num2;

    return result;
}

```

in: 5  
 out:  
 k:  
 j: 2  
 i: 5

in:  
 out:



liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoum

animation

## Trace Call Stack

Invoke max(i, j)

```

public static void main(String[] args) {
    int i = 5;
    int j = 2;
    int k = max(i, j);

    System.out.println(
        "The maximum between " + i +
        " and " + j + " is " + k);
}

```

```

public static int max(int num1, int num2) {
    int result;

    if (num1 > num2)
        result = num1;
    else
        result = num2;

    return result;
}

```

in: 5  
 out:  
 k:  
 j: 2  
 i: 5

in:  
 out:



liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoum

animation

## Trace Call Stack

```
public static void main(String[] args) {
    int i = 5;
    int j = 2;
    int k = max(i, j);

    System.out.println(
        "The maximum between " + i +
        " and " + j + " is " + k);
}
```

```
public static int max(int num1, int num2) {
    int result;

    if (num1 > num2)
        result = num1;
    else
        result = num2;

    return result;
}
```

pass the values of i and j to num1  
and num2



liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoum

animation

## Trace Call Stack

```
public static void main(String[] args) {
    int i = 5;
    int j = 2;
    int k = max(i, j);

    System.out.println(
        "The maximum between " + i +
        " and " + j + " is " + k);
}
```

```
public static int max(int num1, int num2) {
    int result;

    if (num1 > num2)
        result = num1;
    else
        result = num2;

    return result;
}
```

Declare result



liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoum

animation

## Trace Call Stack

```
public static void main(String[] args) {
    int i = 5;
    int j = 2;
    int k = max(i, j);

    System.out.println(
        "The maximum between " + i +
        " and " + j + " is " + k);
}
```

```
public static int max(int num1, int num2) {
    int result;

    if (num1 > num2)
        result = num1;
    else
        result = num2;

    return result;
}
```

(num1 &gt; num2) is true



liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoum

animation

## Trace Call Stack

```
public static void main(String[] args) {
    int i = 5;
    int j = 2;
    int k = max(i, j);

    System.out.println(
        "The maximum between " + i +
        " and " + j + " is " + k);
}
```

```
public static int max(int num1, int num2) {
    int result;

    if (num1 > num2)
        result = num1;
    else
        result = num2;

    return result;
}
```

Assign num1 to result



liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoum

animation

## Trace Call Stack

```

public static void main(String[] args) {
    int i = 5;
    int j = 2;
    int k = max(i, j);

    System.out.println(
        "The maximum between " + i +
        " and " + j + " is " + k);
}

```

```

public static int max(int num1, int num2) {
    int result;

    if (num1 > num2)
        result = num1;
    else
        result = num2;

    return result;
}

```

Return result and assign it to k

liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoum



animation

## Trace Call Stack

```

public static void main(String[] args) {
    int i = 5;
    int j = 2;
    int k = max(i, j);

    System.out.println(
        "The maximum between " + i +
        " and " + j + " is " + k);
}

```

```

public static int max(int num1, int num2) {
    int result;

    if (num1 > num2)
        result = num1;
    else
        result = num2;

    return result;
}

```

Execute print statement

liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoum



## Passing Parameters

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

Suppose you invoke the method using  
`nPrintln("Welcome to Java", 5);`

What is the output?

Suppose you invoke the method using  
`nPrintln("Computer Science", 15);`

What is the output?

Can you invoke the method using  
`nPrintln(15, "Computer Science");`

liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoun



## Pass by Value

This program demonstrates passing values  
to the methods.

Before the call, x is 1  
n inside the method is 2  
After the call, x is 1

```
public class Increment {
    public static void main(String[] args) {
        int x = 1;
        System.out.println("Before the call, x is " + x);
        increment(x);
        System.out.println("After the call, x is " + x);
    }

    public static void increment(int n) {
        n++;
        System.out.println("n inside the method is " + n);
    }
}
```

liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoun



## Case Study: Converting Hexadecimals to Decimals

Write a method that converts a hexadecimal number into a decimal number.

ABCD =>

$$A*16^3 + B*16^2 + C*16^1 + D*16^0$$

$$= ((A*16 + B)*16 + C)*16 + D$$

$$= ((10*16 + 11)*16 + 12)*16 + 13 = ?$$



liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoum

## Overloading Methods

```
public class TestMethodOverloading {
    /** Main method */
    public static void main(String[] args) {
        // Invoke the max method with int parameters
        System.out.println("The maximum of 3 and 4 is "
            + max(3, 4));

        // Invoke the max method with the double parameters
        System.out.println("The maximum of 3.0 and 5.4 is "
            + max(3.0, 5.4));

        // Invoke the max method with three double parameters
        System.out.println("The maximum of 3.0, 5.4, and 10.14 is "
            + max(3.0, 5.4, 10.14));
    }

    /** Return the max of two int values */
    public static int max(int num1, int num2) {
        if (num1 > num2)
            return num1;
        else
            return num2;
    }

    /** Find the max of two double values */
    public static double max(double num1, double num2) {
        if (num1 > num2)
            return num1;
        else
            return num2;
    }

    /** Return the max of three double values */
    public static double max(double num1, double num2, double num3) {
        return max(max(num1, num2), num3);
    }
}
```

The maximum of 3 and 4 is 4  
 The maximum of 3.0 and 5.4 is 5.4  
 The maximum of 3.0, 5.4, and 10.14  
 is 10.14



liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoum

## 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 compile error.***



liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoum

## 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;
    }
}
```



liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoum

## 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.



liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoun

## 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.



liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoun



## Scope of Local Variables, cont.

A variable declared in the initial action part of a for loop **header has its scope in the entire loop**. But a variable declared inside a for loop body has its scope limited in the loop body from its declaration and to the end of the block that contains the variable.

```

public static void method1() {
    .
    .
    for (int i = 1; i < 10; i++) {
        .
        .
        int j;
        .
        .
    }
}

```

The scope of i →

The scope of j →

liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoum



## Scope of Local Variables, cont.

It is fine to declare i in two non-nesting blocks

```

public static void method1() {
    int x = 1;
    int y = 1;
    for (int i = 1; i < 10; i++) {
        x += i;
    }
    for (int i = 1; i < 10; i++) {
        y += i;
    }
}

```

It is wrong to declare i in two nesting blocks

```

public static void method2() {
    int i = 1;
    int sum = 0;
    for (int i = 1; i < 10; i++)
        sum += i;
}

```

liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoum



## 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;
    }
}
```

liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoum



## Scope of Local Variables, cont.

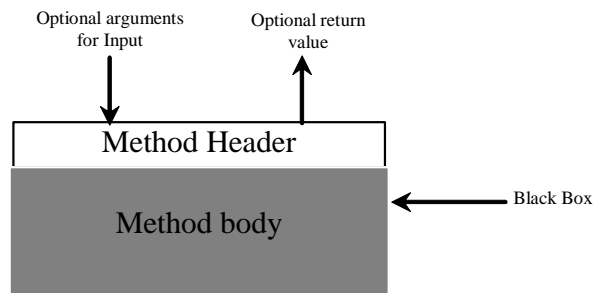
```
// With errors
public static void incorrectMethod() {
    int x = 1;
    int y = 1;
    for (int i = 1; i < 10; i++) {
        int x = 0;
        x += i;
    }
}
```

liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoum



## Method Abstraction

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



liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoum



## Benefits of Methods

- Write a method once and reuse it anywhere.
- Information hiding. Hide the implementation from the user.
- Reduce complexity.

liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoum



## Case Study: Generating Random Characters

Computer programs process numerical data and characters. You have seen many examples that involve numerical data. It is also important to understand characters and how to process them.

As introduced in Section 4.3, each character has a unique Unicode between 0 and FFFF in hexadecimal (65535 in decimal). To generate a random character is to generate a random integer between 0 and 65535 using the following expression: (note that since  $0 \leq \text{Math.random()} < 1.0$ , you have to add 1 to 65535.)

```
(int)(Math.random() * (65535 + 1))
```

liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoum



## Case Study: Generating Random Characters, cont.

Now let us consider how to generate a random lowercase letter. The Unicode for lowercase letters are consecutive integers starting from the Unicode for 'a', then for 'b', 'c', ..., and 'z'. The Unicode for 'a' is

```
(int)'a'
```

So, a random integer between  $(\text{int})'a'$  and  $(\text{int})'z'$  is

```
(int)((int)'a' + Math.random() * ((int)'z' - (int)'a' + 1))
```

liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoum



## Case Study: Generating Random Characters, cont.

Now let us consider how to generate a random lowercase letter. The Unicode for lowercase letters are consecutive integers starting from the Unicode for 'a', then for 'b', 'c', ..., and 'z'. The Unicode for 'a' is

```
(int)'a'
```

So, a random integer between (int)'a' and (int)'z' is

```
(int)((int)'a' + Math.random() * ((int)'z' - (int)'a' + 1))
```



liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoun

## Case Study: Generating Random Characters, cont.

As discussed in Chapter 2, all numeric operators can be applied to the char operands. The char operand is cast into a number if the other operand is a number or a character. So, the preceding expression can be simplified as follows:

```
'a' + Math.random() * ('z' - 'a' + 1)
```

So a random lowercase letter is

```
(char)('a' + Math.random() * ('z' - 'a' + 1))
```



liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoun

## Case Study: Generating Random Characters, cont.

To generalize the foregoing discussion, a random character between any two characters `ch1` and `ch2` with `ch1 < ch2` can be generated as follows:

$$(\text{char})(\text{ch1} + \text{Math.random()} * (\text{ch2} - \text{ch1} + 1))$$


liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoum

## The RandomCharacter Class

```
// RandomCharacter.java: Generate random characters
public class RandomCharacter {
    /** Generate a random character between ch1 and ch2 */
    public static char getRandomCharacter(char ch1, char ch2) {
        return (char)(ch1 + Math.random() * (ch2 - ch1 + 1));
    }

    /** Generate a random lowercase letter */
    public static char getRandomLowerCaseLetter() {
        return getRandomCharacter('a', 'z');
    }

    /** Generate a random uppercase letter */
    public static char getRandomUpperCaseLetter() {
        return getRandomCharacter('A', 'Z');
    }

    /** Generate a random digit character */
    public static char getRandomDigitCharacter() {
        return getRandomCharacter('0', '9');
    }

    /** Generate a random character */
    public static char getRandomCharacter() {
        return getRandomCharacter('\u0000',
            '\uFFFF');
    }
}
```



liang introduction to java programming 11th edition ,2019 , Edit By : Mr.Murad Njoum

## Stepwise Refinement (Optional)

The concept of method abstraction can be applied to the process of developing programs. When writing a large program, you can use the “divide and conquer” strategy, also known as *stepwise refinement*, to decompose it into subproblems. The subproblems can be further decomposed into smaller, more manageable problems.

