King Mongkut's University of Technology Thonburi

Basic of Java Programming

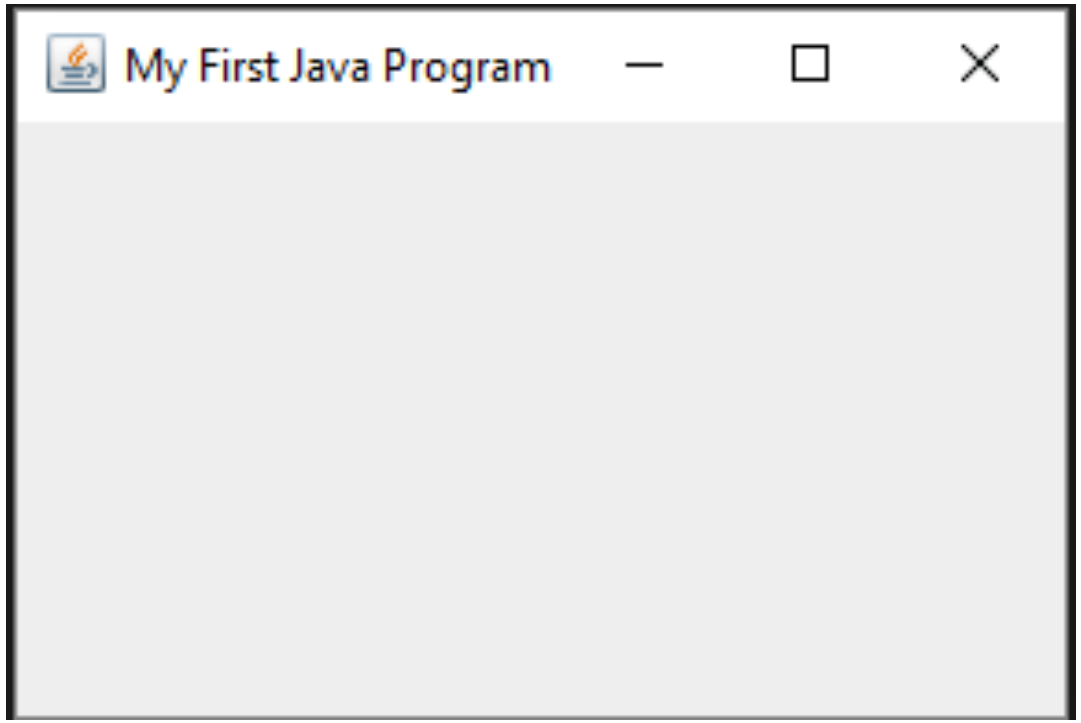
Lecturer: Dr. Thittaporn Ganokratanaa

Java Tutorial

- First look of Java Program
- Comments
- Variables and Data Types
- Type Casting
- Operators
- Strings
- Conditions and If Statements
- Loop
- Break/Continue
- Arrays
- Methods

First look of Java Program

- Let’s build a Java application program displays a window on the screen, as shown in Figure below.



First look of Java Program

❖ Here's the program code:

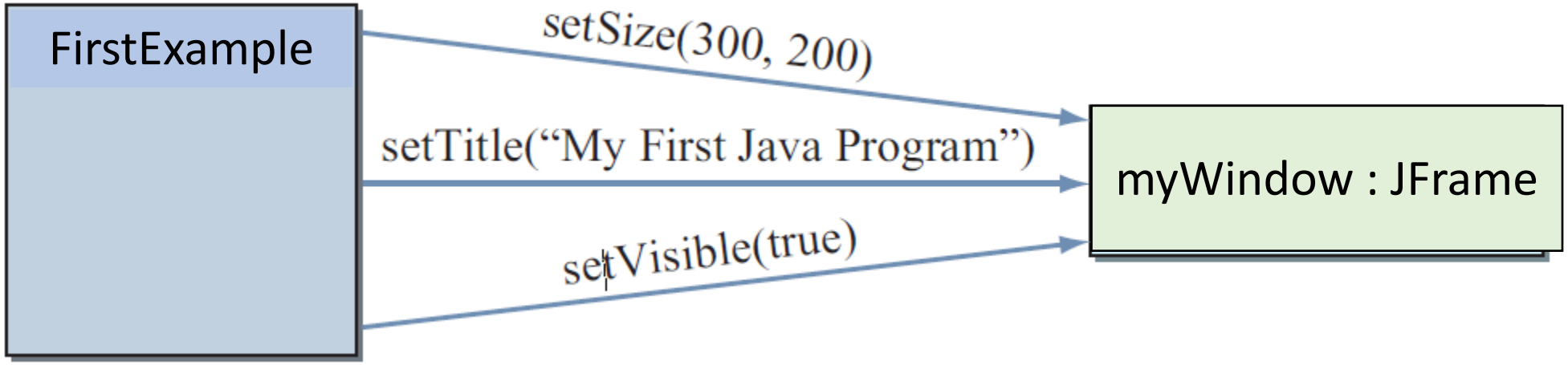
```java
// Import library to generate the frame window
import javax.swing.*;

// Create public calss name FirstExample the same as the filename
public class FirstExample
{
    Run | Debug
    public static void main (String[] args)
    {
        JFrame myWindow;
        myWindow = new JFrame();
        myWindow.setSize(300, 200);
        myWindow.setTitle("My First Java Program");
        myWindow.setVisible(true);
    }
}
```

First look of Java Program

- This program declares one class called “FirstExample”, and the class includes one method called “main”.
- From this “main” method, the “FirstExample” class creates and uses a JFrame in which its object named “myWindow” to send the three messages setSize, setTitle, and setVisible to the object.
- JFrame class is one of many classes that come with the Java system.
- An instance of this JFrame class is used to represent a single window on the computer screen.

First look of Java Program

❖ Program diagram for our first example can be drawn as,

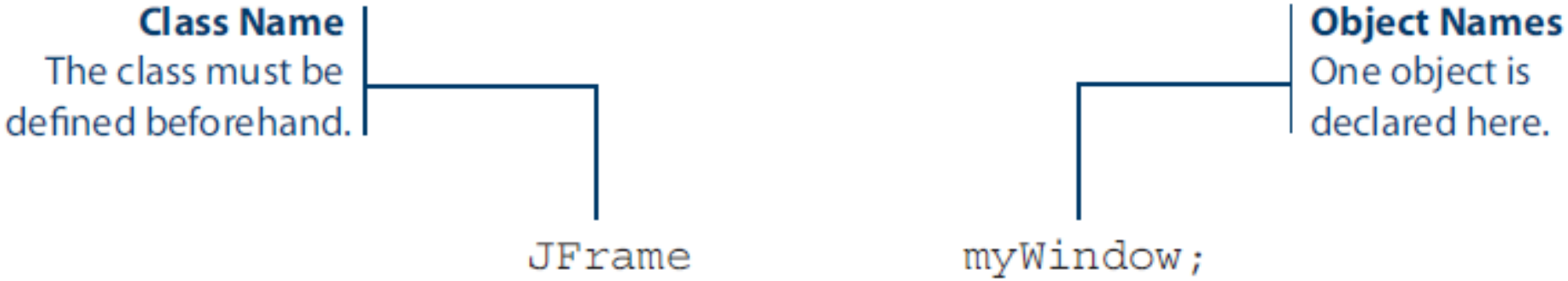


Object Declaration

- Every object we use in a program must be declared. An object declaration designates the name of an object and the class which the object belongs to.
- Syntax is:

```
<class name>      <object names>;
```

- `<object names>` is a sequence of object names separated by commas.
- `<class name>` is the name of a class which these objects belong to.



Object Declaration

- Examples:

```
Account    checking;
Customer   jonh, jack, jill;
```

The first declaration declares an `Account` class with its object named `checking`, and the second declaration declares three `Cutomer` class with its objects, including `jonh`, `jack`, `jill`.

Object Creation

- No objects are actually created by the declaration. An object declaration simply declares the name (**identifier**) that we use to refer to an object. Take a look at the following declaration

```
JFrame      myWindow;
```

This designates that the name `myWindow` is used to refer to an instance of `JFrame`, but the actual object of `JFrame` is not yet created. We create an object by invoking the new *operator*.

- The syntax for new is as follows.

```
<object name> =  new <class name> (<arguments>);
```

  - `<object name>` is the name of a declared object.
  - `<class name>` is the name of the class which the object belongs to.
  - `<arguments>` is a sequence of values passed to the new operation.

Object Creation

❖ Let's match the syntax to the actual statement in the sample program:

Object Name
Name of the object we are creating

Class Name
An instance of this class is created.

```
myWindow = new JFrame ( ) ;
```

Argument
No arguments are used here.

Hints, & Tips, Pitfalls

Instead of writing statements for object declaration and creation separately, we can combine them into one statement. We can write, for example,

```
Student john = new Student();
```

instead of

```
Student john;
john = new Student();
```

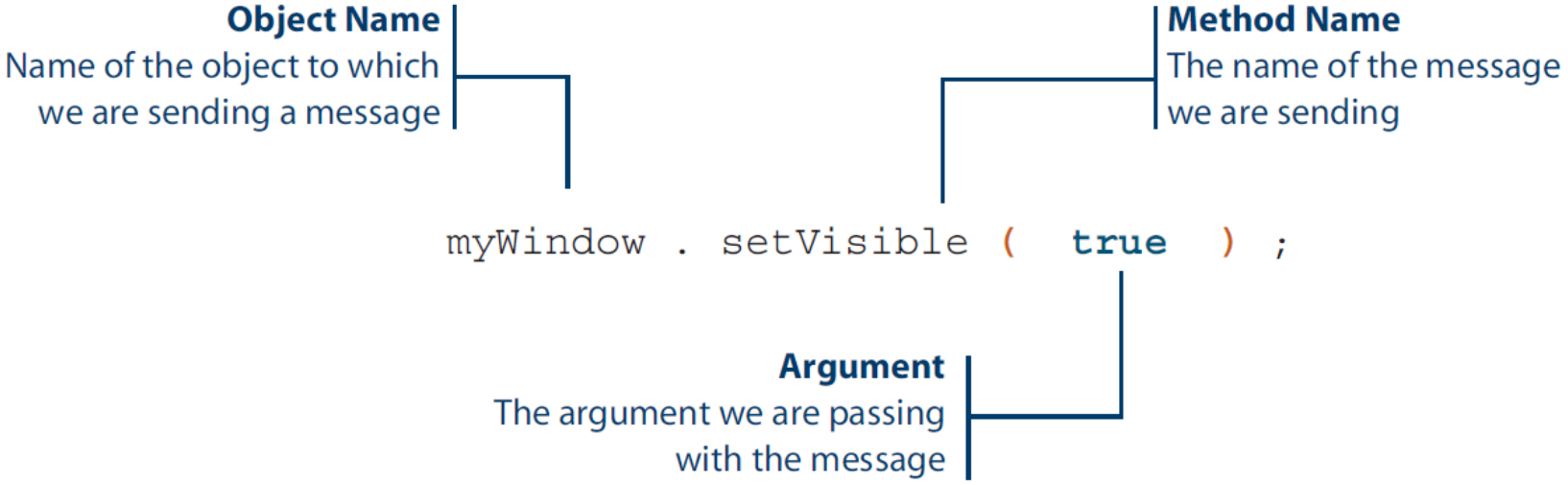
Message Sending

- After the object is created, we can start sending messages to it. The syntax for sending a message to an object is:

```
<object name> . <method name> (<arguments>);
```

where `<object name>` is an object name, `<method name>` is the name of a method of the object (known as function), `<arguments>` is a sequence of values passed to the method.

- Examples



Import

- In Java, classes are grouped into packages, and the Java system comes with numerous packages. To use a class from a package, we refer to the class in our program by using the following format:

```
<package name> . <class name>
```

- Example:

```
javax.swing.JFrame
```

Refer to the class `JFrame` in the `javax.swing` package; the swing package is inside the `javax` package.

Import

```java
// Import library to generate the frame window
import javax.swing.*;

// Create public calss name FirstExample the same as the filename
public class FirstExample
{
    Run | Debug
    public static void main (String[] args)
    {
        JFrame myWindow;
        myWindow = new JFrame();
        myWindow.setSize(300, 200);
        myWindow.setTitle("My First Java Program");
        myWindow.setVisible(true);
    }
}
```

Import **Statement**

The **import** statement allows the program to refer to classes defined in the designated package without using the fully qualified class name.

Class Declaration

- A Java program is composed of one or more classes; some are predefined classes, while others are defined by us.
- In the first sample program, there are two classes, `JFrame` and `FirstExample`.
- The `JFrame` class is one of the standard classes, and the `FirstExample` class is the class we define ourselves.
- Syntax:

```
class <class name> {
    <class member declarations>
}
```

Method Declaration

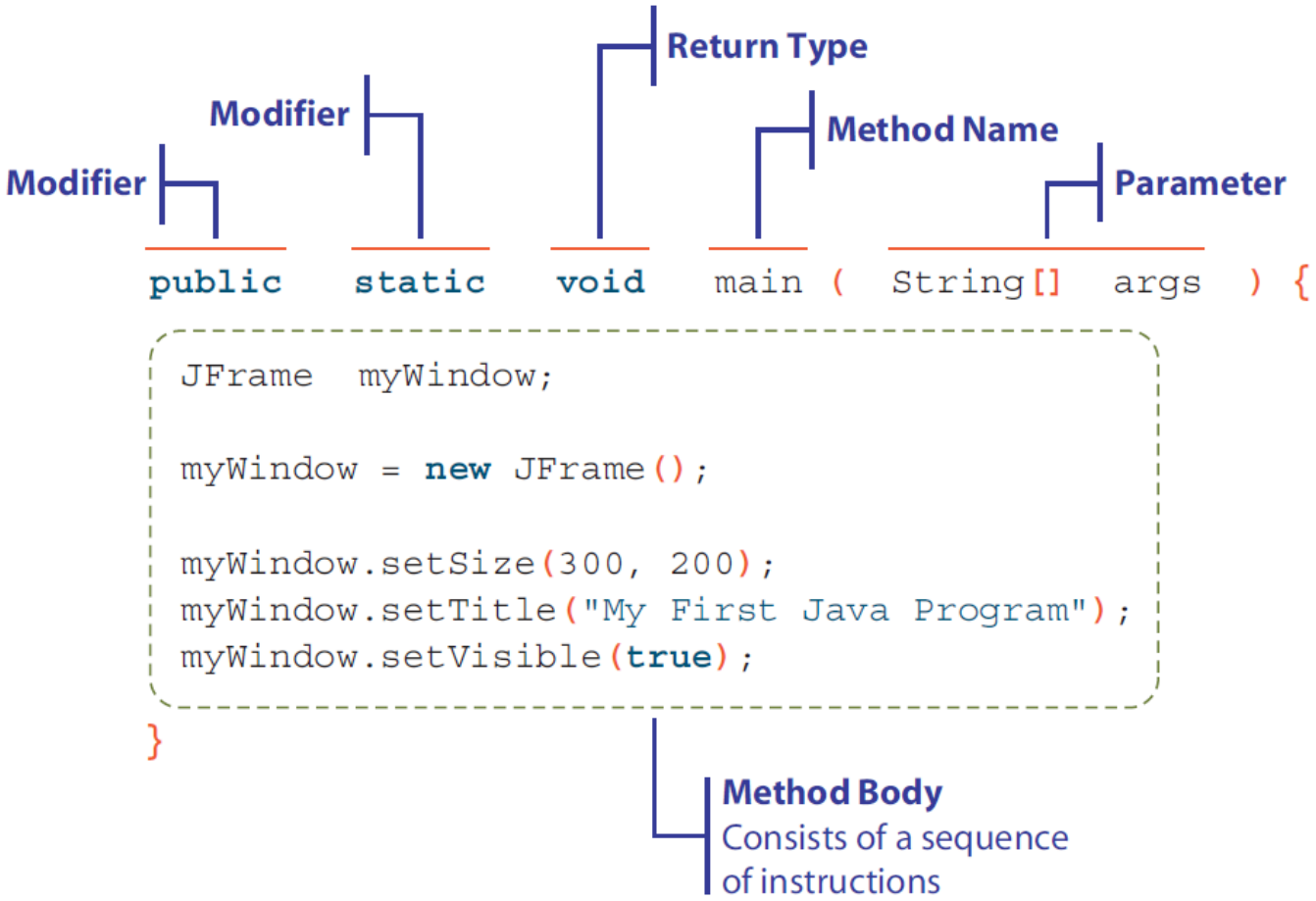
- Syntax:

```
<modifiers> <return type> <method name> ( <parameters> ) {

    <method body>
}
```

  - 〈modifiers〉 is a sequence of terms designating different kinds of methods
  - 〈return type〉 is the type of data value returned by a method,
  - 〈method name〉 is the name of a method
  - 〈parameters〉 is a sequence of values passed to a method
  - 〈method body〉 is a sequence of instructions

Actual Method Declaration

- Actual method declaration of the sample program consists of components as following



Standard naming convention for Java

Category	Convention	Example

Class	Use an uppercase letter for the first letter of the class names. If the name consists of multiple words, the first letter of every word is capitalized.	`Customer`
		`MainWindow`
		`MyInputHandler`

Instance	Use a lowercase letter for the first letter of the object names. If the name consists of multiple words, the first letter of every word (except the first word) is capitalized.	`customer`
		`inputHandler`
		`myFirstApplication`

Constant	(Note: Sample use of a constant will appear in Chap. 4. We include it here for completeness and easy reference later.) Use all uppercase letters. If the constant consists of multiple words, the underscore characters are used to separate the words.	`DEFAULT_RATE`
		`DEG_TO_RAD`
		`CANCEL`

Package	Use all lowercase letters.	`java`
		`game`
		`finance`

A Program Template for Simple Java Programs

Comment
Use a comment to describe the program.

Import Statements
Include a sequence of import statements.

Class Name
Give a descriptive name to the main class.

Method Body
Include a sequence of instructions.

```
class                                  {

  public static void main(String[] args) {

  }

}
```

Java Execution Procedure

Editor

FirstExample.java

```java
// Import library to generate the frame window
import javax.swing.*;

// Create public calss name FirstExample the same as the filename
public class FirstExample
{
    Run | Debug
    public static void main (String[] args)
    {
        JFrame myWindow;
        myWindow = new JFrame();
        myWindow.setSize(300, 200);
        myWindow.setTitle("My First Java Program");
        myWindow.setVisible(true);
    }
}
```

(source file)

Compiler

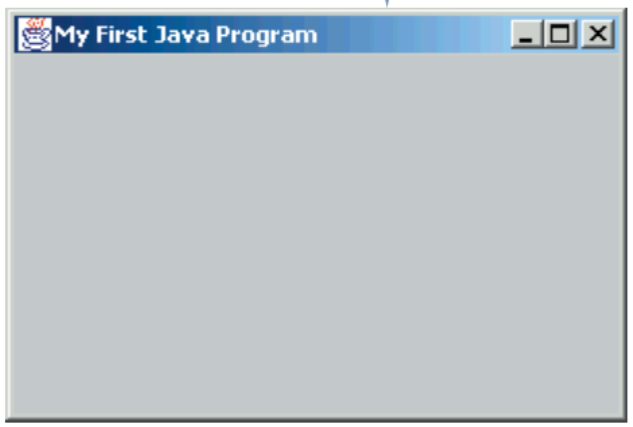
FirstExample.class

```
be 00 03 00 2d 00 1f 08 00 12 07 00 0c
..∫....-........
000010  07 00 15 07 00 13 0a 00 04 00 08
0a 00 03 00 07  ................
000020  0c 00 19 00 1c 0c 00 17 00 14 01
00 04 74 68 69  .............thi
000030  73 01 00 0d 43 6f 6e 73 74 61 6e
74 56 61 6c 75  s...ConstantValu
000040  65 01 00 12 4c 6f 63 61 6c 56 61
72 69 61 62 6c  e...LocalVariabl
000050  65 54 61 62 6c 65 01 00 0e 6d 79
46 69 72 73 74  eTable...myFirst
000060  50 72 6f 67 72 61 6d 01 00 0a 45
78 63 65 70 74  Program...Except
000070  69 6f 6e 73 01 00 0f 4c 69 6e 65
4e 75 6d 62 65  ions...LineNumbe
000080  72 54 61 62 6c 65 01 00 0a 53 6f
75 72 63 65 46   rTable...Source
```

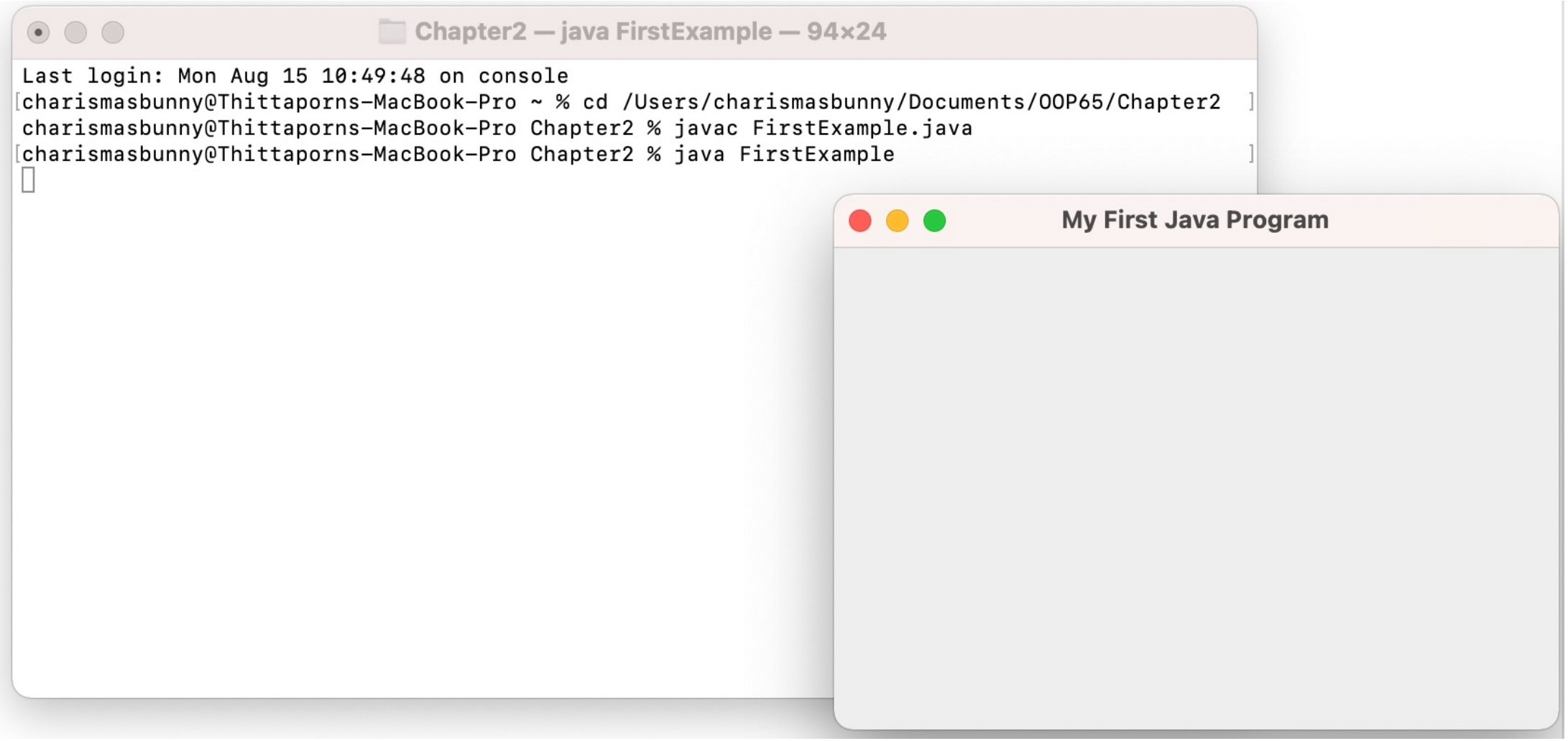
(bytecode file)

Interpreter

Running Program



Java Execution Procedure (cont.)



Comments

- Single-line comments start with two forward slashes (//).
- Multi-line comments start with /* and ends with */.
- Any text between // and the end of the line is ignored by Java (will not be executed).
- Any text between /* and */ will be ignored by Java.

Comments

❖ Example:

```java
// Create class CommentsTest
public class CommentsTest
{
    public static void main(String[] args)
    {
        // This is a single-line comment
        System.out.println("Hello World!");
        /* This is multi-line comment. The code below will print the words Hello World
        to the screen, and it is amazing */
        System.out.println("Hello World!");
    }
}
```

Variables and Data Types

- Variables are containers for storing data values. There are different types of variable:
  - String – stores text, such as “Hello”. String values are surrounded by double quotes
  - char – stores single characters, such as ‘a’ or ‘b’. char values are surrounded by single quotes
  - int – stores integers (whole numbers), without decimals, such as 123 or -123
  - float – stores floating point numbers, with decimals, such as 19.19 or -19.99
  - boolean – stores values with two states: true or false

Variables Examples

```java
// Create public calss name VariablesTest the same as the filename
public class VariablesTest
{
    Run | Debug
    public static void main (String[] args)
    {
        // String
        String name = "Jaidee";
        // char
        char grade = 'A';
        // int
        int classroom = 215;
        // float
        float score = 90.9f;
        // boolean
        boolean checkStatus = true;

        // Print
        System.out.println("Name : " + name);
        System.out.println("Grade : " + grade);
        System.out.println("Classroom : " + classroom);
        System.out.println("Score : " + score);
        System.out.println("Check : " + checkStatus);
    }
}
```

❖ Syntax:

type variable = value;

Command Prompt

```
D:\Lessons\JavaOOP\Lesson2>java VariablesTest.java
Name : Jaidee
Grade : A
Classroom : 215
Score : 90.9
Check : true
```

Variables and Data Types

- Data types are divided into two groups:
  - Primitive data types – includes *byte*, *short*, *int*, *long*, *float*, *double*, *Boolean*, and *char.*
  - Non-primitive data types – such as *String*, *Arrays*, and *Classes.*

Variables and Data Types

❖ Primitive Data Types

Data Type	Size	Description
byte	1 byte	Stores whole numbers from -128 to 127
short	2 bytes	Stores whole numbers from -32,768 to 32,767
int	4 bytes	Stores whole numbers from -2,147,483,648 to 2,147,483,647
long	8 bytes	Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4 bytes	Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits
double	8 bytes	Stores fractional numbers. Sufficient for storing 15 decimal digits
boolean	1 bit	Stores true or false values
char	2 bytes	Stores a single character/letter or ASCII values

Data Types Examples

```java
// Create public calss name DataTypeTest the same as the filename
public class DataTypeTest
{
    Run | Debug
    public static void main (String[] args)
    {
        // Byte (-128 to 127)
        byte a = 100;
        // Short (-32768 to 32767)
        short b = 5000;
        // Int (-2147483648 to 2147483647)
        int c = 100000; // or 10e5;
        // Long (-9223372036854775808 to 9223372036854775807)
        long d = 15000000000L;
        // Float (3.4e-038 to 3.4e+038)
        float e = 5.75f;
        // Double (1.7e-308 to 1.7e+308)
        double f = 19.99d;
        // Booleans
        boolean isJavaFun = true;
        // Char
        char g = 'B';
        // String
        String h = "Run ...";

        // Print
        System.out.println("Byte : " + a);
        System.out.println("Short : " + b);
        System.out.println("Int : " + c);
        System.out.println("Long : " + d);
        System.out.println("Float : " + e);
        System.out.println("Double : " + f);
        System.out.println("Booleans : " + isJavaFun);
        System.out.println("Char : " + g);
        System.out.println("String : " + h);

    }
}
```

Command Prompt

```
D:\Lessons\JavaOOP\Lesson2>java DataTypeTest.java
Byte : 100
Short : 5000
Int : 100000
Long : 15000000000
Float : 5.75
Double : 19.99
Booleans : true
Char : B
String : Run ...
```

Exercise

Add the correct data type for the following variables:

```
myNum = 9;
myFloatNum = 8.99f;
myLetter = 'A';
myBool = false;
myText = "Hello World";
```

Type Casting

- Type casting is when you assign a value of one primitive data type to another type.
- In Java, there are two types of casting:
  - **Widening Casting** (automatically) – converting a smaller type to a larger type size

    byte -> *short* -> *char* -> *int* -> *long* -> *float* -> *double*
  - **Narrowing Casting** (manually) – converting a larger type to a smaller size type

    double -> *float* -> *long* -> *int* -> *char* -> *short* -> *byte*

Widening and Narrowing Casting Example

```java
// Create public calss name CastingTest the same as the filename
public class CastingTest
{
    Run | Debug
    public static void main (String[] args)
    {
        //========================= Widening Casting ====================
        // Create an integer number
        int a = 9;
        // Convert int to double data type, this convertion is automatic
        double b = a;

        //========================= Narrowing Casting ===================
        // Create a double number
        double c = 9.8;
        // Convert doulbe to int data type, we need to do manual casting
        int d = (int) c;

        // Print value
        System.out.println("Integer : " + a);
        System.out.println("After converting to double : " + b);
        System.out.println("Double : " + c);
        System.out.println("After converting to integer : " + d);
    }
}
```

Command Prompt

```
D:\Lessons\JavaOOP\Lesson2>java CastingTest.java
Integer : 9
After converting to double : 9.0
Double : 9.8
After converting to integer : 9
```

Operators

- Operators are used to perform operations on variables and values.
- Java divides the operators into the following groups:
  - Arithmetic operators
  - Assignment and Bitwise operators
  - Comparison operators
  - Logical operators

Operators

- Arithmetic operators are used to perform common mathematical operations.

Operator	Name	Description	Example
+	Addition	Adds together two values	x + y
-	Subtraction	Subtracts one value from another	x - y
*	Multiplication	Multiplies two values	x * y
/	Division	Divides one value by another	x / y
%	Modulus	Returns the division remainder	x % y
++	Increment	Increases the value of a variable by 1	++x
--	Decrement	Decreases the value of a variable by 1	--x

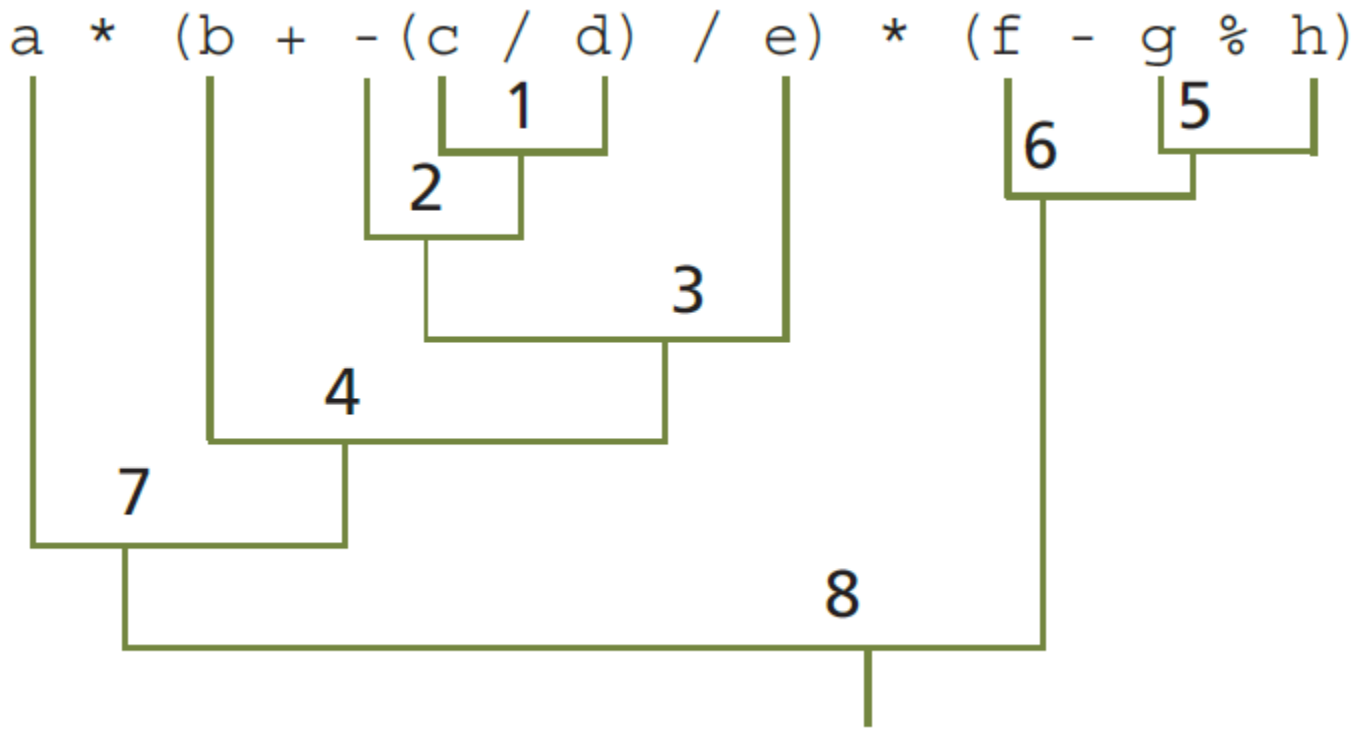
Operators

❖ Precedence rules for arithmetic operators and parentheses.

Order	Group	Operator	Rule
High	Subexpression	( )	Subexpressions are evaluated first. If parentheses are nested, the innermost subexpression is evaluated first. If two or more pairs of parentheses are on the same level, then they are evaluated from left to right.
↕	Unary operator	-, +	Unary minuses and pluses are evaluated second.
↕	Multiplicative operator	*, /, %	Multiplicative operators are evaluated third. If two or more multiplicative operators are in an expression, then they are evaluated from left to right.
Low	Additive operator	+, -	Additive operators are evaluated last. If two or more additive operators are in an expression, then they are evaluated from left to right.

Operators

❖ Examples



Operators

- Assignment and Bitwise operators are used to assign values to variables.

	Operator	Example	Same As
	=	x = 5	x = 5
	+=	x += 3	x = x + 3
	-=	x -= 3	x = x - 3
	*=	x *= 3	x = x * 3
	/=	x /= 3	x = x / 3
	%=	x %= 3	x = x % 3
Bitwise AND(&)	&=	x &= 3	x = x & 3
Bitwise OR(|)	|=	x |= 3	x = x | 3
Bitwise XOR(^)	^=	x ^= 3	x = x ^ 3
Bit Right Shift	>>=	x >>= 3	x = x >> 3
Bit Left Shift	<<=	x <<= 3	x = x << 3

Operators

- Comparison operators are used to compare two values.

Operator	Name	Example
==	Equal to	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Operators

- Logical operators are used to determine the logic between variables or values.

Operator	Name	Description	Example
&&	Logical and	Returns true if both statements are true	x < 5 && x < 10
||	Logical or	Returns true if one of the statements is true	x < 5 || x < 4
!	Logical not	Reverse the result, returns false if the result is true	!(x < 5 && x < 10)

Example of Operators

```java
// Create public calss name OperatorsTest the same as the filename
public class OperatorsTest
{
    Run | Debug
    public static void main (String[] args)
    {
        // Initial some value
        int a = 9;
        int b = 9;

        //====================== Arithmetic ======================
        // Addition
        int result_add = a + b;
        System.out.println("Addition Result = " + result_add);
        // Increment result_add (result_add = result_add + 1)
        ++result_add;
        System.out.println("Increment Result of addition = " + result_add);
        // Modulus
        double result_mod = result_add % b;
        System.out.println("Modulus Result = " + result_mod);

        //================ Assignment and Bitwise ================
        // Add 5 to variable a
        a += 5;
        System.out.println("Result of variable a = " + a);
        // Example: 2 in binary is 10, 1 in binary is 01, 10 | 01 = 11 = 3
        System.out.println("Bitwise of 2 | 1 = " + (2 | 1));

        //====================== Comparison ======================
        boolean compare_ab = (a == b);
        System.out.println("Compare a and b = " + compare_ab);

        //======================== Logical ========================
        boolean logical_ab = (a<5 && b>8);
        System.out.println("Logical a and b = " + logical_ab);
    }
}
```

Command Prompt

```
D:\Lessons\JavaOOP\Lesson2>java OperatorsTest.java
Addition Result = 18
Increment Result of addition = 19
Modulus Result = 1.0
Result of variable a = 14
Bitwise of 2 | 1 = 3
Compare a and b = false
Logical a and b = false
```

Strings

- Strings are used for storing text. A String variable contains a collection of characters surrounded by double quotes.
- A String in Java is actually an object (*known as class*), which contain methods (*known as internal function*) that can perform certain operations on strings.
- Examples: Find length of string, convert string to uppercase or lowercase, find the index of character or word inside string, and concatenate strings.

```
String text;
text = "Espresso";
```

0	1	2	3	4	5	6	7
E	s	p	r	e	s	s	o

Example of Strings

```java
// Create public calss name StringsTest the same as the filename
public class StringsTest
{
    Run | Debug
    public static void main (String[] args)
    {
        // Initial strings
        String questions = "Sabay Dee Mai?";
        String answers = "Sabay Dee Kha!";

        // Find length of questions, count all characters, space,..
        int k = questions.length();
        System.out.println("Total length of questions = " + k);

        // Convert questions to uppercase and lowercase
        String upQuestions = questions.toUpperCase();
        String lowQuestions = questions.toLowerCase();
        System.out.println("Uppercase of questions = " + upQuestions);
        System.out.println("Lowercase of questions = " + lowQuestions);

        // Find index of character 'a', it counts from left to right,
        // Start from 0, So the first character 'a' is at index 1
        int index_a = questions.indexOf('a');
        System.out.println("index of a = " + index_a);

        // Concatenation strings
        String combines = questions.concat(answers);
        System.out.println("Concatenate strings = " + combines);

        // Special case, we want to put quotes or enter in String
        String example1 = "\'a\' is a char.";
        String example2 = "Are you hungry? \nYes, I am.";
        System.out.println("Add quotes in string : " + example1);
        System.out.println("Add enter in string : \n" + example2);
    }
}
```

Command Prompt

```
D:\Lessons\JavaOOP\Lesson2>java StringsTest.java
Total length of questions = 14
Uppercase of questions = SABAY DEE MAI?
Lowercase of questions = sabay dee mai?
index of a = 1
Concatenate strings = Sabay Dee Mai?Sabay Dee Kha!
Add quotes in string : 'a' is a char.
Add enter in string :
Are you hungry?
Yes, I am.
```

Strings

❖ Special Characters

Escape character	Result	Description
\'	'	Single quote
\"	"	Double quote
\\	\	Backslash

Code	Result
\n	New Line
\r	Carriage Return
\t	Tab
\b	Backspace
\f	Form Feed

Conditions and If Statements

- Java has the following conditional statements:
  - Use if to specify a block of code to be executed, if a specified condition is true
  - Use else to specify a block of code to be executed, if the same condition is false
  - Use else if to specify a new condition to test, if the first condition is false
  - Use switch to specify many alternative blocks of code to be executed

Conditions and If Statements

❖ Syntax of if:

```
if (condition1) {
  // block of code to be executed if condition1 is true
} else if (condition2) {
  // block of code to be executed if the condition1 is false and condition2 is true
} else {
  // block of code to be executed if the condition1 is false and condition2 is false
}
```

Conditions and If Statements

❖ Syntax of switch:

```
switch(expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```

Example of Conditions and If Statements

```java
// Import a library to allow you to input information by keyboard
import java.util.Scanner;

// Create public calss name ConditionTest the same as the filename
public class ConditionTest
{
    public static void main (String[] args)
    {
        // Create a Scanner to obtain input from the command window
        Scanner input = new Scanner(System.in);

        System.out.println("Enter current hour (24h) : ");
        int hour = input.nextInt(); // read an int value from the user

        if (hour < 12)
        {
        System.out.println("Good morning.");
        }
        else if (hour < 16)
        {
        System.out.println("Good day.");
        }
        else
        {
        System.out.println("Good evening.");
        }
    }
}
```

Command Prompt

```
D:\Lessons\JavaOOP\Lesson2>java ConditionTest.java
Enter current hour (24h) :
11
Good morning.
```

Example of switch

```java
// Import a library to allow you to input information by keyboard
import java.util.Scanner;

// Create public calss name SwitchTest the same as the filename
public class SwitchTest
{
    Run | Debug
    public static void main (String[] args)
    {
        // Create a Scanner to obtain input from the command window
        Scanner input = new Scanner(System.in);

        System.out.println("Enter day (1,2,3,...,7) : ");
        int day = input.nextInt(); // read an int value from the user
        switch (day)
        {
            case 1:
                System.out.println("Monday");
                break;
            case 2:
                System.out.println("Tuesday");
                break;
            case 3:
                System.out.println("Wednesday");
                break;
            case 4:
                System.out.println("Thursday");
                break;
            case 5:
                System.out.println("Friday");
                break;
            case 6:
                System.out.println("Saturday");
                break;
            case 7:
                System.out.println("Sunday");
                break;
        }
    }
}
```

Command Prompt

```
D:\Lessons\JavaOOP\Lesson2>java SwitchTest.java
Enter day (1,2,3,...,7) :
1
Monday
```

Loop

- Loops can execute a block of code as long as a specified condition is reached. There are three loops:
  - while loop
  - do/while loop
  - for loop
- Syntax of while:

```
while (condition) {
  // code block to be executed
}
```

Example of while loop

```java
// Create public calss name WhileTest the same as the filename
public class WhileTest
{
    Run | Debug
    public static void main (String[] args)
    {
        int i = 0;
        // Loop will stop before i equals to 5
        while (i < 5)
        {
            System.out.println(i);
            // Increase i by 1 (i = i + 1)
            i++;
        }
    }
}
```

Command Prompt

```
D:\Lessons\JavaOOP\Lesson2>java WhileTest.java
0
1
2
3
4
```

do/while Loop

- The do/while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.
- Syntax of do/while:

```
do {
  // code block to be executed
}
while (condition);
```

Example of do/while loop

```java
// Create public calss name DoWhileTest the same as the filename
public class DoWhileTest
{
    Run | Debug
    public static void main (String[] args)
    {
        int i = 0;
        // execute the code block inside do, then check condition in while
        do
        {
            System.out.println(i);
            i++;
        }
        while (i < 5);
    }
}
```

Command Prompt

```
D:\Lessons\JavaOOP\Lesson2>java DoWhileTest.java
0
1
2
3
4
```

for Loop

- The for loop is used when you know exactly how many times you want to loop through a block of code.
- Syntax of for:

```
for (statement 1; statement 2; statement 3) {
  // code block to be executed
}
```

- **Statement 1** is executed (one time) before the execution of the code block.
- **Statement 2** defines the condition for executing the code block.
- **Statement 3** is executed (every time) after the code block has been executed.

Example of for loop

```java
// Create public calss name ForTest the same as the filename
public class ForTest
{
    Run | Debug
    public static void main (String[] args)
    {
        // Statement 1 sets a variable before the loop start
        // int i = 0
        // Statement 2 define the condition for the loop to run
        // i < 5
        // Statment 3 increase a value i by 1
        // i++
        // If we want to increase a value i by 2, we just put i = i + 2
        for (int i = 0; i < 5; i++)
        {
            System.out.println(i);
        }
    }
}
```

Command Prompt

```
D:\Lessons\JavaOOP\Lesson2>java ForTest.java
0
1
2
3
4
```

for-each Loop

- There is also a “for-each” loop, which is used exclusively to loop through elements in an array.
- Syntax :

```
for (type variableName : arrayName) {
  // code block to be executed
}
```

Example of for-each loop

```java
// Create public calss name ForEachTest the same as the filename
public class ForEachTest
{
    Run | Debug
    public static void main (String[] args)
    {
        // Create an array of string
        String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
        // Loop element inside string cars
        for (String i : cars)
        {
            System.out.println(i);
        }
    }
}
```

Command Prompt

```
D:\Lessons\JavaOOP\Lesson2>java ForEachTest.java
Volvo
BMW
Ford
Mazda
```

Exercise

Use a `for` loop to print "Yes" 5 times.

```
    (int i = 0; i < 5;    ) {
  System.out.println(      );
}
```

Break/Continue

- Break statement is used to jump out of a loop.
- Continue statement is used to breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

Example of break and continue

```java
// Create public calss name BreakContinueTest the same as the filename
public class BreakContinueTest
{
    Run | Debug
    public static void main (String[] args)
    {
        // Using break in while loop
        System.out.println("Break Example");
        int i = 0;
        while (i < 10)
        {
            System.out.println(i);
            i++;
            // Break code if condition i == 4
            if (i == 4)
            {
                break;
            }
        }
        // Using break in while loop
        System.out.println("Continue Example");
        int j = 0;
        while (j < 10)
        {
            // Skip code if condition j == 4
            if (j == 4)
            {
                j++;
                continue;
            }
            System.out.println(j);
            j++;
        }
    }
}
```

Command Prompt

```
D:\Lessons\JavaOOP\Lesson2>java BreakContinueTest.java
Break Example
0
1
2
3
Continue Example
0
1
2
3
5
6
7
8
9
```

Exercise

Stop the loop if `i` is 5.

```
for (int i = 0; i < 10; i++) {
  if (i == 5) {
      ;
  }
  System.out.println(i);
}
```

Array

- Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.
- To declare an array, define the variable type with square brackets ([]).

40	55	63	17	22	68	89	97	89
0	1	2	3	4	5	6	7	8

<- **Array Indices**

Array Length = 9
First Index = 0
Last Index = 8

Example of Array

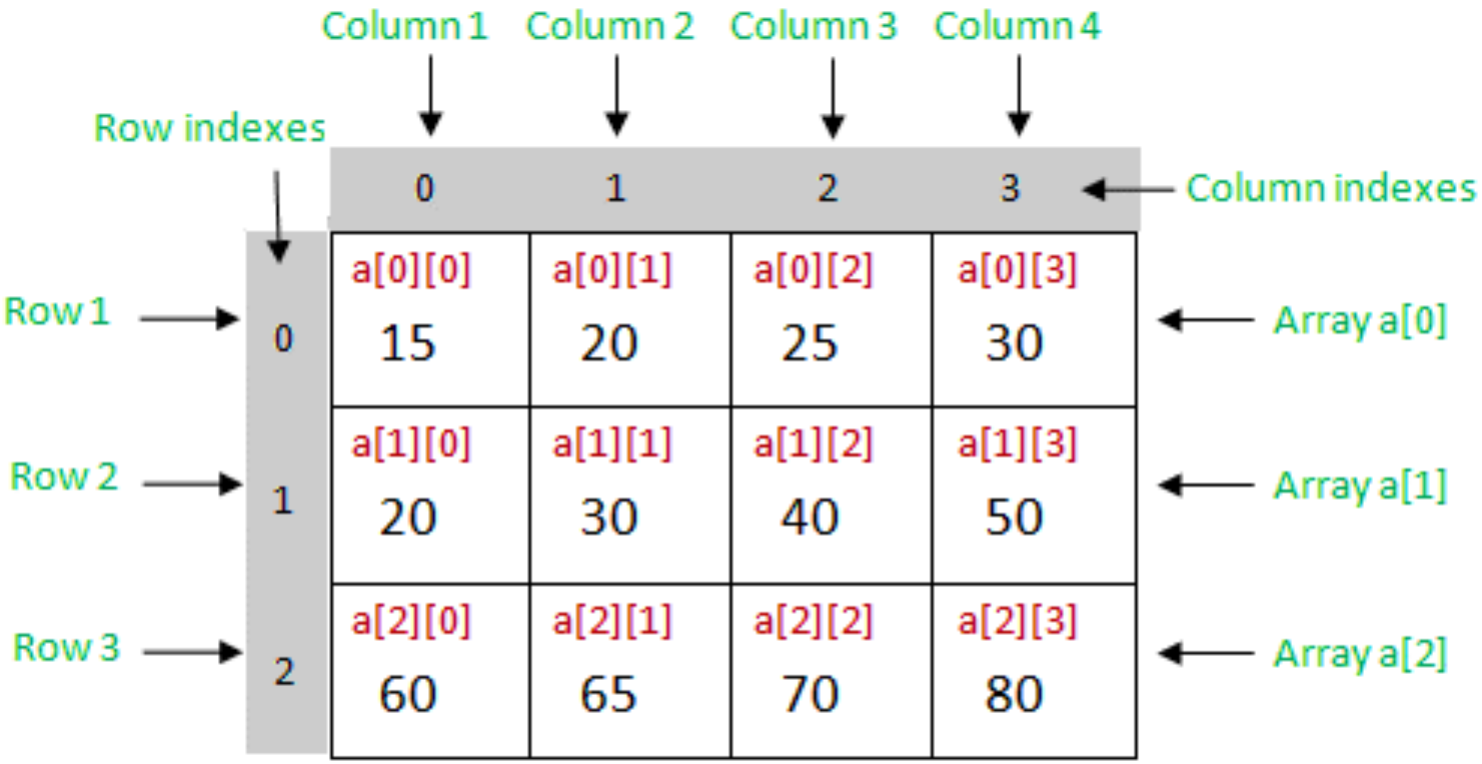
```java
// Create public calss name ArrayTest the same as the filename
public class ArrayTest
{
    Run | Debug
    public static void main (String[] args)
    {
        // Create an array of string
        String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
        // Print index 0 of cars array
        System.out.println("Index 0 of cars array is " + cars[0]);

        // Create an array of integers
        int[] age = {10,20,30,40};
        // Print index 0 of age array
        System.out.println("Index 0 of age array is " + age[0]);

        // Change an array element
        cars[0] = "Honda";
        System.out.println("Change element 0 of cars array to " + cars[0])

        // Find the length of array
        System.out.println("Length of cars array is " + cars.length);

        // Loop array according to index and length
        for (int i = 0; i < cars.length; i++)
        {
            System.out.println("Element " + i + " is " + cars[i]);
        }
    }
}
```

Command Prompt

```
D:\Lessons\JavaOOP\Lesson2>java ArrayTest.java
Index 0 of cars array is Volvo
Index 0 of age array is 10
Change element 0 of cars array to Honda
Length of cars array is 4
Element 0 is Honda
Element 1 is BMW
Element 2 is Ford
Element 3 is Mazda
```

Muti-Array

- A multidimensional array is an array containing one or more arrays.



Example of Multi-Array

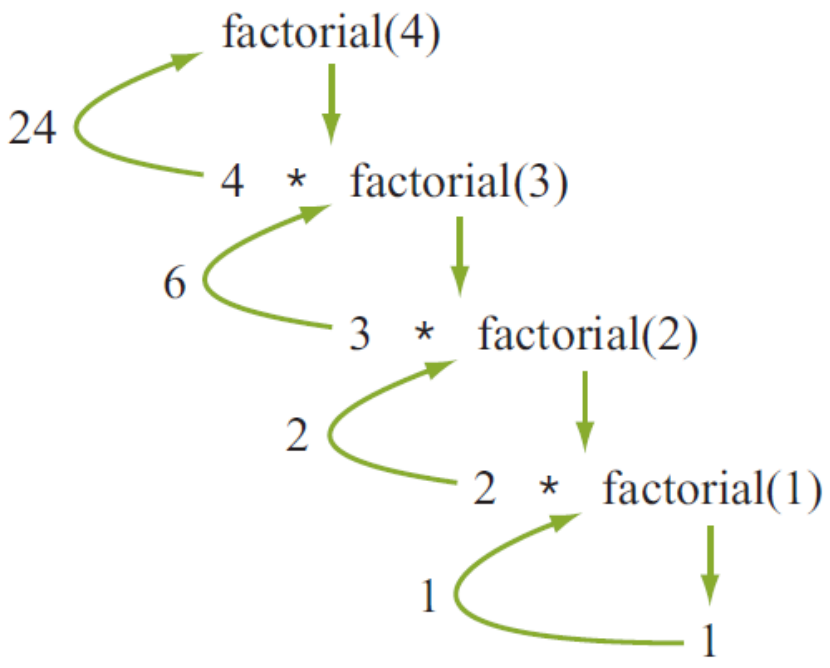
```java
// Create public calss name MultiArrayTest the same as the filename
public class MultiArrayTest
{
    Run | Debug
    public static void main (String[] args)
    {
        // Create a multidimensional array of string
        int[][] numbers = { {1, 2, 3, 4}, {5, 6, 7, 9} };
        System.out.println("Row length is " + numbers.length);
        System.out.println("Column length of first raw is " + numbers[0].length);

        // Loop inside array
        for (int i = 0; i < numbers.length; ++i)
        {
            for (int j = 0; j < numbers[i].length; ++j)
            {
                System.out.println("Element [" + i + "][" + j + "] is " + numbers[i][j]);
            }
        }
    }
}
```

Command Prompt

```
D:\Lessons\JavaOOP\Lesson2>java MultiArrayTest.java
Row length is 2
Column length of first raw is 4
Element [0][0] is 1
Element [0][1] is 2
Element [0][2] is 3
Element [0][3] is 4
Element [1][0] is 5
Element [1][1] is 6
Element [1][2] is 7
Element [1][3] is 9
```

Java Method Recursion

- Recursion is the technique of making a function call itself. This technique provides a way to break complicated problems down into simple problems which are easier to solve.
- Example, the function factorial(4) is evaluated as follows:



Java Method Recursion

❖ The recursive factorial method parallels the preceding mathematical definition. The method is defined as:

```
//Assume N is greater than 0
public int factorial(int N) {
    if (N == 1)
        return 1;
    else
        return N * factorial(N-1);
}
```

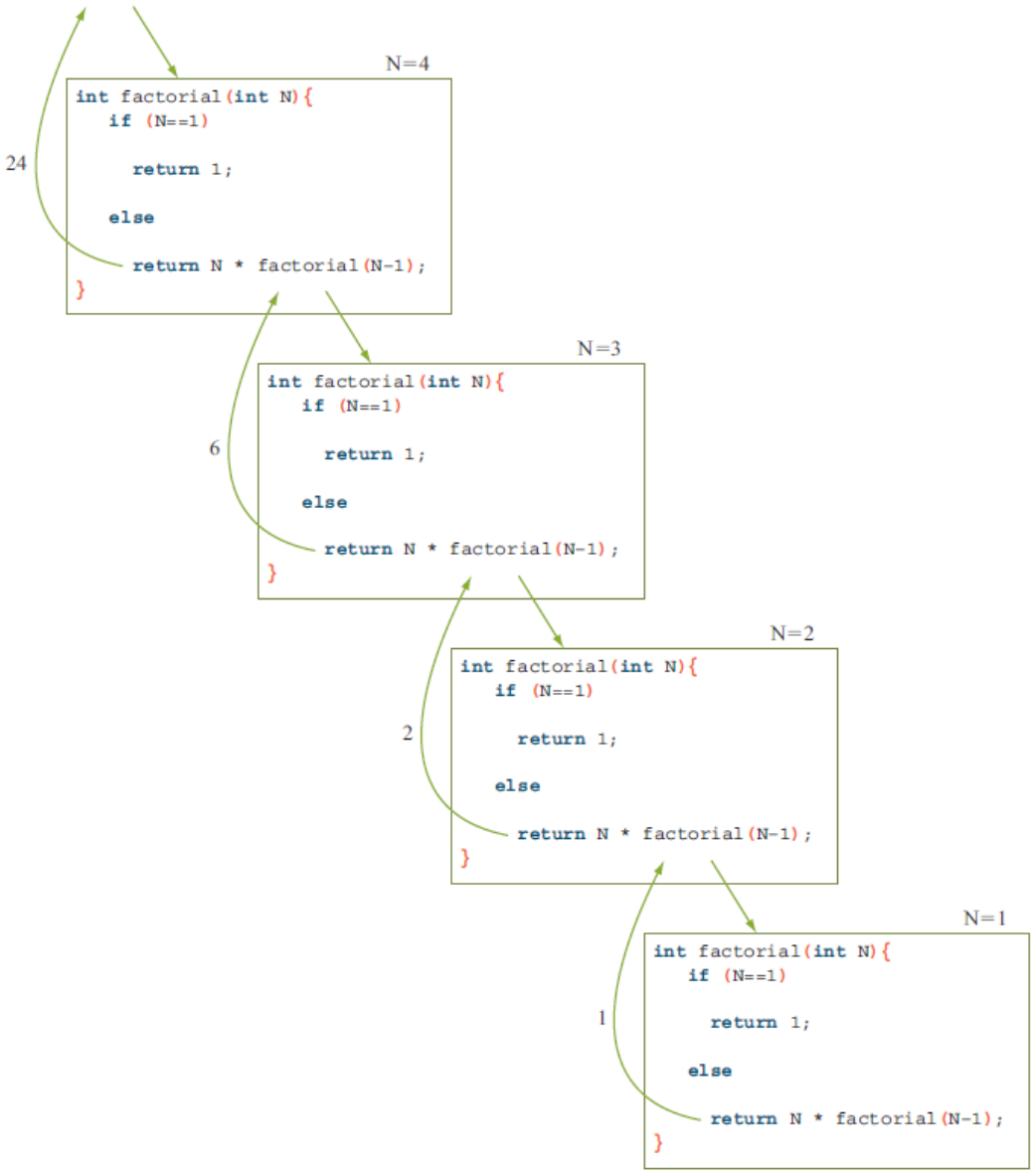
Test to stop or continue.

End case: recursion stops.

Recursive case: recursion continues with another recursive call.

Java Method Recursion

❖ The sequence of calls for the recursive factorial method can be illustrated as,



Finish

Q&A