



**BACHELOR OF COMPUTER  
APPLICATIONS  
SEMESTER 4**

**DCA2202  
JAVA PROGRAMMING**

# Unit 2

## Java Basics:

### Table of Contents

SL No	Topic	Fig No / Table / Graph	SAQ / Activity	Page No
1	<a href="#">Introduction</a>	-	-	3
	1.1 <a href="#">Objectives</a>	-	-	
2	<a href="#">Keywords</a>	-	-	4
3	<a href="#">Working of Java</a>	<a href="#">1,2</a>	<a href="#">1</a>	5 – 6
4	<a href="#">Including Comments</a>	-	<a href="#">2</a>	7
5	<a href="#">Data Types in Java</a>	<a href="#">3</a>	<a href="#">3</a>	8 – 9
	5.1 <a href="#">Primitives Data Types</a>	-	-	
	5.2 <a href="#">Abstract / Derived Data Types</a>	-	-	
6	<a href="#">Variables in Java</a>	-	<a href="#">4</a>	10 – 11
7	<a href="#">Using Classes in Java</a>	-	-	12
8	<a href="#">Declaring Methods in Java</a>	-	<a href="#">5</a>	13
9	<a href="#">Code to Display Test Value</a>	-	-	14
10	<a href="#">The <i>main()</i> Method</a>	-	-	15
11	<a href="#">Invoking a Method in Java</a>	-	-	16
12	<a href="#">Java Generics</a>	-	<a href="#">6</a>	17 – 19
	12.1 <a href="#">Generic Methods</a>	-	-	
	12.2 <a href="#">Generic Classes</a>	-	-	
13	<a href="#">Saving, Compiling and Executing a Java Programs Saving</a>	-	<a href="#">7</a>	20
14	<a href="#">Summary</a>	-	-	21
15	<a href="#">Terminal Questions</a>	-	-	22
16	<a href="#">Answers</a>	-	-	22 – 23

## 1. INTRODUCTION

In the last unit, we have discussed various features of Java. In this unit, we will discuss the basic programming structure in Java. Before that, we all know that the English language has a vocabulary – a set of words that have certain meanings. It also provides us with rules for using the vocabulary – English grammar. The Java language also provides a vocabulary and a set of rules to use the vocabulary. The vocabulary is represented through a set of keywords and the grammar is the syntax of the language.

This unit explains how to write object-oriented programs using the Java language syntax.

### 1.1 Objectives:

*After studying this unit, you should be able to:*

- *List the keywords significant to Java compiler*
- *Explain the working of Java*
- *Describe different data types in Java*
- *Explain the variables and their naming in Java*
- *Use classes and coding standards in Java*
- *Discuss declaring methods in Java*
- *Use code to display test value*
- *Discuss the main methods with its rules*
- *Describe invoking a method in Java*
- *Save, compile and execute a Java program*

## 2. KEYWORDS

Keywords are special words that are of significance to the Java compiler. Currently, JAVA language has 50 keywords (see Table 1). The Java language is defined by these keywords combined with operators and separators. Keywords cannot be used as variable, class, or method names.

**Table 1: Java Reserved Keywords**

Abstract	Boolean	Break	byte	case
Catch	char	Class	const	continue
Default	do	Double	else	extends
Final	Finally	Float	For	goto
If	Implements	Import	instanceof	int
interface	Long	Native	New	package
Private	Protected	Public	Return	short
static	Strictfp	Super	Switch	synchronized
this	Throw	Throws	transient	try
void	Volatile	While	Assert	enum

The **const** and **goto** are reserved keywords but these are not used. In addition to the above keywords Java also reserves the following: **true**, **false**, and **null**.

### 3. WORKING OF JAVA

The compiler converts the given program in Java into an intermediate language representation called **bytecode**. It is platform-independent. A Java file will have the extension **.java**, similar to a word file having the extension .doc, a Pascal file having the extension .pas and a text file having the extension .txt.

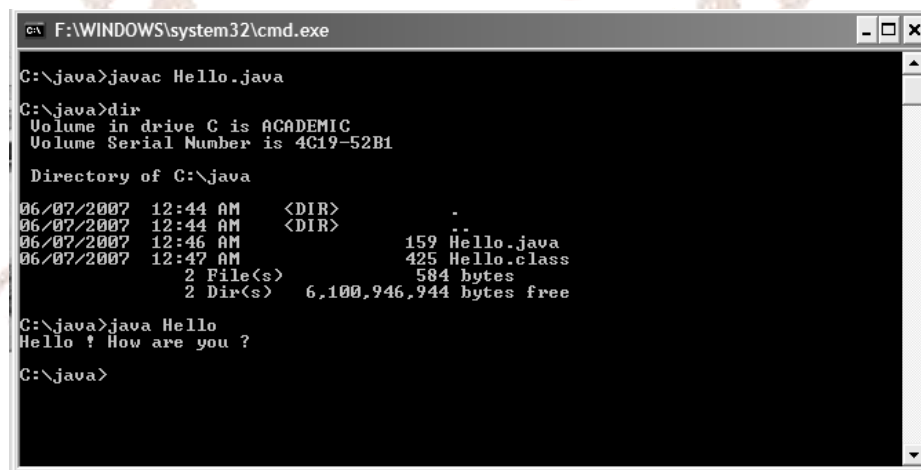
Let us assume that there exists a Java file named **Hello.java**. When this file is compiled we get a file called as **Hello.class**.

This class file is run using an interpreter as and when necessary. The Figure 1 shows the java program saved as Hello.java.

```
public class Hello {  
    public static void main( String args[])  
    {  
        System.out.println("Hello! How are you ?");  
    }  
}
```

**Figure 1:** A Sample Java Program

The steps for compiling and running the program are shown in Figure 2. The program is stored in a subdirectory called java. The above program prints “Hello! How are you? “



```
C:\F:\WINDOWS\system32\cmd.exe  
C:\java>javac Hello.java  
C:\java>dir  
Volume in drive C is ACADEMIC  
Volume Serial Number is 4C19-52B1  
  
Directory of C:\java  
06/07/2007  12:44 AM  <DIR>      .  
06/07/2007  12:44 AM  <DIR>      ..  
06/07/2007  12:46 AM                159 Hello.java  
06/07/2007  12:47 AM                425 Hello.class  
                2 File(s)          584 bytes  
                2 Dir(s)  6,100,946,944 bytes free  
  
C:\java>java Hello  
Hello ! How are you ?  
C:\java>
```

**Figure 2:** Compiling and Executing

The concept of “write once, run anywhere” is possible in Java. The Java program can be compiled on any platform having a Java compiler. The resulting *bytecode* can then be run on Window NT or Solaris or Macintosh or any other machine. The machine should have a Java platform to run Java code. The platform for Java contains Java Virtual Machine (JVM)

and a package of ready- made software components. These packages are known as Java Application Programming Interface (Java API). Any machine having JVM (Java Virtual Machine) can run the compiled Java codes.

**Integrated Development Environment(IDE) :**

Platforms are available on which Java code can be compiled, executed, and debugged in the same environment. These are referred to as Integrated Development Environment(IDE). Though it is possible to write the code on a notepad, compile it externally and execute it, Typically Java developers use IDE to develop, collaborate and integrate the code. Examples of the most common IDE for Java are

1. Eclipse
2. JetBrains IntelliJ IDEA
3. Microsoft Visual Studio etc

**Self-Assessment Questions - 1**

1. The JAVA program converted into an intermediate language representation by a compiler is called \_\_\_\_\_.
2. The concept of \_\_\_\_\_ is possible in Java.

## 4. INCLUDING COMMENTS

Comments can be included in a Java program as follows:

### Type 1

```
/* Comments go here  
More comments here  
*/
```

### Type 2

```
// This information is ignored by the compiler till the end of the line
```

### Type 3

```
/* Documentation comment */
```

In the first type comments can spread over multiple lines. In Type 2 compiler ignores any instruction written after the symbol `//`. It is used for writing single line comments. In Type 3 **Javadoc** tool uses the comment to automatic generate documentation.

### Self-Assessment Questions - 2

3. The Type 3 comments are used for automatic generation of documentation by a tool called \_\_\_\_\_.



## 5. DATA TYPES IN JAVA

In JAVA there are two different types of datatypes as mentioned below:

- Primitive / Standard data types.
- Abstract / Derived data types.

### 5.1 Primitives Data Types

Primitive data types are inbuilt in Java language. It is also called standard data types. Detailed instructions on each legal operation that the data type can support are contained in a compiler. There are eight primitive data types in Java (see Table 2).

**Table 2:** Primitive / Standard Data Types

Data Type	Size/Format (bits)	Description	Range
Byte	8	Byte length integer	-128 to 128 (for signed) 0 to 255 (for unsigned)
Short	16	Short integer	$-2^{15}$ to $2^{15} - 1$
Int	32	Integer	$-2^{31}$ to $2^{31} - 1$
Long	64	Long integer	$-2^{63}$ to $2^{63} - 1$
Float	32	Single precision floating point	+/- about $10^{39}$
Double	64	Double precision floating point	+/- about $10^{317}$
Char	16	A single character	Any single character
boolean	1	A boolean value	<b>true</b> or <b>false</b>

The byte, int, short, float, long and double are numeric data types. The first four of these can hold only whole numbers whereas the last two (float and double) can hold decimal values like 5.05. All these data types can hold negative values. However, the keyword **unsigned** an



be used to restrict the range of values to positive numbers. Amongst others, boolean can hold only the value **true** or **false** and char can hold only a single character.

## 5.2 Abstract / Derived Data Types

Based on primitive data types, the Abstract Datatypes have more functionality in comparison primitive data types. An example of Abstract datatypes is **String**, it can store digits, alphabets and special characters such as /, (), :\$#. Normal numerical operations cannot be performed on a variable of String datatype even if the data stored in it has digits.

### Self-Assessment Questions - 3

4. String is a \_\_\_\_\_ data type in Java.

## 6. VARIABLES IN JAVA

Since school days you have learned about variables and constants in mathematics. You have often used x and y to store values in an expression that were not constant and then there was some value like that of pi which was constant throughout. Java also allows declaring variables in the program to perform calculations and store values.

When variables and constants are declared in Java it is assigned some memory. Like mathematics variables, values may change but constant value remains the same for the entire program. Unique names are to be assigned for the variables following the instruction for variable declaration in Java. It follows the similar convention that we followed for declaring variables name in mathematics.

The variables needed in our program will have to be declared before using it. The datatypes of the variables must be precisely defined. The type declaration helps the compiler identify what kind of values the variable will store. For example, if we declare any datatype as an integer it will contain only integer values. If a variable is declared as float it will contain only decimal values.

### **Naming Variables**

There are set of rules for naming a variable in Java. The rules are mentioned below and it must be followed while choosing a variable name.

#### ***Rules for Naming Variables in Java***

A variable name:

- Keyword cannot be a variable name in Java.
- Variable can be alphanumeric but must not begin with a numeric value (number). It must always begin with an alphabet.
- There must not be spaces in a variable name.
- It may contain alphabets from other languages too like Japanese, Greek, etc.

#### ***Syntax for Defining Variables***

All the attributes of a class are defined as data members. The syntax for declaring a class variable is:

```
<data_type> <variable_name>;
```

The brackets “{ }” are used to mark the beginning and end of a class, loop etc. A semicolon “;” is used to mark the end of a statement.

#### **Self-Assessment Questions - 4**

5. Keywords can be used as a variable name. True or False.



## 7. USING CLASSES IN JAVA

In Java the body of the class is contained with the curly brackets {}. The body contains the methods and functions. The opening bracket { marks the beginning of the class body and the closing bracket } denotes the end of the class. Any program that you write in Java can be written using a simple text editor.

Example:

```
class Employee {  
}
```

Coding standards and conventions will have to be followed while writing Java Code. This will help in better understanding of the code and easier collaboration.

One of the conventions is indentation. Notice that the code is written below clearly indicates the data members defined in the class.

**Example:**

```
class Employee {  
    String employeeName;  
    String employeeAddress;  
}
```

## 8. DECLARING METHODS IN JAVA

The syntax for declaring a method in Java is as follows:

```
<access_specifier><return_type><method_name> ([argument_list])  
{  
}
```

### **access\_specifier**

An access specifier states where you can access a method. A 'public' specifier allows the method to be executed from another class. A 'private' provides access to methods for only the current class.

### **return\_type**

The return\_type of a method is the data type of the value that is returned by the method.

### **Example:**

```
public void displayEmpName(); // returns no value, therefore, the return  
                             // type of the method is void.  
public float calculateAllowance(); // returns a value of float data type  
                                   // therefore, the return type of the method  
                                   // is float
```

### **Self-Assessment Questions - 5**

6. A \_\_\_\_\_ access specifier allows the method to be executed from another class.

## 9. CODE TO DISPLAY TEST VALUE

### The *System* class

To communicate with the computer, a program needs to use certain system resources such as the display device and the keyboard.

Java makes use of all these resources with the help of a class called ***System***, which contains all the methods that are required to work with these resources.

*System* is one of the most important and useful classes provided by Java. It provides a standard interface to common system resources like the display device and the keyboard.

### The *out* Object

It is a static class encapsulated inside the ***System*** class, and represents the standard output device. This class contains the *println()* method.

### The *println()* method

The *println()* method displays the data on the screen.

### Example:

```
System.out.println("Hello World");
```

The above statement will display "Hello World" on the screen.



## 10. THE *main()* METHOD

In a Java application, you may have many classes. Within those classes, you may have many methods. The method that you need to execute first should be the ***main()*** method.

Syntax for the *main()* method:

```
public static void main(String args[])  
{  
}
```

The *main ()* method should exist in a class that is declared as public.

### Rules for the *main ()* method:

- The primary name of the file in which the code is written, and the name of the class that has the *main()* method should be exactly the same.
- If you try to execute a Java application that does not have a *main ()*
- method, the following error message will be printed:

Exception in thread "main" java.lang.NoSuchMethodError: main

## 11. INVOKING A METHOD IN JAVA

### Creating an Object of a class

The **new** operator is used to create a class object.

#### Example:

```
Employee emp = new Employee ();
```

### Invoking a Method

To invoke a method, the method name must be followed by parentheses and a semicolon. One method of a class can invoke another method of the same class using the name of the method.

#### Example:

```
class Employee {
    String employeeName; String
    employeeAddress;

    public Employee()
    {
        employeeName="Bala";
        employeeAddress = "Manipal";
    }

    public void display()
    {
        System.out.println ("Name: "+employeeName);
        System.out.println("Address: "+employeeAddress);
    }

    public static void main(String args[])
    {
        Employee emp = new Employee();
        emp.display();
    }
}
```

## 12. JAVA GENERICS

Java Generics helps programmers to write a single method or class that can be used with different related methods or related class types respectively.

It provides compile – time safety by allowing programmers to find invalid types during compile time. For example, Using the Generics concept we can write a method for sorting an array of objects. This generic method can then be used to sort an Integer array, string array and so on.

### 12.1 Generic Methods

A generic method declaration can be called with different types of arguments. The compiler, based on the types of arguments passed to the generic classes, handles the method appropriately. Following set of rules defines a Generic Methods:

- In every generic method you have a type parameter section which is delimited by angle brackets (< and >) and it precedes the method's return type declaration.
- Type parameter section contains one or more type parameters as shown in the example which are separated using commas. These type parameters are also called type variable and is used as an identifier to specify a generic type.
- Type parameters can also be used to declare the return type. It acts as placeholders for the argument types passed to a generic method, also called actual type arguments.
- A generic method's body is declared like other method's body, however the type parameters can only represent reference types and not primitive data types.

```
public class GenericMethodTest {
    // generic method printArray
    public static <E> void printArray(E[] inputArray) {
        // Display array elements
        for (E element : inputArray) {
            System.out.printf("%s ", element);
        }
        System.out.println();
    }

    public static void main(String args[]) {
        // Create arrays of Integer, Double and Character
        Integer[] intArray = {100, 200, 300, 400, 500 };
        Double[] doubleArray = { 1.1, 2.2, 3.3, 4.4 };
        Character[] charArray = {'H','E','L','L','O' };
        System.out.println("Array integerArray contains:");

        // pass an Integer array
        printArray(intArray);

        System.out.println("\nArray,doubleArray contains:");
        printArray(doubleArray); // pass a Double array

        System.out.println("\nArray,characterArray contains:");
        printArray(charArray); // pass a Character array
    }
}
```

**Example:****Output:****Compiling the source code....**

\$javac GenericMethodTest.java 2>&1

**Executing the program....**

\$java -Xmx128M -Xms16M GenericMethodTest

Array integerArray contains:

100 200 300 400 500

Array doubleArray contains:

1.1 2.2 3.3 4.4

Array characterArray contains:

H E L L O W O R L D

## 12.2 Generic Classes

A generic class declaration is similar to non-generic class declaration but like generic method the class name is followed by type parameter section.

Similar to generic methods the type parameter section of a generic class can also have one or more type parameters which are separated commas. Generic classes are also called parameterized classes or parameterized types as they accept one or more parameters.

### Example

```
public class Box<T> {  
    private T t;  
  
    public void add(T t) {  
        this.t = t;  
    }  
  
    public T get() {  
        return t;  
    }  
  
    public static void main(String[] args) {  
        Box<Integer> integerBox = new Box<Integer>();  
        Box<String> stringBox = new Box<String>();  
        integerBox.add(new Integer(10));  
        stringBox.add(new String("Hello World"));  
  
        System.out.printf("Integer Value :%d\n\n", \n            integerBox.get());  
        System.out.printf("String Value :%s\n", \n            stringBox.get());  
    }  
}
```

### Output:

#### Compiling the source code....

```
$javac Box.java 2>&1
```

#### Executing the program....

```
$java -Xmx128M -Xms16M Box
```

```
Integer Value :10
```

```
String Value :Hello World
```

### Self-Assessment Questions - 6

7. \_\_\_\_\_ operator is used to create an object.

### 13. SAVING, COMPILING AND EXECUTING A JAVA PROGRAMS SAVING

The programs that you write in Java should be saved in a file, which has the following name format:

**<class\_name>.java**

#### Compiling

A program is a set of instructions. To execute a program, the operating system needs to understand the language. The only language an operating system understands is in terms of 0's and 1's i.e. the binary language. Programs written in language such as C and C++ are converted to binary code during the compilation process. However, that binary code can be understood only by the operating system for which the program is compiled. This makes the program or application as operating system dependent.

In Java, the program is compiled into bytecode (.class file) that run on the Java Virtual Machine, which can interpret and run the program on any operating system. This makes Java programs platform-independent.

At the command prompt, type

**javac <filename>.java**

to compile the Java program.

#### Executing

When the code is compiled and error-free, the program can be executed using the command:

**java <class filename>**

In an integrated environment (IDE) options are available within the environment to compile and execute the code.

#### Self-Assessment Questions - 7

8. \_\_\_\_\_ is the extension for Java source code files.
9. \_\_\_\_\_ command is used to compile the Java source code.
10. \_\_\_\_\_ command is used to execute the Java class file.



## 14. SUMMARY

In this unit, you have learnt the following:

- **Creating Classes Using Java**

The data members and methods of a class are defined inside the class body. In Java, brackets {} mark the beginning and end of a class or method. The **class** keyword is used to declare a class.

- **Coding Methods of a Class**

Methods provide functionality to classes. In Java, methods are declared in the class body.

- **Declaring Objects**

The **new** operator is used to create a class object.

- **Displaying Data on Screen**

The **System** class is one of the most important and useful classes provided by Java. It provides a standard interface to common system resources like the display device and the keyboard. The **println()** method displays the data on the screen.

- **Compiling a Java Program**

In Java, the program is compiled into bytecode (.class file) that runs on the Java Virtual Machine, which can interpret and run the program on any operating system. This makes Java programs platform-independent.

- **Java Generics**

An introduction to Java Generics methods and class. Usage of type parameters. Using generics same methods and class can be used for similar types of methods or class types respectively.

- **Executing a Java Program**

When the code is compiled and error-free, the program can be executed by issuing the following command:

```
java <class filename>
```

## 15. TERMINAL QUESTIONS

1. The candidate class for the employee referral process has the following attributes and behaviors:

Candidate
candidateName
candidateAddress
candidateCourse
displayDetails()

2. Create the candidate class with its attribute and methods.
3. What do you mean by the statement : `System.out.println(" ");`
4. How do you compile a Java program?
5. How do you execute a Java program?

## 16. ANSWERS

### Self Assessment Questions

1. bytecode
2. write once, run anywhere
3. javadoc
4. abstract / derived
5. false
6. public
7. new
8. .java
9. Javac
10. java

**Terminal Questions**

1. class Candidate

```
{  
  
    String candidateName;  
    String candidateAddress;  
    String candidateCourse;  
  
    public Candidate()  
    {  
        candidateName="Arun";  
        candidateAddress="Manipal";  
        candidateCourse="BScIT";  
    }  
    public void displayDetails()  
    {  
        System.out.println("Name: "+candidateName);  
        System.out.println("Address: "+candidateAddress);  
        System.out.println("Course: "+candidateCourse);  
    }  
}
```

} (Refer section 11)

2. System.out.println(" "); will print a blank line. (Refer section 9)
3. Java programs are compiled using the command javac. (Refer section 12)
4. Java class files are executed using the command java. (Refer section 12)