

Unit 1: Basics of Java

1

Unit Structure

- 1.1. Learning Objectives
- 1.2. Introduction
- 1.3. Implementation of O.O.P concept in java
- 1.4. Java Environment
- 1.5. Java Features and support
- 1.6. Sample program & Compilation
- 1.7. Using block of code
- 1.8. Lexical Issues
- 1.9. Java Class Library
- 1.10. Data type
- 1.11. Operators
- 1.12. Control Structures
- 1.13. Let us sum up
- 1.14. Check your Progress

- 1.15. Check your Progress: Possible Answers
- 1.16. Further Reading
- 1.17. Assignments

1.1 LEARNING OBJECTIVE

After studying this unit student should be able to:

- Understand the structure of java program
- Implement and run a “hello world” program.
- Differentiate among various types of tokens.
- Understand the basic data types, operators, arrays, libraries etc.

1.2 INTRODUCTION

Java is an object oriented programming language. Before starting with how to do programming using java, we will briefly discuss about the object oriented concepts and their implementation in java.

1.3 IMPLEMENTATION OF O.O.P CONCEPT IN JAVA

Object-oriented language uses a unique programming pattern compare to structural programming like C. Object Oriented programming supports a programming using the concepts like class/object, Inheritance, Encapsulation, Abstraction, and Polymorphism. The structural programming is mainly based on data. It uses various data structures and write program to perform action on those data. However object oriented programming language like java uses concept of encapsulation which combines data and program code together in object. Thus here the data and program are combined. Also using abstraction concept, class attributes or behavior can be made hidden from the other external objects. Polymorphism concept use to represent an object in many forms. One task can be performed in different ways. A common use of polymorphism is use when a parent class reference is used to refer to a child class object in a program.

An object is an entity which has several attributes and behavior. A number of objects sharing same attributes and behavior form a Class. For example: parrot, peacock, hen, dove are objects of class birds. They share attributes like color of eyes, size, shape of beak, their food habit etc. and behavior like laying eggs, flying, constructing nest etc. An object/class may also relate with other objects/classes through parent-child relationship which is called inheritance. A child object looks

similar to its parent but with some unique specialized attributes. Hence child inherits the properties of parents. Parents have all common attributes and behavior of the child class. And each child class has its additional attributes and behavior. For example:

Birds can be parent with size, weight, eating habit and living place as common attribute. The class Water and Land can be children of Birds class with based on where they live. Birds live in water have unique attributes than birds live on land. The pictorial representation of this example shown in figure 1.1.

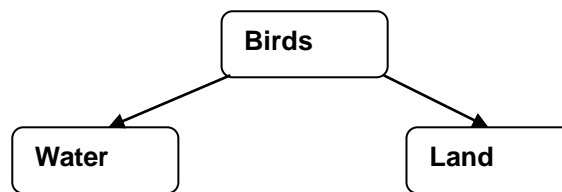


Figure-1 Attributes of Class

1.4 JAVA ENVIRONMENT

Java Runtime Environment (JRE) consists of various software tools used for java application development. It is also called Java Runtime. It has the Java Virtual Machine (JVM), core classes and supporting libraries. To develop a java application we must install Java Development Kit (JDK) in our computer. It includes JRE as its part. It was developed by Sun Microsystems which is now owned by Oracle Corporation. You can download latest version JDK from Website of Oracle Corporation. The list of components which are part of JRE are deployment tools, user interface toolkits, integration and other base libraries, language and utility base library, and Java Virtual Machine.

1.5 JAVA FEATURES AND SUPPORT

We can develop four types of applications using Java. They are desktop applications, web applications, enterprise application and mobile applications. These applications can be developed using four types of java platforms such as Java SE, Java EE, and Java ME. Java Standard Edition(Java SE) is a programming platform for java application development. It supports core concept like implementation of

OOPs, String, Exception, Inner classes, Multithreading, I/O Stream, Networking, AWT, Swing, Reflection, Collection, etc. Java Enterprise Edition(Java EE) mainly used to develop web applications and enterprise application. It consists of libraries for servlet, JSP, JDBC, Web Service, EJB etc. Java Micro Edition(Java ME) is used to develop mobile applications using java.

➤ **Features of java**

Java includes various features like simple, object oriented, distributed, compiled and interpreted, robust, secure, platform independent, multithreaded, portable and dynamic.

- **Simple:** Java is easy to learn compared to C++. For C/C++ programmer, it will be easier to learn java as basic syntax is almost same.
- **Object oriented:** Java syntax supports the concept of object oriented programming.
- **Distributed:** we can develop distributed application using java.
- **Compiled and interpreted:** Java first compile the program (.java file) and generate the bytecode (.class file). This byte code is then interpreted using java interpreter.
- **Robust:** Java develops robust applications using strong memory management and error handling using garbage collector and exception handling.
- **Platform independent:** Platform means Operating system and hardware. Java programs run in same way with any Operating system and hardware combination.
- **Secure:** Java is secure because it runs inside Java Virtual Machine(JVM) which verifies code before execution. It also handles run time errors using exception handling mechanism.
- **Multithreading:** This feature enables a java program to perform multiple task simultaneously.
- **Portable:** Due to platform independent nature of java program, it is portable. We can shift program to any environment without any side effect.

- **Dynamic:** As java supports dynamic loading of classes, it is dynamic. The classes are loaded run time when needed. Java also executes functions from its native languages like C and C++.

1.6 SAMPLE PROGRAM & COMPILATION

For running any java program we must install java(Java SE) in our computer. For installing java we must download latest version of java using Oracle Corporation website (<https://www.oracle.com>). After installation you can find java in java/jdk folder. In the jdk folder you have java compiler(javac) and java interpreter(java), which are used to run java program. For writing your first java application, you can use any text editor and write following code in file. Save this file as “MyFirst.java”. if you are creating class in this java file the class name should be same as file name(here MyFirst). This is not compulsory if class is not declared as public. Then compile this java code using java compiler,

```
/javac MyFirst.java
```

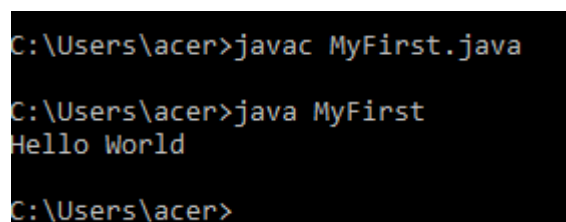
This command will create MyFirst.class file, which is a bytecode. To run this program we have to use following command,

```
/java MyFirst
```

Hello World

This will run your program and print “HelloWorld” as an output.

```
class MyFirst
{
    public static void main(String args[])
    {
        System.out.println("Hello World");
    }
}
```



```
C:\Users\acer>javac MyFirst.java
C:\Users\acer>java MyFirst
Hello World
C:\Users\acer>
```

Figure-2 Compiling and Running First Java Program

A java program must have at-least one class. Each class has a class name. In above example MyFirst is a class name. After compilation of MyFirst.java file, we get MyFirst.class. We have to use this class file to run the program.

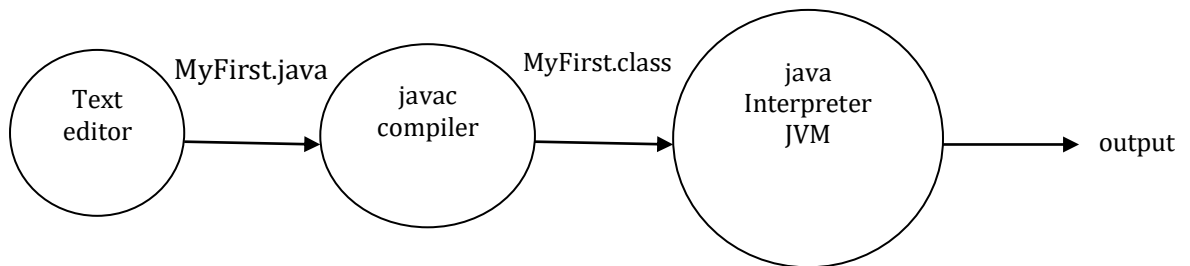


Figure-3 Flow of Program

The first line in the program defines a class. After that the second line has curly bracket which shows starting of class definition. The last line of program must be end of the curly bracket. Within the class we have defined a main() method. Same as C/C++, when we run program the execution starts by calling main method. The main method is declared public, which means this method can be called by code outside this program. The method is static, which means method can be called using class name without using any object. Void means main function return nothing. The main function accepts array of string(args[]) as an argument, which is use to stored command line arguments when we run the program.

In main method we can write our program instructions. In above example we have used System.out.println function to print a message(Hello World) on output screen. This function is same as cout and printf of C and C++ respectively. Here System is a class available in java.lang package. out is an object of PrintStream class which is declared static in System class. println is a method of PrintStream class which accepts a string and print it.

1.7 USING BLOCK OF CODE

<pre>/* comments section */ // documentation</pre>
Package declaration
Import statements which use to specifies the external packages used in program

class definitions
class with main() method

The above diagram shows the block structure of the program written in java.

The first part of the program is documentation section. We can write explanation of the program or usage of program in this section. It can start with `/*` and end with `*/`(Multiline comment). We can write number of line in between them. If you want to write only one line(single line comment), you can use `//` at the beginning of the line. For example

```
/* this program is for adding n elements of an array. We have to provide n numbers as a command line arguments */
```

```
//This program adds n numbers passing as command line argument
```

In second part of the block, we can declare packages. This is used only if we want to create a class in a specific package. In java, packages are used to create a group of classes which are related.

In third part we have to import all the packages whose classes we are using in our program. It is like include statement of C/C++. In java library functions are available as a part of classes inside the package. if we want to use library function, we have to import appropriate package.

After import statements, we can define classes. For this we have to use class key word followed by class name. we can define class within curly braces as shown in above example. A class can have variable names and methods defined in it.

The last section is the definition of class which has main method. It is the method from where execution of our program starts.

1.8 LEXICAL ISSUES

While compiling java program statements, the words/characters in source code are separated as tokens. Tokens are the atomic elements of java program. The java compiler identified the tokens as white space or saperators, identifiers, comments, keywords, operators, literals

During compilation, the characters in Java source code are reduced to a series of tokens. The Java compiler recognizes five kinds of tokens: identifiers, keywords, literals, operators, and miscellaneous separators. Comments and white space such as blanks, tabs, line feeds, and are not tokens, but they often are used to separate tokens.

➤ **White space**

White space such as blanks, tabs, line feeds, form feed, carriage return, new line etc are not the tokens. They are used to separate tokens.

➤ **Identifiers**

In java, identifies are the names given to variables, methods, class, interfaces and packages. In the above mentioned sample program (figure 1), MyFirst, main, String, args, println etc. are identifiers. Rules for identifier in java are listed below:

- 1) An identifier must begin with letter, underscore or '\$'
- 2) Identifiers can not start with digits
- 3) There is no limit for length of identifiers
- 4) A keyword (reserved word) can not be identifier. i.e. class is invalid identifier.
- 5) Identifiers are case sensitive. i.e. Name and name are different

➤ **Literals**

Literals are the representation of data. They can be numeric, boolean, character or string data. They are actually represents the value of the variables. For example in program statement `int x=25;` x is a variable and 25 is numeric literal.

Following are the example of various literals

- **Numeric literals** : 12, 45, 0x345, 101, 12.4, 0.980, -2
- **String literals**: "hello", "hello\nworld"
- **Character literals**: 'a', '\n', '\t', 'c'
- **Boolean literals**: true, false

➤ **Comments**

Comments are the text in program which is not compiled. They are used in a program for documentation. In java, comment can be written in between /* and */ or after //. We can write number of line in between /* and */. If you want to write only one line (single line comment), you can use // at the beginning of the line. For example

```
/* this program is for adding n elements of an array. We have to provide n numbers
as a command line arguments */
```

```
//This program adds n numbers passing as command line argument
```

➤ **Keywords**

Keyword are the words reserved for java compiler. They have a pre-defined meaning in java, hence they can not be used as identifiers. The following is the list of java keywords.

abstract	Continue	for	new	switch
assert	Default	goto	package	synchronized
boolean	Do	if	private	this
break	Double	implements	protected	throw
byte	Else	import	public	throws
case	Enum	instanceof	return	transient
catch	Extends	int	short	try
char	Final	interface	static	void
class	Finally	long	strictfp	volatile
const*	Float	native	super	while

1.9 JAVA CLASS LIBRARY

Java Class Library(JCL) is a collection of libraries that java programs can call at run time. Java is platform independent, so it does not use any Operating System library. Java provides a standard class library which contains the functions commonly available in all operating systems. All JCL implementations are available in single jar file (rt.jar). This jar file is available with JDK/JRE installation and always located in bootstrap classpath.

JCL can be used through classes provided in following packages. One must use import statement for using them in program.

- **java.lang** : for fundamental classes and interfaces related to the language and runtime system.
- **I/O and networking access**: They access the file system, and networks through the java.io, java.nio and java.net packages.
- **Mathematics package**: java.math provides mathematical expressions and evaluation
- **Collections and Utilities** : java.util for Regular expressions, Concurrency, logging and Data compression.
- **GUI**: The java.awt package for basic GUI components/operations which bound to the underlying operating system. It also contains the 2D Graphics API. The package (javax.swing) is built on AWT and provides a platform-independent GUI components/operations.
- **Applets**: java.applet are the java class stored on web server and downloaded over a network for execution on client machine.
- **Introspection and reflection**: The package java.lang.Class use to represent a class, but other classes such as Method and Constructor are available in java.lang.reflect package.

1.10 DATA TYPES

Data types can specify the sizes and values stored in any variable. Data types can be identified value type and reference type. The value type has a value stored in a stack. The reference type stores a reference of data and stored in heap. In java data types are classified into two categories

- **Primitive data types**: byte, int, short, long, float, double, boolean and char are primitive data types in java.
- **Non- primitive data types**: class, array, String, Vector, LinkedList etc are non primitive data types.

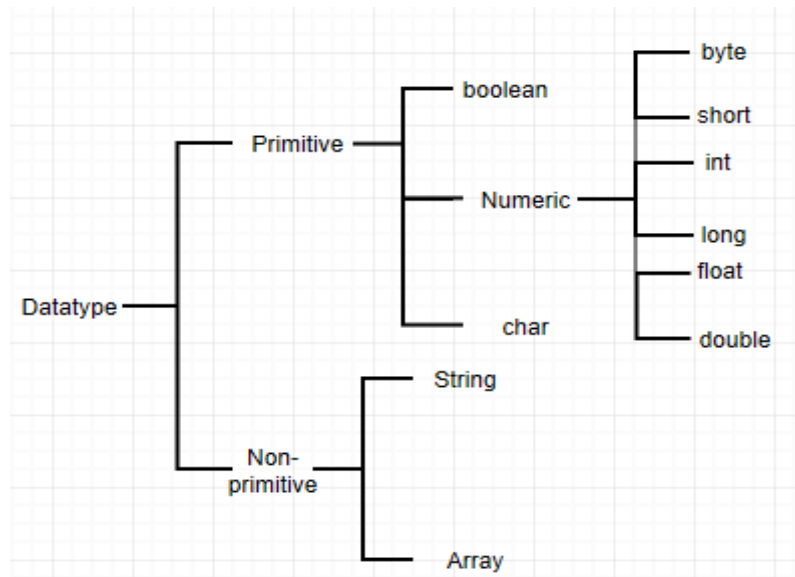


Figure-4 Data Types in Java

- **boolean:** The boolean data type can be use to store two values: true or false. This data types are used to defined a flag. It occupies 1 bit space in memory.
- **byte:** it is used to store 8 bit integer value. It can store value from -128 to 127 in it.
- **short:** it is used to store 16 bit number. It can store value from -32768 to 32767.
- **int:** it occupies 32 bit area in memory. It can store a number between - 2,147,483,648 and 2,147,483,647.
- **long:** it occupies 64 bit memory area. It can store a number between -264 to (264)-1.
- **char:** It occupies 16bit and can store 0 to 65536 representing Unicode for different characters.
- **float :** it is of 32 bit and store number in 1.4×10^{-45} to 3.4×10^{38} range.
- **double:** it occupies 64 bit memory and can store a number in 4.9×10^{-324} to 1.8×10^{308} .

Example:

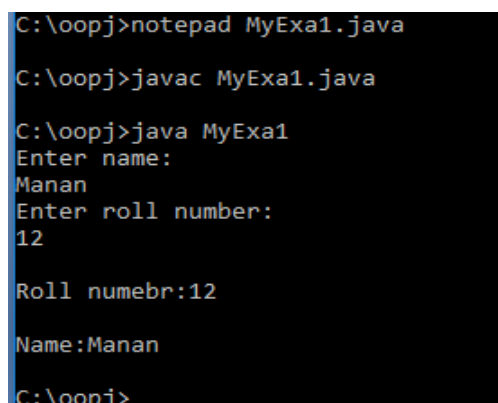
In this example we have used Scanner class to read value from keyboard. Scanner class is available in java.util package which is used to obtain input of the primitive types like int, double etc. and Strings from input stream. It is the easiest way to read input in a Java program. To use this class we have to keep following points in mind.

- 1) We need to create an object of Scanner class with System.in(for standard input stream).
- 2) To read integer value we have to use nextInt() function. We can also use nextLong(), nextShort(), nextByte() etc. for String input we have to use nextLine() function of Scanner class.

The following is the program to get roll number and name of the student through keyboard and display them as output. The program is written in MyExa1.java file.

```
import java.util.Scanner;
class MyExa1
{
    public static void main(String args[])
    {
        int rno=null;
        String name=null;
        Scanner x=new Scanner(System.in);
        System.out.println("Enter name:");
        name=x.nextLine();
        System.out.println("Enter number:");
        rno=x.nextInt();
        System.out.println("\nRoll number:"+rno);
        System.out.println("\Name:"+name);
    }
}
```

The output of the above program is shown in Figure-5.



```
C:\oopj>notepad MyExa1.java
C:\oopj>javac MyExa1.java
C:\oopj>java MyExa1
Enter name:
Manan
Enter roll number:
12
Roll numebr:12
Name:Manan
C:\oopj>
```

Figure-5 Output of Program

1.11 OPERATORS

They are the characters/symbols used to manipulate data. Operators can have one or more operand on which they perform a function. The operators in java can be classified in to following categories:

➤ **Arithmetic Operators**

Operator	Use
+	Addition of two values Ex: 20+10 gives 30
-	Subtraction of two values Ex: 20-10 gives 10
*	Multiplication of two values Ex: 20*10 gives 200
/	Division of two values Ex: 20/10 gives 2
%	Reminder/Modulus gives reminder of division of two numbers Ex: 21%2 gives 1
++	Increment operator increase value by 1 **
--	Decrement operator decrease value by 1 **

Table-1 Arithmetic Operators

** if we use ++/-- before operand, the increment/decrement is performed first before using the operand and if we use ++/-- after operand, the increment/decrement is performed first after using the operand.

For Example ,

```
a=4;
b=++a; //give 5 in b and 5 in a;
a=4;
b=a--; //gives 4 in b and 3 in a
```

Example:

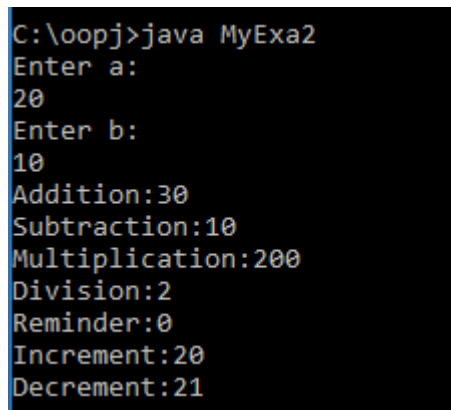
```
import java.util.Scanner;
public class MyExa2
{
public static void main(String args[])
```

```

{
    int a,b;
    Scanner sc=new Scanner(System.in);
    System.out.println("Enter a:");
    a=sc.nextInt();
    System.out.println("Enter b:");
    b=sc.nextInt();

    System.out.println("Addition:"+(a+b));
    System.out.println("Subtraction:"+(a-b));
    System.out.println("Multiplication:"+(a*b));
    System.out.println("Division:"+(a/b));
    System.out.println("Reminder:"+(a%b));
    System.out.println("Increment:"+(a++));
    System.out.println("Decrement:"+(a--));
}
}

```



```

C:\oopj>java MyExa2
Enter a:
20
Enter b:
10
Addition:30
Subtraction:10
Multiplication:200
Division:2
Reminder:0
Increment:20
Decrement:21

```

Figure-6 Output of Program

➤ **Assignment Operators**

This operators are used to assign value to the operand.= is assignment operator. It assigns value to its operand for Ex: a=5;

+=, -=, *=, /= and %= are the shorthand operators. They perform operation as shown below:

Operator	Use	Meaning
=	a=5;	value 5 is assigned to a
+=	a+=5;	it performs a=a+5.
-=	a-=5;	it performs a=a-5.
=	a=5;	it performs a=a*5.
/=	a/=5;	it performs a=a/5.
%=	a%=5;	it performs a=a%5.

Table-2 Assignment Operators

➤ **Relational Operators**

They are also called comparison operators. They are used to compare two operands and returns Boolean value.

Operator	Meaning	Use
==	equality	a==b
!=	not equal	a!=b
>	greater than	a>b
<	less than	a=	greater than or equal to	a>=b
<=	less than or equal to	a<=b

Table-3 Relational Operators

They are used with if...else statement to build a condition. For example (a>b) returns true if a is greater than b else it returns false.

➤ **Logical Operators**

&&, || and ! are the logical operators. They are used to check for two conditions simultaneously.

Operator	Meaning	Usage
&&	logical and	(a>b && a>c) check both condition

	logical or	(a>b a>c) check either of one condition
!	logical not	!(a>b) check not of condition

Table-4 Logical Operators

➤ **Bitwise Operators**

&, |, ^, << and >> are bitwise operators. They are used to perform bitwise operations.

Operator	Meaning	Usage
&	AND	a&b
	OR	a b
^	EXOR	a^b
<<	left shift	a<>	right shift	a>>b

Table-5 Bitwise Operators

The AND, OR and EXOR operations are shown below in truth table.

A	b	a&b	a b	a^b
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Table-6 Truth Table

The shift operator shift the value of operand specific number of time in left(<<) or right(>>). The left operand specifies the value to be shifted and right operand specifies number of shift.

➤ **Miscellaneous Operators**

- **instance of operator**

it is used to check whether an object is of a specific class type or not.

For example,

```
String s="hello";  
if( s instance of String)  
{  
    System.out.println("s is of String type");  
}
```

- **Ternary operator**

?: is used as a ternary operator. It has three operands. It is shorter replacement of if...else statement.

Syntax: var=(expression)?value1:value2;

Example: c=(a>b)?a:b; It means c is largest of a or b.

1.12 CONTROL STRUCTURES

1.12.1 CONDITIONAL STATEMENTS

Conditional statements are used to run block of java code based on a condition. The java has various ways to execute conditional statements. They are using if, if...else, if else ladder, nested if...else, and switch...case. All can be used same as C/C++ syntax.

➤ **if...else and its variations**

The syntax of if...else is,

```
if(condition)  
{  
    Code block  
}  
else  
{  
    Code block  
}
```

We can omit the braces ({...}), if the code block has only one program statement.

if...else statements of java are identical to C/C++. We can use if without else. For example to check whether a is even we can use following statements .

```
if(a%2==0)
System.out.println(a+" is even");
```

We can also use if...else together. For example to check whether a is even or odd we can use following code.

```
if(a%2==0)
System.out.println(a+" is even");
else
System.out.println(a+" is odd");
```

If we use if...else inside if or else block, it will be nested if... else. For example to find out largest of three numbers a, b and c the following code can be used.

```
if(a>b)
{
    if(a>c)
        System.out.println("a is greatest");
    else
        System.out.println("c is greatest");
}
else
{
    if(b>c)
        System.out.println("a is greatest");
    else
        System.out.println("c is greatest");
}
```

We can also use if...else in ladder pattern. For example from current time if you want a java program to wish "good morning", "good afternoon", "good evening" or "good night", we can use following if...else ladder.

```
if(current_time>5 && current_time<12)
System.out.println("good morning");
else if(current_time>12 && current_time<5)
System.out.println("good afternoon");
else if(current_time>5 && current_time<8)
System.out.println("good evening");
else
System.out.println("good night");
```

switch...case can be used to execute different code block for different value of input. For example if based on input value of arithmetic operator we want to perform the operation, we may use following code in java. In switch...case, each case should end with break statement. And default case is match if input is not match with any case.

```
switch(opr)
{
case '+':
System.out.println(a+b);
break;
case '-':
System.out.println(a-b);
break;
case '*':
System.out.println(a*b);
break;
case '/':
System.out.println(a/b);
break;
case '%':
System.out.println(a%b);
break;
default:
System.out.println("Invalid operation");}
```

Examples

A program to which reads two integers and perform the arithmetic operation on them based on user's choice.

```
import java.util.Scanner;
public class Ex_if
{
    public static void main(String args[])
    {
        int ch=0;
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter a:");
        int a=sc.nextInt();
        System.out.println("Enter b:");
        int b=sc.nextInt();

        System.out.println("1. add");
        System.out.println("2. subtract");
        System.out.println("3. multiply");
        System.out.println("4. divide");
        System.out.println("Enter your choice:");
        ch=sc.nextInt();
        if(ch!=5)
        {
            switch(ch)
            {
                case 1: System.out.println(a+b); break;
                case 2: System.out.println(a-b); break;
                case 3: System.out.println(a*b); break;
                case 4: System.out.println(a/b); break;
                default: System.out.println("Invalid choice");
            }
        }
    }
}
```

```
C:\ajava\oopj>javac Ex_if.java
C:\ajava\oopj>java Ex_if
Enter a:
23
Enter b:
12
1. add
2. subtract
3. multiply
4. divide
Enter your choice:
2
11
```

1.12.2 LOOPING

In a program when we want to execute a code block more than once, we need to put it in a loop. In java loop can be a for loop, while loop and do...while loop. The syntax of these loop are same as C/C++. The Java 5 introduce foreach loop. It is used to access the array or collection elements.

➤ For

The for loop executes a statement or block of statements repeatedly until a condition is matched. For loops are normally used to execute the code block for more than one number of times. The syntax of for loop is given below.

```
for (initialization; test; increment)
{
    statements;
}
```

We can omit the braces if for loop has only one statement. As you can see in the syntax for loop has three parts in bracket.

- **initialization** is used to initialize the counter used in loop to keep track on number of iteration. -for example, int i=0 OR i=0.
- **test** must be the condition which must be true to enter in the loop. If the condition is false the loop terminates. Test is used to control the iteration count. For example i<10 terminates the loop when i is greater or equal to 10.
- **increment** is used to change value of variable used in initialization

For example the below for loop prints "Hello" 10 times with value of i each time. The output will print Hello0, Hello1,.....Hello9.

```
for(int i=0;i<10;i++)
    System.out.println("Hello"+i);
```

➤ while and do...while

while and do...while loops are also used to repeatedly execute a block of Java code until a condition is true. The syntax of these loops are same as C/C++.

The only difference between while and do...while loop is the timing of checking the condition. The while loop checks the condition before entering the loop.

If condition is true it enters. The do...while loop first enter into the loop and check condition at the end.

The syntax of these loops are

```
while(test)
{
    Statements;
}
do
{
    Statements;
}while(test);
```

The example in above section can be implemented using while and do...while as below.

```
int i=0;
while(i<10)
{
    System.out.println("Hello"+i);
    i++;
}
```

OR

```
int i=0;
do
{
    System.out.println("Hello"+i);
    i++;
}
while(i<10);
```

➤ **for-each**

This loop is not available in C/C++. The Java 5 introduce foreach loop. It is used to access the array or collection elements. The purpose of this loop is to make our program code bug free and more readable. The syntax of this loop is :

```
for(data_type variable : array | collection)
{
```

```
Statements;  
}
```

For example, the following code will print content of array arr. The for-each loop will execute 3 time. First time i will be 18, second time i will be 23 and then 45.

```
int[] arr={18,23,45};  
for(int i:arr)  
{  
    System.out.println(i);  
}
```

Example of loop:

A program to find factorial of a number.

Note: factorial of 5 is $1*2*3*4*5$

```
import java.util.Scanner;  
public class Ex_loop  
{  
    public static void main(String args[])  
    {  
        long fact=1;  
        Scanner sc=new Scanner(System.in);  
        System.out.println("Enter n:");  
        int n=sc.nextInt();  
        //using for loop  
        for(int i=1;i<=n;i++)  
            fact*=i;  
        System.out.println("for:Factorial of "+n+" is :"+fact);  
        //using while loop  
        fact=1;  
        int i=1;  
        while(i<=5)  
        {  
            fact*=i;  
            i++;  
        }  
        System.out.println("while:Factorial of "+n+" is :"+fact);  
    }  
}
```

```
C:\ajava\oopj>javac Ex_loop.java
C:\ajava\oopj>java Ex_loop
Enter n:
5
for:Factorial of 5 is :120
while:Factorial of 5 is :120
```

Figure-8 Output of Program

➤ Use of continue and break in loops

In any loop, we can use break to terminate the loop and continue to skip existing iteration and start new iteration of the loop.

We can further understand the break and continue using example.

```
for(int i = 0; i < 5; i++)
{
    if ( i < 3 )
        System.out.println( "Hello" + i );
    else
        break;
}
```

In above loop the Hello will be printed for i = 0, 1 and 2. The loop terminates as soon as (i >= 3) because we used break in else part. Here loop will be executed three times only.

The use of continue explained in following code block.

```
for(int i = 0; i < 5; i++)
{
    if ( i ==3 )
        continue;
    else
        System.out.println( "Hello" + i );
}
```

In above example, the loop will be executed 5 times. However hello will be print only four times. Because when i==3 we use continue that means all the statements in a loop after continue will not be executed and next iteration is started after increasing i.

➤ Labeled loops

Loop can also have a loop inside it. This is called nesting of loop. When we are using nest loop the inside loop is called inner loop and outside loop is called outer loop. When we use break in inner loop the inner loop will be terminated. But if we want to terminate outer loop by using break statement in inner loop, we have to used the concept of labeled loop and continue/break with label.

For example

```
i = 0;
while( i < 3)
{j = 0;
 while( j < 3)
 {
  if(j==2)
   break;
  j++;
 }
 i++;
 }
```

In above example, the inner while loop will be break when j is 2. Inner loop will execute twice. Now if we want to break outer loop when in inner loop j is 2, we should use following code

```
i = 0;
outer: while( i < 3)
 {
  j = 0;
  while( j < 3)
  {
   if(j==2)
    break outer;
   j++;
  }
  i++;
 }
```

Here we have labeled outer loop with label outer: and with break w have to used label of outer loop.

Similarly continue can also be used with labeled loop.

Example:

```
public class Exa2
{
public static void main(String args[])
{
first: for (int i = 0; i < 3; i++)
```

```

    {
      for (int j = 0; j < 3; j++)
      {if(i == 1)
        continue first;
        System.out.print(" [i = " + i + ", j = " + j + " ] ");
      }
    }
  System.out.println();
  second: for (int i = 0; i < 3; i++)
  {
    for (int j = 0; j < 3; j++)
    {if(i == 1)
      break second;
      System.out.print(" [i = " + i + ", j = " + j + " ] ");
    }
  }
}
}

```

```

C:\ajava\oopj>javac Exa2.java

C:\ajava\oopj>java Exa2
[i = 0, j = 0] [i = 0, j = 1] [i = 0, j = 2] [i = 2, j = 0] [i = 2, j = 1
] [i = 2, j = 2]

```

Figure-9 Output of Program

1.13 LET US SUM UP

Java compiler and **interpreter** using javac.exe and java.exe

Java virtual machine: runs a byte code

Features of java: simple, object oriented, distributed, compiled and interpreted, robust, secure, platform independent, multithreaded, portable and dynamic

Running sample java program: program file has .java extension and class name should equal to file name.

Java program structure: various block of java programs

Java tokens: whitespaces, keywords, literals, identifiers/variables etc

Java class libraries: readily available class file which can be imported in program

Data types: byte, short, int, long, float, double, char, and boolean

Operators: arithmetic, assignment, logical, relational, and miscellaneous operators

Conditional statement: if...else, if...else ladder, nested if...else, switch...case

Loops: for loop, while loop do...while loop and for-each loop

Array: one dimensional and multi dimensional arrays

1.14 CHECK YOUR PROGRESS

➤ True-False with reason.

1. Keyword can be identifier.
2. = is assignment operator.
3. ++ will increment operators.
4. For loop can not be terminated until condition is false.
5. Conditional operator can be used using if...else.
6. javac is compile and java is interpreter.
7. While loop is entry control loop.
8. ?: can be replaced with if...else.

➤ Which of the following is valid identifier?

- | | | |
|----------|----------|----------|
| 1. abc | 4. a12 | 7. XYZ |
| 2. Xyx | 5. a_23 | 8. 1plus |
| 3. \$abc | 6. int_1 | |

➤ Match **A** and **B**.

- | A | B |
|--------------------|------------|
| 1) Variable | a) "hello" |
| 2) int literal | b) abc |
| 3) String | c) 23 |
| 4) Boolean literal | d) for |
| 5) Keyword | e) false |

1.15 CHECK YOUR PROGRESS: POSSIBLE ANSWERS

➤ True-False with reason.

1. Keyword can be identifier.
2. = is assignment operator.

3. ++ will increment value of a variable.
4. For loop can not be terminated until condition is false.
5. Conditional operator can be used using if...else.
6. javac is compiler and java is interpreter.
7. While loop is entry control loop.
8. ?: can be replaced with if...else.

➤ Which of the following is valid identifier?

- | | | |
|-----------|-----------|-----------|
| 9. abc | 12. a12 | 15. XYZ |
| 10. Xyx | 13. a_23 | 16. 1plus |
| 11. \$abc | 14. int_1 | |

Answer: abc, Xyz, a12, a_23, int_1 and XYZ are valid identifiers.

➤ Match **A** and **B**.

- | A | B |
|--------------------|------------|
| 1) Variable | a) "hello" |
| 2) int literal | b) abc |
| 3) String | c) 23 |
| 4) Boolean literal | d) for |
| 5) Keyword | e) false |

Answer:

- 1) – b, 2)- c, 3)- a, 4) – e, 5) – d

1.16 FURTHER READING

1. "Java 2: The Complete Reference" by Herbert Schildt, McGraw Hill Publications.
 2. "Effective Java" by Joshua Bloch, Pearson Education.
-

1.17 ASSIGNMENTS

➤ Write java program for following:

- 1) Print largest of two numbers.
- 2) Print largest of three numbers.
- 3) Check number is even or odd.

- 4) Print first five even numbers.
- 5) Print a number in reverse.
- 6) Add n numbers.
- 7) Print first 10 prime numbers.
- 8) Find factorial of a number n.
- 9) Print Fibonacci series upto n elements.
- 10) Print sum of first 10 odd numbers.