Introduction

Origin of Java

Java is the first purely object oriented programming language. It was developed by James Gosling, Patrick Naughton, Chris Warth, Ed Frank and Mike Sheridan of Sun Microsystems in 1991

It was initially called "Oak", but was later renamed as "Java" in 1995.

The motivation behind the development of Java was the need to develop a language that was platform-independent, could be used to develop software that could be embedded on different devices and could adapt to changes in the operating environments. The outcome of the efforts to develop such a language is JAVA

So, what makes Java platform-independent?

The answer is Byte-Code.The source code that we write in Java, is compiled and converted into Byte-Code. This format can be run on any machine irrespective of its environment.

One of the most fundamental things we need to remember while learning Java is that the source file has a .java extension which is loaded into a compiler to be compiled. The output of the compiler is a file that has a .class extension. This class file that is generated is made up of Byte-Code and is responsible for platform independence.

Therefore, using Java, we can develop codes that can be termed as *"Write Once - Run Anywhere"*.

Test Yourself

Q1. What is the type of file that the compiler generates?

Show/Hide Answer

Ans. The output of the compiler is a ".class" file. It has Byte-Code, that is responsible for the platform independence of Java.

First Java program

Now, lets get started with your first Java program. Before that, we must know the three important components that make up Java - JDK, JRE andJVM.

Java Development Kit (JDK) contains all the tools that are required to write and run Java programs. It includes the JRE.

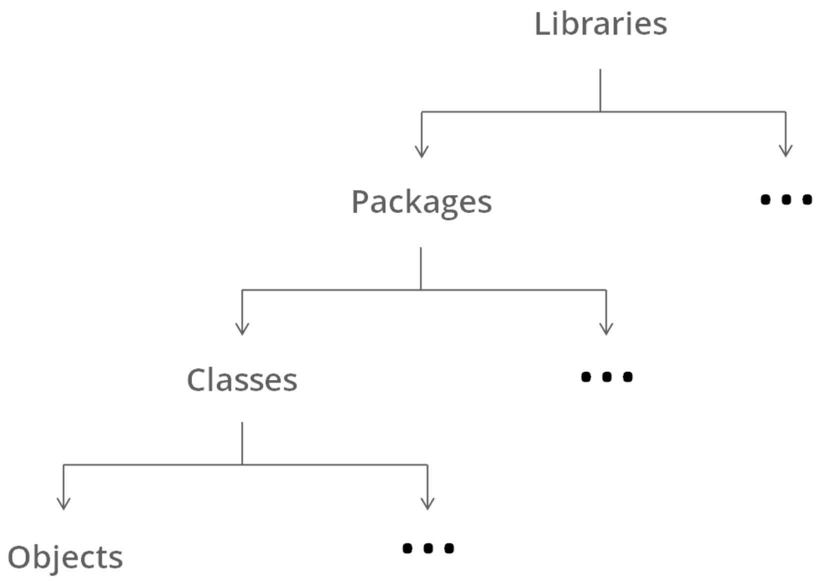
Java Runtime Environment (JRE) is the component that runs all the programs. It consists of JVM, class libraries and supporting tools.

Java Virtual Machine (JVM) is responsible for converting the Byte-Code to machine code depending on the underlying Operating System and Hardware combination.

Note that we must install JDK, JRE and JVM to run Java codes on our systems. You have already installed JDK, and by extension, JRE and JVM, on your system if you have followed the steps in the previous tutorial.

How do we run a Java program?

We use the command javac to compile the source code that we write, and then use the command java to run it.



The above images shows how Java is structured and is to be used. Java consists of several libraries. Each library is a collection of packages. A package consists of a set of related classes, which are essentially blueprints for objects. Each class can have multiple objects. We shall learn more about classes and objects in the next module.

Importing Libraries

We begin every program with an import statement. The Java API has a set of packages that include classes which define the basic functionalities like the Input/Output operations. For instance, the statement "import java.io.*" will import all the classes that are a part of the IO (input/output) package. This is used the code that follows. Similarly, all the graphical functionalities can be used by importing the awt package ie., "import java.awt.*"

The "*" at the end is used when we want to include all the classes present in the package. But we can also make use of a particular class by specifying the name of the class. For e.g. "import java.util.zip" only includes the classes useful for reading and writing in standard ZIP formats.

If we don't specify any import file at the beginning of our code, Java will automatically import the 'java.lang' library. This library is included by default.

Consider the following code:

```
javac filename.java
java filename
```

The following is the code for the classical Hello World program

```
import java.io.*;
public class HelloWorld
{
  public static void main(String args[])
     {
        System.out.println("Hello World!");
     }
}
```

The line System.out.println is used to print the content that is passed as an argument to it. Also make sure that the file has the same name as that of the class that has main() method and is public.

Try it out yourself. Try changing the content that is passed as an argument to the print function.

More exercises

Write a program similar to the one presented to print "Hello" and "World" on two different lines. Hint: The command 'print()' prints the content on a single line, while 'println()' prints content on a new line.

OOP 1

Object oriented programming is one of many ways to organize source code. Java is a purely Object Oriented programming language. But, what are the factors that make a programming language object oriented?

We will discuss three important OOP Principles:

- Encapsulation
- Inheritance
- Polymorphism

Don't worry if you don't understand these concepts completely right now. These will become more clear during the course of this program.

Encapsulation

Instead of defining variables and the functions that operate on them separately and hoping they will be used correctly, object oriented programming explicitly groups them together. Object oriented programming languages have syntax that enables this grouping. This grouping is called Encapsulation

For example, consider the example of a car. In a car, one function 'shifting of gears' does not effect another function, the 'signalling system' and vice versa. However, a driver uses these together. He does not bother about their individual functionalities and how these functionalities are implemented.

Using encapsulation, any programmer can use the code without knowing any details of implementation. In Java, encapsulation is achieved through classes and objects.

Inheritance

Programmers create templates that describe how data and methods will be grouped, called "classes". Specific instances of the data based on these templates are called "objects". Inheritance is a process by which one object acquires the properties of another object. In the hierarchy, the object from which properties are acquired is the parent and that which acquires becomes the child.

We shall deal with inheritance in 'Classes' module.

Test Yourself

Q1. What is the principle of OOP that provides data privacy?

Show/Hide Answer

Ans. The principle of OOP that provides data privacy is Encapsulation.

OOP 2

Polymorphism

Polymorphism literally means the ability to take many forms. In Java, Polymorphism is a property using which, a particular interface can be used for a general range of actions. The particular action that is performed will depend on the scenario.

For example, consider the "move" functionality of human beings. A person, depending on the situation, may move by walking, running or even crawling. This means, depending on the scenario, the same functionality may have different implementations and may result in a different set of actions.

Encapsulation, Inheritance and Polymorphism, when used properly, create a more scalable and robust environment than the available traditional environments.

In the next set of topics, we will plunge into the technical details of Java.

Test Yourself

Q1. What is the principle of OOP through which we can achieve multiple tasks through a single interface?

Show/Hide Answer

Ans. The principle of OOP through which we can multiple functionalities through single interface is Polymorphism.

Data Types

We shall now look at the different data types used in Java. Remember, Java is a strongly-typed language. This means that, in Java, each data type has its own strict definition. There are no implicit data type conversions when any conflicts occur between the data types. Any change in data types should be explicitly declared by the programmer.

Java defines 8 simple data types. They are divided into the following categories:

- Integers

- Floating-Point Numbers
- Characters
- Boolean Type

The details of each of the data types is given below

- Integers: These are of four types: byte, short, int, long. It is important to note that these are signed positive and negative values. Java does not support positive-only numbers. The size of long is 64-bits, int is 32-bits, short is 16-bits and byte is 8-bits.

- Floating-Point: These are also called real numbers and are used for expressions involving fractional precision. These are of two types:float, double. The width of double is 64-bits and that of float is 32-bits.

- Character: We use this data type to store characters. This is not the same as the char in C/C++. Java uses an UNICODE, an internationally accepted character set. Char in Java is 16-bits long while that in C/C++ is 8-bits long.

- Boolean: This is used for storing logical values. A boolean type can have a value of either true or false. This type is generally returned by relational operators.

The following code demonstrates how a variable of any data type is declared.

```
int a;
char ch;
double f1;
boolean b;
```

In the above snippet of code, the values a, ch, f1 and b are variables of the respective data types. We will learn more about variables in the next topic.

Test Yourself

Q1. How many data types does Java have?

Show/Hide Answer
Ans. 8

Variables

A variable is a unit of storage in a Java program. It has a data type, an identifier, and an optional initializer.

- The type of a variable gives information of the data type of the variable.

- Identifier is the name by which any variable is referred to.

The variable when assigned any value is said to be initialized. This can be done at the time of declaration of the variable or/and can be done at any later point of time.

Remember that, any conversion between data types should be explicitly mentioned by the programmer in Java.

The following code shows the division of two integer type variables and the result being assigned to a float type variable.

```
import java.io.*;
public class Division
{
  public static void main(String args[])
  {
    int a, b=2;
    a=9;
    float result;
    result = (float) a/b;
    System.out.println(result);
  }
}
```

Note here, that in line 9, we have given "float" within parenthesis. This is because, Java is a strongly-typed language and implicit type conversion is not possible. The result of operation on two integers will not give the fractional part by default and we have made the type conversion explicitly. Try this code without the (float in line 9) and observe the difference betweeen the two outputs.

The result of operation without type conversion is 4.0, whereas with type conversion, the result will be 4.5.

Play with the code. Try changing the variables' values and data types and perform other mathematical operations.

Inputs

There are various ways to read input from the keyboard, the java.util.Scanner class is one of them. The Scanner class breaks the input into tokens using a delimiter which is whitespace bydefault. It provides many methods to read and parse various primitive values.

There is a list of commonly used Scanner class methods:

```
public String next(): it returns the next token from the scanner.
public String nextLine(): it moves the scanner position to the next
line and returns the value as a string.
public byte nextByte(): it scans the next token as a byte.
public short nextShort(): it scans the next token as a short value.
public int nextInt(): it scans the next token as an int value.
public long nextLong(): it scans the next token as a long value.
public float nextFloat(): it scans the next token as a float value.
public double nextDouble(): it scans the next token as a double value.
```

Example of java.util.Scanner class:

Let's see the simple example of the Scanner class which reads the int, string and double value as an input:

```
 import java.util.Scanner;

class ScannerTest{

       public static void main(String args[]){



       Scanner sc=new Scanner(System.in);
       System.out.println("Enter your rollno");

       int rollno=sc.nextInt();

       System.out.println("Enter your name");

       String name=sc.next();

       System.out.println("Enter your fee");

       double fee=sc.nextDouble();


       System.out.println("Rollno:"+rollno+" name:"+name+" fee:"+fee);


  }

}
```

We are using Scanner class to get input from user. The next program firstly asks the user to enter a string and then the string is printed, then an integer and entered integer is also printed and finally a float and it is also printed on the screen. Scanner class is present in java.util package so we import this package in our program. We first create an object of Scanner class and then we use the methods of Scanner class.

Consider the statement Scanner a = new Scanner(System.in); Here Scanner is the class name, a is the name of object, new keyword is used to allocate the memory and System.in is the input stream. Following methods of Scanner class are used in the program below :-

1. nextInt to input an integer
2. nextFloat to input a float
3. nextLine to input a string

```
import java.util.Scanner;
class GetInputFromUser
{ public static void main(String args[])
{ int a;
float b;
String s;
Scanner in = new Scanner(System.in);
System.out.println("Enter a string");
s = in.nextLine();
System.out.println("You entered string "+s);
System.out.println("Enter an integer");
a = in.nextInt();
System.out.println("You entered integer "+a);
System.out.println("Enter a float");
b = in.nextFloat();
System.out.println("You entered float "+b);
}
}
```

Command Line Arguments

Your Java application can accept any number of arguments from the command line. Command line arguments allow the user to affect the operation of an application.

When invoking an application, the user types the command line arguments after the application name. For example, suppose you had a Java application, called Sort, that sorted lines in a file, and that the data you want sorted is in a file named ListOfFriends. If you were using DOS, you would invoke the Sort application on your data file like this:

C:\> java Sort ListOfFriends

In the Java language, when you invoke an application, the runtime system passes the command line arguments to the application's main method via an array of Strings. Each String in the array contains one of the command line arguments. In the previous example, the command line arguments passed to the Sort application is an array that contains a single string: "ListOfFriends".

This simple application displays each of its command line arguments on a line by itself:

```
class Echo {
public static void main (String args[]) {
for (int i = 0; i < args.length; i++)
System.out.println(args[i]);
}
}
```

Try this: Invoke the Echo application with the command line shown in this DOS example:

```
C:\> java Echo Drink Hot Java
Drink
Hot
Java
```

You'll notice that the application displays each word--Drink, Hot, and Java--on a line by itself. This is because The Space Character Separates Command Line Arguments.

Test Yourself

Q1. Which other class we can use in place of Scanner class for user input?

Show/Hide Answer
Ans. Buffered Reader

Arrays

Until now, we have seen data types and variables that are only a single element. Let us now look at Arrays. An Array is a collection of homogeneous data elements. They are referred to by a collective common name, which is the name of the array. We access a particular element of the array using the index element of the array.

We can broadly classify arrays into two sets. These are:

- One Dimensional Arrays
- Multi-Dimensional Arrays

One Dimensional Arrays

A one-dimensional array can simply be considered as a list of homogeneous data elements. We create an array by creating an array variable of the required data type. The following code shows how an array is defined in Java.

```
int sample_array[];
sample_array = new int[10];
```

The above code shows the declaration of an array in Java. In the first line, an array variable 'sample_array' is created but no space is allocated to it. In case of the second line, the array 'sample_array' is allocated with ten address spaces.

Remember that, to use an array, it should be allocated some space. This is done using the new keyword as demonstrated.

The following code snipped shows another way to declare the same array.

```
int sample_array[] = new int[10];
```

The index used for accessing the array ranges from 0 to (size of the array)-1. For instance, if we want to access the third element from an array A, then we can use use A[2].

Multi-Dimensional Arrays

For the scope of this program, we shall consider 2-Dimensional arrays. This can be thought of as an array of one-dimensional arrays. An example of a two-dimensional array is a matrix. The declaration and usage of a multi-dimensional array are similar to those of a one-dimensional array. The following code demonstrates the declaration of a 2-dimensional array.

```
int a[][];
a = new int[2][3];
```

In case of the two-dimensional array as a matrix, in the second line of declaration above, the first dimension represents the number of rows and the second dimension represents the number of columns. We have two indices here one for one dimension each.

The following code shows how to declare an array and access an element from the array. Here we print the fourth element of the array.

```
import java.io.*;
public class Sample_Array
{
  public static void main(String args[])
  {
    int a[];
    a = new int[5];
    a[0]=1;
    a[1]=2;
    a[2]=3;
    a[3]=4;
    a[4]=5;
    System.out.println(a[3]);
  }
}
```

In the above code, first we have declared the array, then allocated space to it and then initialized the elements of the array. At the end, we have printed the fourth element of the array.

Try to modify the code by changing the data types and dimensions of the array and try to retrieve the element(s) of your choice.

Test Yourself

Q1. What will this code print?

```
int arr[] = new int [5];
System.out.print(arr);
```

Show/Hide Answer

Ans. Garbage Value because arr is an array variable, it is pointing to array if integers. Printing arr will print garbage value. It is not same as printing arr[0].

More Exercises

1. Write a program to add two matrices
2. Write a program to multiply two a matrices
3. Write a program to multiply each element in an array with its position and replace the number with the result in the same array
4. Write a program to allocate memory for 10 integers in an array and initialize and display each element.
5. WAP using multidimensional array to give the output as:
   Mr. Smith
   Ms. Jones
6. Write a program to sort an array.
7. Write a program to find the largest element in an array.

Operators 1

Every action that we perform has an underlying operation associated with it. We will now look at the operators which perform these operations. Java has a rich set of operators that help us perform required tasks with ease. In this section and the following one, we shall study three categories of operators: Arithmetic, Relational and Logical.

Arithmetic Operators

Arithmetic Operators are mainly used in mathematical expressions. The following list describes some of the arithmetic operators and their usage.

- + : This operators adds the values that are provided as operands to it.
- - : This operators subtracts the values that are provided as operands to it.
- * : This operators multiplies the values that are provided as operands to it.
- / : This operators divides the values that are provided as operands to it and returns the quotient.
- % : This operators divides the values that are provided as operands to it and returns the remainder.
- ++ : This operators increments the value of the operand by 1. It operates on only a single operand.
- += : This operators adds the value that is on the LHS to the value on the RHS and assigns the sum to the operand on LHS.
- -- : This operators decrements the value of the operand by 1. It operates on only a single operand.
- -= : This operators subtracts the value that is on the RHS from the value on the LHS and assigns the difference to the operand on LHS.
- *= : This operators multiplies the value that is on the LHS and the value on the RHS and assigns the product to the operand on LHS.
- /= : This operators divides the value that is on the LHS by the value on the RHS and assigns the quotient to the operand on LHS.
- %= : This operators divides the value that is on the LHS by the value on the RHS and assigns the remainder to the operand on LHS.

It is important to note that the above operators can be used only on numeric values. They cannot be used with boolean type. However, they can be used on char type. In such a case, the ASCII values of characters are used while performing the operation.

The following code shows the usage of some of the arithmetic operators that we have discussed above. The results are printed after performing the operations.

```
import java.io.*;
public class Arithmetic
{
  public static void main(String args[])
  {
    int a = 11, b = 2;
    int sum, product, difference, remainder, quotient;
    sum = a + b ;
    difference = a - b;
    product = a * b;
    quotient = a / b;
    remainder = a % b;
    System.out.println("Sum is " + sum);
    System.out.println("Difference is " + difference);
    System.out.println("Product is " + product);
    System.out.println("Quotient is " + quotient);
    System.out.println("Remainder is " + remainder);
  }
}
```

In the above code, we have seen the usage of some of the arithmetic operators. Note that, the + sign in the print statement is used to append the result that is obtained to the string that is provided within double quotes.

Try changing the values and using other arithmetic operators to better understand how operators behave.

Test Yourself

Q1. What is the output of this code snippet?

```
int x=5;
int y=x+++x;
System.out.println(y);
```

Show/Hide Answer

Ans. 11

Operators 2

Relational Operators

These operators are primarily used to determine the relationship between the two operands that they take. They return boolean value as a result.

The following are the list of relational operators and their functionalities.

==	Checks if two operands are equal and returns true if they are equal.
!=	Returns true if the two operands are unequal.
>	Returns true if the operand on LHS is greater than the one on RHS.
<	Returns false if the operand on LHS is greater than or equal to the one on RHS.
>=	Returns true if the operand on LHS is greater than or equal to the one on RHS.
<=	Returns false if the operand on LHS is greater than the one on RHS.

These relational operators are useful when we use control statements, where the program flow is based on some condition. Remember that, any data type in Java can be compared using the equality test. Only numeric types can be compared using the "greater-than" or "lesser than" operators.

Logical Operators

These operators operate solely on the boolean operands. The result that they return is also of Boolean type.

The following are the most widely used Logical Operands.

&	AND - Returns true only if both the operands are true. Returns false in all other cases.
|	OR - Returns false only if both the operands are false. Returns true in all other cases.
!	NOT - It works on a single operand. Returns the complement value of the value of the operand.
^	XOR - Returns false if the both the operands are true or false. In other cases, it returns true.

There are two more operators short-circuit AND (&&), short-circuit OR (||). When we use these, Java will not bother to evaluate the right-hand operand when the result can be determined by the left-hand operand independently.

The following code presents how the relational and logical operators are used.

```
import java.io.*;
public class Operators
{
  public static void main(String args[])
  {
    int a = 3, b = 4;
    boolean c = true, d = true;
    if(a<b)
      System.out.println("Relational Operators!");
    if(c & d)
      System.out.println("Logical Operators!");
  }
}
```

In the above code, first, we check whether the variable a (asssigned value 3) is lesser than variable b (assigned value 4) and if the operation returns true, then we print the text "Relational Operators!". Next, we have checked the logical condition (whether both the boolean variables c and d are true) and have printed the text "Logical Operators!".

Try to change the operators and variables and observe the output.

Test Yourself

Q1. Which of the operators can skip evaluating right hand operand?

Show/Hide Answer

Ans. Operator short circuit and, &&, and short circuit or, ||, skip evaluating right hand operand when output can be determined by left operand alone.

Control Statements 1

In any programming language, the flow of the program is necessary for executing the program and achieving desired results. Control Statements manage the flow of execution of the program and decide the branching of the logical flow.

For the scope of this course, we will deal with four control statements; namely if-else statement, for-loop, while-loop and do-while loop.

if-else statement

if-else statement is a conditional branch statement. It helps us to choose between two available choices. The syntax of if-else statement is shown in the following code.

```
if(condition)
  statement 1;
else
  statement 2;
```

In the above snippet, the condition is first resolved. If it turns out to be true, then the statements that are inside the 'if' condition are executed (statement 1 in this case). If the condition turns out to be false, then the statements inside the 'else' condition are executed (statement 2, in this case).

for-loop

The if-else statement we studied above is used for branching purpose. On the other hand, the for-loop is used for iteration. In other words, we use the for loop to repeat a set of statements a specific number of times. This is a counter controlled loop. The syntax of for-loop is as shown in the following code.

```
for(initialization, condition, iteration)
{
  set_of_statements;
}
```

In the code above, initialization is where we set the loop-counter's initial value. Everytime the loop runs, the condition is checked to see if it still holds true. In the iteration part, the loop-counter's value is updated. If the condition holds even after updating the loop-counter's value (iteration), then the set of statements within the body of the loop are executed. If the condition fails, the body of the loop shall not be executed.

The following code illustrates the use of the two control statements that we discussed above. Here, we will print the first five natural numbers if a condition holds else print a statement that the condition did not hold.

```
import java.io.*
public class ControlStatements
{
  public static void main(String args[])
  {
    int a = 3, b = 4;
    if(a<b)
    {
        System.out.println("a is less than b");
    }
    else
    {
        System.out.println("a is not less than b");
    }
    for(int i=1;i<=5;i++)
    {
        System.out.println(i);
    }
  }
}
```

In the above code, we have defined two variables (a and b), checked which variable is greater than the other (using an if loop) and printed the result accordingly. Also, we have printed the first five natural numbers using a for loop. The value of counter variable i is first set and then the condition is checked for each iteration. The value gets incremented for each iteration.

Now, try out yourself by modifying the conditions and the statements in the loop body. Observe how each of the statements will effect the output.

More Exercises

1. Write a program that prints the first 10 even numbers.
2. Write a program that produces the following output:

```
3, 7, 15, 31, 63, 127
```

3. Make an array which stores the days of the week. Then use a for loop to print the contents of the array.
4. WAP to create a grade sheet where if score is greater than or equal to 90 then grade is A, if it is greater than or equal to 80 grade is B, if greater than or equal to 70 then grade is C, and if greater than or equal to 60 grade is D else grade is F.
5. Calculate the electricity bill of a person,the following conditions are provided:

   Conditions

   ```
   Take the units consumed

   Accordingly calculate amt

   for 1st 100 units @ 1Rs.(any currency)/unit

   for next 100 units @ 2 Rs./unit

   for next 100 units @ 3 Rs./unit

   for next 200 units @ 4 Rs./unit

   for next units @ 5 Rs./unit

   tax to be added in final amount @ 10%.

   Meter charge 50 Rs. extra.

   Print the bill....
   ```

Test Yourself

Q1. Which of these are selection statements in Java?

```
a) for()
b) continue
c) if()
d) break
```

Show/Hide Answer
Ans. c, as for is a looping statement and continue, break statements are jump statements.

Control Statements 2

Next, we shall look at the while loop and the do-while loop.

While loop

This is one of the most basic looping controls. This repeats the set of statements present within the body of the loop until the condition specified in the parenthesis of the loop is true. Once the condition fails, the loop terminates.

The syntax of a while loop is as shown in the following code.

```
while(condition)
{
  set_of_statements;
}
```

The body of the loop iterates only when the looping condition is satisfied. If a counter variable is used, then it is updated in the body and is again checked against the condition.

do-while loop

do-while loop has almost the same structure and function as that of a while loop. The major difference between the two loops is that, in this case, the looping condition is present at the end of the body of the loop. Therefore, the body of the loop is always executed once before the condition is checked. The second iteration may or may not occur depending on the looping condition. The following gives the basic structure ofdo-while loop.

The syntax of a do-while loop is as shown in the following code.

```
do{
  set_of_statements;
}while(condition);
```

So, in this, the body of the loop is first executed and then the condition is checked. it can be seen that the body of the loop is executed at least once irrespective of whether the condition is satisfied or not. This is the major difference between the while and the do-while loops.

Now let us repeat the exercise of printing the first 5 natural numbers using both while and do-while loops. The following code illustrates this.

```
import java.io.*;
public class NaturalNumbers
{
  public static void main(String args[])
  {
    int i=1, j=1;
    while(i<6)
    {
      System.out.println(i);
      i++;
    }
    do
    {
      System.out.println(j);
      j++;
    }while(j<6);
  }
}
```

The first five natural numbers are printed using the while and the do-while loops. Note that, we have used different loop counter variables i,j for the two loops. The reason for using two different variables is that, the value of variable i will be 6 by the time the first loop is terminated. If we use the same variable, the body of do-while loop is executed only once and gets terminated. In order to avoid this, we have used two different loop counter variables. Alternatively, we could have reset the value of variable i after the first loop.

Try to modify the code with other body statements and also use a single loop variable and verify the differences between both the outputs obtained. Play around with these codes as much as possible to get a better understanding of the basic concepts before moving on the the next set of modules.

Test Yourself

Q1. Which looping construct will you use so that the code will execute at least once?

Show/Hide Answer

Ans. do..while.

More Exercises

1. Take the scores of ten students in an array and find out the highest score from the set of scores. Further, find the average of the scores of all the students. Hint: For finding the highest of all the scores, iterative approach can be used and the highest score is found by traversing through all the elements of the array.
2. Write a program to reverse a number
3. Write a program to check whether a given string is a palindrome or not?
4. Write a program to find whether a number is Armstrong number or not? (hint :Armstrong number is a number whose digits when powered with the total number of digits would sum to the original number eg: 153=1^3+5^3+3^3)