UNIT 3: LOOPS AND SELECTION STATEMENTS

Unit Structure

3.0 Learning Objectives

After learning this unit, you will be able to understand:

- Explain loops- for, while and do-while loop
- Describe nested loops
- Discuss Selection statement- if, if else, nested if, switch statement and recursion
- Define Arrays- one dimensional and multidimensional
- Illustrate switch statement

3.1 Introduction

A programming loop is a control structure that allows a programmer to execute the same instruction or group of instructions over and over until some condition is met. All loops have a basic structure to them, though there are many types.

We can use Java JDBC Select statement in a java program to retrieve the data and display it for the respective Tables. JDBC returns results in a Result Set object, so we need to declare an instance of the class Result Set to hold our results. Select is the SQL keyword that performs a query

3.2 Loops

The process of executing a block of statements a number of times is known as looping or iterating. There are mainly three types of loops in Java which are used in programs when the same set of statements are executed a number of times.

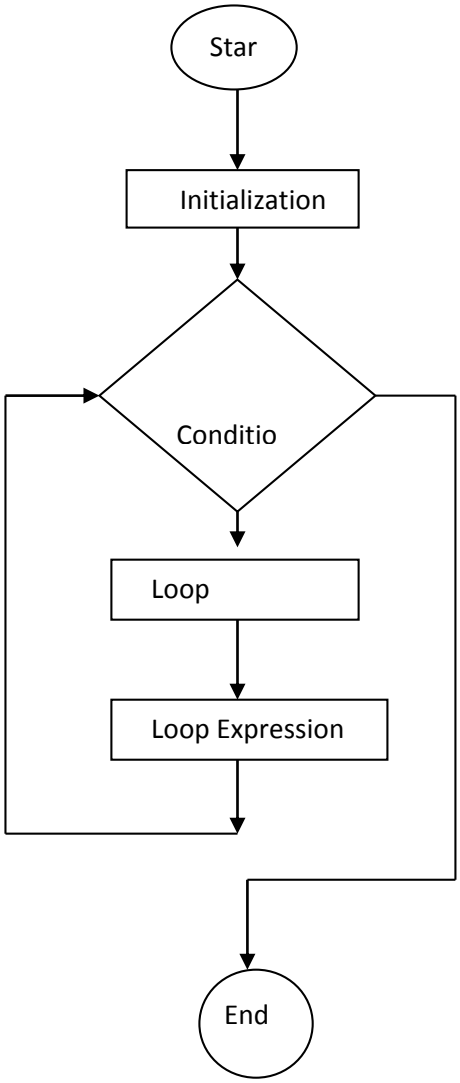
For Loop

The for loop is used to execute the same set of statements a number of times. With for loop, the initial value of variable is specified with the condition which gets checked so that the given set of statements be executed till the value of variable is less than/greater than or equals to it, the increment/decrement value of the variable is also specified which keeps on increasing/decreasing the value of variable on each iteration.

The syntax of for loop is given below:

```
for (initial value; condition; increment/decrement value);
{       // start of for loop
//statements to be executed
}       //end of for loop
```

The flow chart for execution of for loop is given below:



False

True

For example,

```
for ( int i=1; i<=5; i++)
{
System.out.println ( i );
}
```

The above code fragment will display numbers from 1 to 5.

Here is the flow of control of For Loop -

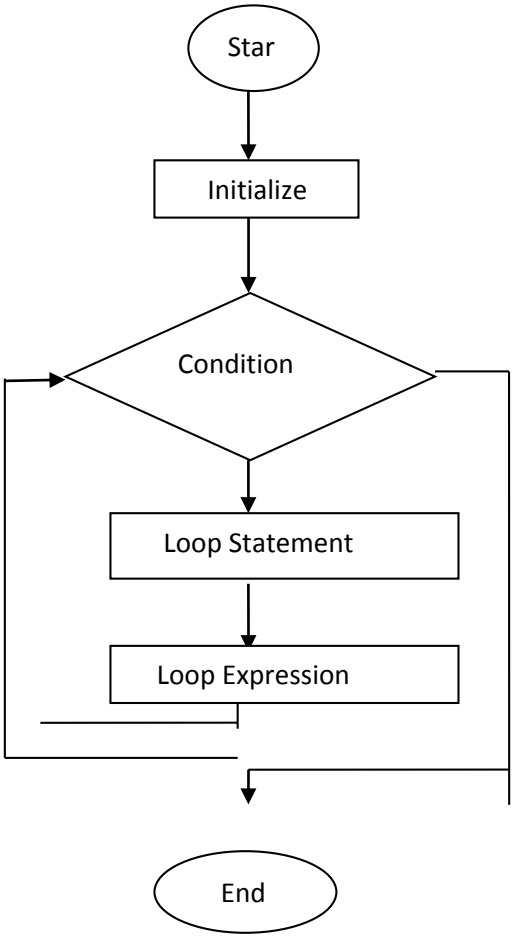
1. The initialization step is executed first and only once. This step allows you to declare and initialize any loop control variables. You are not required to put a statement here, as long as a semicolon appears.
2. Next, the Boolean expression is evaluated. If it is true, the body of the loop is executed. If it is false, the body of the loop does not execute and flow of control jumps to the next statement past the for loop.
3. After the body of the for loop executes, the flow of control jumps back up to the update statement. This statement allows you to update any loop control variables. This statement can be left blank, as long as a semicolon appears after the Boolean expression.
4. The Boolean expression is now evaluated again. If it is true, the loop executes and the process repeats itself (body of loop, then update step, then Boolean expression). After the Boolean expression is false, the for loop terminates.

While Loop

The while loop is used to execute the same set of statements a number of times. In while loop, first the condition is specified with while and the statements specified under it gets executed a number of times. The syntax of same is given below:

```
while (condition)
{
//loop statements
//increment/decrement value
}       //end of while loop
```

The execution of statements in while loop is illustrated using the given flowchart:



Flowchart for while loop

The statements within the while loop gets executed till the condition being tested remains true. As soon as it becomes false, the control of the program passes to the first statement that follows the body of the loop.

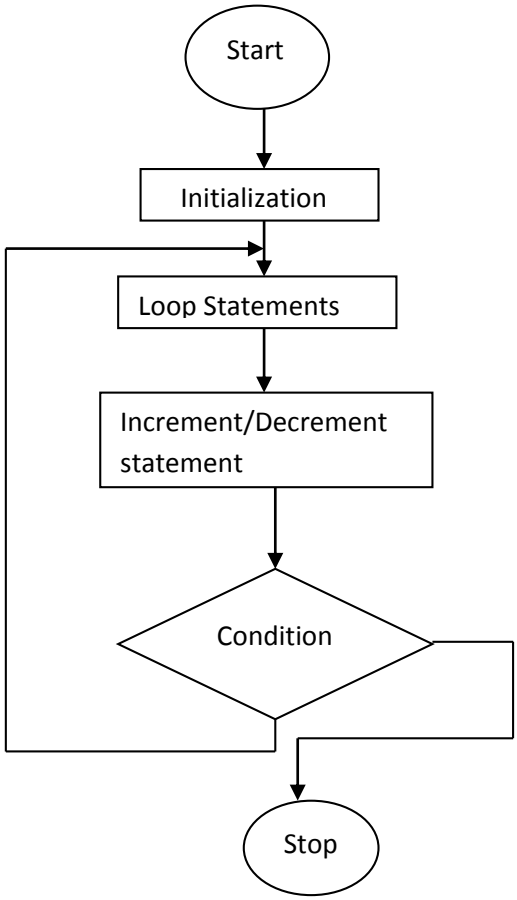
If the statements within the parenthesis of while loop is single then the use of parenthesis is optional.

Do - While Loop

Just like while loop, the do-while loop is also used to execute the same set of statements a number of times. The syntax of do-while is given below:

```
Initialisation
do
{
// loop statements
//increment/decrement value;
} while (condition);
```

The process of execution of statements of do-while loop is illustrated below:



The while loop is also called entry controlled loop whereas the do-while loop is called as exit-controlled loop. There is a small difference between both these loops. The while loop first tests the condition and then executes the statements and do-while loop first executes the statements and then checks the condition. The differences between them are shown below in the given table:

Table 3.1: Do-while loop Vs While loop

Do-while loop	**While loop**
Also called exit-controlled loop	Also called entry controlled loop
In do-while loop, first the statements get executed and then the condition is checked	First the condition gets checked and then the statements are executed
The statements get executed even if the condition is wrong	The statements do not get executed if the condition is wrong

Check your progress 1

1. Give the difference between do while and while loop.
2. Explain for loop with the help of flowchart.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

3.3 Nested Loops

A loop within a loop is called nested loop. When two loops are nested, the outer loop takes control of the number of complete repetitions of the inner loop. While all types of loops may be nested, the most commonly used nested loop is for loop.

When working with nested loop, the outer loop is only changed once the inner loop is completely finished.

Check your progress 2

1. Explain nested loops.
2. Which is the commonly used nested loop?

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

3.4 Selection Statements

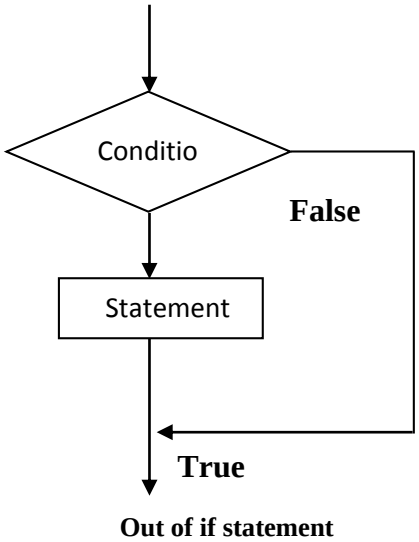
The selection statements are used for decision making purpose. We invoke the jdbc select query (executequery) method, using the jdbc select data statement as the parameter. The tabular results of the query are captured in the ResultSet object, results. Note that executeQuery() always returns a ResultSet although it need not have any rows in it. These statements are discussed below:

If Statement

The 'if' statement is used to check a condition; if that condition is true then the statement specified after if statement gets executed. The syntax of if statement is given below:

```
if (condition)
{
//statement 1
//statemen
……..
……..
//statement n
}
```

If there is a single statement then we need not to specify the braces but if there is more than one statement then it has to be enclosed in pair of braces. The process flow of if statement is explained in the given flowchart:



The If- Else Statement

The if-else statement is also used to check a condition just like if statement but if the given statement is not true then the statements specified in the else part gets executed.

The syntax of if-else statement is given below:

```
if ( condition)
{
//statements
//statements
}

else

{
//statements
//statements
}
}
```

Nested If

An if inside another if is called as nested if statement. The syntax of nested if statement is given below:

```
if (condition)

{

if (condition)
```

```
{

//statements

}

//statements

}

else
{

if (condition)

{

//statements

}

//statements

}
```

Switch Statement

The switch statement tests the value of a variable and based on that executes the corresponding case statement. The last statement of switch-case statement contains a default statement, which gets executed when all the case statements specified does not match with the value of the variable specified.

The syntax of switch-case statement is given below:

```
switch (expression)
{
case                cond1:          code_block_1
case                cond2:          code_block_2
...
case                condn:          code_block_n
default:                            code_block_default
}
```

Here, expression is an integral expression (like int, char, short and byte). In each case statement within the switch statement, a comparison is made with the value entered by the user and the value specified with case. If the comparison evaluates to true, the code specified within that block is executed, otherwise it goes to next case statement. The last default statement gets executed if none of the conditions match.

The following rules apply to a switch statement:

- The variable used in a switch statement can only be a byte, short, int, or char.
- You can have any number of case statements within a switch. Each case is followed by the value to be compared to and a colon.
- The value for a case must be the same data type as the variable in the switch and it must be a constant or a literal.
- When the variable being switched on is equal to a case, the statements following that case will execute until a break statement is reached.
- When a break statement is reached, the switch terminates and the flow of control jumps to the next line following the switch statement.
- Not every case needs to contain a break. If no break appears, the flow of control will fall through to subsequent cases until a break is reached.
- A switch statement can have an optional default case, which must appear at the end of the switch. The default case can be used for performing a task when none of the cases is true. No break is needed in the default case.

The Break Statement

The break statement sends the control of program out of the loop. This statement is useful when in certain instances we want to exit out of the loop instantly. As soon as this keyword is encountered inside any loop, the control of the program automatically passes to the next statement after the loop.

```
public class Test {
   public static void main(String args[]){
      int [] numbers = {10, 20, 30, 40, 50};

      for(int x : numbers ){
         if( x == 30 ){
            break;
         }
         System.out.print( x );
         System.out.print("\n");
      }
   }
}
```

This would produce following result:

```
10
20
```

The Continue Statement

The continue statement sends the control of program to the beginning of the loop. As soon as this statement occurs in the program, the rest of the statements written after continue statement is bypassed and the control of the program is sent to the beginning of the loop.

For example,

//Code to print numbers from 1 to 5 except 4

int i=1

while (i<=5)

{

```
if (i==4)

continue

System.out.println (i)

i++

}
```

Recursion

Recursion is when a function calls itself. That is, in the course of the function definition there is a call to that very same function. At first this may seem like a never ending loop, or like a dog chasing its tail. It can never catch it. So too it seems our method will never finish. This might be true is some cases but in practice, we can check to see if a certain condition is true and in that case exit (return from) our method. The case in which we end our recursion is called a base case. Additionally, just as in a loop, we must change some value and incrementally advance closer to our base case.

Consider this method:

```
void myMethod (int counter)
{
if(counter == 0)
return;
else
{
System.out.println("hello" + counter)
myMethod(--counter)
System.out.println(""+counter)
return
```

```
}
```

```
}
```

If the method is called with the value 4, what will the output be? Explain. The above recursion is essentially a loop like a for loop or a while loop. When do we prefer recursion to an iterative loop? We use recursion when we can see that our problem can be reduced to a simpler problem that can be solved after further reduction.

Every recursion should have the following characteristics:

1. A simple base case which we have a solution for and a return value.
2. A way of getting our problem closer to the base case, i.e., a way to chop out part of the problem to get a somewhat simpler problem.
3. A recursive call which passes the simpler problem back into the method.

The key to thinking recursively is to see the solution to the problem as a smaller version of the same problem. The key to solving recursive programming requirements is to imagine that your method does what its name says it does even before you have actually finish writing it. You must pretend the method does its job and then use it to solve the more complex cases. The same is explained below:

Identify the base case(s) and what the base case(s) do. A base case is the simplest possible problem (or case) your method could be passed. Return the correct value for the base case. Your recursive method will then be comprised of an if-else statement where the base case returns one value and the non-base case(s) recursively call(s) the same method with a smaller parameter or set of data. Thus, you decompose your problem into two parts: (1) The simplest possible case which you can answer (and return for) and (2) all other more complex cases which you will solve by returning the result of a second calling of your method.

This second calling of your method (recursion) will pass on the complex problem but reduced by one increment. This decomposition of the problem will actually be a complete, accurate solution for the problem for all cases other than the base case. Thus, the code of the method actually has the solution on the first recursion.

Check your progress 3

1. Explain the characteristics of recursion.
2. Write the rules for switch statement.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

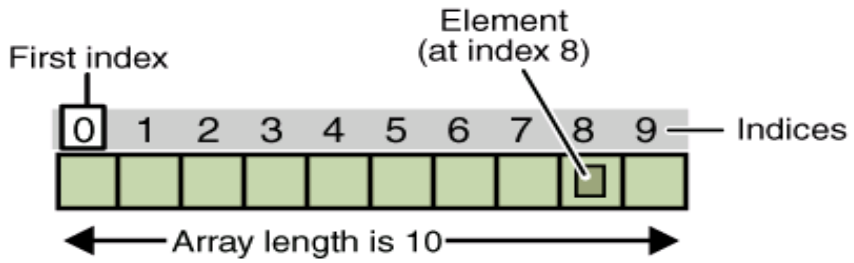
.....

.....

3.5 Arrays

An array is a collection of similar types of variables which are referenced under a single name. It can also be defined as a collection of homogeneous cells inside computer's memory.

To understand the concept, let us take an example. Consider that you want to store the marks of 50 students and display them. Now, the simple way which you have studied till now is to take 50 separate variables, store 50 numbers in them and then display their values. But this will make your program very lengthy and complicated, so, in order to reduce the number of statements in a program, the concept of arrays is used.



Each item of an array is called an element and each element is accessed by its index number. As shown in the above figure, the numbering begins with 0 and the total length of the array is 10. So, the 1st element is stored at 0th index, 2nd at 1st index and so on. The last element that is 10th element is stored at 9th index.

These arrays are classified as:

a. One-dimensional array

b. Multi-dimensional array

a. **One-Dimensional Arrays -** A one-dimensional array contains single row or column. The syntax of declaring single-dimensional array is:

data type varname[size];

Here, type declares the data type of the array, varname is the name of the array and size is the total size of the array.

An array can also be declared as:

int student[]= new int [5];

int [ ] student = new int [5];

Processing Arrays - When processing array elements, we often use either for loop or for each loop because all of the elements in an array are of the same type and the size of the array is known.

Example:

Here, is a complete example of showing how to create, initialise and process arrays:

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

```
{
double mylist [ ] = { 4, 6,2,9,7 };
//Print all the array elements
for (int i=0;i< myList.length;i++)
{
System.out.println (myList[i] + " ");
}
//Summing all elements
double total=0;
for( int i=0; i<myList.length;i++)
{
total+= myList[i];
}
System.out.println ("Total is" + total);

//Finding the largest element
double max=myList[0];
for (int i=1; i<myList.length;i++)
{
if (myList[i] > max)
max = myList[i];
}
System.out.println ("Max is" + max);
}
}
```

This would produce following result:

```
4
6
2
```

9

7

Total is 28

Max is 9

Passing Arrays to Methods - Just as you can pass primitive type values to methods, you can also pass arrays to methods. For example, the following method displays the elements in an int array:

```
public static void printArray (int [ ] array)
{
for (int i=0; i<array.length;i++)
{
System.out.println (array[i] + " " );
}
}
```

You can invoke it by passing an array. For example, the following statement invokes the printArray method to display 3, 1, 2, 6, 4 and 2:

```
printArray (new int [ ] { 3, 2, 5, 6,9} );
```

The Arrays Class - The java. util. Arrays class contains various static methods for sorting and searching arrays, comparing arrays and filling array elements. These methods are overloaded for all primitive types.

SN	Methods with Description
1	public static int binarySearch(Object[] a, Object key) Searches the specified array of Object (Byte, Int , double etc) for the specified value using the binary search algorithm. The array must be sorted prior to making this call. This returns index of the search key, if it is contained in the list; otherwise, (-(insertion point + 1).

2	public static boolean equals(long[] a, long[] a2) Returns true if the two specified arrays of longs are equal to one another. Two arrays are considered equal if both arrays contain the same number of elements and all corresponding pairs of elements in the two arrays are equal. This returns true if the two arrays are equal. Same method could be used by all other primitive data types (Byte, short, Int etc.)
3	public static void fill(int[] a, int val) Assigns the specified int value to each element of the specified array of ints. Same method could be used by all other primitive data types (Byte, short, Int etc.)
4	public static void sort(Object[] a) Sorts the specified array of objects into ascending order, according to the natural ordering of its elements. Same method could be used by all other primitive data types (Byte, short, Int etc.)

b. **Multi-dimensional Arrays -** If an array contains multiple rows and multiple columns then it is called as multi-dimensional array or two-dimensional array.

The syntax of declaring two-dimensional array is:

data type array name[][] = new int [size of row][size of column];

The total number of elements which can be stored in 2-D array can be obtained by the given formula:

Size of the array=number of rows X number of columns

For example,

int student[][] = new int [3][4]

The above statement creates a two-dimensional array of student name of integer data type with 3 rows and 4 columns, that is, the total number of elements which can be stored in this array is 12.

Check your progress 4

1. Explain the process of passing arrays to methods.
2. Discuss the process of using methods in arrays.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

3.6 Let Us Sum Up

This gave us lot of insight in to the Java programming aspects with clarity such as loop, the process of executing a block of statements a number of times is known as looping or iterating. There are mainly three types of loops in Java which are used in programs when the same set of statements are executed a number of times.

1. **FOR LOOP -** The for loop is used to execute the same set of statements a number of times. With for loop, the initial value of variable is specified with the condition which gets checked so that the given set of statements be executed till the value of variable is less than/greater than or equals to it, the increment/decrement value of the variable is also specified which keeps on increasing/decreasing the value of variable on each iteration. We need to understand and apply control of FOR LOOP.

2. **WHILE LOOP -** The while loop is used to execute the same set of statements a number of times. In while loop, first the condition is specified with while and the statements specified under it gets executed a number of times.

3. **DO-WHILE LOOP -** Just like while loop, the do-while loop is also used to execute the same set of statements a number of times. The syntax of do-while is given below:

4. **NESTED LOOP -** A loop within a loop is called nested loop. When two loops are nested, the outer loop takes control of the number of complete repetitions of the inner loop. While all types of loops may be nested, the most commonly used nested loop is for loop. When working with nested loop, the outer loop is only changed once the inner loop is completely finished.

5. **THE IF-ELSE STATEMENT -** The if-else statement is also used to check a condition just like if statement but if the given statement is not true then the statements specified in the else part gets executed.

6. **NESTED IF-** If inside another if is called as nested if statement.

7. **SWITCH STATEMENT -** The switch statement tests the value of a variable and based on that executes the corresponding case statement. The last statement of switch-case statement contains a default statement, which gets executed when all the case statements specified does not match with the value of the variable specified. At this point of time we need to understand and follow rules to apply to a switch statement:

8. **THE BREAK STATEMENT -** The break statement sends the control of program out of the loop. This statement is useful when in certain instances we want to exit out of the loop instantly. As soon as this keyword is encountered inside any loop, the control of the program automatically passes to the next statement after the loop.

9. **THE CONTINUE STATEMENT -** The continue statement sends the control of program to the beginning of the loop. As soon as this statement occurs in the program, the rest of the statements written after continue statement is bypassed and the control of the program is sent to the beginning of the loop.

10. **RECURSION -** Recursion is when a function calls itself. That is, in the course of the function definition there is a call to that very same function. At first this may seem like a never ending loop, or like a dog chasing its tail. It can never catch it. So too it seems our method will never finish. This might be true is some cases but in practice, we can check to see if a certain condition

is true and in that case exit (return from) our method. The case in which we end our recursion is called a base case. Additionally, just as in a loop, we must change some value and incrementally advance closer to our base case.

11. **ARRAY -** An array is a collection of similar types of variables which are referenced under a single name. It can also be defined as a collection of homogeneous cells inside computer's memory. a. One-dimensional array and b. Multi-dimensional array

12. **PROCESSING ARRAYS -** When processing array elements, we often use either for loop or for each loop because all of the elements in an array are of the same type and the size of the array is known.

13. **PASSING ARRAYS TO METHODS -** Just as you can pass primitive type values to methods, you can also pass arrays to methods. For example, the following method displays the elements in an int array:

14. **THE ARRAYS CLASS -** The java. util. Arrays class contains various static methods for sorting and searching arrays, comparing arrays and filling array elements. These methods are overloaded for all primitive types.

15. **MULTI-DIMENSIONAL ARRAYS -** If an array contains multiple rows and multiple columns then it is called as multi-dimensional array or two-dimensional array.

3.7 Suggested Answer for Check Your Progress

Check your progress 1

Answers: See Section 3.2

Check your progress 2

Answers: See Section 3.3

Check your progress 3

Answers: See Section 3.4

Check your progress 4

Answers: See Section 3.5

Check your progress 5

Answers: See Section 3.6

3.8 Glossary

1. **For Loop -** The For loop is used to execute the same set of statements a number of times.
2. **While Loop -** The while loop is used to execute the same set of statements a number of times.
3. **Do-while loop -** Just like while loop, the do-while loop is also used to execute the same set of statements a number of times.
4. **Nested loop -** A loop within a loop is called nested loop. When two loops are nested, the outer loop takes control of the number of complete repetitions of the inner loop.
5. **The if-else Statement -** The if-else statement is also used to check a condition just like if statement but if the given statement is not true then the statements specified in the else part gets executed.
6. **Nested if -** if inside another if is called as nested if statement.

7. **Switch Statement -** The switch statement tests the value of a variable and based on that executes the corresponding case statement
8. **The Break Statement -** The break statement sends the control of program out of the loop.
9. **The Continue Statement -** The continue statement sends the control of program to the beginning of the loop.
10. **Recursion -** Recursion is when a function calls itself. That is, in the course of the function definition there is a call to that very same function.
11. **Array -** An array is a collection of similar types of variables which are referenced under a single name. It can also be defined as a collection of homogeneous cells inside computer's memory. a. One-dimensional array and b. Multi-dimensional array
12. **Processing Arrays -** When processing array elements, we often use either for loop or for each loop because all of the elements in an array are of the same type and the size of the array is known.
13. **Passing Arrays to Methods -** Just as you can pass primitive type values to methods, you can also pass arrays to methods. For example, the following method displays the elements in an int array:
14. **The Arrays Class -** The java.util.Arrays class contains various static methods for sorting and searching arrays, comparing arrays and filling array elements. These methods are overloaded for all primitive types.
15. **Multi-dimensional Arrays -** If an array contains multiple rows and multiple columns then it is called as multi-dimensional array or two-dimensional array.

3.9 Assignment

Explain the flow of control in for loop.

3.10 Activities

Using switch case statement, write a program for calculator.

3.11 Case Study

What are selection statements explain with the help of an example.

3.13 Further Reading

1. Core Java 2, 2 volumes, Cay S. Horstmann, Gary Cornell, The Sun Microsystems Press, 1999, Indian reprint 2000
2. Java 2, The Complete Reference, Patrick Naughton and Herbert Schildt, Tata McGraw Hill, 1999
3. Programming with Java, Ed. 2,E. Balagurusamy, Tata McGraw Hill, 1998, reprint, 2000
4. The Java Tutorial, Ed. 2, 2 volumes, Mary Campione and Kathy Walrath, Addison Wesley Longmans, 1998
5. The Java Language Specification, Ed. 2, James Gosling, Bill Joy, Guy Steele & Gilad Bracha, (Ed. 1 1996, Ed. 2 2000), Sun Microsystems, 2000
6. Using Java 2, Joseph L. Weber, Prentice Hall, Eastern Economy Edition, 2000