1.6 Flow Control

1.6.1 Java Decision Making

- Java decision-making statements allow you to make a decision, based upon the result of a condition.
- There may be a situation when you need to execute a block of code several number of times.
- In general, statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on.
- Programming languages provide various control structures that allow for more complicated execution paths.

- **Decision Making Statements in Java**
  1) if Statement
  2) else-if statement
  3) if-else statement
  4) switch statement

1. if statement:

- If statements in Java is used to control the program flow based on some condition, it's used to execute some statement code block if the expression is evaluated to true, otherwise, it will get skipped.
- This is the simplest way to modify the control flow of the program.

Syntax:
```
if(test_expression)
{
statement 1;
statement 2;
...
statement n;
}
```

Example:if statement

```
public class Sample{
     public static void main(String args[]){
      int a=20, b=30;
      if(b>a)
System.out.println("b is greater");
      }
}
```

2. **else if statement:**
   - else if statements in Java is like another if condition, it's used in the program when if statement having multiple decisions.

Syntax:
```
if(test_expression){
//execute your code
}
else if(test_expression n){
//execute your code
}
else{
//execute your code
}
```

Example: else if statement

```
public class Sample {
    public static void main(String args[]) {
        int a = 30, b = 30;
        if (b > a) {
System.out.println("b is greater");
        }
```

```
       else if(a > b){
System.out.println("a is greater");
       }
       else {
System.out.println("Both are equal");
      }
   }
}
```

3. if else statement:

- If else statements in Java is also used to control the program flow based on some condition, only the difference is: it’s used to execute some statement code block if the expression is evaluated to true, otherwise executes else statement code block.

Syntax:if(test_expression){

```
//execute your code
}
else{
//execute your code
}
```

Example:

```
public class Sample {
 public static void main(String args[]) {
  int a = 80, b = 30;
  if (b &gt; a) {
System.out.println("b is greater");
  }
else {
System.out.println("a is greater");
  }
 }
}
```

4. **switch statement:**

- Java switch statement is used when you have multiple possibilities for the ‘if statement’.
- After the end of each block it is necessary to insert a break statement because if the programmers do not use the break statement, all consecutive blocks of codes will get executed from each and every case onwards after matching the case block.
- When none of the cases is evaluated to true, the default case will be executed, and break statement is not required for default statement.

Syntax:
```
switch(variable){
case 1:
//execute your code
break;
case n:
//execute your code
break;
default case:
//execute your code
break;
}
```

Example:
```
public class Sample {
 public static void main(String args[]) {
  int a = 5;
  switch (a) {
   case 1:
System.out.println("You chose One");
    break;
   case 2:
System.out.println("You chose Two");
```

```
    break;
   case 3:
System.out.println("You chose Three");
 break;
   case 4:
System.out.println("You chose Four");
    break;
   case 5:
System.out.println("You chose Five");
    break;
default case:
System.out.println("Invalid Choice. Enter a no between 1 and 5");
    break;
  }
 }
}
```

1.6.2 Loop statements in java

Sometimes it is necessary for the program to execute the statement several times, and Java loops execute a block of commands a specified number of times until a condition is met.
In this chapter, you will learn about all the looping statements of Java along with their use.

❑What is Loop?

- A computer is the most suitable machine to perform repetitive tasks and can tirelessly do a task tens of thousands of times.
- Every programming language has the feature to instruct to do such repetitive tasks with the help of certain form of statements.
- The process of repeatedly executing a collection of statement is called ***looping.***

- ➢ Java supports many looping features which enable programmers to develop concise Java programs with repetitive processes.

❑ **Java supports following types of loops:**

1) while loops
2) do while loops
3) for loops

❑ **Java Loop Control Statements:**

Loop control statements are used to change the normal sequence of execution of the loop.

Statement	Syntax	Description
break statement	break;	Is used to terminate loop or switch statements.
continue statement	continue;	Is used to suspend the execution of current loop iteration and transfer control to the loop for the next iteration.
goto statement	gotolabelName;labelName: statement;	It transfers current program execution sequence to some other part of the program.

1. while loop:

- ➢ Java while loops statement allows to repeatedly run the same block of code until a condition is met.
- ➢ while loop is the most basic loop in Java. It has one control condition and executes as long the condition is true.
- ➢ The condition of the loop is tested before the body of the loop is executed, hence it is called an entry-controlled loop.

Syntax:while (condition)
{

```
statement(s);
Incrementation;
}
```

Example:

```
public class Sample {
    public static void main(String args[]) {
        int n = 1, times = 5;/* local variable Initialization */
        /* while loops execution */
        while (n <= times) {
System.out.println("Java while loops:" + n);
            n++;
        }
    }
}
```

2. do while loop:

- ➢ Java do-while loops are very similar to the while loops, but it always executes the code block at least once and furthermore as long as the condition remains true.
- ➢ This is an exit-controlled loop.

Syntax:do{

```
statement(s);
}while( condition );
```

Example:

```
public class Sample {
    public static void main(String args[]) {
        int n = 1, times = 0; //local variable Initialization
        /* do-while loops execution */
        do {
System.out.println("Java do while loops:" + n);
            n++;
        } while (n <= times);
    }
}
```

3. For loop:

- Java for loops is very similar to Java while loops in that it continues to process a block of code until a statement becomes false, and everything is defined in a single line.

Syntax: for (init; condition; increment)

```
{
statement(s);
}
```

Example:

```
public class Sample {
    public static void main(String args[]) {
            int n = 1, times = 5;   // local variable Initialization
        /* for loops execution */
        for (n = 1; n <= times; n = n + 1) {
System.out.println("Java for loops:" + n);
        }
    }
}
```