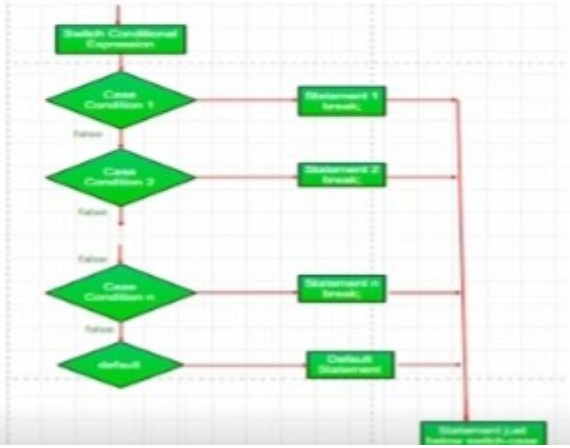
Switch Statement

DSCE

- The **switch statement is Java's multiway branch statement.**
- **It provides an easy way to dispatch** execution to different parts of code based on the value of an expression.
- As such, it often provides a better alternative than a large series of **if-else-if statements. Here is the general form** of a **switch statement:**

```
switch (expression) {
case value1:
// statement sequence
break;
case value2:
// statement sequence
break;
...
case valueN:
```



Control Statements

- Programming language uses *control statements to cause the flow of execution to* advance and branch based on changes to the state of a program.
- Java's programcontrol statements can be put into the following categories:
- selection,
- iteration, and
- jump.

DSCE

Java's Selection Statements

2. Nested ifs

- A *nested* ***if is an if statement that is the target of another if or else.***
- ***Nested ifs are very common*** in programming.
- **Here is an example:**

```
if(i == 10) {
if(j < 20) a = b;
if(k > 100) c = d; // this if is
else a = c; // associated with this else
}
else a = d; // this else refers to if(i == 10)
```

DSCE

Java's Selection Statements

2. Nested ifs

- A *nested* ***if is an if statement that is the target of another if or else.***
- ***Nested ifs are very common*** in programming.
- **Here is an example:**

```
if(i == 10) {
if(j < 20) a = b;
if(k > 100) c = d; // this if is
else a = c; // associated with this else
}
else a = d; // this else refers to if(i == 10)
```

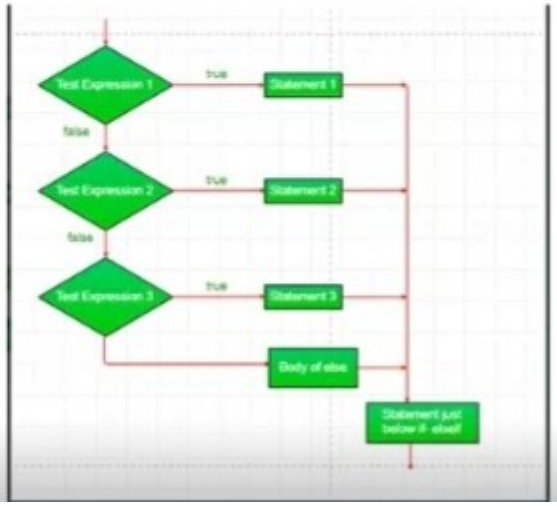
DSCE

Java's Selection Statements

3. The if-else-if Ladder

- A common programming construct that is based upon a sequence of nested **ifs is the** *if-else-if ladder. It looks like this:*

if(*condition)*
statement;
else if(*condition)*
statement;
else if(*condition)*
statement;
...
else
statement;



Example of if-else-if ladder

DSCE

```
// Java program to illustrate if-else-if ladder
class ifelseifDemo
{
    public static void main(String args[])
    {
        int i = 20;

        if (i == 10)
            System.out.println("i is 10");
        else if (i == 15)
            System.out.println("i is 15");
        else if (i == 20)
            System.out.println("i is 20");
        else
            System.out.println("i is not present");
    }
}
```

Example 2-Switch statements

DSCE

```
//Java Switch Example where we are omitting
    //the break statement
public class SwitchExample2 {
public static void main(String[] args) {
   int number=20;
   //switch expression with int value
   switch(number){
   //switch cases without break statements
   case 10: System.out.println("10");
   case 20: System.out.println("20");
   case 30: System.out.println("30");
   default:System.out.println("Not in 10, 20 or
     30");
   }
}
}
```

Output:

```
20
30
Not in 10, 20 or 30
```

Example 2-Switch statements

DSCE

```
//Java Switch Example where we are omitting
    //the  break statement
public class SwitchExample2 {
public static void main(String[] args) {
   int number=20;
   //switch expression with int value
   switch(number){
   //switch cases without break statements
   case 10: System.out.println("10");
   case 20: System.out.println("20");
   case 30: System.out.println("30");
   default:System.out.println("Not in 10, 20 or
     30");
   }
}
}
```

Output:

20
30
Not in 10, 20 or 30

Nested Switch Statement

DSCE

We can use switch statement inside other switch statement in Java.

Example:

```
public class NestedSwitchExample {
  public static void main(String args[])
   {
   //C - CSE, E - ECE, M - Mechanical
    char branch = 'C';
    int Semester = 4;
    switch( Semester )
    {
      case 1:
        System.out.println("English, Maths, Science");
        break;
      case 2:
        switch( branch )
        {
          case 'C':
            System.out.println("Operating System, Java, Data Structure");
            break;
```

Cntd..

DSCE

```
case 4:
        switch( branch )
        {
          case 'C':
            System.out.println("Data Communication and Networks, MultiMedia");

            break;
          case 'E':
            System.out.println("Embedded System, Image Processing");
            break;
          case 'M':
            System.out.println("Production Technology, Thermal Engineering");
            break;
        }
        break;
    }
  }
}
```

Output:
Data Communication and Networks, MultiMedia

Nested Switch Statement

DSCE

We can use switch statement inside other switch statement in Java.

Example:

```
public class NestedSwitchExample {
   public static void main(String args[])
    {
    //C - CSE, E - ECE, M - Mechanical
     char branch = 'M';
     int Semester = 3;
     switch( Semester )
     {
        case 1:
           System.out.println("English, Maths, Science");
           break;
        case 2:
           switch( branch )
           {
              case 'C':
                 System.out.println("Operating System, Java, Data Structure");
                 break;
```

DSCE

Iteration Statements

- Java's iteration statements are **for, while, and do-while.**

while

- The **while loop is Java's most fundamental loop statement. It repeats a statement or block**
- while its controlling expression is true. Here is its general form:

```
while(condition) {
// body of loop
}
```

  - The condition can be any Boolean expression.

DSCE

While loop Example

```
public class Test
{
public static void main(String args[])
 {     int x = 10;
     while( x < 20 )
     {
     System.out.print("value of x : " + x );
      x++;
     System.out.print("\n");
  }  }}
```

Output:

```
value of x : 10
value of x : 11
value of x : 12
value of x : 13
value of x : 14
value of x : 15
value of x : 16
```

DSCE

do-while Loop

- A do-while loop is similar to a while loop, except that a do-while loop is guaranteed to execute at least one time.

Syntax:

The syntax of a do-while loop is:

```
do{
 //Statements
}while(Boolean_expression);
```

- Notice that the Boolean expression appears at the end of the loop, so the statements in the loop execute once before the Boolean is tested.

The for Loop:

- A for loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.
- A for loop is useful when you know how many times a task is to be repeated.
- The syntax of a for loop is:

```
for(initialization; Boolean_expression; update)
{
 //Statements
}
```

DSCE

for Loop-Example

```
public class Test
{
public static void main(String args[])
 {
 for(int x = 10; x < 20; x = x+1)
 {
   System.out.print("value of x : " + x );
   System.out.print("\n");
    }
 }
}
```

Output:

```
value of x : 10
value of x : 11
value of x : 12
value of x : 13
value of x : 14
value of x : 15
value of x : 16
value of x : 17
value of x : 18
value of x : 19
```