Limitation of if condition :

```
int x = 90;

if(x == 10)
{
}
else if(x ==20)
{
}
else if(x ==30)
{
}
else if(x ==40)
{
}
else if(x ==50)
{
}
else
{
}
```

The major drawback with if condition is, it checks the condition again and again so It increases the burdon over CPU so we introduced switch-case statement to reduce the overhead of the CPU.

Switch Statement in java (Till JDK 17V) :

Switch Statement in java (Till JDK 17V) :

It is a selective statement in java so, we can select one statement among the available statements.

While writing the cases, break is optional but if we use break then the control will move from out of the switch body.

We can write default so if any statement is not matching then default will be executed but It is optional. [default is compulsory, If we take switch statement as an expression]

Valid types of switch expression :
----
byte
short
int
char
Byte
Short
Integer
Character
enum (JDK 1.5)
String (JDK 1.7)
Pattern Matching (JDK 17) [instanceof]

Invalid types are :
----
long
float
double
boolean

Some important points :
----
a) We cannot accept duplicate case value.

b) The label of case must be constant OR literals.

c) The break statement is used to exit from the switch block. It is optional but recommended [JDK 17 (->) break is not required]

d) The default case is optional and executes if no case matches in the switch expression.

//Programs :

```java
package com.ravi.switch_case;

import java.util.Scanner;

public class SwitchDemo1
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
         System.out.print("Please Enter a Character :");
        char colour = sc.next().toLowerCase().charAt(0);

        switch(colour)
        {
        case 'r' : System.out.println("Red") ; break;
        case 'g' : System.out.println("Green");break;
        case 'b' : System.out.println("Blue"); break;
        case 'w' : System.out.println("White"); break;
        default : System.out.println("No colour");
        }
        System.out.println("Completed") ;
        sc.close();
    }
}
```

```java
package com.ravi.switch_case;

import java.util.Scanner;

public class SwitchDemo2
{
public static void main(String args[])
  {
        System.out.println("\t\t**Main Menu**\n");
        System.out.println("\t\t**100 Police**\n");
        System.out.println("\t\t**101 Fire**\n");
        System.out.println("\t\t**102 Ambulance**\n");
        System.out.println("\t\t**139 Railway**\n");
        System.out.println("\t\t**181 Women's Helpline**\n");

        System.out.print("Enter your choice :");
        Scanner sc = new Scanner(System.in);
        int choice = Integer.parseInt(sc.nextLine());

        switch(choice)
        {
        case 100 -> System.out.println("Police Services");

        case 101 -> System.out.println("Fire Services");

        case 102 -> System.out.println("Ambulance Services");

        case 139 -> System.out.println("Railway Enquiry");

        case 181 -> System.out.println("Women's Helpline ");

        default -> System.out.println("Your choice is wrong");

        }
        sc.close();
      }
}
```

Note : Here break is not required

Passing String in the switch case :
----
* We can pass String in switch statement from JDK 1.7V.
* As we know the label of switch must be constant that means in java String are constants (immutable)

```java
package com.ravi.switch_case;

import java.util.Scanner;

public class SwitchDemo3
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the language you want to learn :");
         String language = sc.nextLine().toUpperCase();

         switch (language) //JDK 1.7v It is allowed
         {
            case "C" -> System.out.println("I want to learn C language");
            case "JAVA" -> System.out.println("I want to learn Java language");
            case "HTML" -> System.out.println("I want to learn HTML language");
            case "CSS" -> System.out.println("I want to learn CSS language");
            default -> System.out.println("Language is not available");
         }

         sc.close();
    }
}
```

How to provide grouping in switch case :

```java
package com.ravi.switch_case;

import java.util.Scanner;

public class SwitchDemo4
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter an animal Name :");
        String animal = sc.nextLine().toLowerCase();

        switch(animal)
        {
            case "dog", "cat", "cow", "sheep" -> System.out.println("It is a domestic animal");


            case "lion", "tiger", "elephant", "wolf" ->
            {
                System.out.println("It is a wild animal");
            }

            default ->
            {
                System.out.println("Unknown animal");
            }
        }
        sc.close();
    }
}
```

Switch case with return value (Expression Style) :
----
* From JDK 14V onwards, A switch statement can also return some value that means switch we can use as an expression.

* This is basically known as switch case with return value OR switch case with expression style.

Example :

```
int num =   switch(number)
{
    case 1 -> 1;
    case 2 -> 2;
    case 3 -> 3;
    default -> 0;       //default is compulsory

 };                     //semicolon is compulsory because It will return the vale
```

* Some important Points :
----
a) While writing switch body, semicolon is compulsory at the end to return the value.
b) default is compulsory here because If all the cases will not match then default will return the value.
c) From switch body, We need to return appropriate value.

```java
package com.ravi.switch_case;

import java.util.Scanner;

public class SwitchDemo5
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter your Subject grade [A/B/C/D/E]");
        char grade = sc.nextLine().toUpperCase().charAt(0);

        String result = switch(grade)
        {
        case 'A','B' -> "Excellent";
        case 'C' -> "Very good";
        case 'D' -> "Good";
        case 'E' -> "Average";
        default -> "Invalid";
        };

        System.out.println("Your grade is :"+result);
        sc.close();
    }
}
```

From JDK 14v onwards, switch can be directly used as an expression so It can directly return value hence return keyword is not required.

On the other hand, If we use switch expression with {} blocks then
compulsory we should use yield (keyword) not return statement as shown in the program below :

```java
package com.ravi.switch_case;
/*Bonus for a Employee based on the performance
  Grade A - 20%
  Grade B - 15%
  Grade C - 10%

*/
import java.util.Scanner;

public class SwitchDemo6
{
    public static void main(String[] args)
    {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter your performance grade [A/B/C] : ");
      char grade = sc.nextLine().toUpperCase().charAt(0);

       double salary = 40000;

       double bonus = switch(grade)
       {
       case 'A' ->
       {
       System.out.println("Grade A performer!!!");
       yield salary*0.20;
       }

       case 'B' ->
       {
       System.out.println("Grade B performer!!!");
       yield salary*0.15;
       }

       case 'C' ->
       {
       System.out.println("Grade C performer!!!");
       yield salary*0.10;
       }

       default -> 0.0;

       };

       System.out.println("Your bonus amount is :"+bonus);
       sc.close();
    }

}
```

Note :If we have block of statements while working with switch cases
then we should compulsory use yield keyword.

return keyword is used to return a value and terminate the flow
from a method but switch is not a method (does not contain return type) It is an expression so, yield keyword is compulsory.

```java
package com.ravi.switch_statement;

public class SwitchDemo7
{
    public static void main(String[] args)
    {
        int x = 12;

        switch(x)
        {

        }
    }
}
```

Note : We can't pass long, float and double and boolean value.

```java
package com.ravi.switch_statement;

public class SwitchDemo8
{
    public static void main(String[] args)
    {
        int x = 10;
        final int y = 12;

        switch(x)
        {
        case y -> System.out.println("It is case 12");
        }

    }

}
```

The label of case must be constant.

```java
package com.ravi.switch_statement;

public class SwitchDemo9 {

    public static void main(String[] args)
    {
        byte b = 12;

        switch(b)
        {
        case 127 -> System.out.println();
        case 128 -> System.out.println();  //error

        }

    }

}
```

While working with switch case byte and short case value must be within the range only.