Java Decision Making

- There are two types of decision making statements in Java.

  They are:

1. if statements
2. switch statements

if Statement:

Syntax:

if(Boolean_expression)

{

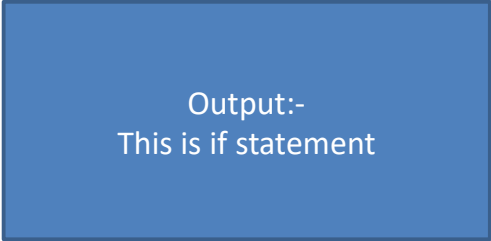   //Statements will execute if the Boolean expression is true

}

   If the boolean expression evaluates to true then the block of code inside the if statement will be executed. If not the first set of code after the end of the if statement(after the closing curly brace) will be executed.

if Statement:

```
public class Test
{
        public static void main(String args[])
        {
            int x = 10;
            if( x < 20 )
            {
            System.out.print("This is if statement");
            }
        }
}
```

Output:-
This is if statement

3

if..else Statement:

- An if statement can be followed by an optional *else* statement, which executes when the Boolean expression is false.

Syntax:

if(Boolean_expression)

{

   //Executes when the Boolean expression is true

}

else

{

   //Executes when the Boolean expression is false

 }

- If the boolean expression evaluates to true then the block of code inside the if statement will be executed. If not the first set of code after the end of the if statement(after the closing curly brace) will be executed.

4

If...else Statement:

```java
public class Test
{
    public static void main(String args[])
    {
     int x = 30;
    if( x < 20 )
    {
     System.out.print("This is if statement");
    }
    else
    {
     System.out.print("This is else statement");
    }
    }
}
```

Output:-
This is else statement

if...else if...else Statement:

* An if statement can be followed by an optional *else if...else* statement, which is very usefull to test various conditions using single if...else if statement.
* **Syntax:**

if(Boolean_expression 1)

{

   //Executes when the Boolean expression 1 is true

}

else if(Boolean_expression 2)

{

   //Executes when the Boolean expression 2 is true

}

else if(Boolean_expression 3)

{

   //Executes when the Boolean expression 3 is true

 }

else

{

   //Executes when the none of the above condition is true.

 }

if...else if...else Statement

```java
public class Test
{
    public static void main(String args[])
    {
        int x = 30;
        if( x == 10 )
        {
         System.out.print("Value of X is 10");
        }
        else if( x == 20 )
        {
         System.out.print("Value of X is 20");
        }
        else if( x == 30 )
        {
        System.out.print("Value of X is 30");
        }
        else
        {
        System.out.print("This is else statement");
        }
    }
}
```

Output:-
Value of X is 30

Nested if...else Statement

- It is always legal to nest if-else statements, which means you can use one if or else if statement inside another if or else if statement.

- **Syntax:**

if(Boolean_expression 1)

{

//Executes when the Boolean expression 1 is true

if(Boolean_expression 2)

{

//Executes when the Boolean expression 2 is true

}

}

You can nest *else if...else* in the similar way as we have nested *if* statement.

Nested if...else Statement

```
public class Test
{
    public static void main(String args[])
    {
        int x = 30;
        int y = 10;
        if( x == 30 )
        {
            if( y == 10 )
            {
                System.out.print("X = 30 and Y = 10");
            }
        }
    }
}
```

Output:-
X = 30 and Y = 10

Switch case

- A *switch* statement allows a variable to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each case.

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.

Switch case

- 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.

11

Syntax of switch case

```
switch(expression)
{
        case value :
            //Statements break;//optional
        case value :
            //Statements break; //optional
            //You can have any number of cases.
        default : //Optional //Statements
}
```

Example of switch case

```java
public static void main(String args[])
{
 char grade = args[0].charAt(0);   switch(grade)
{
case 'A' : System.out.println("Excellent!");  break;
case 'B' :
case 'C' : System.out.println("Well done");
break;
case 'D' : System.out.println("You passed");
case 'F' : System.out.println("Better try again");
 break;
default : System.out.println("Invalid grade");
}
System.out.println("Your grade is " + grade);
}
```

$ java Test a
Invalid grade
Your grade is a a

$ java Test A
Excellent!
Your grade is a A

$ java Test C
Well done
Your grade is a C

13