HNDIT 3012- Object Oriented programing

Lecture 03 - Control Flow Statements

Control flow

control flow (or **flow** of **control**) is the order in which individual statements of an **program** are executed .

Control flow structures

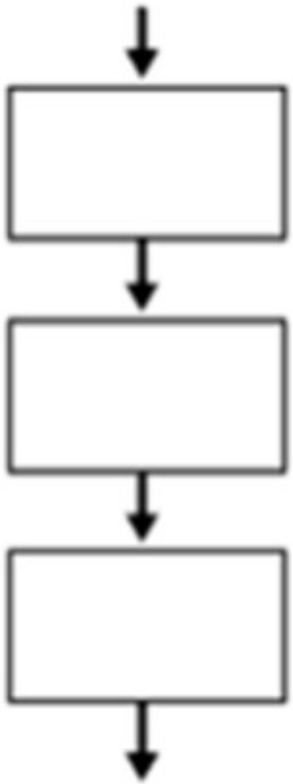
- **Sequential structure :**
  - Default mode. Statements are executed from top to bottom of program one by one.

- **Selection structure:**
  - **Selection structure** or conditional **structure**, is different blocks of code are performed based on whether a Boolean condition is true or false
  - If-else, switch-case

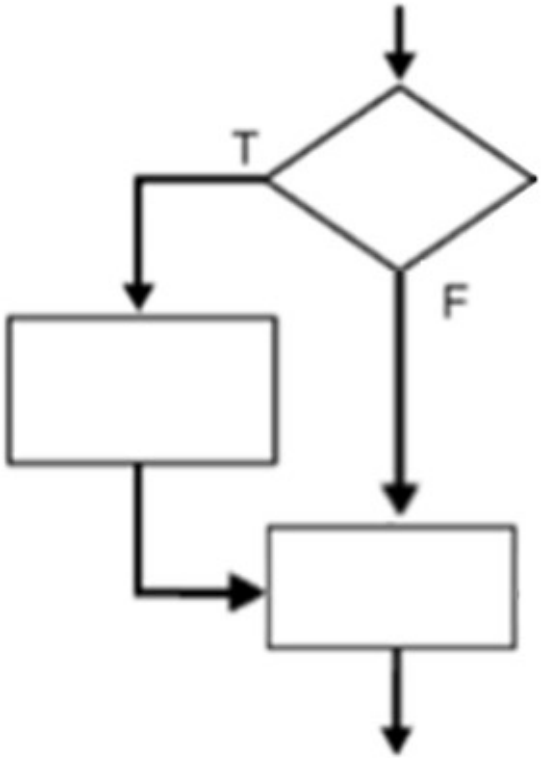
- **Repetition structure:**
  - Same block of code is executed again and again based on whether a Boolean condition is true or false
  - while, do-while, for

Control Flow

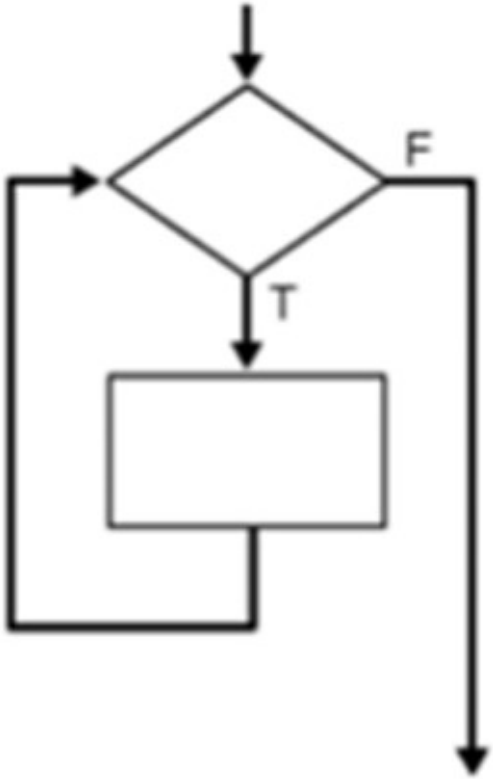
Sequence



Selection



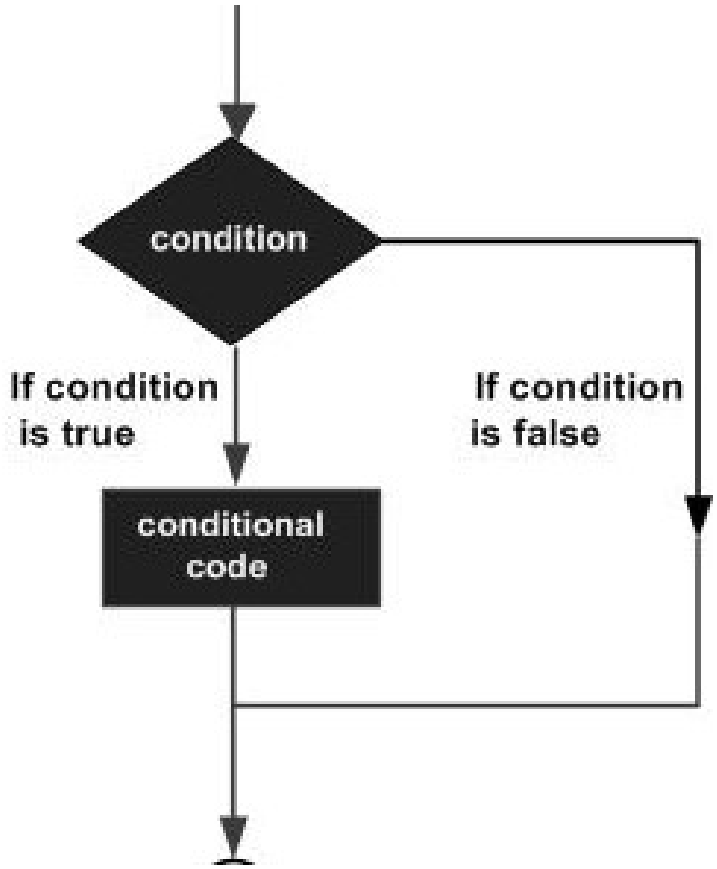
Iteration



Selection → If condition

- Syntax

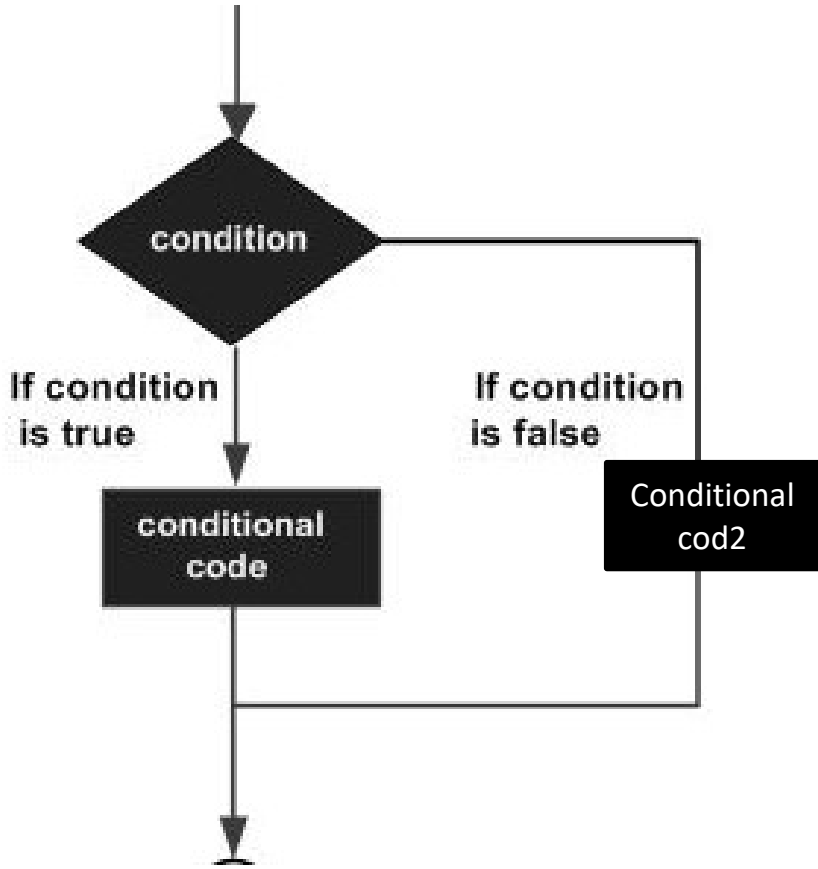
```
If(expression)
{
statement(s)
}
```



Selection →If else condition

- Syntax

```
If(expression)
{
Conditional code
}
Else
{
Conditional code 2
}
```



If Else Condition Example

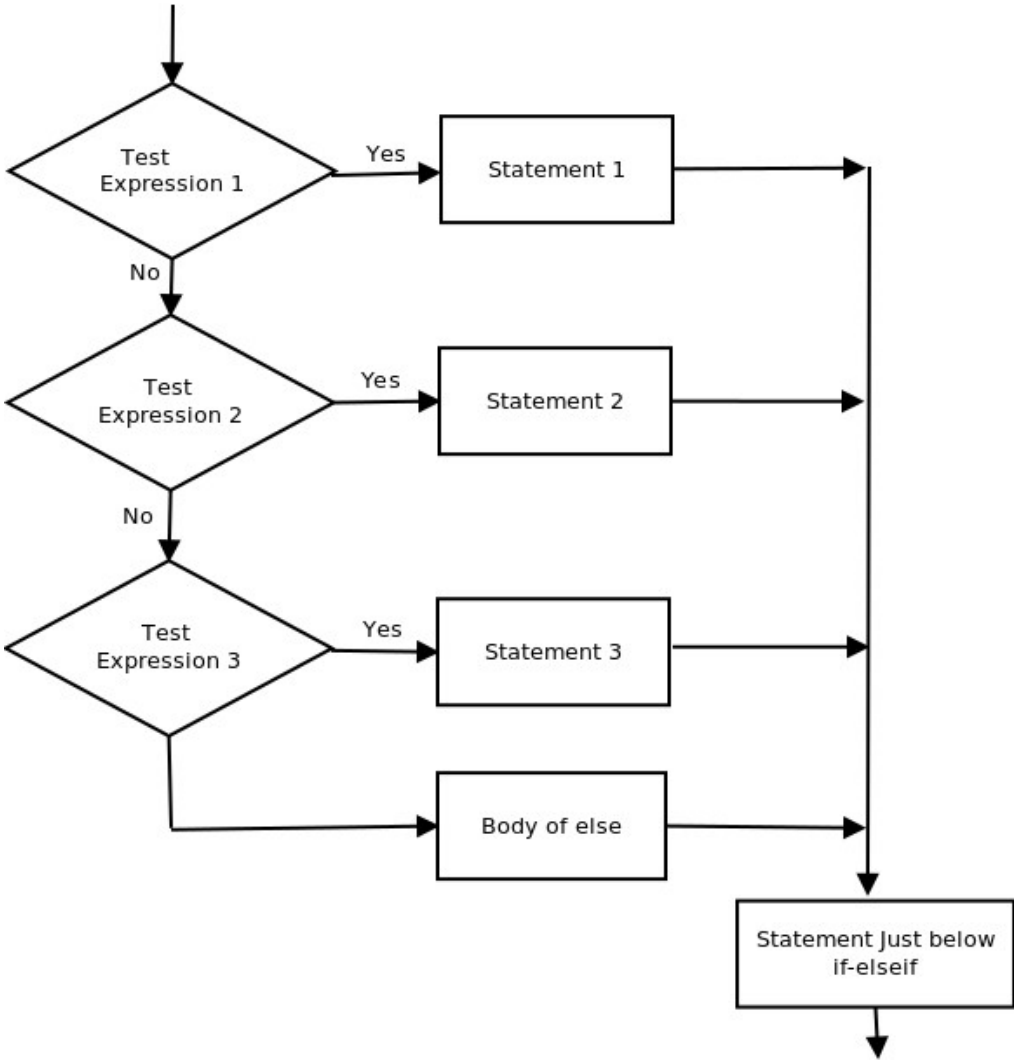
- Write down a java code to display whether a student is passed from a subject. Minimum requirement is to score 40 marks.

```
public static void main(String[] args)
   {
      int Marks=92;
      if (Marks<40 )
            System.out.print("Result : Fail");
      else
            {

            System.out.print("Result :  Pass");

            }
}
```

Selection → Nested if

- Syntax

```
if(condition){
    //code to be executed
        if(condition){
            //code to be executed
    }
}
```



Nested If condition Example

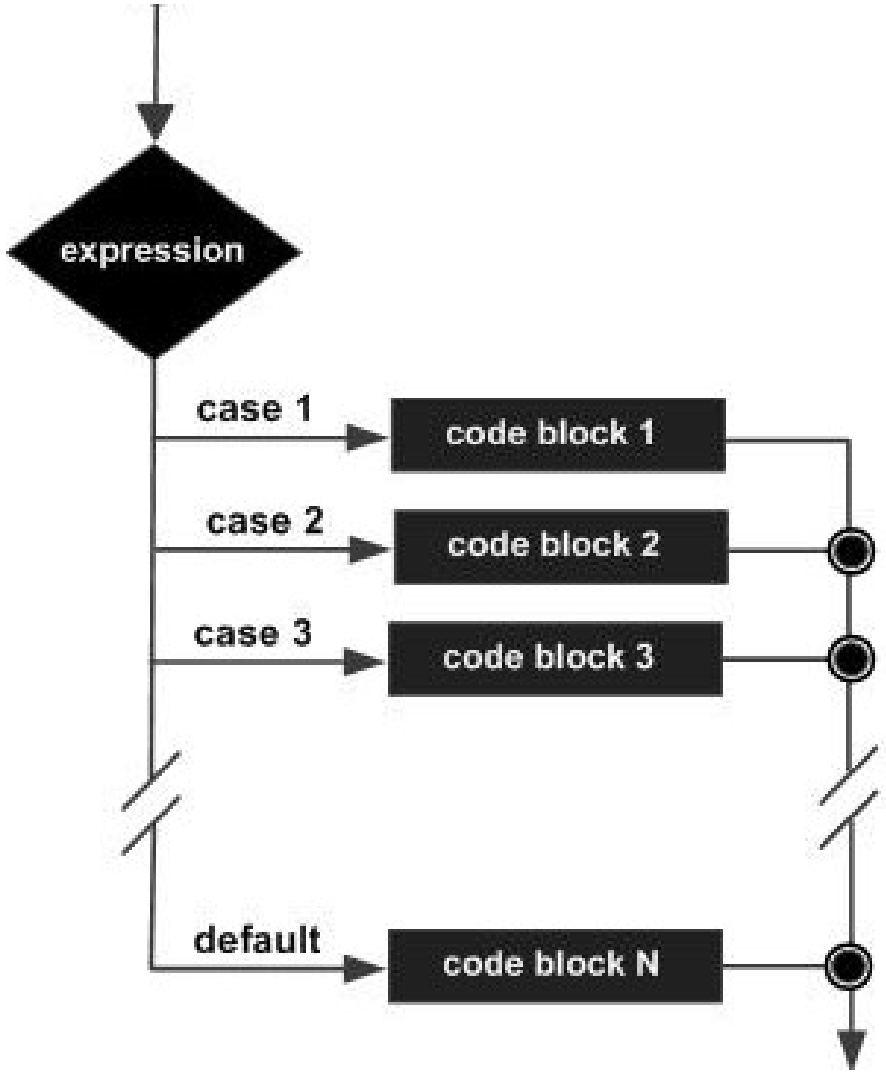
- Example

```
public static void main(String[] args)
   {
      int Marks=92;
      if (Marks<40 )
            System.out.print("Result : Fail");
      else
            {
            if (Marks<60 )
                        System.out.print("Result : Simple Pass");
                  else
                        System.out.print("Result : Credit Pass");
            }
 }
```

Selection → switch Statement

- Syntax

```
switch ( expression )
{
    case value:
            statement(s);
            break;
    case value:
            statement(s);
            break;
    default:
            statement(s);
}
```



- keyword **break** is needed to break out of each case.
- **default** will execute, only if the execution skip from all the cases

switch Statement

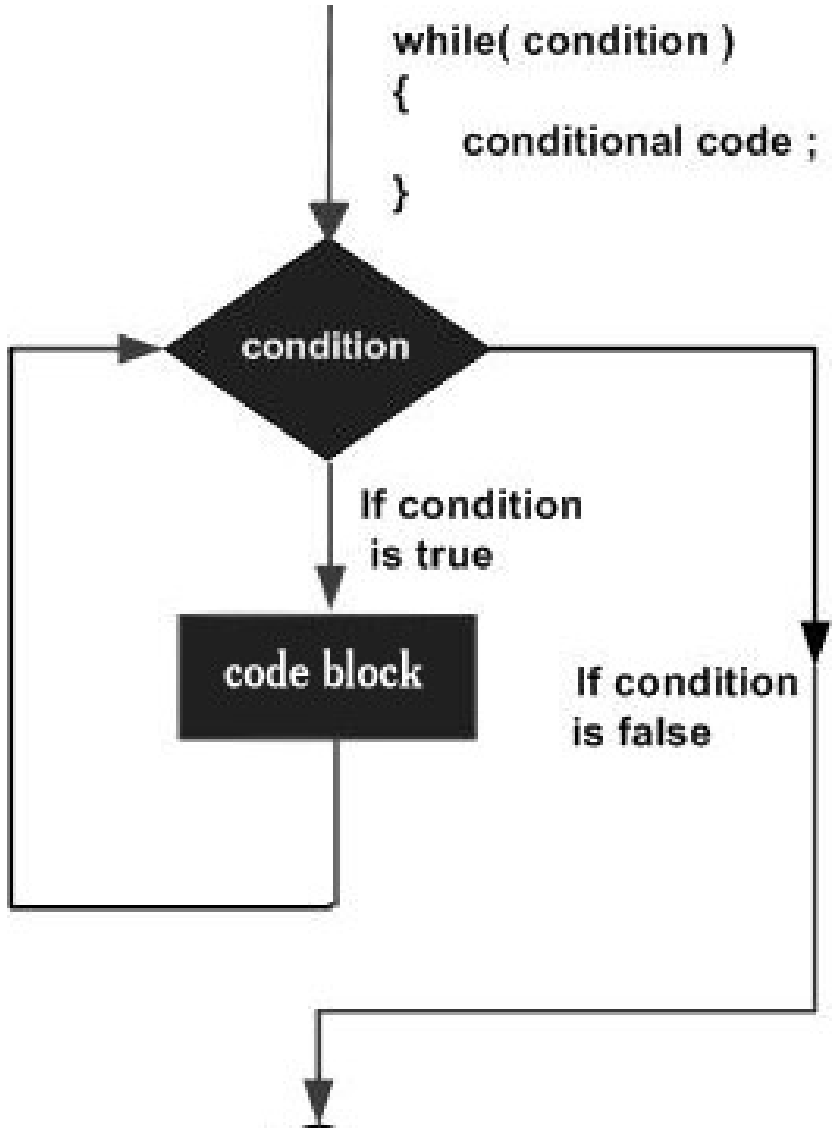
Display the day of a week according to the user input day no.
Example Day 3 is Tuesdays

```
public static void main(String[] args) {
        int Day = 4;
       String DayString;
       switch (Day) {
          case 1:  DayString = "Sunday";      break;
          case 2:  DayString = "Monday";      break;
          case 3:  DayString = "Tuesday";     break;
          case 4:  DayString = "Wednesday";   break;
          case 5:  DayString = "Thursday";    break;
          case 6:  DayString = "Friday";      break;
          case 7:  DayString = "Saturday";    break;
          default: DayString = "Invalid Day"; break;
       }
       System.out.println(DayString);
    }
}
```

Iteration→ while loop

- Syntax

```
while (expression)
{
statement(s)
}
```



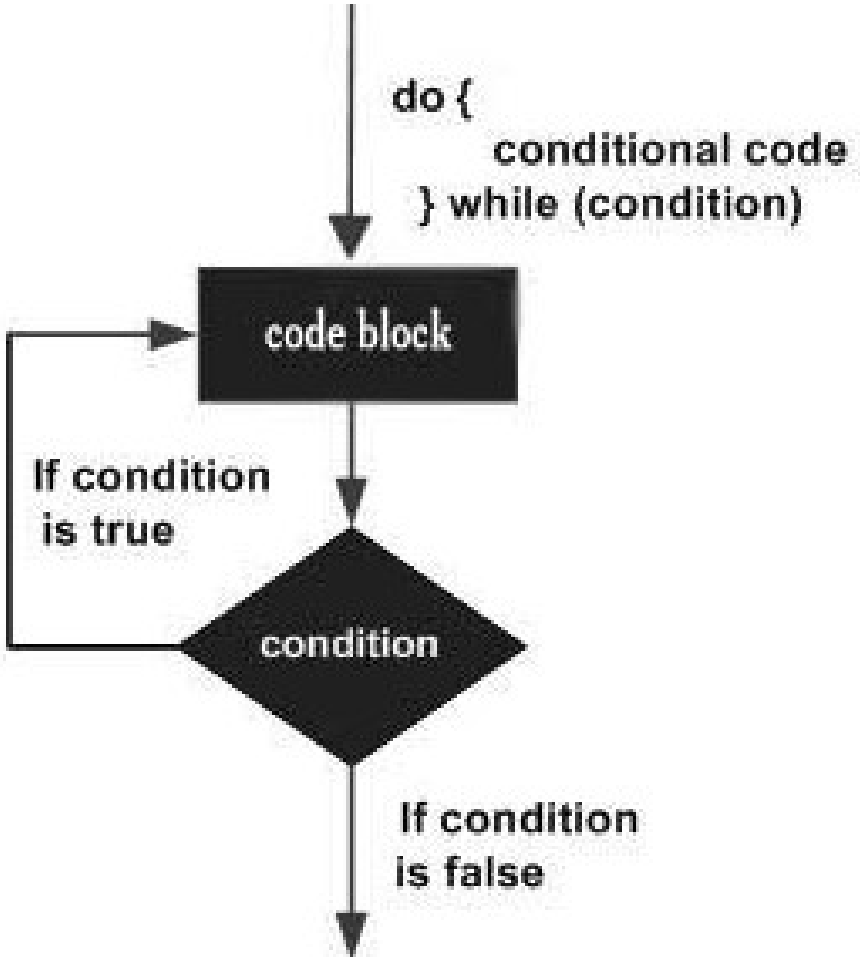
While Loop

```
public static void main(String[] args)
   {
            int count = 1;
            while (count < 11)
            {
                     System.out.println("Count is: " + count);
                     count++;
            }
}
```

Iteration→ Do while loop

- Syntax

```
do
 {
statement(s)
}
 while (expression)
```



do while loop

```
public static void main(String[] args)
  {
        int count = 1;
        do
        {
              System.out.println("Count is: " + count);
              count++;
        }
        while (count < 11)
 }
```

statements within the do block are always executed at least once

Question

```
public static void main(String args[])
    {
        int x = 21, sum = 0;
         do {
                sum += x;
               x--;
        }
          while (x > 10);
         System.out.println("Summation: " + sum);
    }
```

Output

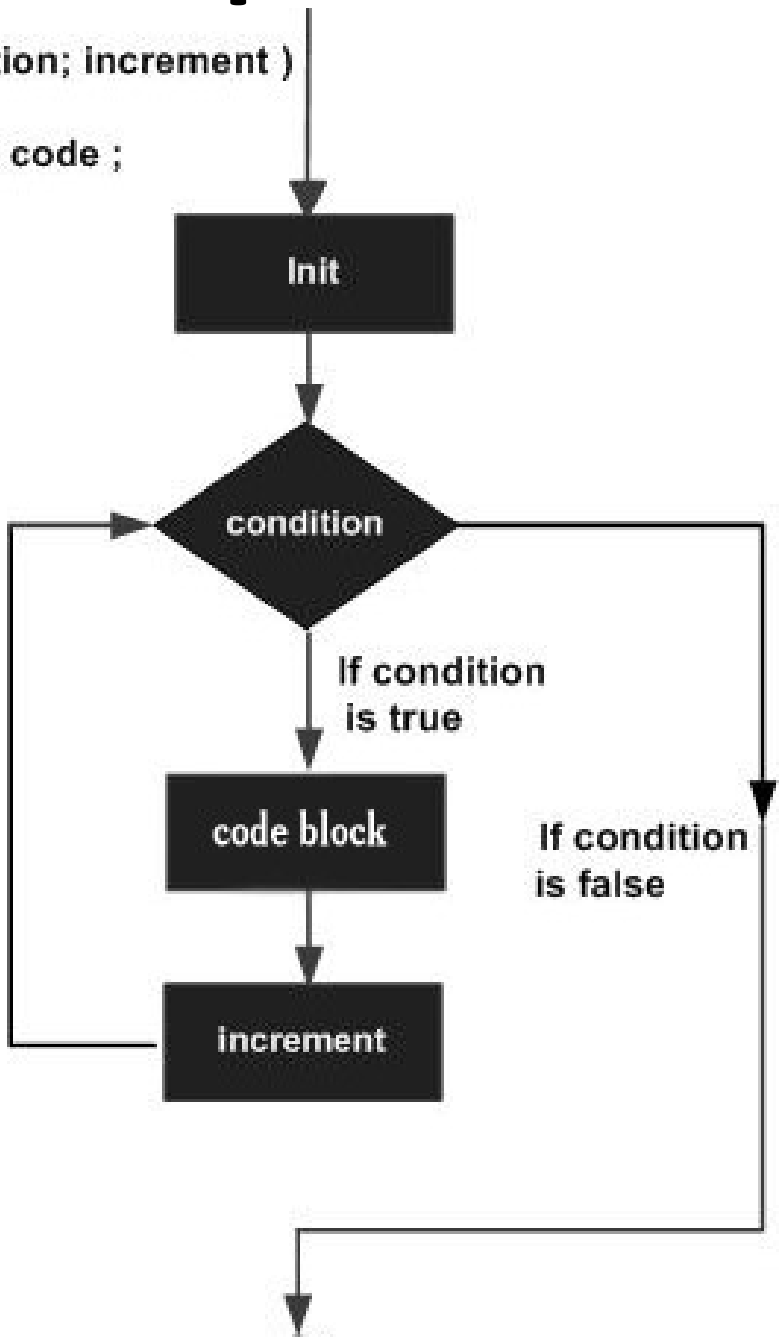
Summation: 176

Iteration→ For loop

- Syntax

```
for(initialization; expression; change)
 {
        statement/s
}
```

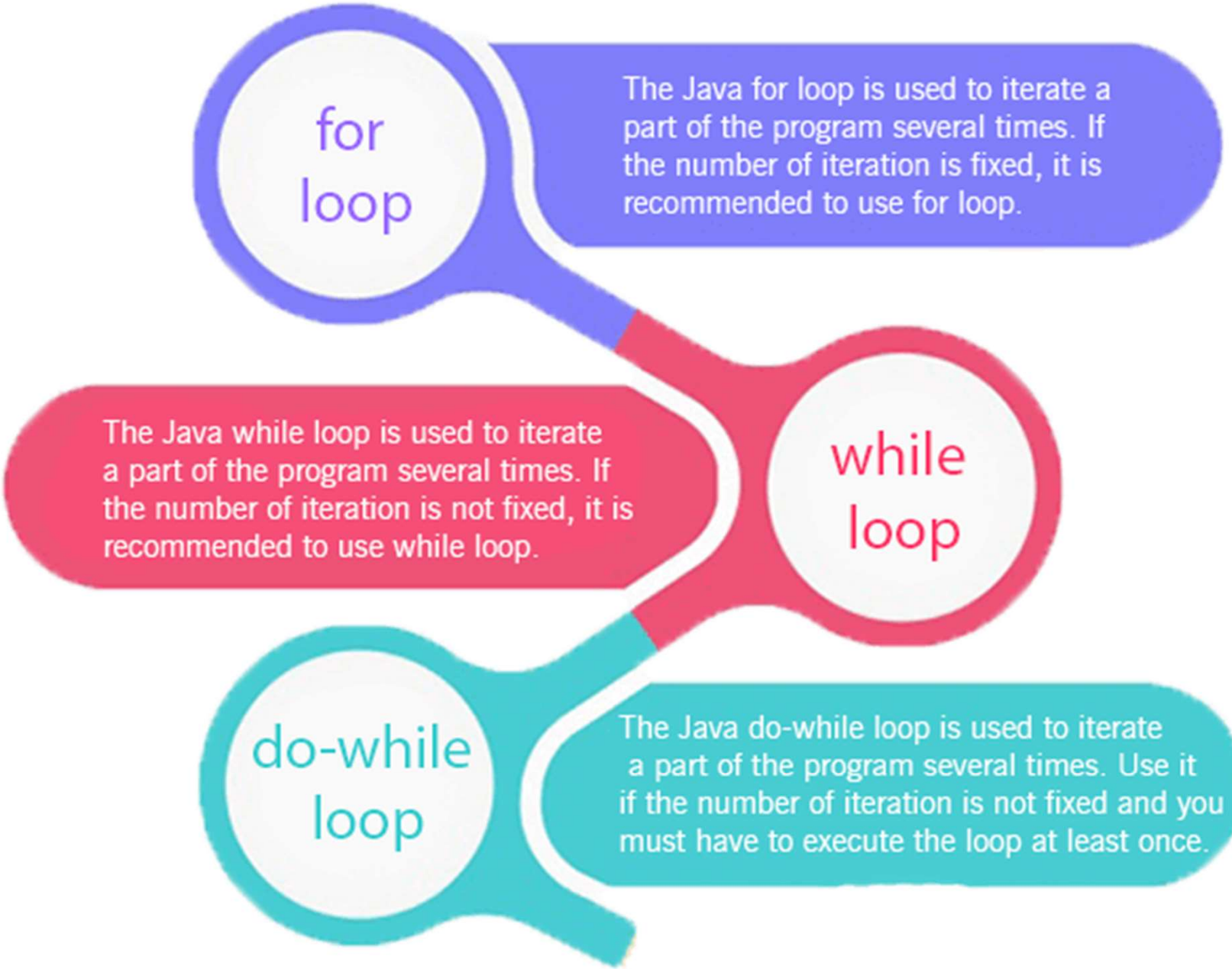
```
for( init; condition; increment )
{
    conditional code ;
}
```



For Loop

```
public static void main(String[] args)
   {
          for(count=0;count < 10;count++) )
                    {
                     System.out.println("Count is: " + count);
                    }
}
```

- The ***initialization*** expression initializes the loop; it's executed once, as the loop begins.
- When the ***termination*** **expression** evaluates to false, the loop terminates.
- The ***change*** expression (increment/decrement) is invoked after each iteration through the loop; it is perfectly acceptable for this expression to increment *or* decrement a value

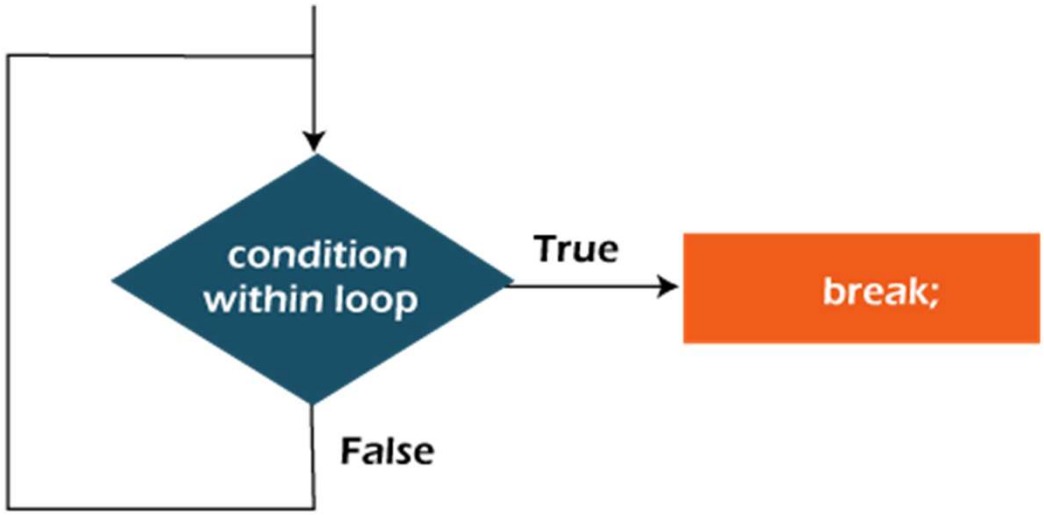

for loop
The Java for loop is used to iterate a part of the program several times. If the number of iteration is fixed, it is recommended to use for loop.
while loop
The Java while loop is used to iterate a part of the program several times. If the number of iteration is not fixed, it is recommended to use while loop.
do-while loop
The Java do-while loop is used to iterate a part of the program several times. Use it if the number of iteration is not fixed and you must have to execute the loop at least once.

Break Statement

- Terminates the **loop** or **switch** statement and transfers execution to the statement immediately following the loop or switch.

- The break statement has two forms:
  - labeled and unlabeled.

- Unlabeled break
  - Break out from a case
  - Terminate a for, while,
  - or do-while loop



Break Statement

Example

```
class Test
 {
 public static void main(String[] args)
    {
            int count ;
            for(count=0;count < 10;count++)
                    {
                    if (count==5)
                    {
                            break;
                    }
                    System.out.println("Counting: " + count);
                    }
    }
 }
```

Continue statement

- Continue statement skips the current iteration of a for, while , or do-while loop

```
class Test
  {
  public static void main(String[] args)
      {
        int count ;
        for(count=0;count < 10;count++)
              {
              if (count==5)
             {
                      continue;
              }
           System.out.println("Counting: " + count);
              }
      }
  }
```