SNS COLLEGE OF ENGINEERING

Coimbatore-35
An Autonomous Institution

Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

COURSE NAME : 23CSB101 & Object Oriented Programming

I YEAR/ II SEMESTER

UNIT – I INTRODUCTION TO OOP & JAVA

Topic: Control structures

Dr.P.Poonkodi

Assistant Professor(SG)

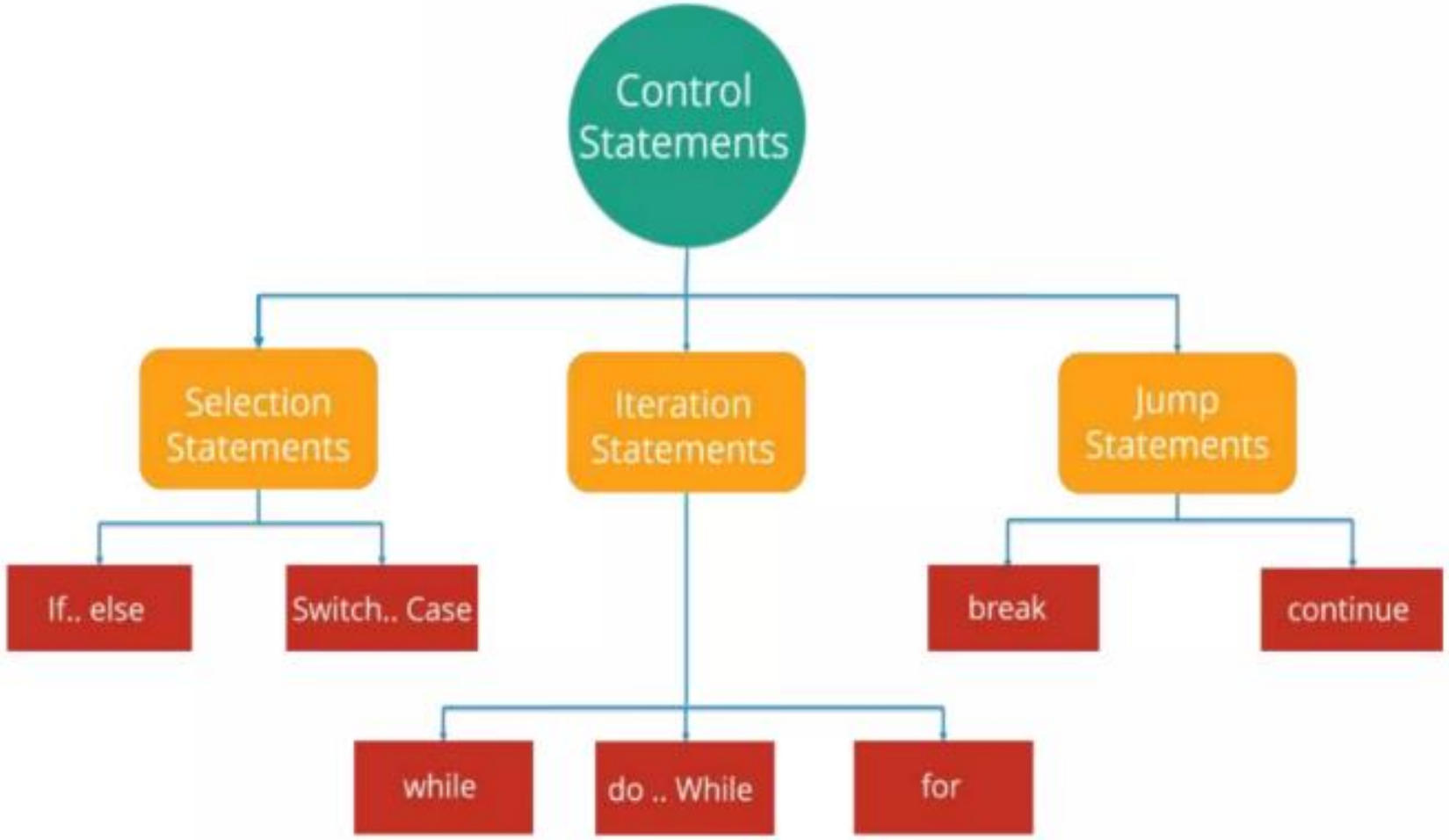
Department of Computer Science and Technology

Control Statements

- The control statement are used to **control the flow of execution** of the program.
- This execution order depends on the supplied **data values** and the **conditional logic**.
- In java program, control structure is can divide in three parts:

1. **Selection statement**
2. **Iteration statement**
3. **Jumps in statement**

Control Statements



Selection Statements

- statements allow a program to make decisions based on conditions
- Also called as Decision making Statements
- 2 Types
  - If Statement
  - Switch Statement
- Based on condition it control the program execution
- Provide power & Flexibility

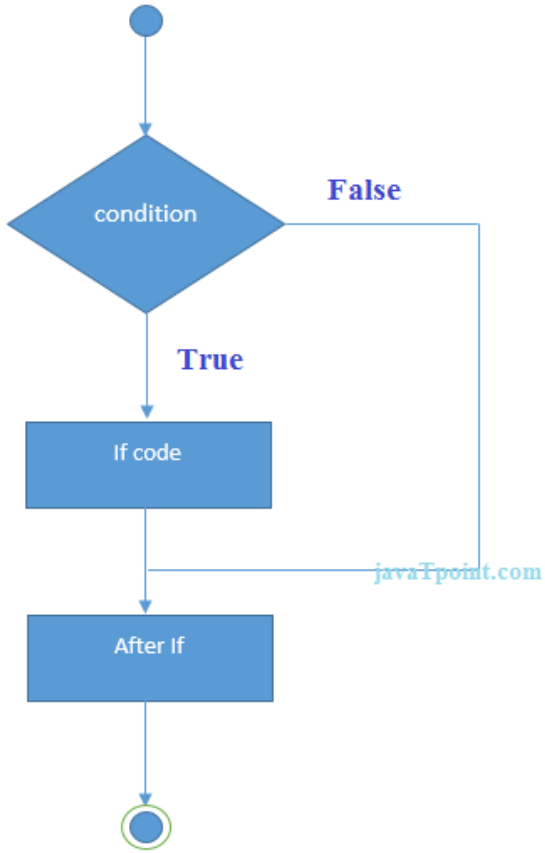
If Statements

- Used to test the condition
- It check Boolean condition: true or false
- Executes a block of code if the condition is true
- Route program execute through 2 different paths
- If types
  - If
  - If-else
  - Nested if else
  - If-else-if

If Statements

- Used to test the condition
- It executes if block if condition is true
- Syntax

```
if(condition)
{
        // Code block
}
```



If Statements

- Example

```
class Main {
  public static void main(String[] args) {
    int x = 20;
    int y = 18;
    if (x > y) {
      System.out.println("x is greater than y");
    }
  }
}
```

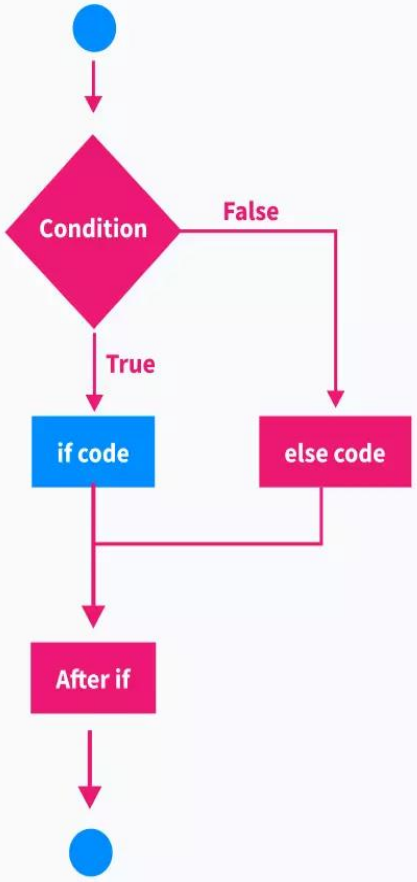
- Output

```
x is greater than y
```

If-else Statements

- Used to test the condition
- It executes if block condition is true otherwise else block will be executed
- Syntax

```
if(condition)
{
        // Code block
}
else
{
        // Code block
}
```



If-else Statements

- Example

```
class Main
{
  public static void main(String[] args)
  {
        int num = 10;
        if (num%2==0)
       {
            System.out.println(num+ “is even");
       }
       else
       {
            System.out.println(num+ “is odd");
       }
    }
}
```

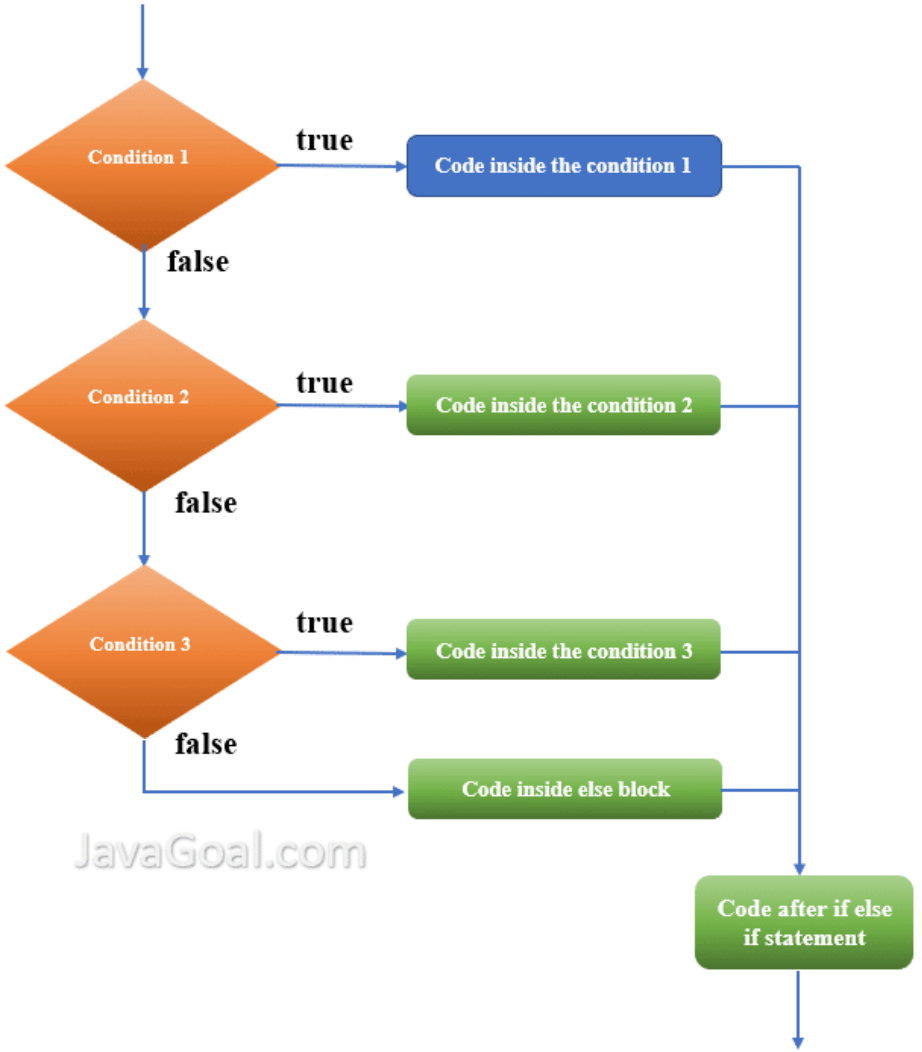
- Output

  10 is even

Nested If-else Statements

- Executes one if or else if statement inside another if or else if statement
- Syntax

```
if(condition1)
{
        // Code block
        if(condition2)
        {
                // Code block
        }
        else
        {
                // Code block
        }
}
```



Nested If-else Statements

- Example

```
class largest {
  public static void main(String[] args) {
    double n1=-4.5, n2=3.9, n3=5.5;
    if (n1>=n2) {
            if (n1>=n2)
                    System.out.println(n1 is largest no);
            else
                    System.out.println(n3 is largest no);
    }
  else {
   if (n2>=n3) {
            if (n1>=n2)
                    System.out.println(n2 is largest no);
            else
                    System.out.println(n3 is largest no);
    }
  } }
```

- Output

  n3 is largest no

if-else-if Statements

- Executes one if or else if statement inside another if or else if statement
- Syntax

```
if(condition)
        Statement1;
elseif(condition)
        Statement2;
elseif(condition)
        Statement3;
elseif(condition)
        Statement4;
else
        Statement5;
```

}

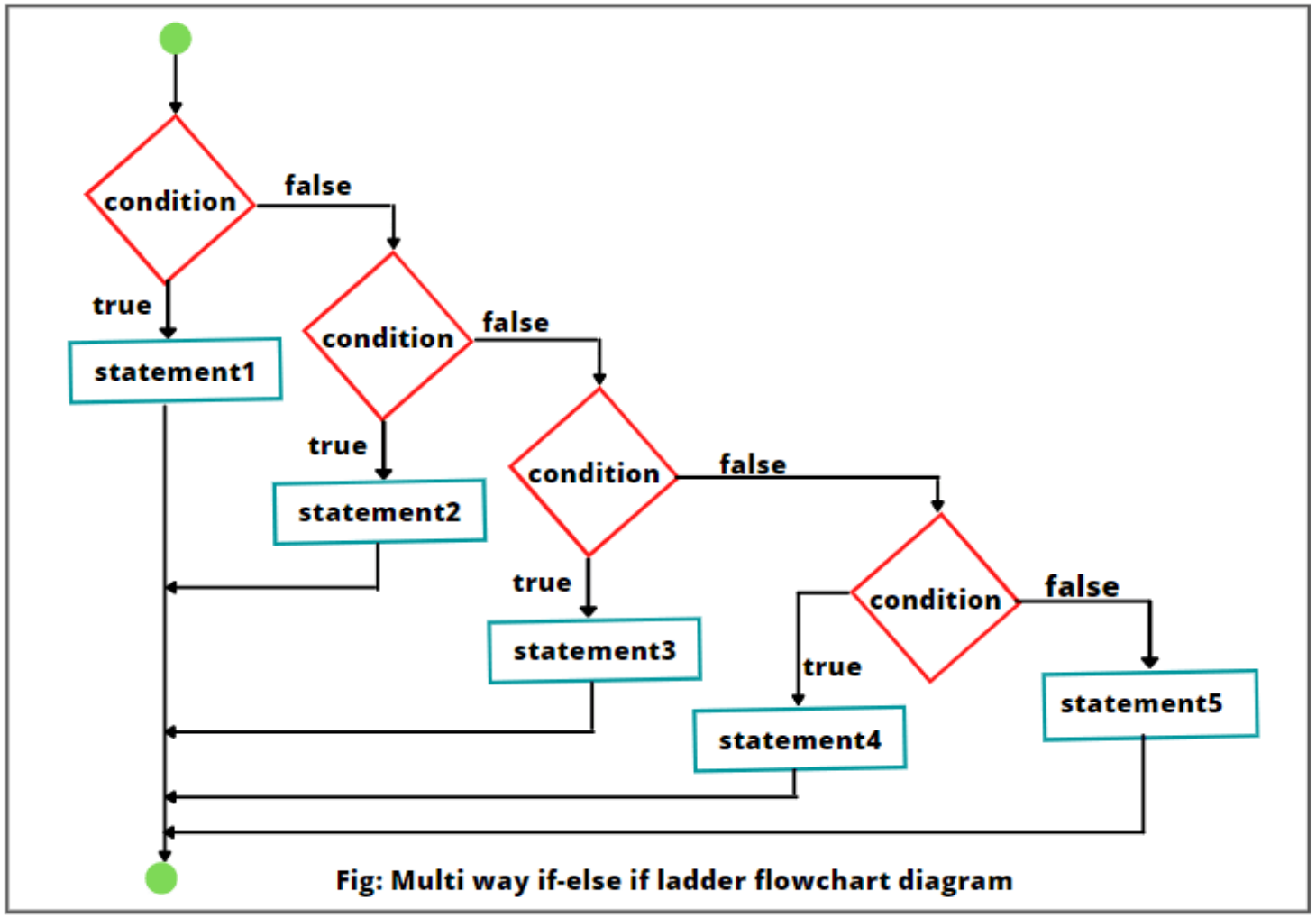


Fig: Multi way if-else if ladder flowchart diagram

if-else-if Statements

- Example

```
class largest
{
  public static void main(String[] args) {
   int  n1=10, n2=20, n3=15;
    if (n1>=n2 && n1>=n3)
            System.out.println(n1 is largest no);
    elseif (n2>=n1 && n2>=n3)
            System.out.println(n2 is largest no);
     else
            System.out.println(n3 is largest no);
  }
}
```

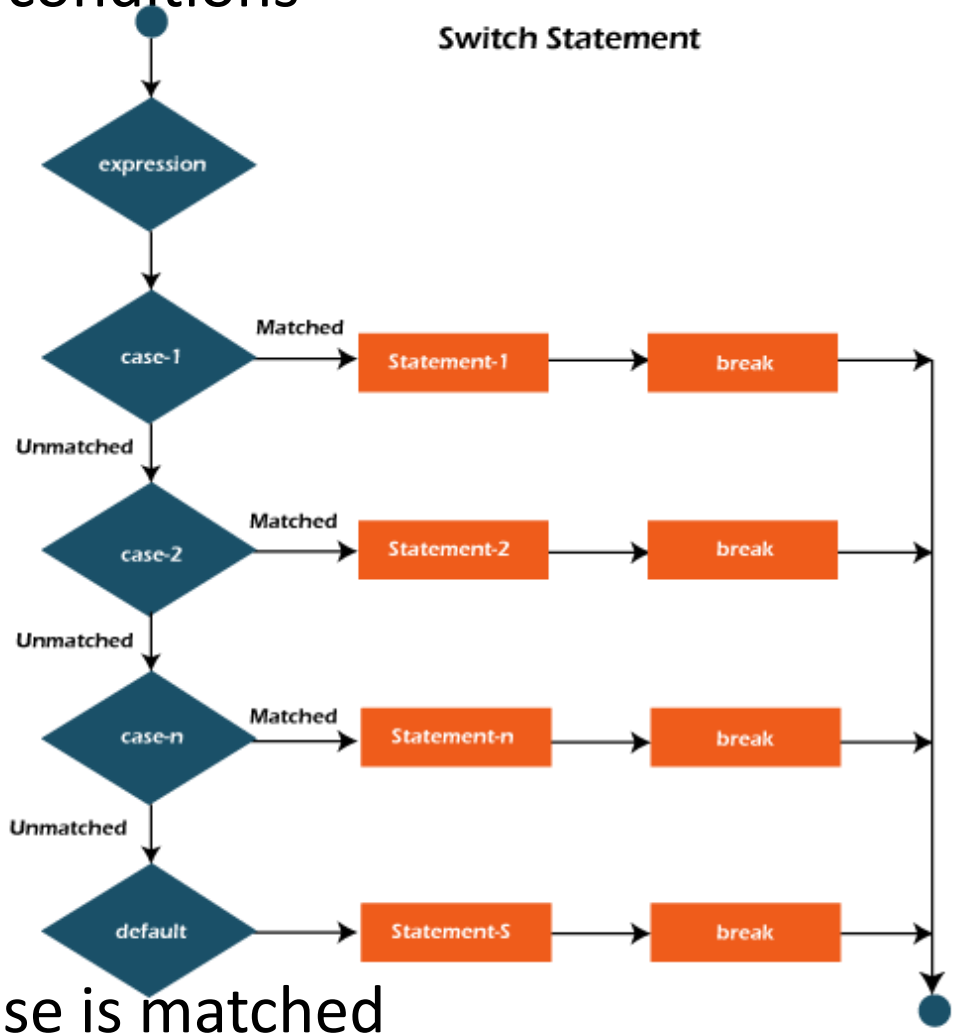
- Output

```
n3 is largest no
```

Switch Statements

- Executes one statement from multiple conditions
- Syntax

```
switch(expression)
{
        Case 1:
                Statements;
                Break;
        .
        .
        .
        default:
                execute when no case is matched
}
```



switch Statements

- Example

```
public class Main {
  public static void main(String[]
args) {
    int day = 4;
    switch (day) {
      case 1:

System.out.println("Monday");
        break;
      case 2:

System.out.println("Tuesday");
        break;
      case 3:

System.out.println("Wednesday")
;
        break;
      case 4:

System.out.println("Thursday");
        break;
      case 5:
        System.out.println("Friday");
        break;
      case 6:

System.out.println("Saturday");
        break;
      case 7:

System.out.println("Sunday");
        break;
    }
  }
}
```

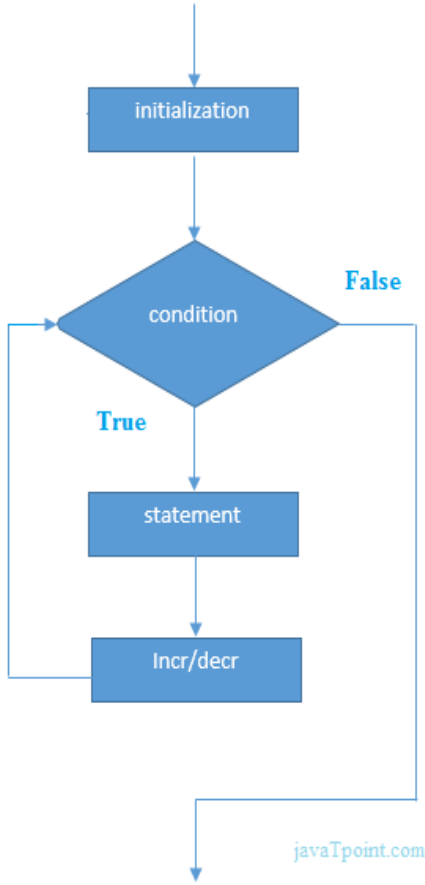
- Output

  Thursday

For Loop (Iteration Statement)

- Initialize variable , check condition and increment / decrement value
- Syntax

```
for(initialization; condition; increment/decrement)
{
    //statement or code to be executed
}
```



For loop

- Example

```
public class Example
{
public static void main(String[] args)
{
   //Code of Java for loop
   for(int i=1;i<=10;i++){
      System.out.println(i);
   }
}
}
```

- Output

1 2 3 4 5 6 7 8 9 10

While loop

- Used when the number of iterations is unknown

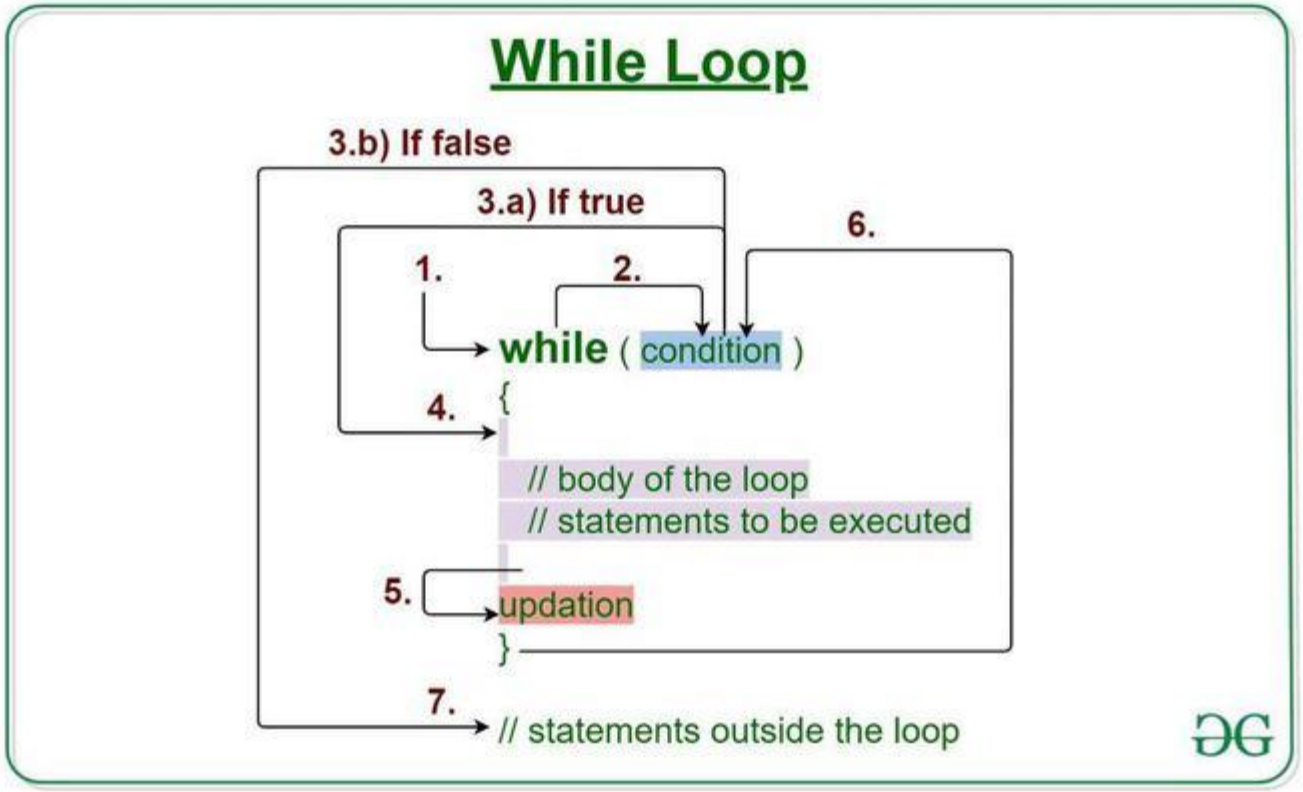
Syntax

```
While(Condition)
{
        //statement or code to be executed

}
```

Example

```
int i = 0;
while (i < 5)
{
     System.out.println(i);
    i++;
 }
```

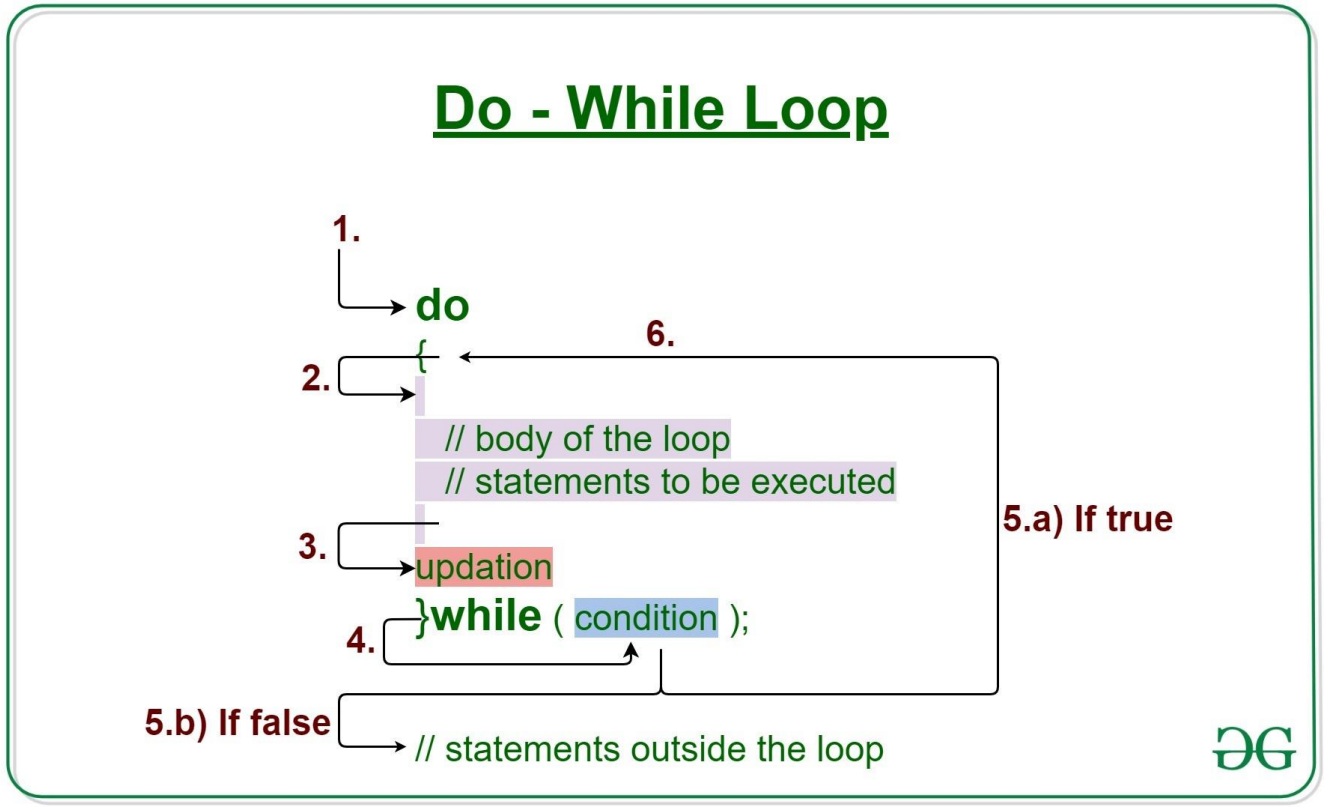


Do While loop

- Executes at least once before checking the condition

Syntax

```
Do
{
//statement or code to be executed
}
While(Condition);
```



Example:

```
int i = 0;
Do
 {
System.out.println(i);
i++;
} while (i < 5);
```

MCQ

What's wrong with this code?

```
public class WhilePuzzle {
    public static void main(String[] args) {
        int i = 5;

        while (i > 0) {

            System.out.println("i = " + i);

            i++;
        }}}
```

MCQ

```
public class BreakPuzzle {
    public static void main(String[] args) {
        for (int i = 1; i <= 3; i++) {

            for (int j = 1; j <= 3; j++) {

                if (j == 2) break; Stop when j = 2


                System.out.println(i + " " + j);

            }}}}
```

MCQ

```
public class BreakPuzzle {
    public static void main(String[] args) {
        for (int i = 1; i <= 3; i++) {

            for (int j = 1; j <= 3; j++) {

                if (j == 2) break; Stop when j = 2

                System.out.println(i + " " + j);

            }}}}
```

Output

1 1
2 1
3 1

MCQ

```
public class ContinuePuzzle {
    public static void main(String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {

                if (j == 2) continue; // Skip when j = 2
                System.out.println(i + " " + j);
            }
        }
    }
}
```

MCQ

```
public class ContinuePuzzle {
    public static void main(String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                if (j == 2)
            continue; // Skip when j = 2
                    System.out.println(i + " " + j);
                }
            }
        }
}
```

Output

```
1 1
1 3
2 1
2 3
3 1
3 3
```

MCQ

```
public class BreakContinueExample {
    public static void main(String[] args) {
System.out.println("Printing numbers from 1 to 10, skipping 5 and stopping at 8:");

  for (int i = 1; i <= 10; i++) {
          if (i == 5) {
              continue; // Skip 5
          }
          if (i == 8) {
              break; // Stop at 8
          }
          System.out.println(i);
      }

      System.out.println("Loop ended.");
    }
}
```

Output:

Printing numbers from 1 to 10, skipping 5 and stopping at 8:
1
2
3
4
6
7
Loop ended.

MCQ

```
public class ContinuePuzzle {
    public static void main(String[] args) {
        for (int i = 1; i <= 5; i++) {
            if (i % 2 == 0) continue;
            System.out.print(i + " ");
        }
    }
}
```

MCQ

```
public class ContinuePuzzle {
    public static void main(String[] args) {
        for (int i = 1; i <= 5; i++) {
            if (i % 2 == 0) continue;
            System.out.print(i + " ");
        }
    }
}
```

Output:

1 3 5

MCQ

```
int x = 2;
switch (x) {
    case 1: System.out.println("One");
    case 2: System.out.println("Two");
    case 3: System.out.println("Three");
}
```

MCQ

Output:

A) Two
B) Two Three ✓ (Because there's no break)
C) Error
D) No output

MCQ

What happens if a break statement is used inside a loop?

A) It skips the current iteration and moves to the next
B) It stops the entire loop execution
C) It throws an error
D) It restarts the loop

MCQ

What happens if a break statement is used inside a loop?

A) It skips the current iteration and moves to the next
B) It stops the entire loop execution ✓
C) It throws an error
D) It restarts the loop

Problems

You have a sorted array of numbers. Remove all duplicates in-place so that each number appears only once. Keep the order the same.

Input :[1, 1, 2, 2, 3, 4, 5, 5]

Output :[1, 2, 3, 4, 5]

References

- Java : the complete Reference (Eleventh Edition), Herbert Schildt, 2018.


Thank You!