Web Technology Presentation

GROUP - 6

CONTROL STATEMENTS IN JAVA

UNIVERSITY ROLL	NAMES
13000318045	Sohini Roy
13000318075	Rajashree Ghosh
13000318082	Moupiya Ghosh
13000318087	Madhusmita Misra

Why CONTROL Statements ??

Q Write the word "JAVA" 1000 times…

LOOP

for

while

do while

for each

Q If Sunny => Play
Or Cloudy => No play

DECISION

if

switch

Q Jump over a pit on the way…

JUMP

break

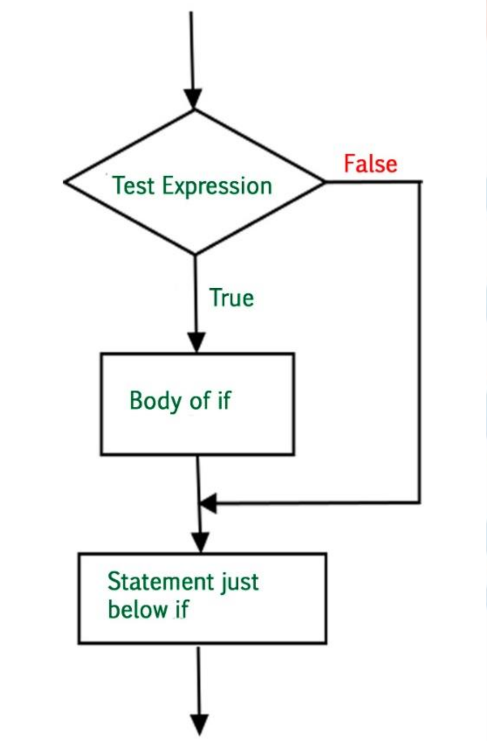
Continue

return

Decision Making Statements…

- decide **which statement** to execute and **when**
- **evaluate the Boolean expression** and **control the program flow** depending upon the **result of the condition** provided

If statement



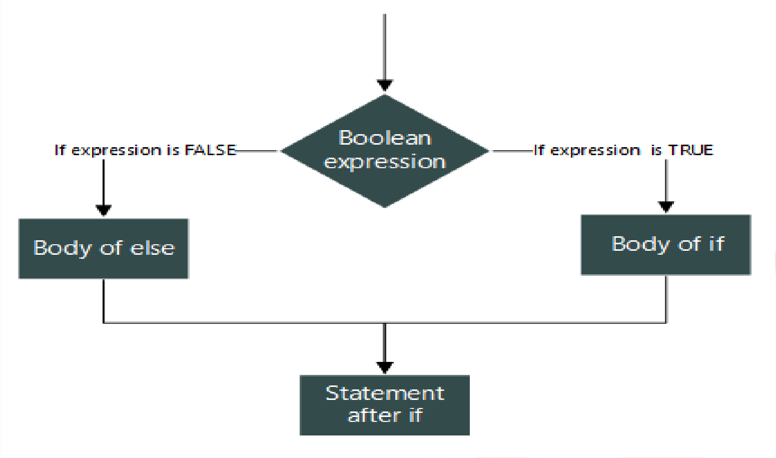
```
if(condition) {
    statement 1;
    //executes when
    condition is true
}
```

```
class Main {
  public static void main(String[] args) {
    // create a string variable
    String language = "Java";

    // if statement
    if (language == "Java") {
      System.out.println("Best Programming Language");
    }
  }
}
```

```
OUTPUT  :   Best Programming Language
```

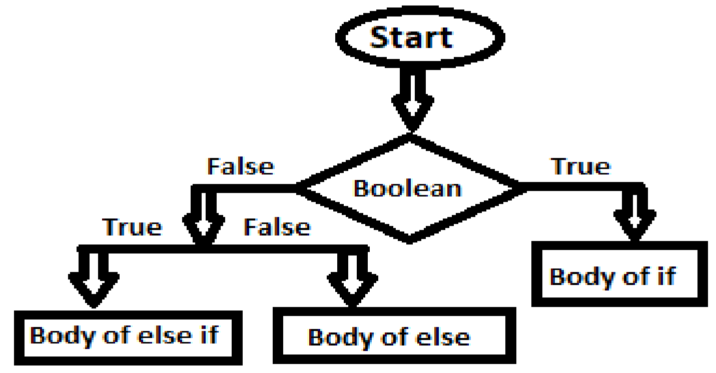
If - else Statement



```
class Main {
  public static void main(String[] args) {
    int number = 10;
    if (number > 0) {
      System.out.println("The number is positive.");
    }
    else {
      System.out.println("The number is not positive.");
    }
    System.out.println("Statement outside if...else block");
  }
}
```

```
OUTPUT  :   The number is positive
            Statement outside if...else block
```

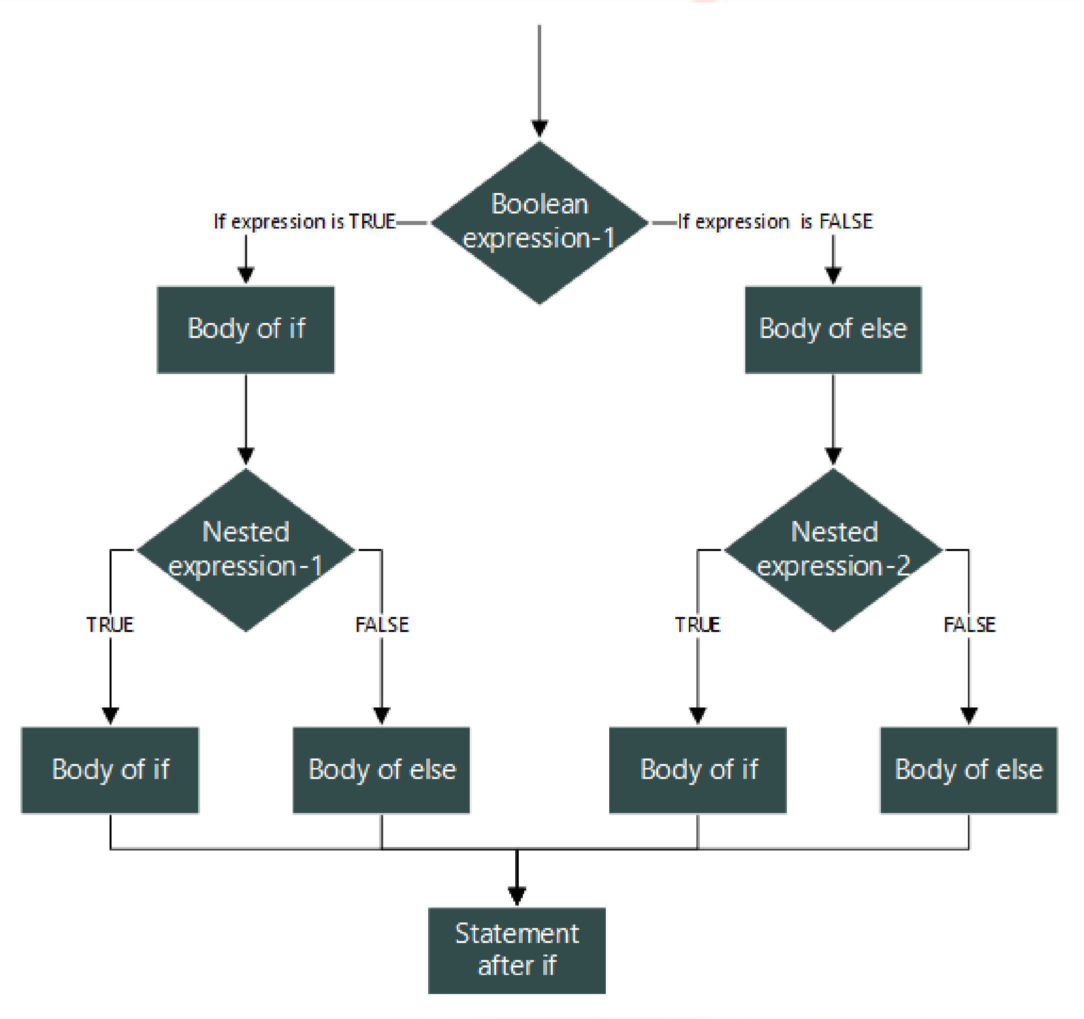
If - else - if Statement



```
class Main {
  public static void main(String[] args) {
    int number = 0;
    if (number > 0) {
      System.out.println("The number is positive.");
    }
    else if (number < 0) {
      System.out.println("The number is negative.");
    }
    else {
      System.out.println("The number is 0.");
    }
  }
}
```

```
OUTPUT  :   The number is 0.
```

Nested if Statement

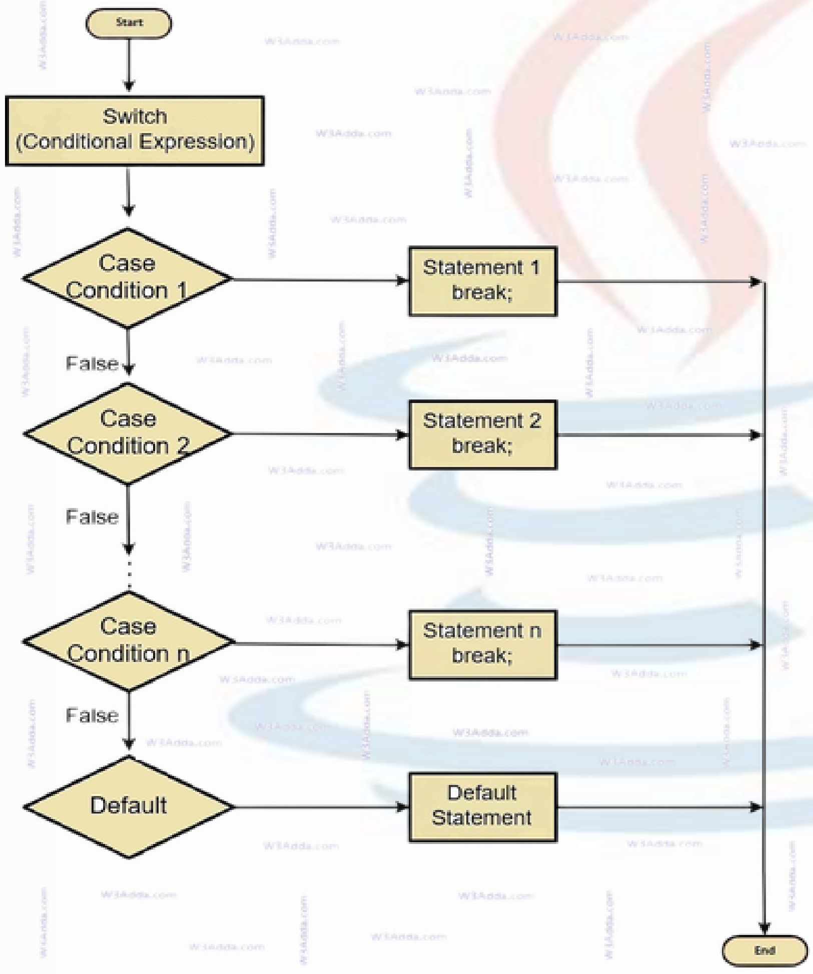


```
class Main {
  public static void main(String[] args) {
    Double n1 = -1.0, n2 = 4.5, n3 = -5.3, largest;
    if (n1 >= n2) {
        if (n1 >= n3) {
            largest = n1;
        }
        else {
            largest = n3;
        }
    } else {
        if (n2 >= n3) {
            largest = n2;
        }
        else {
            largest = n3;
        }
    }
    System.out.println("Largest Number: " + largest);
  }
}
```

OUTPUT : Largest Number : 4.5

Switch Statement

Switch is a Multi-Branch Statement,used to make selection of a choice from a number of options.It is useful for writing Menu-Driven Programs.



```
switch(constant expression)
//Integer or Character type
{
case value1:
      statement(s);
       break; //optional
case value2:
      statement(s);
       break; //optional
case value3:
      statement(s);
       break; //optional
default:
    default statement(s);

}
```

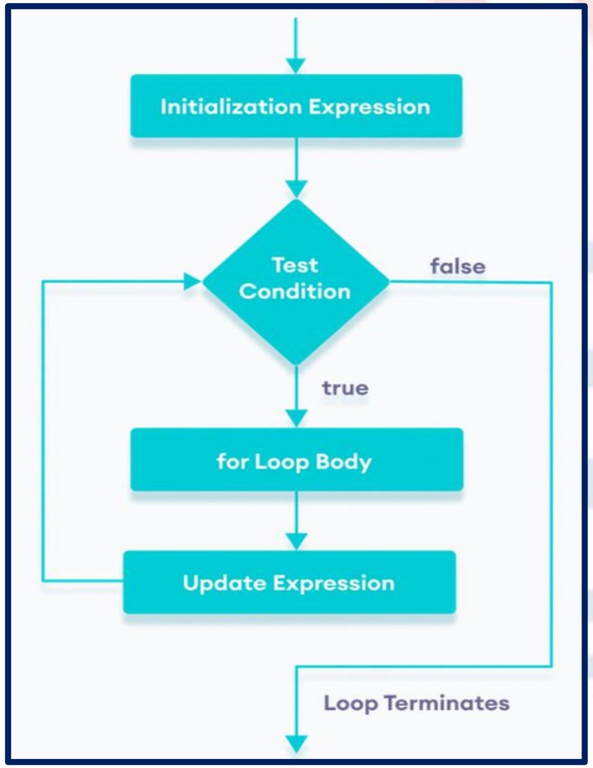
If expression matches value3,control jumps to here

```
char mygrade = 'B';
 switch(mygrade)
 {
      case 'A' :
       System.out.println("Your grade
is A" );
       break;
     case 'B' :
       System.out.println("Your grade
is B" );
       break;
      case 'C' :
       System.out.println("Your grade
is C" );
       break;
      default :
       System.out.println("Invalid
grade " );
   }
```

Output: Your grade is B

For Loop

```
for(initialization;condition;update)
{
  //body of the loop
 //statements to be executed
}
```



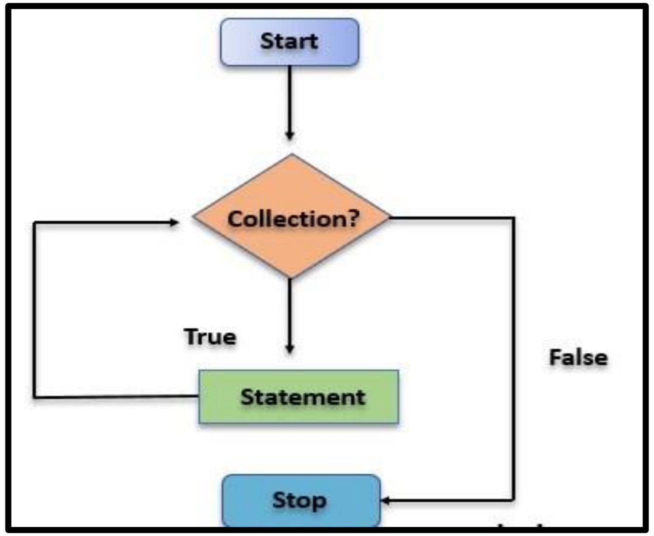
For-each Loop

```
for(data type variable : array | collection)
 {
  //body of for-each loop
 }
```

- ❑ No loop Counter
- ❑ No condition Check

```
 int arr[]={12,25,74,104};
//traversing the array with for loop
 for (int i=0; i<arr.length; i++)
  {
    System.out.print(arr[i] + " ");
   }
// using for-each loop
  for(int i:arr)
  {
   System.out.println(i);
   }
```

OUTPUT: 12 25 74 104



Nested Loop

```
for (int i = 1; i <= 5; i++)
//outer loop
{
 for (int j= 1; j <= i; J++)
    //inner loop
 {
     System.out.print(j+' ');
   }
 System.out.println();
}
```

```
OUTPUT:

1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

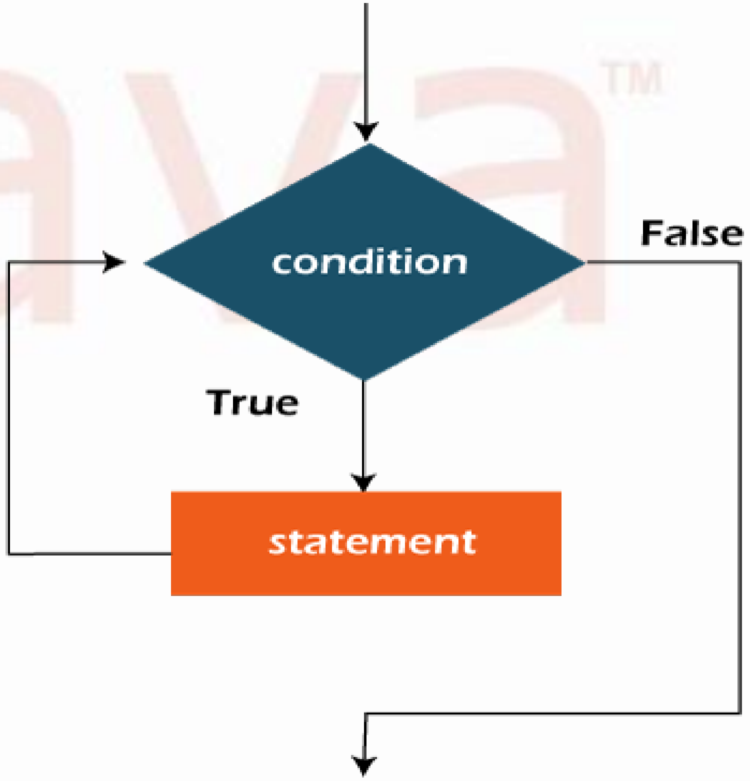
while Loop

- **Java while loop is a control flow statement that allows code to be executed repeatedly based on a given Boolean condition.**
- **While loop is used to iterate a part of the program repeatedly until the specified Boolean condition is true.**
- **If the Boolean condition becomes false, the loop automatically stops.**

Syntax :

```
while (test_expression)
{
        // statements
         update_expression;
}
```

Workflow of While loop



do-while Loop

- In do-while loop condition is evaluated after the execution of loop’s body.

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

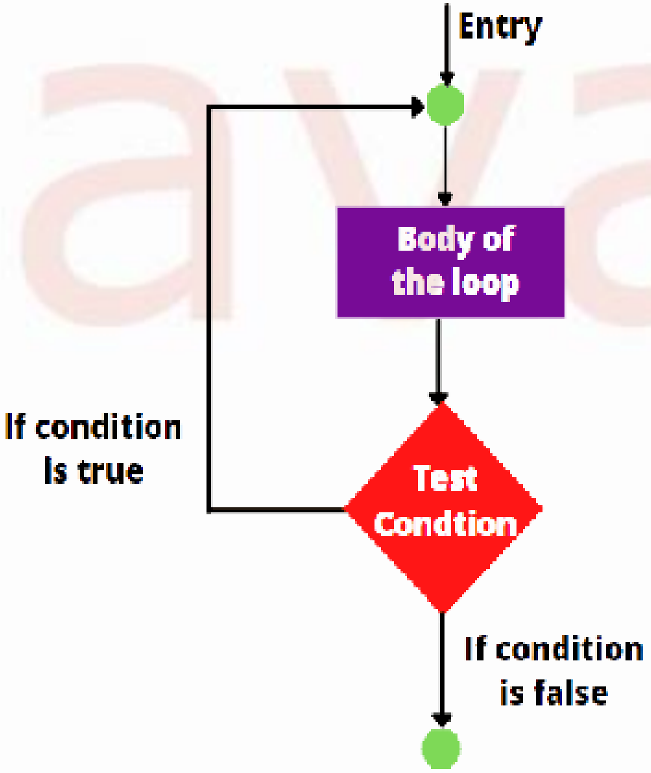
- Loop body is executed first, and then the loop conditional expression is evaluated to determine whether to continue or terminate the loop.

Syntax :

```
do {
        // Statements

}while(Boolean_expression);
```

Workflow of do-while loop



JUMP STATEMENTS

3 TYPES OF JUMP STATEMENTS

- *break*
- *continue*
- *return*

- ✓ **Jump statements** are used to unconditionally **transfer program control** from one point to elsewhere in the program.
- ✓ *Jump statements* are primarily **used to interrupt loop or switch-case instantly.**

BREAK STATEMENTS

- ❑ The break construct is used to break out of the middle of loops: **for, do, or while loop.**
- ❑ Execution of the current loops immediately stops and resumes at the first statement following the current loop.
- ❑ It is mostly used to exit early from the loop by skipping the remaining statements of loop or switch control structures.
- ❑ We can have more than one break statement in a loop.
- ❑ The break command terminates only the current loop and not any enclosing loops.

AN EXAMPLE...

```
class BreakStatement {
public static void main(String args[]){
     for(int i =1; i<=10; i++){
         System.out.println(i+" ");

          if(i==5){

          System.out.println(“\nYOU HAVE REACHED 5”);

          break;
          }
      }
      System.out.println(“\nLOOP ENDED DUE TO BREAK”);
   }
}
```

OUTPUT:-

```
1 2 3 4 5
YOU HAVE REACHED 5
LOOP ENDED DUE TO BREAK
```

CONTINUE STATEMENT

- ❑ Continue statement also skips the remaining statements of the body of the loop where it is defined.
- ❑ Instead of terminating the loop, the control is transferred to the beginning of the loop for next iteration.
- ❑ The loop continues until the test condition of the loop becomes false.

AN EXAMPLE...

```
class NumberExcept {
      public static void main(String args[] ) {
      int i;
      for(i=1;i<=10;i++)  {

          if(i==5)  continue;

          System.out.print(i +” “);
      }
   }
}
```

OUTPUT:-

```
1 2 3 4 6 7 8 9 10
```

RETURN STATEMENT

- ❑ This statement is mainly used in methods in order to terminate a method in between and return back to the caller method. It is an optional statement.
- ❑ That is, even if a method doesn’t include a return statement, control returns back to the caller method after execution of the method.
- ❑ Return statement may or may not return parameters to the caller method. For methods that **define** a return type, return statement **must be** immediately followed by return value.

AN EXAMPLE...

```
class ReturnExample{
      int Sum(int a, int b){

          return a+b;
      }
      public static void main(String args[]) {
       int c = Sum(2,5);
       System.out.print(c);
     }
}
```

OUTPUT:-

```
7
```

Thank You