Decision Making Statements

Ratna Mufidah, S.Kom., M.Kom.

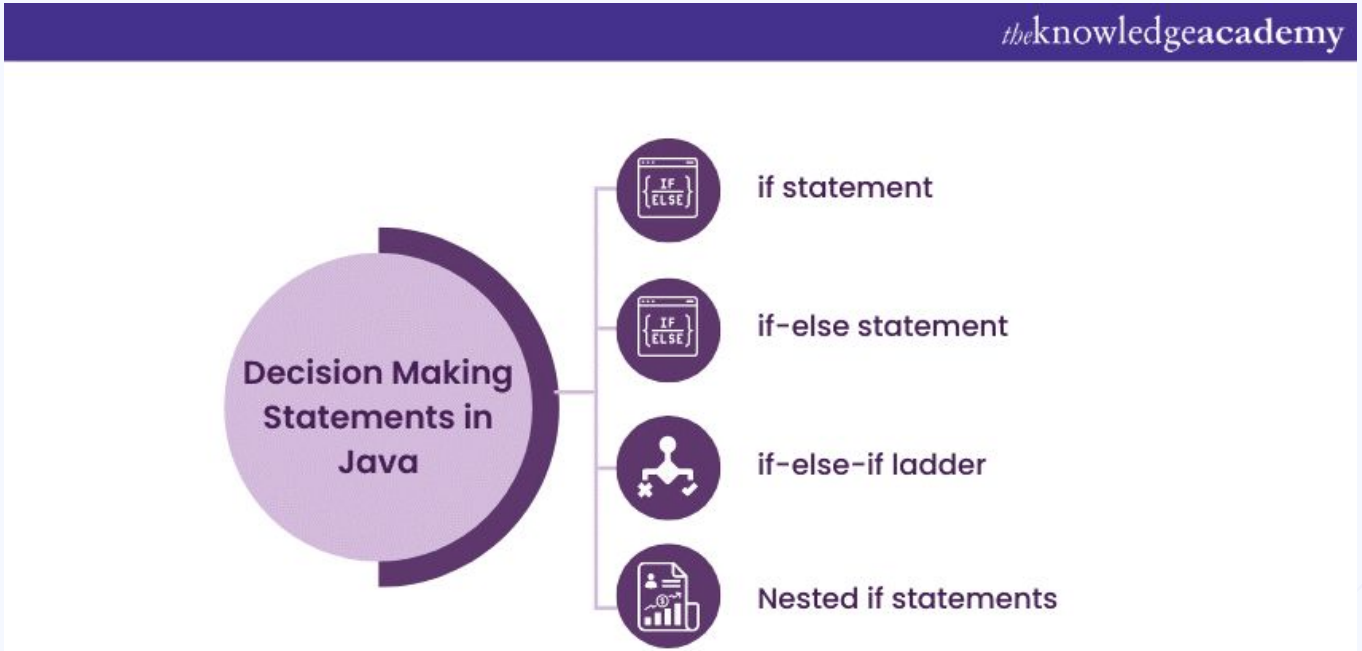
Decision Making Statements in Java

Decision Making is also known as “Control Flow”, “Condition Structure”, “IF Structure”, “Selection”, “Branching”, etc.

A branching or selection statement is a statement that **will be executed based on a certain condition**.

These statements enable Java programs to make informed choices and adapt to different scenarios dynamically.

Branching/ Selection Statements in Java

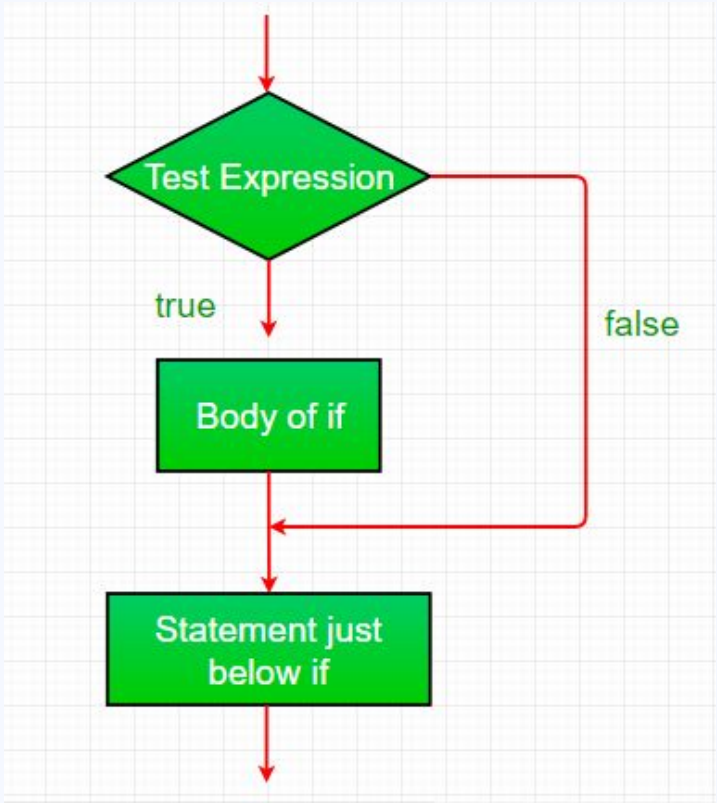


https://www.theknowledgeacademy.com/blog/decision-making-statements-in-java/

IF Statement

- Simple decision-making statement.
- “If statement” executes a block of code if a condition is true.

IF Statement



https://www.geeksforgeeks.org/decision-making-javaif-else-switch-break-continue-jump/

IF Statement

Syntax

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

IF Statement

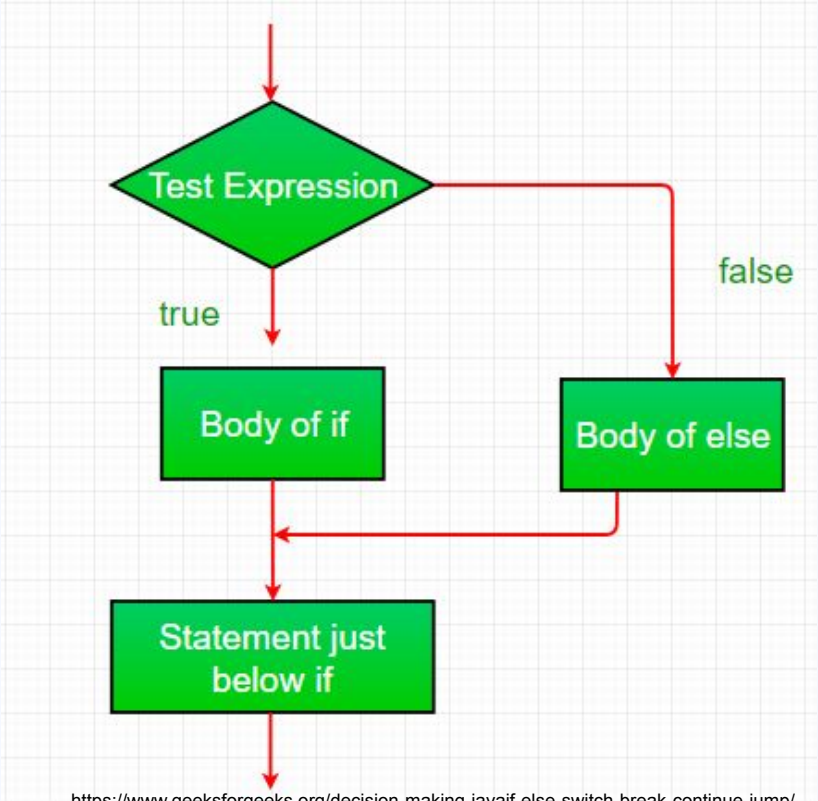
IfExample.java

```
1  public class IfExample {
2      public static void main(String[] args) {
3          int number = 10;
4          if (number > 5) {
5              System.out.println("The number is greater than 5.");
6          }
7
8          System.out.println("End of the program.");
9      }
10 }
```

IF-ELSE Statement

- What if we want to do something else if the condition is false? **Here comes the else statement**
- “If-else statement” executes the **if block** if condition is true otherwise **else block** is executed.

IF-ELSE Statement



https://www.geeksforgeeks.org/decision-making-javaif-else-switch-break-continue-jump/

IF-ELSE Statement

Syntax

```
if(condition){
    //code if condition is true
}
else{
    //code if condition is false
}
```

IF-ELSE Statement

IfElseExample.java

```
1  public class IfElseExample {
2      public static void main(String[] args) {
3          //defining a variable
4          int number=13;
5          //Check if the number is divisible by 2 or not
6          if(number%2==0){
7              System.out.println("even number");
8          }
9          else{
10             System.out.println("odd number");
11         }
12     }
13 }
```

IF-ELSE Statement using ternary operator (? :)

IfElseTerExample.java

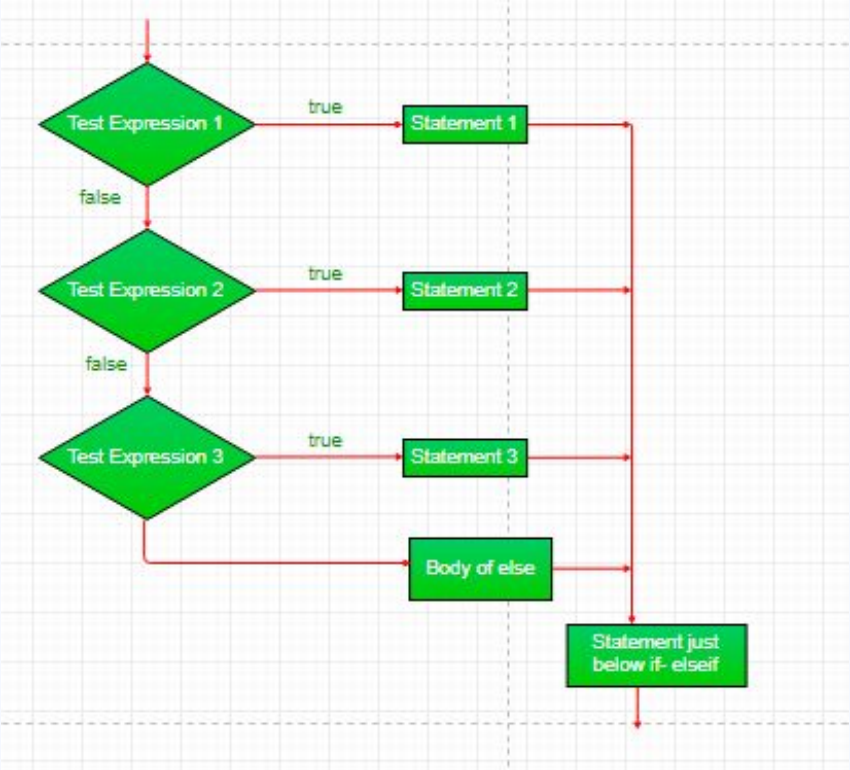
```
1  public class IfElseTerExample {
2
3      public static void main(String[] args) {
4          // TODO Auto-generated method stub
5          int number=13;
6          //Using ternary operator
7          String output=(number%2==0)?"even number":"odd number";
8          System.out.println(output);
9      }
10
11 }
```

IF-ELSE-IF Ladder Statement

- The "if-else-if ladder" is a sequence of conditional statements that allow developers to **check multiple conditions one by one** and execute the corresponding block of code based on the first true condition encountered.

- Each "if" condition is followed by an "else if" condition, except for the last one, which can be an "else" statement if none of the previous conditions evaluates to true.

IF-ELSE-IF Ladder Statement



https://www.geeksforgeeks.org/decision-making-javaif-else-switch-break-continue-jump/

IF-ELSE-IF Ladder Statement

Syntax

```
if(condition1){
    //code to be executed if condition1 is true
}else if(condition2){
    //code to be executed if condition2 is true
}
...
else{
    //code to be executed if all the conditions are false
}
```

IF-ELSE-IF Ladder Statement

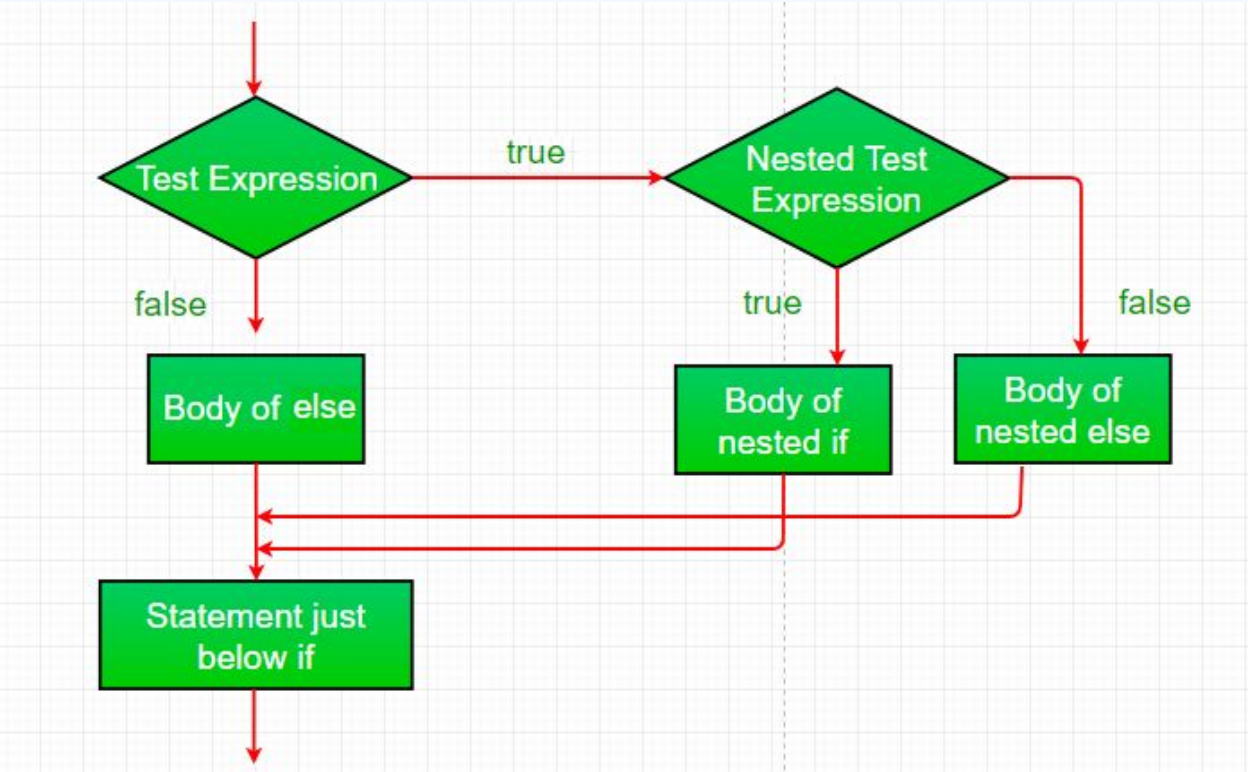
IfElseIfExample.java

```
1  public class IfElseIfExample {
2
3      public static void main(String[] args) {
4          // TODO Auto-generated method stub
5          int number = 10;
6
7          if (number > 0) {
8            System.out.println("The number is positive.");
9          }else if (number < 0) {
10           System.out.println("The number is negative.");
11         }else {
12           System.out.println("The number is zero.");
13         }
14
15         System.out.println("End of the program.");
16     }
17 }
```

Nested-IF Statement

- The nested if statement represents the if block within another if block.
- The **inner if block condition executes** only **when outer if block condition is true**.

Nested-IF Statement



https://www.geeksforgeeks.org/decision-making-javaif-else-switch-break-continue-jump/

Nested-IF Statement

Syntax

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

Nested-IF Statement

NestedIfExample.java

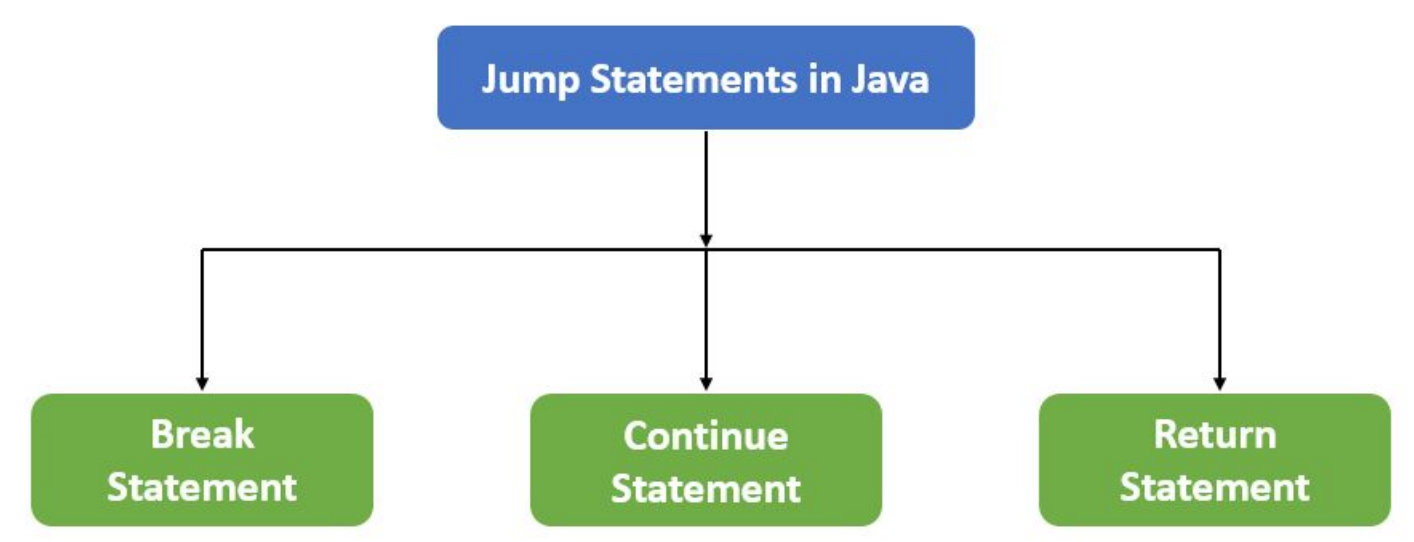
```
1  public class NestedIfExample {
2
3      public static void main(String[] args) {
4          // TODO Auto-generated method stub
5          //Creating two variables for age and weight
6          int age=20;
7          int weight=80;
8          //applying condition on age and weight
9          if(age>=18){
10             if(weight>50){
11                 System.out.println("You are eligible to donate blood");
12             }
13         }
14     }
15
16 }
```

Jump Statements

Jump statements are **used to jump the control to another part of our program depending on the conditions**.

Jump statements are typically **used to abruptly end a loop or switch-case**.

Jump Statements in Java



https://www.shiksha.com/online-courses/articles/jump-statements-in-java/

Jump Statements

-Break Statement-

The break statement is used to **exit loops prematurely**.

The break statement is widely used with the **switch** statement, **for** loop, **while** loop, **do-while** loop.

Jump Statements

-Break Statement-

Syntax 1

```
break;
```

Syntax 2

```
break label;
```

Jump Statements
-Break Statement-

BreakExample.java

```java
public class BreakExample {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        for (int i = 1; i <= 10; i++) {
            if (i == 5) {
                break; // The loop will terminate when i becomes 5
            }

            System.out.print(i + " ");
        }

    }

}
```

Jump Statements

-Break Statement-

BreakLabelExample.java

```java
public class BreakLabelExample {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        for (int i = 0; i < 3; i++) {
            one : { // label one
            two : { // label two
            three : { // label three
                System.out.println("i=" + i);
                if (i == 0)
                    break one; // break to label one
                if (i == 1)
                    break two; // break to label two
                if (i == 2)
                    break three; // break to label three
            }
                System.out.println("after label three");
            }
                System.out.println("after label two");
            }
                System.out.println("after label one");
        }
    }
}
```

Jump Statements

-Continue Statement-

The continue statement is a jump statement **used when a loop's current iteration is to be skipped**.

It allows developers to o**ptimize loops by excluding unnecessary computations**.

Jump Statements

-Continue Statement-

Syntax

```
continue;
```

Jump Statements

-Continue Statement--

ContinueExample.java

```java
public class ContinueExample {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        for (int i = 1; i <= 5; i++) {
            if (i == 3) {
                continue; // Skip iteration when i is 3
            }
            System.out.println(i + " ");
        }

    }

}
```

Jump Statements

-Return Statement-

The “return” keyword can help you **transfer control from one method to the method that called it**.

The return statement is **used for returning a value** when **the execution of the block is completed**.

Jump Statements
-Return Statement-

Syntax

```
return returnvalue;
```

Jump Statements
-Return Statement--

ReturnIntExample.java

```java
public class ReturnIntExample {

    public int CompareNum()
    {
        int x = 3;
        int y = 8;
        System.out.println("x = " + x + "\ny = " + y);
        if(x>y)
            return x;
        else
            return y;
    }

        /* Driver Code */
     public static void main(String ar[])
     {
        ReturnIntExample obj = new ReturnIntExample();
        int result = obj.CompareNum();
        System.out.println("The greater number among x and y is: " + result);
     }
}
```

Jump Statements

-Return Statement--

ReturnExample.java

```java
public class ReturnExample {
    private int a;
    public ReturnExample(int i)
    {
        a = i;
    }
    //The addition method returns a ReturnExample object
    //with adding 100 into it.
    public ReturnExample addition()
    {
        ReturnExample result = new ReturnExample(a + 100);

        return result;
    }

    public void display()
    {
        System.out.println("Additon result: " + a);
    }
}
```

ReturnDemo.java

```java
public class ReturnDemo {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        ReturnExample obj1 = new ReturnExample(50);
        ReturnExample obj2;
        /* addition method returns a reference
         * of ReturnExample class */
        obj2 = obj1.addition();
        obj2.display();
    }

}
```

Thanks!

Ada pertanyaan?

References

https://www.geeksforgeeks.org/
https://www.theknowledgeacademy.com/
https://www.w3schools.com/
https://www.javatpoint.com/


slidesgo