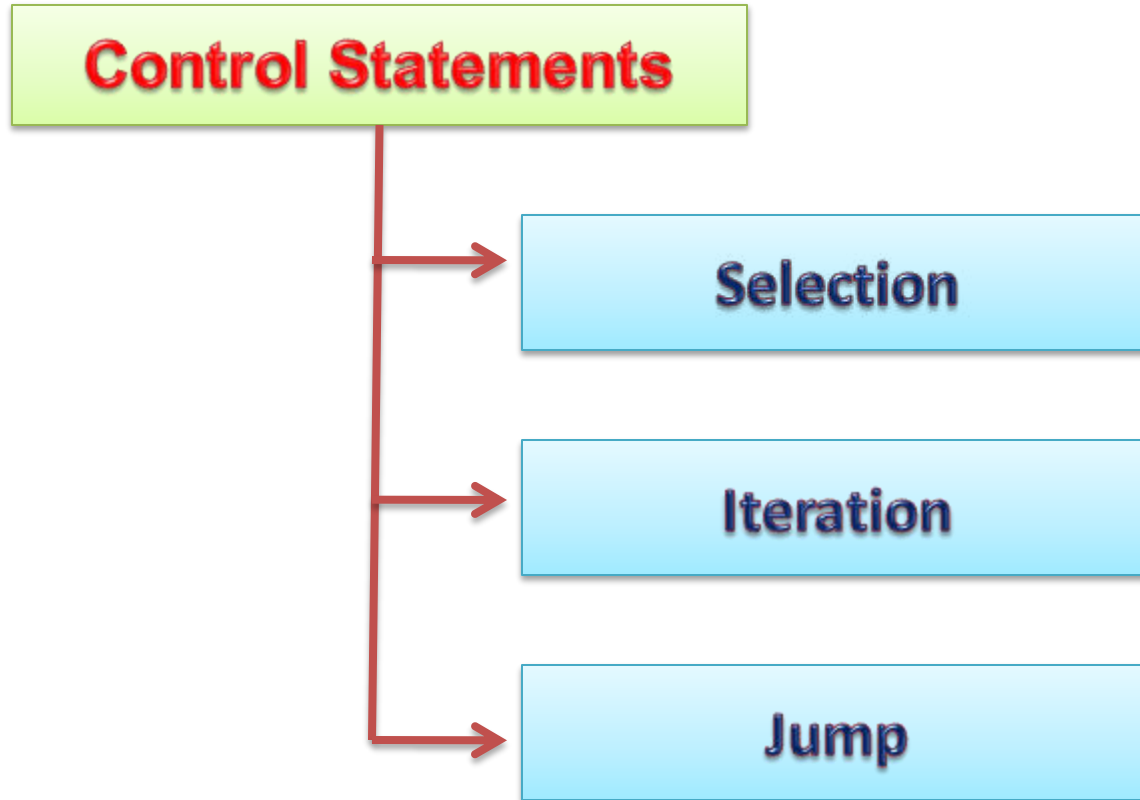


# CONTROL FLOW STATEMENTS IN JAVA





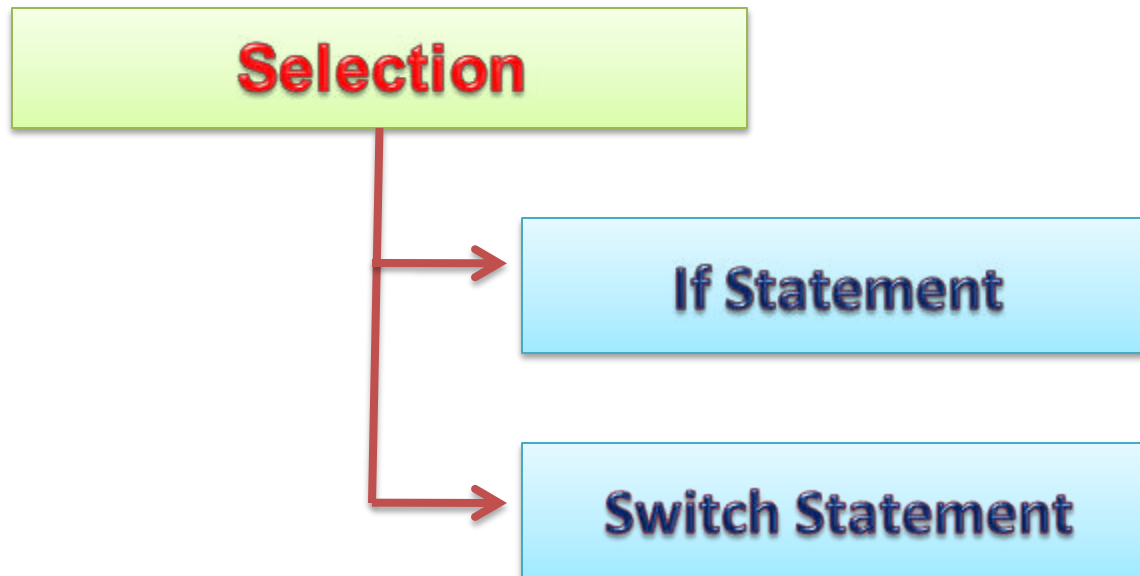
# Java – Control Flow Statements





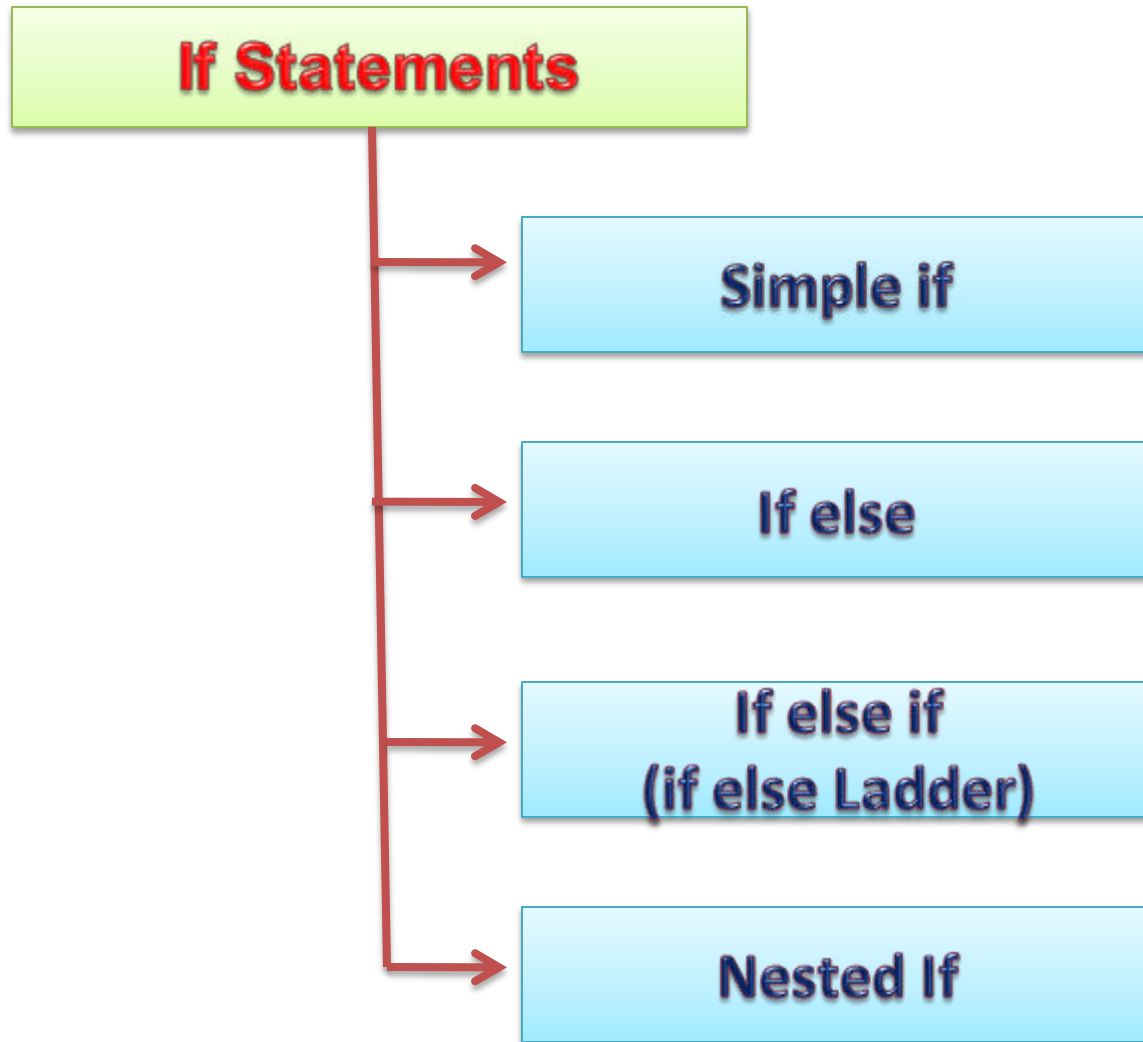
# Java – Control Flow Statements

Selection Statements are also called **Decision Making Statements**





# Java – Control Flow Statements



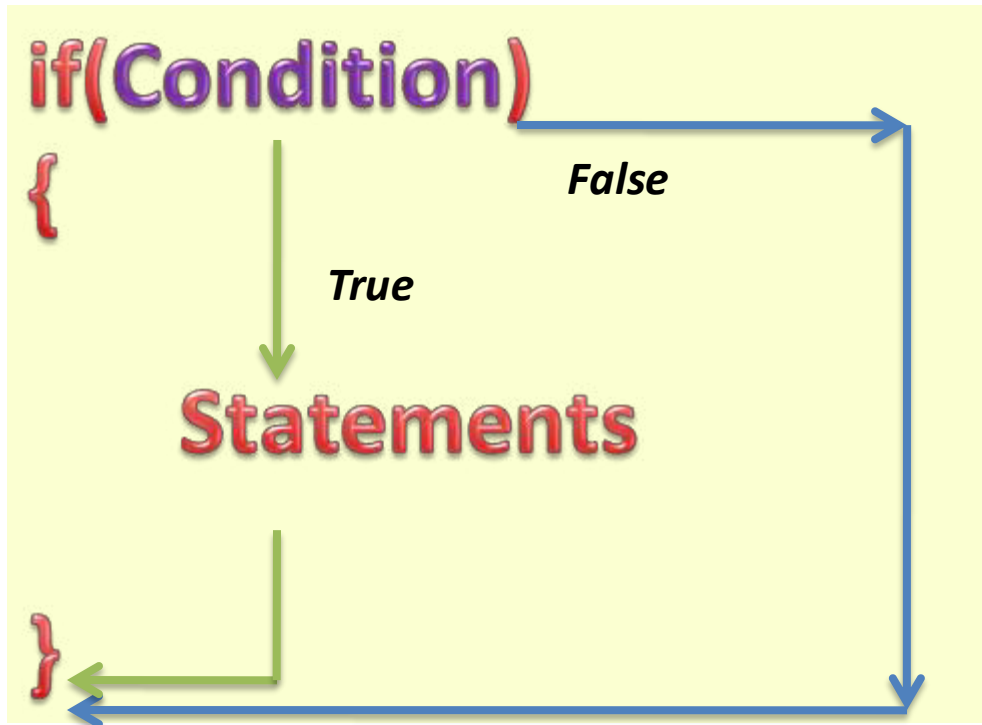


# Java – Control Flow Statements

## If Statement

It is used to take decision based on a single condition

Syntax:



If condition is True; control will enter the if block

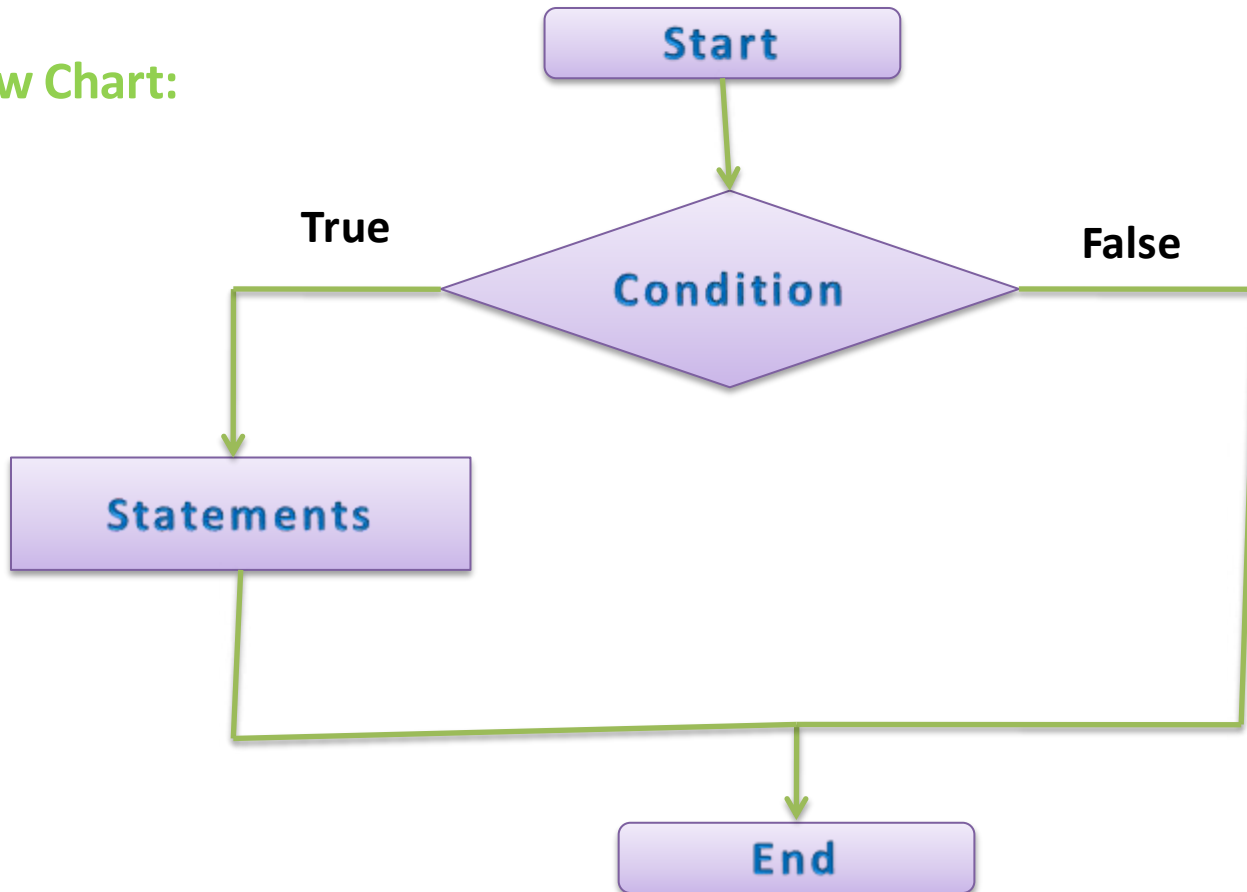
If condition is False; control will execute statement followed by if block



# Java – Control Flow Statements

## If Statement

Flow Chart:





# Java – Control Flow Statements

## If Statement

**Example:** Read marks from user and state only if user is pass.

```
Eg_if - Notepad
File Edit Format View Help
import java.util.Scanner;

class Eg_if
{
    public static void main(String[] args)
    {
        int marks;

        Scanner in=new Scanner(System.in);

        System.out.println("Enter Your Marks: ");

        marks=in.nextInt();

        if(marks >= 36)
        {
            System.out.println("You are Pass.");
        }
    }
}
```

Jyoti Lakhani

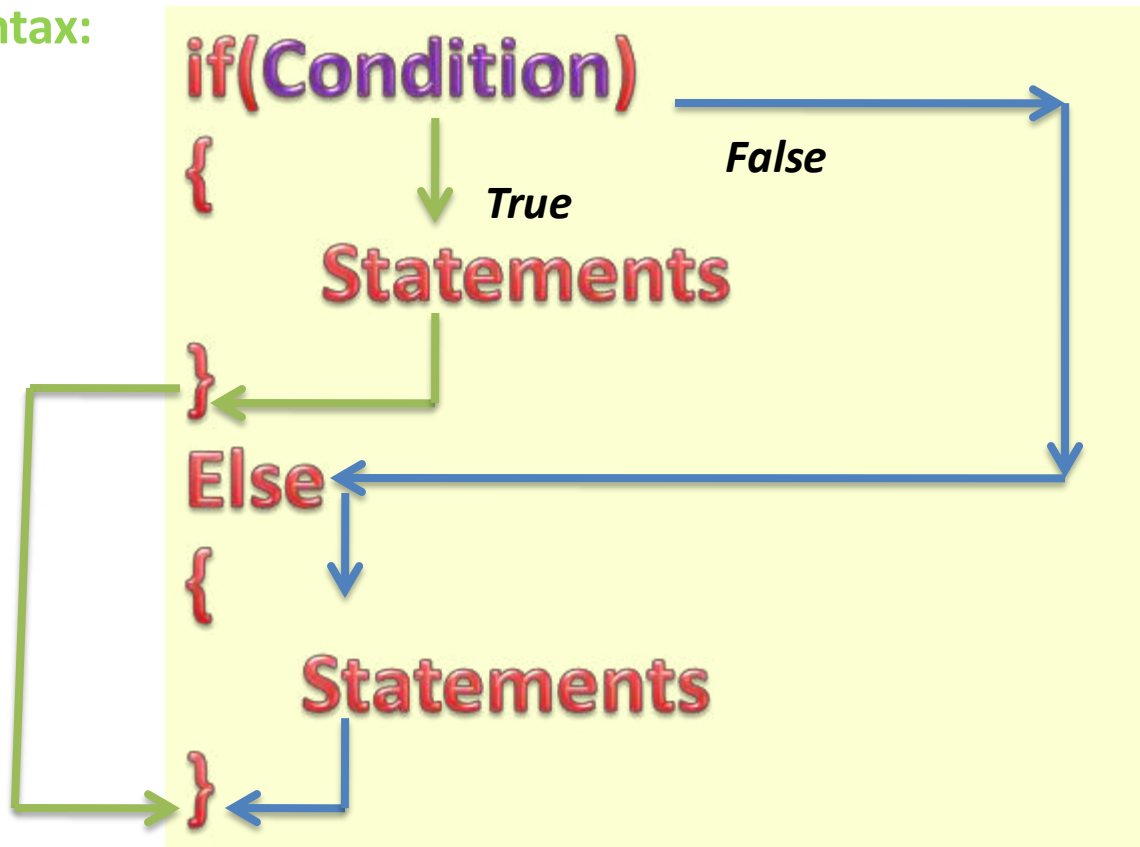


# Java – Control Flow Statements

## *if-else Statement*

It is used to take decision based on a single condition

Syntax:



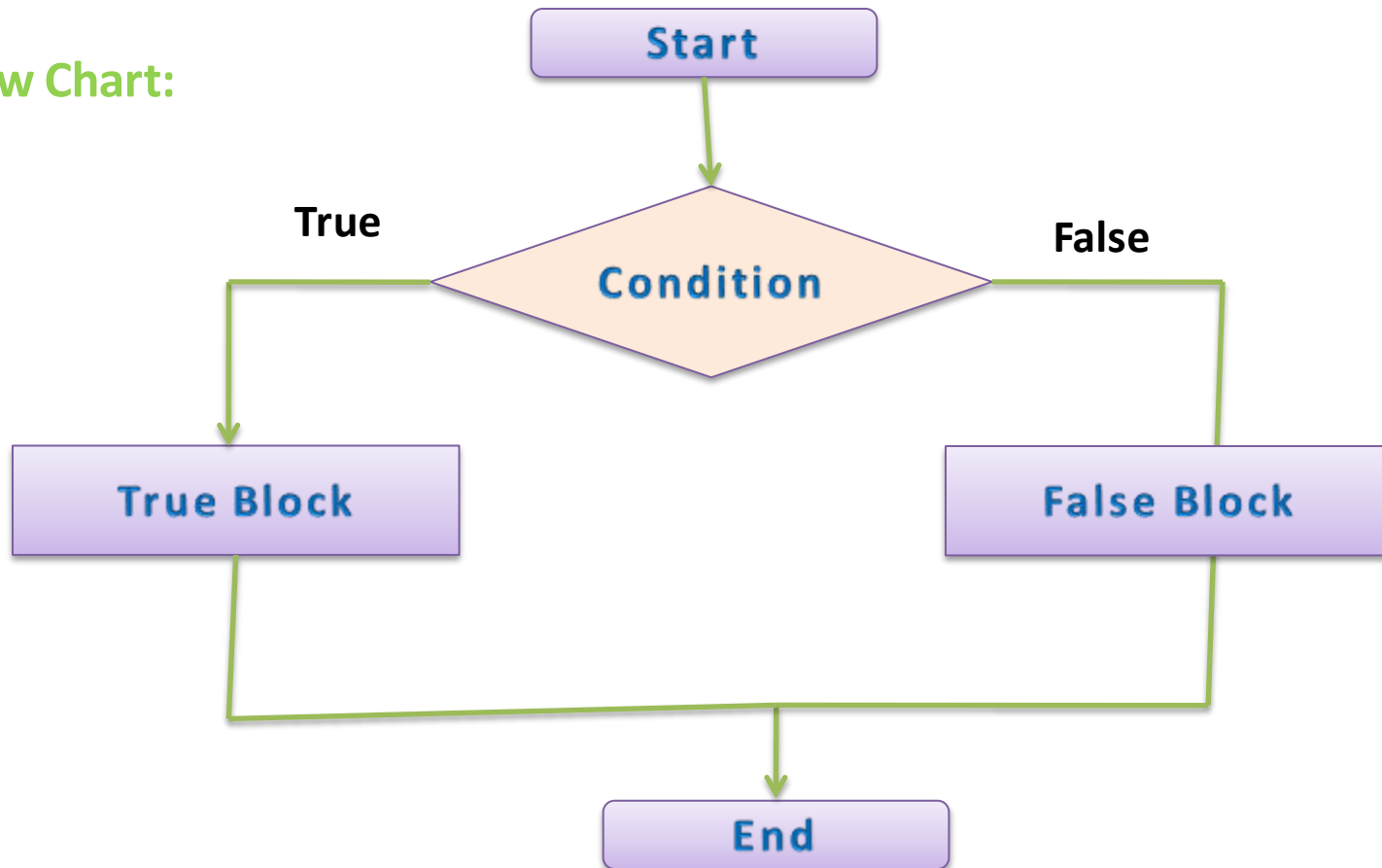




# Java – Control Flow Statements

## If-else Statement

Flow Chart:





# Java – Control Flow Statements

## *If-else Statement*

The if else statement has one condition and two statement blocks-  
True block and False block

If condition is True; control will execute the statement in the true block

If condition is False; control will execute the statement in false block



# Java – Control Statements

## *if-else Statement*

**Example:** Read marks from user and state whether student is Pass or Fail

```
Eq_if - Notepad
File Edit Format View Help
import java.util.Scanner;

class Eg_if
{
    public static void main(String[] args)
    {
        int marks;

        Scanner in=new Scanner(System.in);

        System.out.println("Enter Your Marks: ");

        marks=in.nextInt();

        if(marks>=36)
        {
            System.out.println("You are Pass.");
        }
        else
        {
            System.out.println("You are Fai");
        }
    }
}
```

Jyoti Lakhani



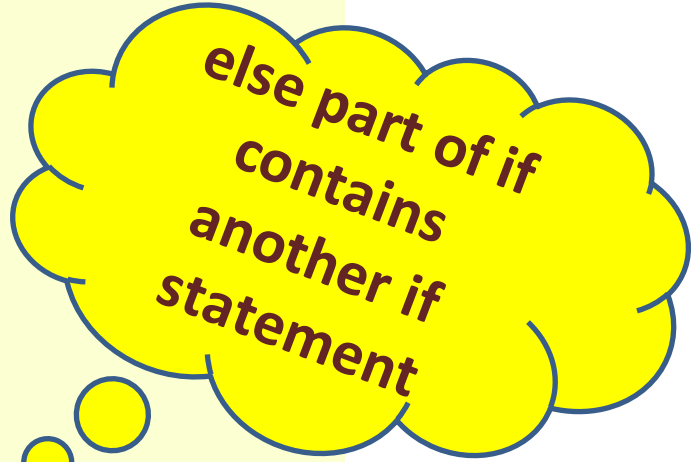
# Java – Control Statements

## *if-else-if Statement (Ladder if)*

It is used to take decision based on two conditions.

Syntax:

```
if(Condition1)
{
    Statements
}
else
{
    if(Condition2)
    {
        Statements
    }
}
```



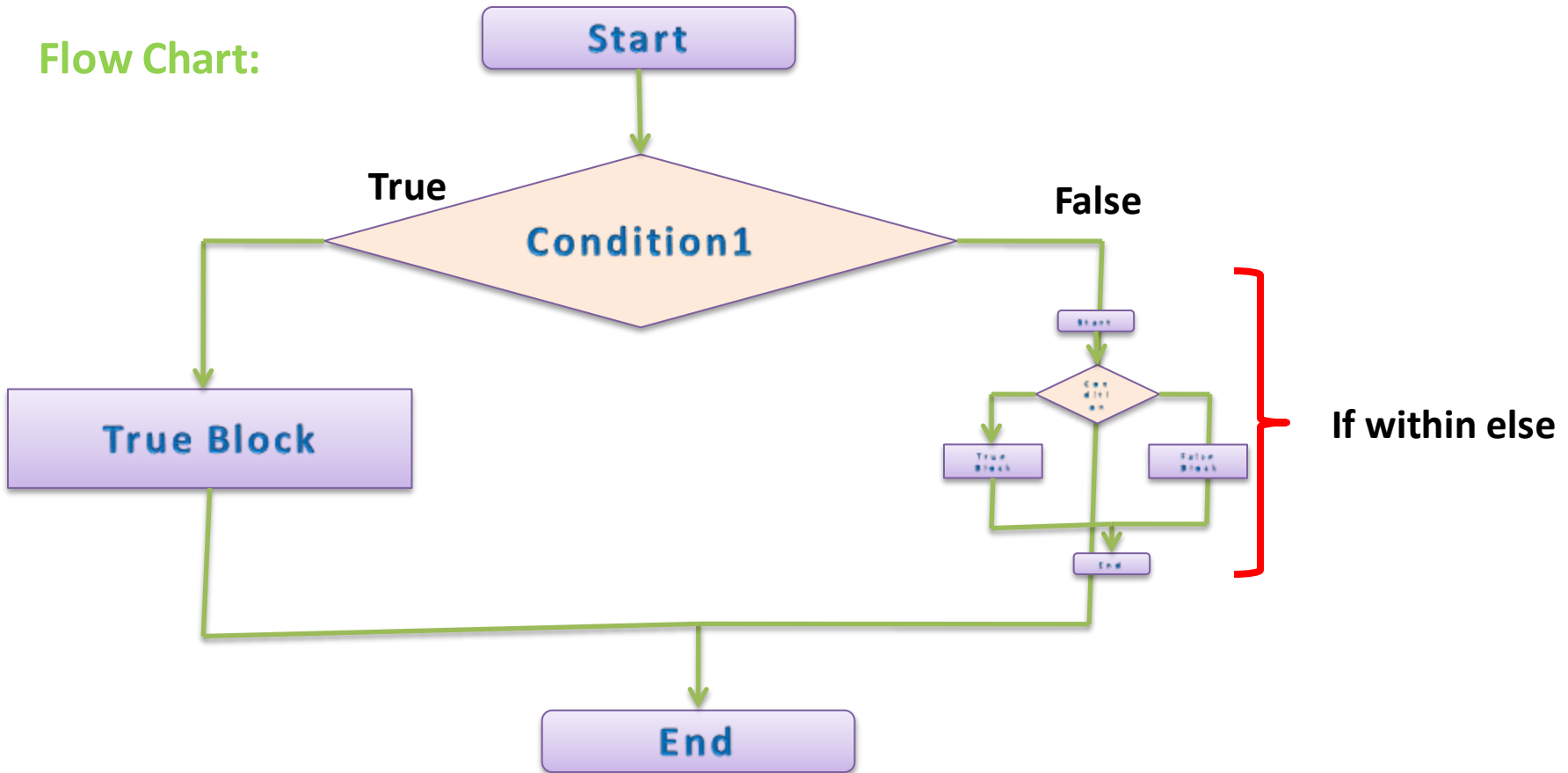
else part of if  
contains  
another if  
statement



# Java – Control Statements

## If-else-if Statement (Ladder if)

Flow Chart:





# Java – Control Statements

## *if-else-if Statement (Ladder if)*

Example

```
Eg_if - Notepad
File Edit Format View Help
import java.util.Scanner;

class Eg_if
{
    public static void main(String[] args)
    {
        int marks;

        Scanner in=new Scanner(System.in);

        System.out.println("Enter Your Marks: ");

        marks=in.nextInt();

        if(marks<36)
        {
            System.out.println("You are Fail.");
        }
        else
        {
            if(mark >=60)
            {
                System.out.println("You have got Ist Division.");
            }
            else
            {
                System.out.println("You have got IInd Division.");
            }
        }
    }
}

Jyoti Lakhani
```



# Java – Control Statements

## Nested-if Statement

The Nested if can be inside the if-part or else-part

Syntax:

```
if(Condition1)
{
    Statements
}
else
{
    if(Condition2)
    {
        Statements
    }
}
```

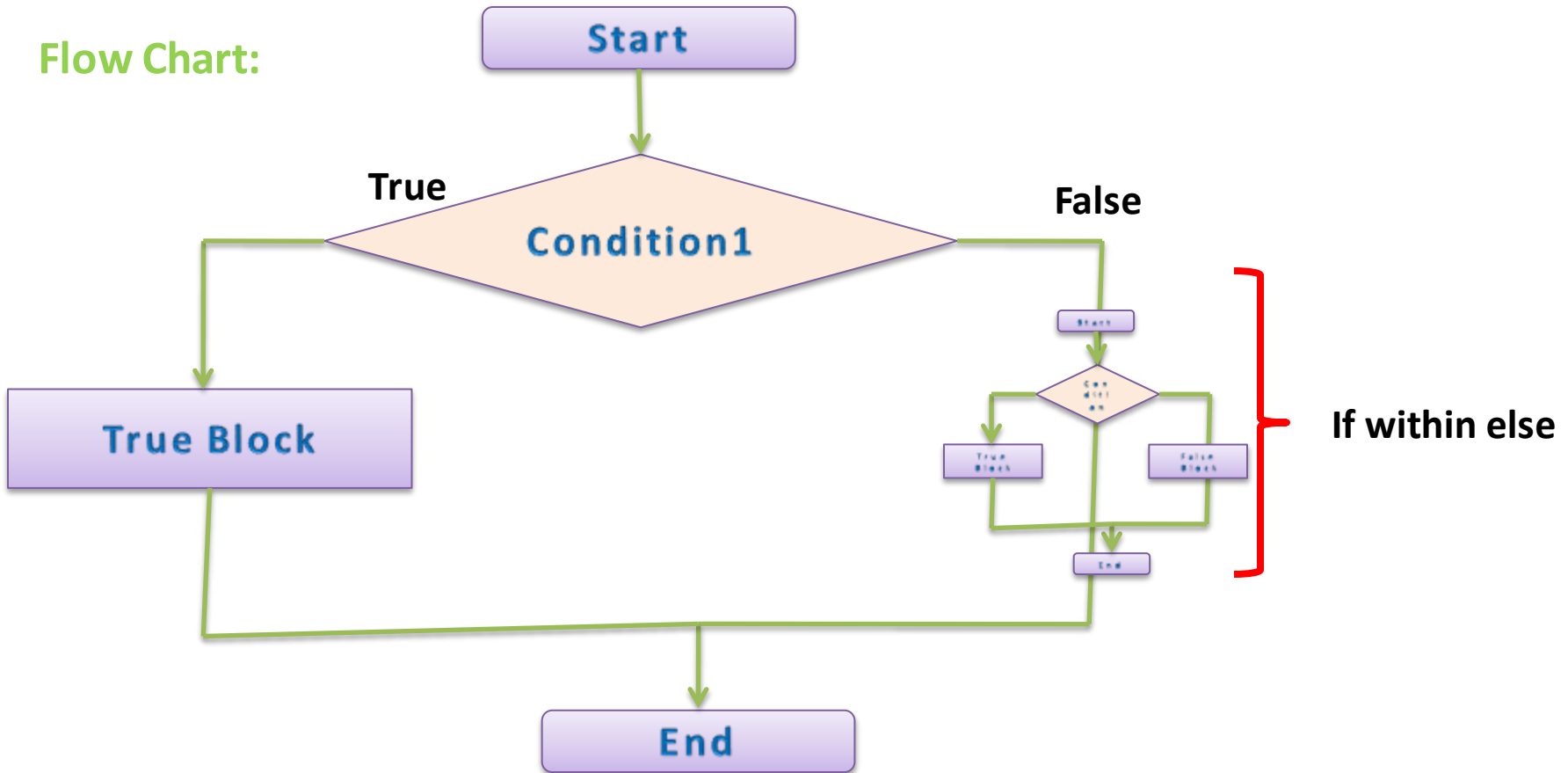
```
if(Condition1)
{
    if(Condition2)
    {
        Statements
    }
}
else
{
    Statements
}
```



# Java – Control Statements

## Nested-if Statement

Flow Chart:



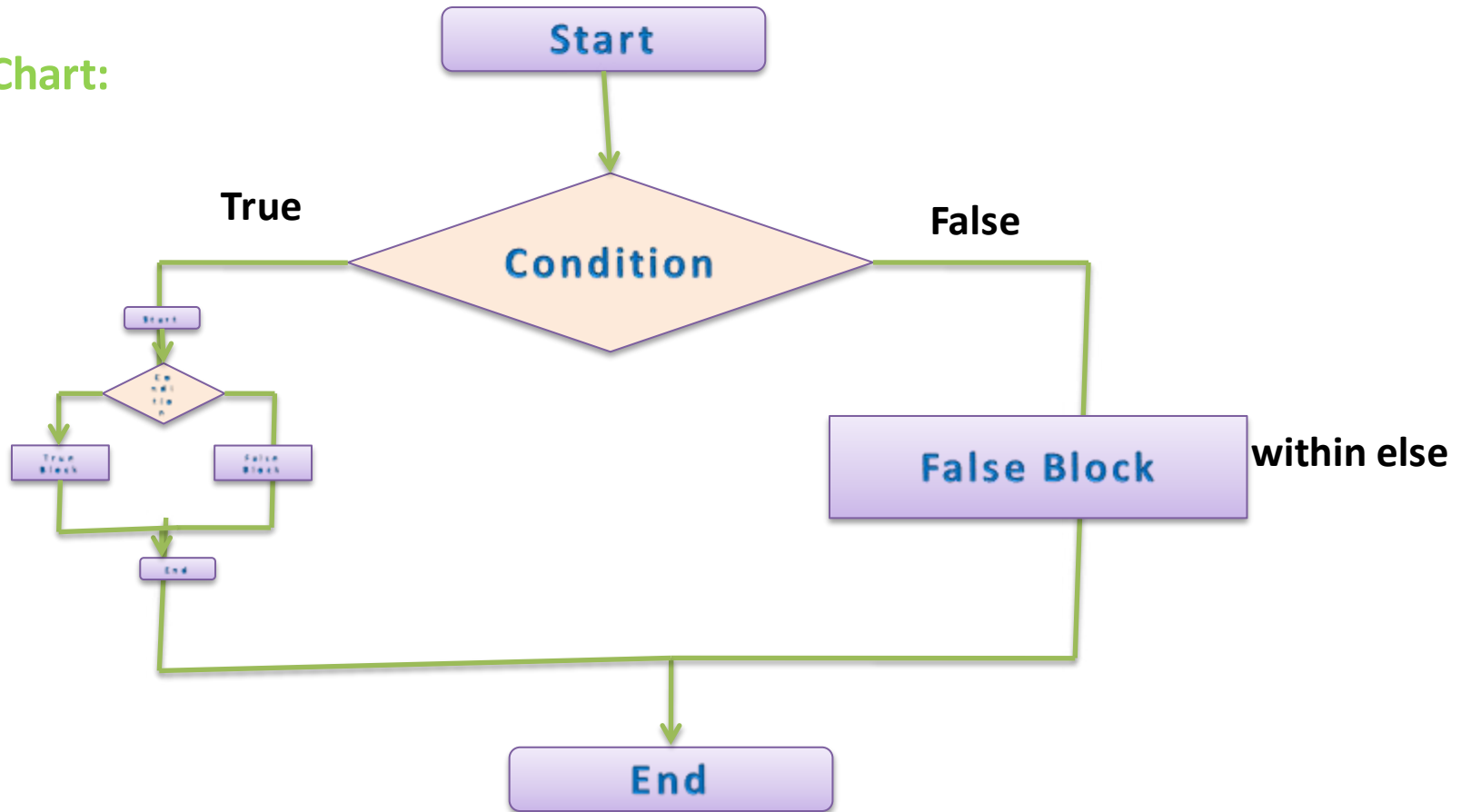




# Java – Control Statements

## Nested-if Statement

Flow Chart:





# Java – Control Statements

## Nested-if Statement

Example

```
Eg_if - Notepad
File Edit Format View Help
import java.util.Scanner;

class Eg_if
{
    public static void main(String[] args)
    {
        int marks;

        Scanner in=new Scanner(System.in);

        System.out.println("Enter Your Marks: ");

        marks=in.nextInt();

        if(marks<36)
        {
            System.out.println("You are Fail.");
        }
        else
        {
            if(mark >=60)
            {
                System.out.println("You have got Ist Division.");
            }
            else
            {
                System.out.println("You have got IInd Division.");
            }
        }
    }
}

Jyoti Lakhani
```



# Java – Control Statements

## Nested-if Statement

Example:

```
Eg_if - Notepad
File Edit Format View Help
import java.util.Scanner;

class Eg_if
{
    public static void main(String[] args)
    {
        int marks;

        Scanner in=new Scanner(System.in);

        System.out.println("Enter Your Marks: ");
        marks=in.nextInt();

        if(marks>=36)
        {
            if(mark >=60)
            {
                System.out.println("You have got Ist Division.");
            }
            else
            {
                System.out.println("You have got IInd Division.");
            }
        }
        else
        {
            System.out.println("You are Fail.");
        }
    }
}

Jyoti Lakhani
```



# Java – Control Statements

## switch Statement

A switch statement is used to test many conditions

Syntax:

```
switch(expression/variable)
{
    Case value:
        case statements
        break // optional
    Case value:
        case statements
        break// optional
        :
    default://optional
        default statements
}
```

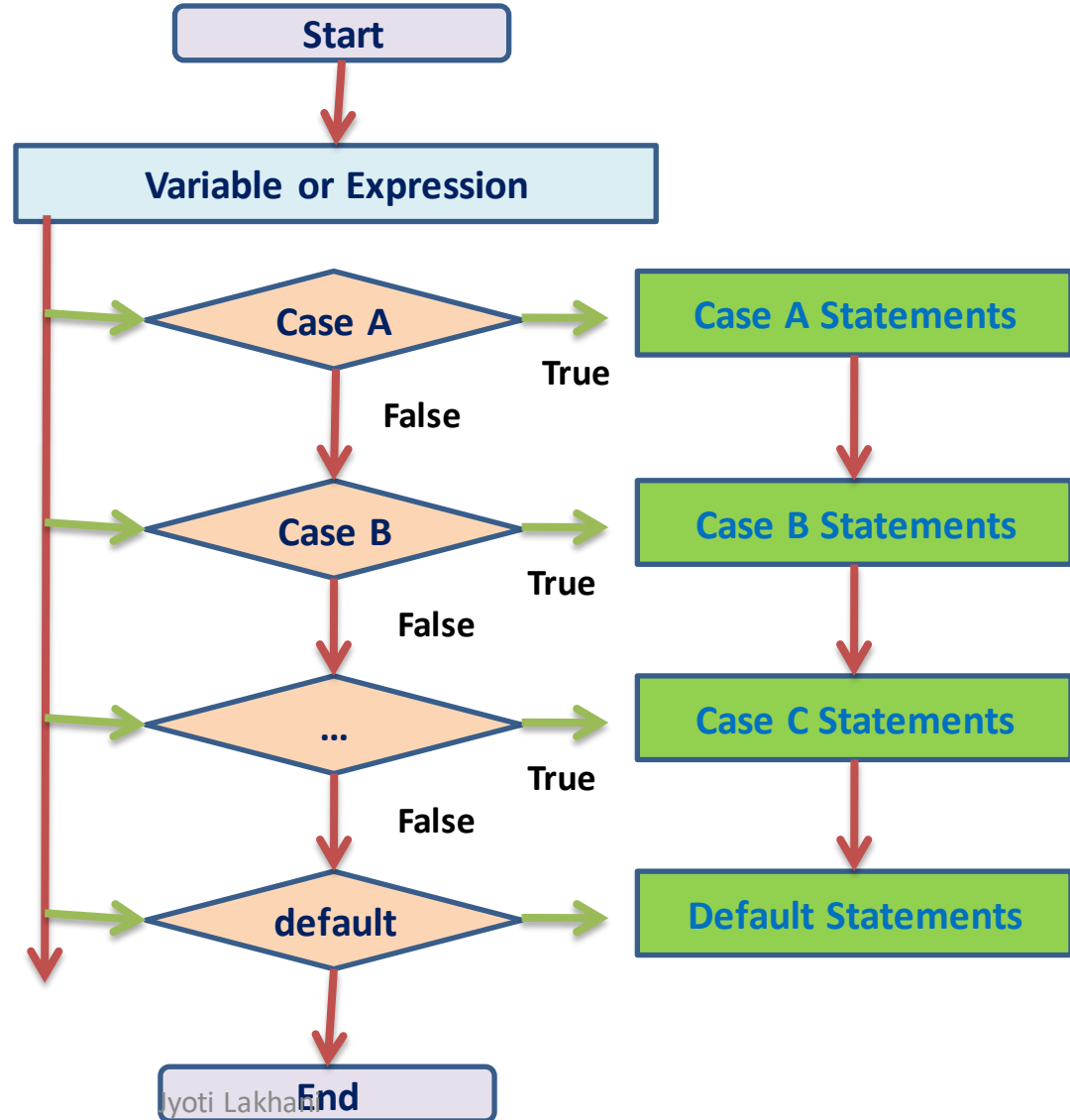


# Java – Control Statements

## switch Statement

Flow Chart:

Switch Statement without “break”



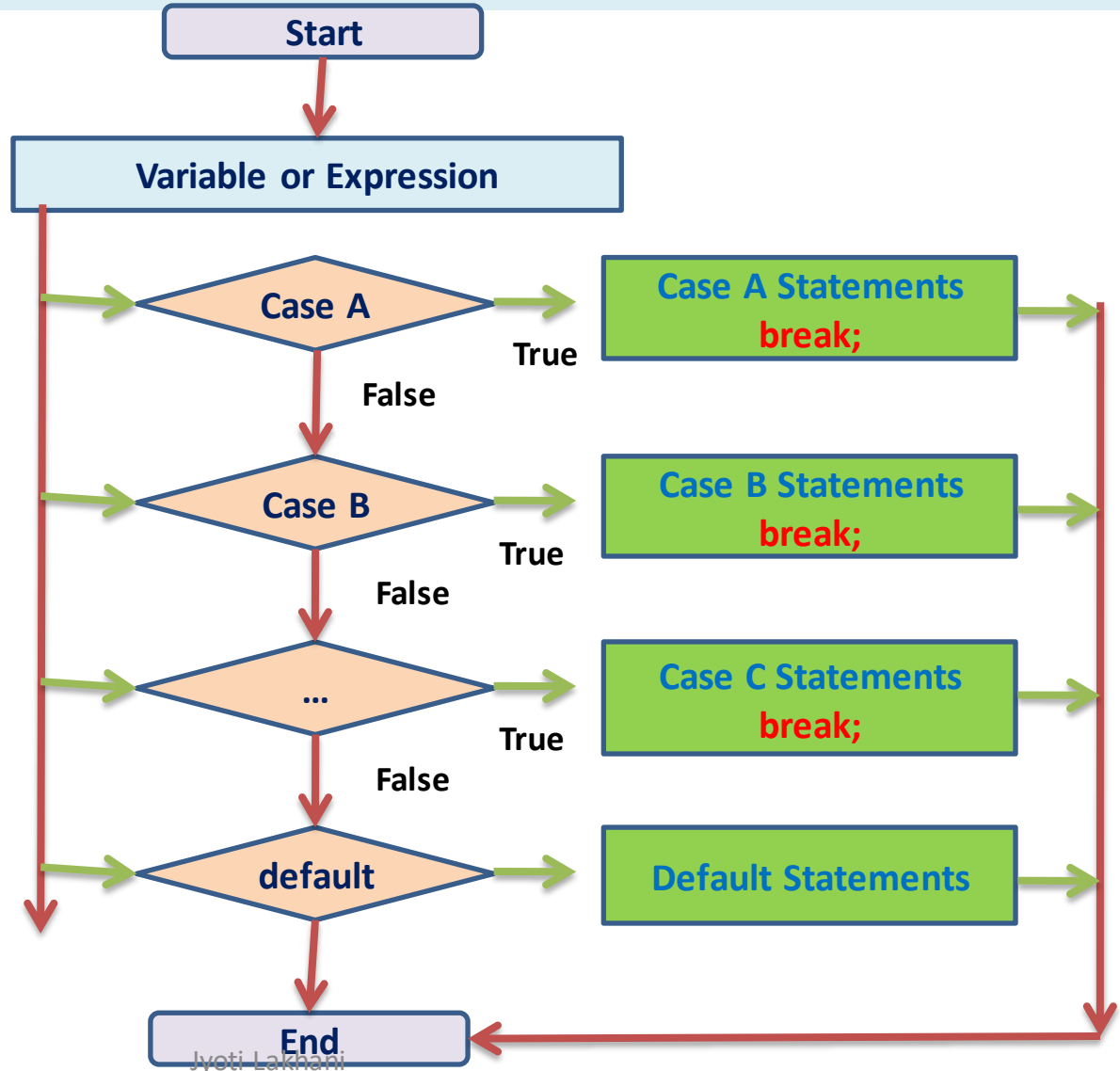


# Java – Control Statements

## switch Statement

Flow Chart:

Switch Statement with “break”





# Java – Control Statements

## switch Statement

Example:

```
Eg_Switch - Notepad
File Edit Format View Help
import java.util.Scanner;
class Eg_Switch
{
    public static void main(String[] args)
    {
        int days;
        Scanner in=new Scanner(System.in);
        System.out.println("Enter any day of Week: ");
        days=in.nextInt();
        switch(days)
        {
            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;
            default:
                System.out.println("Wrong Input");
        }
    }
}
Jyoti Lakhani
```





# Java – Control Statements

## *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
- When the variable being switched on is equal to a case, the statements following that case will execute until a *break statement* is reached





# Java – Control Statements

## *switch Statement*

- ***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***



# Java – Control Statements

**Iterations / Loops**

Iterations / Loops



# Java – Control Statements

**The iterative statements/ loop executes a block of statements when a particular condition is true**

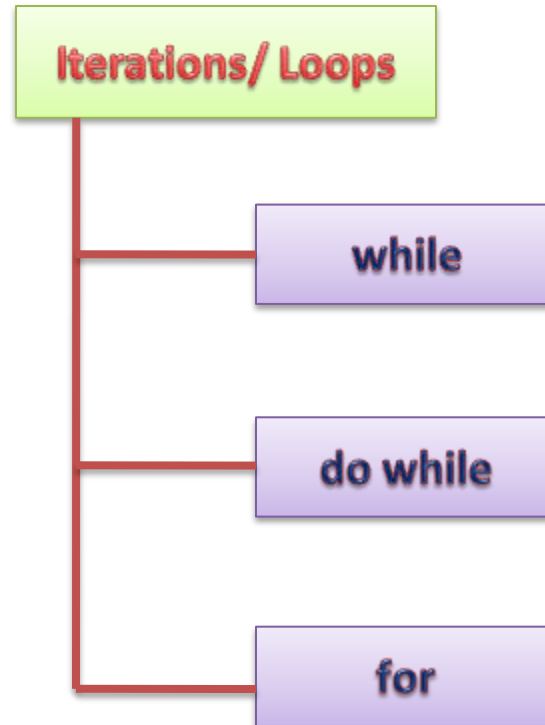
**Each loop has four types of statements :**

- Initialization**
- Condition checking**
- Execution**
- Increment / Decrement**



# Java – Control Statements

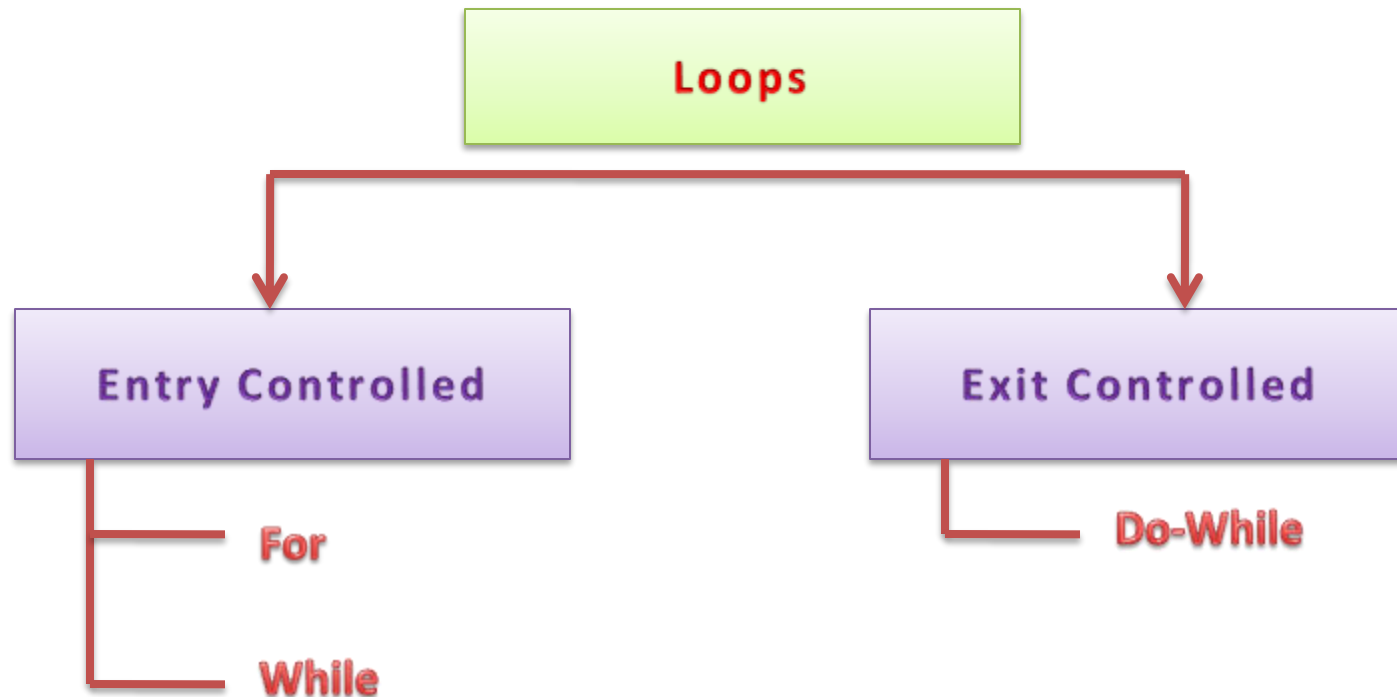
## Types of Loops





# Java – Control Statements

Loops can be categorized on the basis of place where condition checking is being done:





# Java – Control Statements

## Difference Between Entry Controlled and Exit Controlled Loops

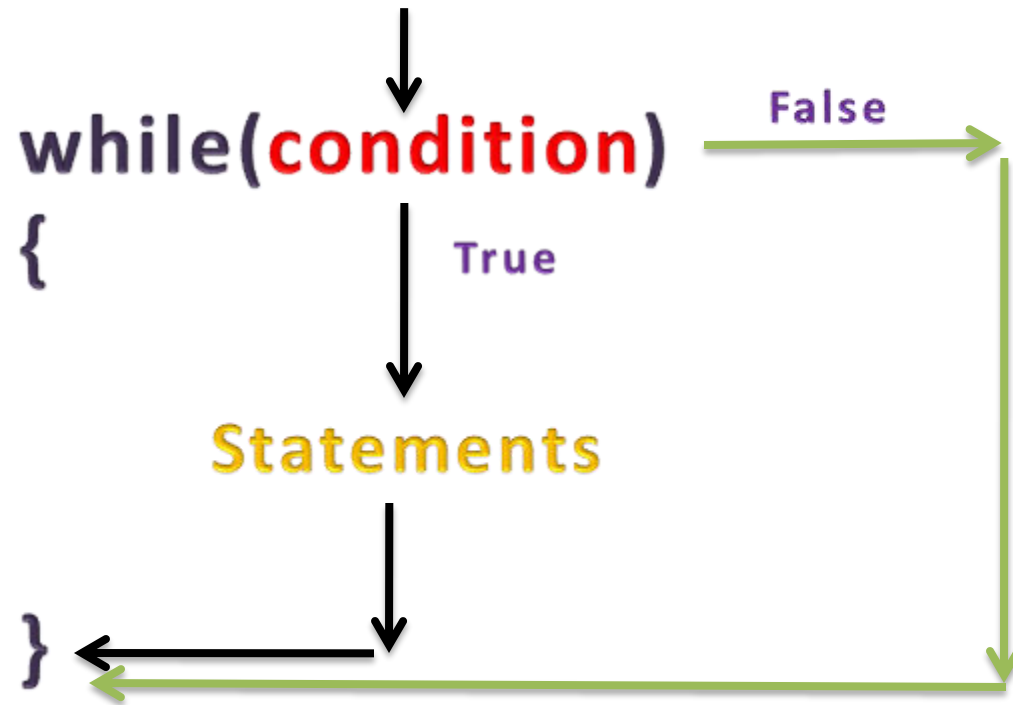
Entry Controlled	Exit Controlled
Condition is checked at the entry of the loop	Condition is checked at the exit of the loop
If condition is initially false, the loop never executes	If condition is initially false, then also the loop executes at least once
<pre><b>i=1;</b> <b>while(i==0)</b> <b>{</b>   <b>System.out.println("In While loop");</b> <b>}</b> <b>System.out.println("out of the loop");</b></pre>	<pre><b>i=1;</b> <b>do</b> <b>{</b>   <b>System.out.println("In While loop");</b> <b>} while(i==0);</b> <b>System.out.println("out of the loop");</b></pre>
Output: <b>Out of the loop</b>	Output: <b>In while loop</b> <b>Out of the loop</b>
Example- for, while	Example – do-while



# Java – Control Statements

## *while Loop*

Syntax





# Java – Control Statements

## *while Loop*

**Example: print values from 1 to ten**

```
Eg_while - Notepad
File Edit Format View Help
import java.util.Scanner;
class Eg_while
{
    public static void main(String[] args)
    {
        int i;
        Scanner in=new Scanner(System.in);
        i=1;
        while(i<=10)
        {
            System.out.println(i);
            i=i+1;
        }
    }
}
```





# Java – Control Statements

## *while Loop*

**Example: print values from 1 to ten**

```
Command Prompt
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Jyoti>D:
D:\>cd java programs
D:\Java Programs>javac Eg_while.java
D:\Java Programs>java Eg_while
1
2
3
4
5
6
7
8
9
10
D:\Java Programs>
```



# Java – Control Statements

## while Loop

### Functionality

Initialization Statement is used to initialize a variable/ counter.

```
i=1;
```

Initialization  
statement

```
while(i<=10)  
{
```

```
    System.out.println(i);  
    i=i+1;
```

```
}
```



# Java – Control Statements

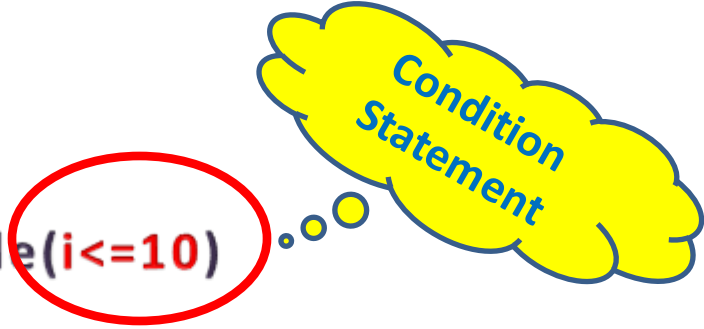
## while Loop

### Functionality

The condition statement controls the execution of loop

The loop executes till the condition statement is true

```
i=1;  
while(i<=10)  
{  
  
    System.out.println(i);  
    i=i+1;  
  
}
```





# Java – Control Statements

## while Loop

### Functionality

The execution statements are the main body of a loop

All action statements of loop are written here

```
i=1;  
while(i<=10)  
{  
    System.out.println(i);  
    i=i+1;  
}
```





# Java – Control Statements

## while Loop

### Functionality

This section is used to increment or decrement the variable value

```
i=1;  
  
while(i<=10)  
{  
  
    System.out.  
  
    i=i+1;  
  
}
```

Increment/  
Decrement

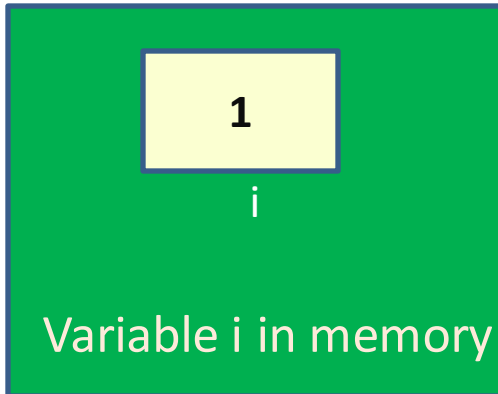
i=i+1;



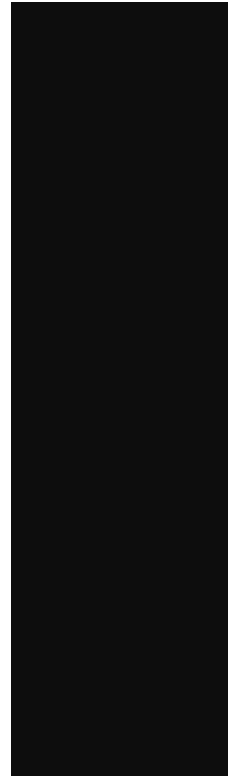
# Java – Control Statements

## *while Loop*

### Functionality



### Output



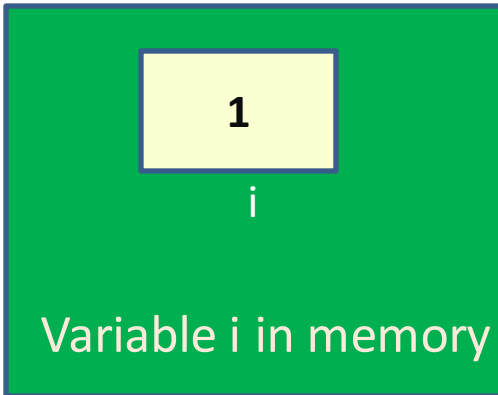
```
i=1;  
while(i<=10)  
{  
  
    System.out.println(i);  
  
    i=i+1;  
}
```



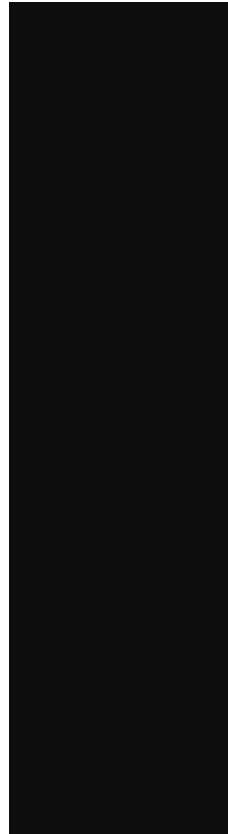
# Java – Control Statements

## while Loop

### Functionality



### Output



```
i=1;  
while(i<=10)  
{  
    System.out.println(i);  
    i=i+1;  
}
```

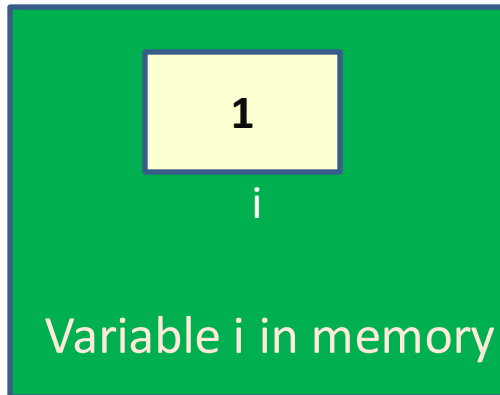




# Java – Control Statements

## while Loop

### Functionality



Output

1

Check it  
out here

```
i=1;
```

```
while(i<=10)
```

```
{
```

```
System.out.println(i);
```

```
i=i+1;
```

```
}
```

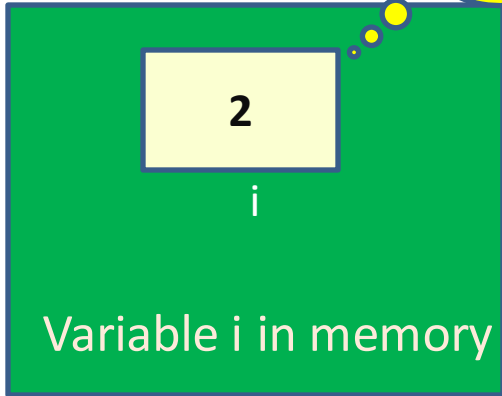




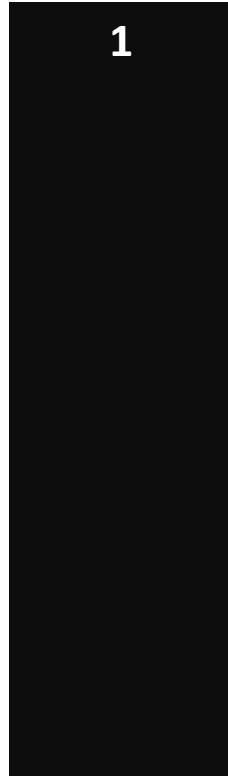
# Java – Control Statements

## while Loop

### Functionality



### Output



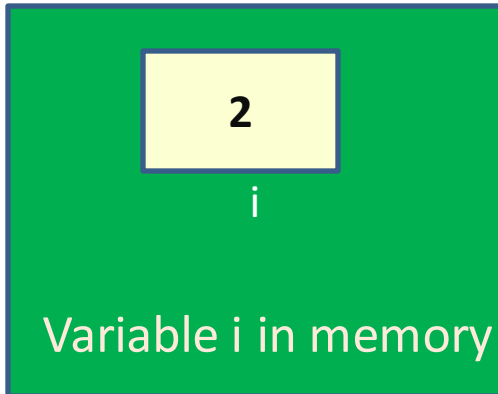
```
i=1;  
  
while(i<=10)  
{  
  
    System.out.println(i);  
  
    i=i+1;  
  
}
```



# Java – Control Statements

## while Loop

### Functionality



### Output

1

```
i=1;  
while(i<=10)  
{  
  
    System.out.println(i);  
  
    i=i+1;  
}
```

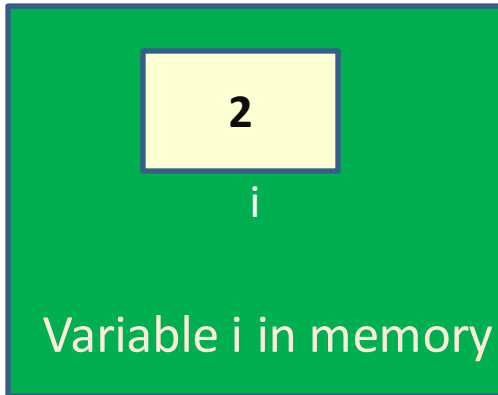
Condition is checked again



# Java – Control Statements

## while Loop

### Functionality



### Output

1

```
i=1;  
while(i<=10)  
{  
  
    System.out.println(i);  
  
    i=i+1;  
  
}
```

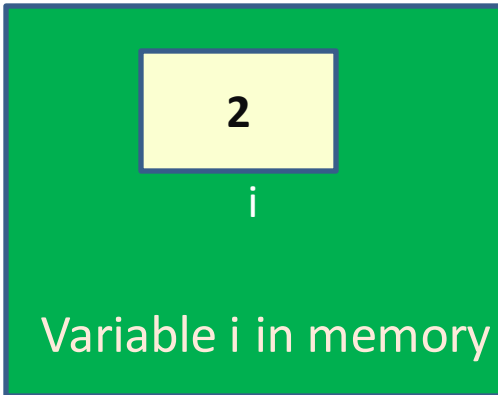
Condition  
is True



# Java – Control Statements

## while Loop

### Functionality



Output

1  
2



```
i=1;
```

```
while(i<=10)
```

```
{
```

```
    System.out.println(i);
```

```
    i=i+1;
```

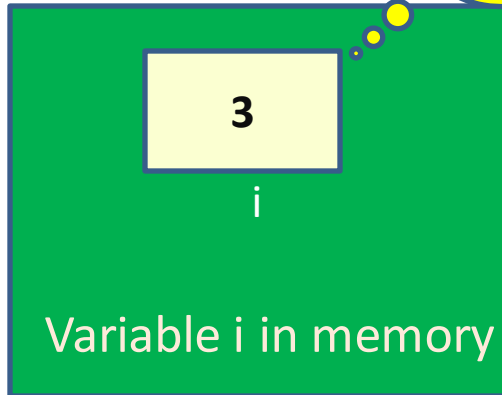
```
}
```



# Java – Control Statements

## while Loop

### Functionality



### Output

```
1  
2
```

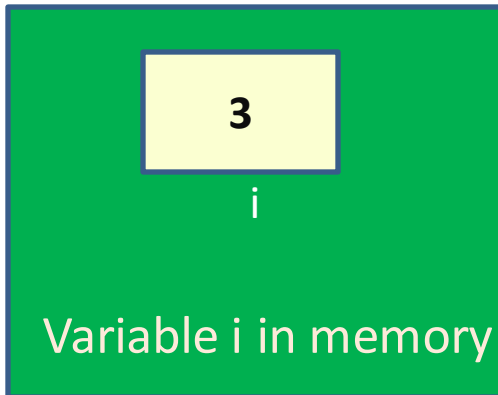
```
i=1;  
  
while(i<=10)  
{  
  
    System.out.println(i);  
  
    i=i+1;  
  
}
```



# Java – Control Statements

## while Loop

### Functionality



### Output

```
1
2
```

```
i=1;
while(i<=10)
{
    System.out.println(i);
    i=i+1;
}
```

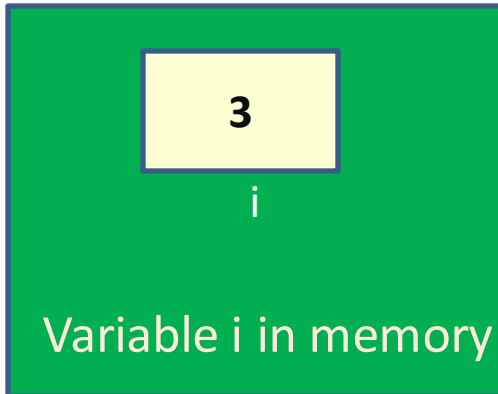
Condition is checked again



# Java – Control Statements

## while Loop

### Functionality



### Output

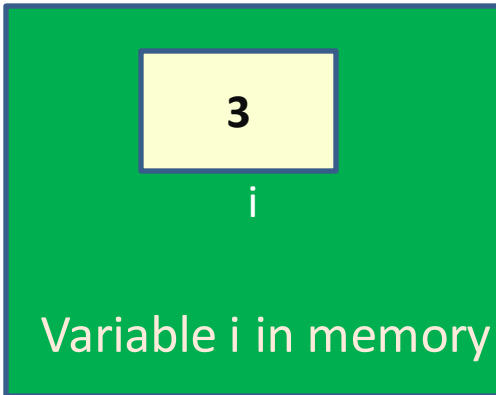
```
1
2
```

```
i=1;
while(i<=10)
{
    System.out.println(i);
    i=i+1;
}
```

Condition  
is True

## while Loop

### Functionality



Output

1  
2  
3

Check it  
out here

```
i=1;
```

```
while(i<=10)
```

```
{
```

```
    System.out.println(i);
```

```
    i=i+1;
```

```
}
```

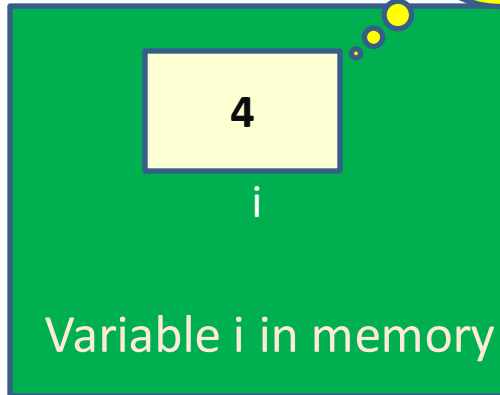




# Java – Control Statements

## while Loop

### Functionality



### Output

```
1  
2  
3
```

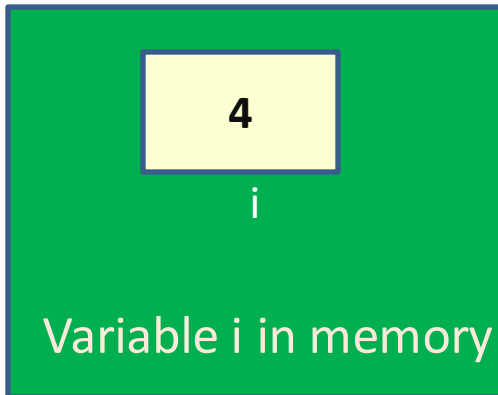
```
i=1;  
  
while(i<=10)  
{  
  
    System.out.println(i);  
  
    i=i+1;  
  
}
```



# Java – Control Statements

## while Loop

### Functionality



### Output

1  
2  
3

```
i=1;  
while(i<=10)  
{  
  
    System.out.println(i);  
  
    i=i+1;  
}
```

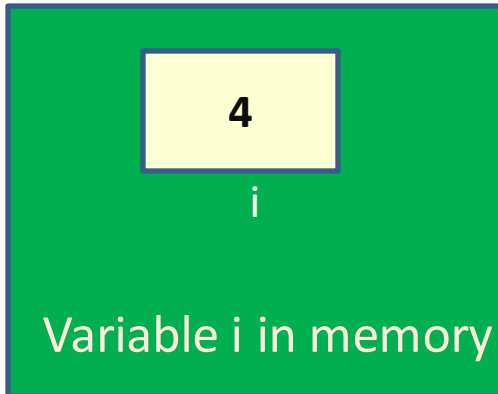
Condition is checked again



# Java – Control Statements

## while Loop

### Functionality



### Output

```
1
2
3
```

```
i=1;
```

```
while(i<=10)
{
```

```
    System.out.println(i);
```

```
    i=i+1;
```

```
}
```

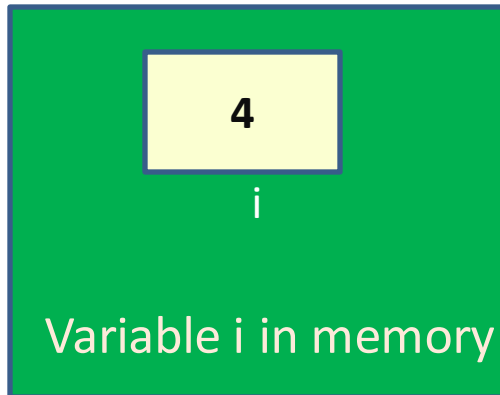
Condition  
is True



# Java – Control Statements

## while Loop

### Functionality



Output

1  
2  
3  
4

Check it  
out here

```
i=1;
```

```
while(i<=10)
```

```
{
```

```
    System.out.println(i);
```

```
    i=i+1;
```

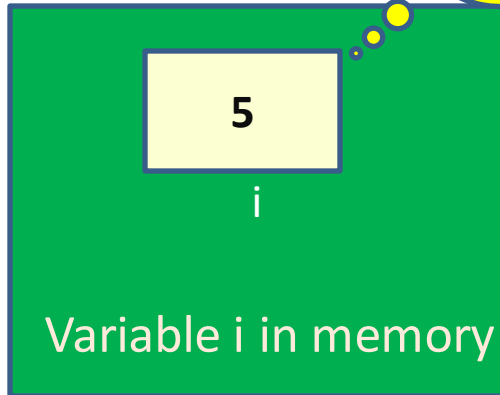
```
}
```



# Java – Control Statements

## while Loop

### Functionality



### Output

```
1  
2  
3  
4
```

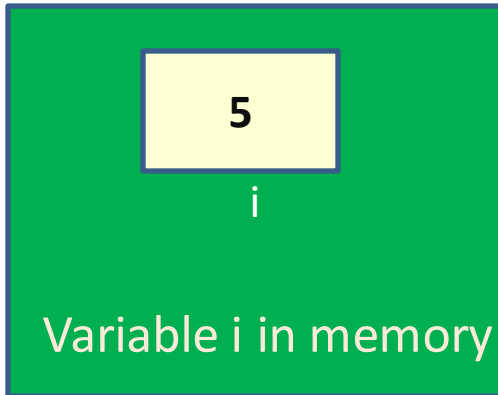
```
i=1;  
  
while(i<=10)  
{  
  
    System.out.println(i);  
  
    i=i+1;  
  
}
```



# Java – Control Statements

## while Loop

### Functionality



### Output

1  
2  
3  
4

```
i=1;  
while(i<=10)  
{  
  
    System.out.println(i);  
  
    i=i+1;  
  
}
```

Condition is checked again



# Java – Control Statements

## *while Loop*

This process will continue till the condition become false

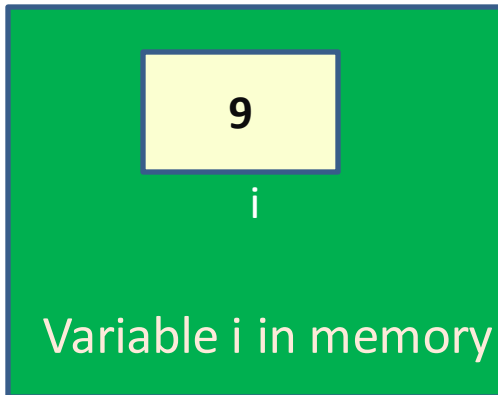
Suppose value of i is 9 now



# Java – Control Statements

## while Loop

### Functionality



### Output

```
1
2
3
4
5
6
7
8
```

```
i=1;
```

```
while(i<=10)
{
```

```
    System.out.println(i);
```

```
    i=i+1;
```

```
}
```

Condition  
is True

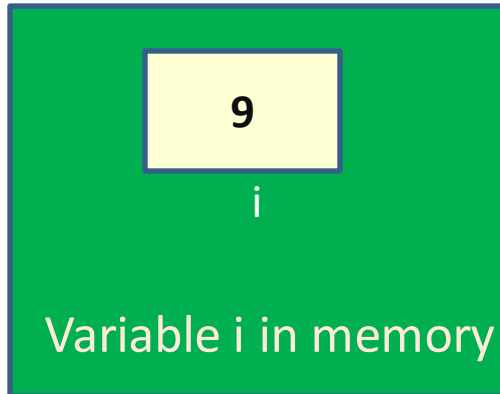




# Java – Control Statements

## while Loop

### Functionality



Output

1  
2  
3  
4  
5  
6  
7  
8  
9

Check it  
out here

```
i=1;
```

```
while(i<=10)
```

```
{
```

```
System.out.println(i);
```

```
i=i+1;
```

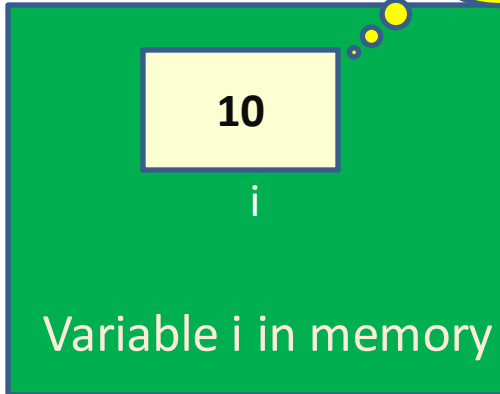
```
}
```



# Java – Control Statements

## while Loop

### Functionality



### Output

1  
2  
3  
4  
5  
6  
7  
8  
9

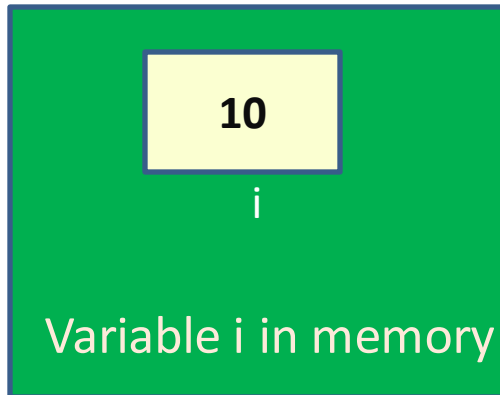
```
i=1;  
while(i<=10)  
{  
  
    System.out.println(i);  
  
    i=i+1;  
}
```



# Java – Control Statements

## while Loop

### Functionality



### Output

```
1
2
3
4
5
6
7
8
9
```

```
i=1;
while(i<=10)
{
    System.out.println(i);
    i=i+1;
}
```

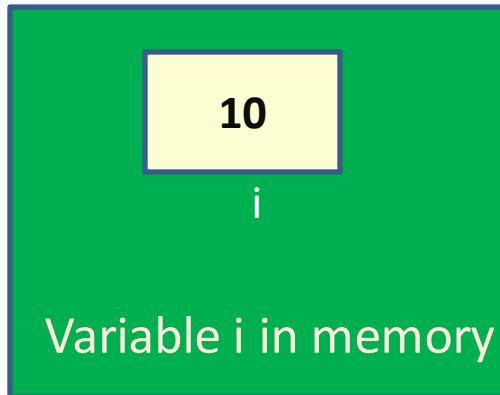
Condition is checked again



# Java – Control Statements

## while Loop

### Functionality



### Output

```
1
2
3
4
5
6
7
8
9
```

```
i=1;
```

```
while(i<=10)
{
```

```
    System.out.println(i);
```

```
    i=i+1;
```

```
}
```

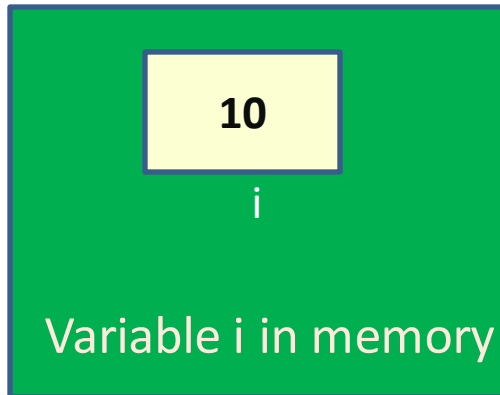
Condition  
is Still True



# Java – Control Statements

## while Loop

### Functionality



Output

1  
2  
3  
4  
5  
6  
7  
8  
9  
10

Check it  
out here

```
i=1;
```

```
while(i<=10)
```

```
{
```

```
System.out.println(i);
```

```
i=i+1;
```

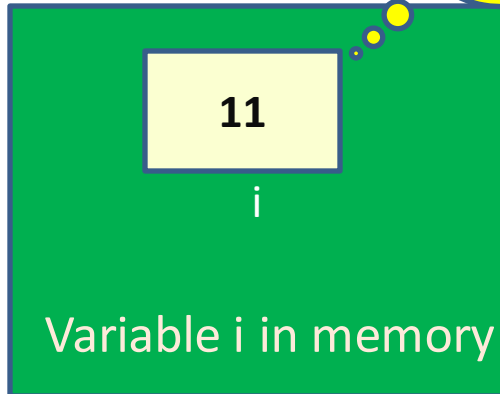
```
}
```



# Java – Control Statements

## while Loop

### Functionality



### Output

```
1
2
3
4
5
6
7
8
9
10
```

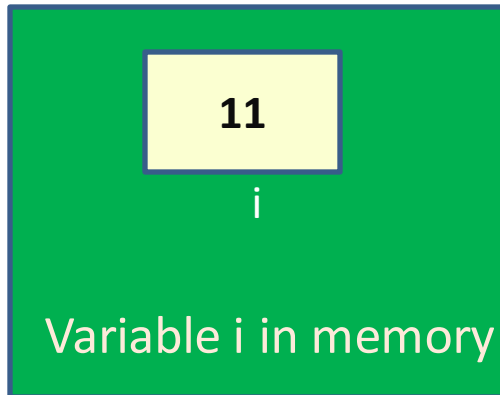
```
i=1;
while(i<=10)
{
    System.out.println(i);
    i=i+1;
}
```



# Java – Control Statements

## while Loop

### Functionality



### Output

```
1
2
3
4
5
6
7
8
9
10
```

```
i=1;
while(i<=10)
{
    System.out.println(i);
    i=i+1;
}
```

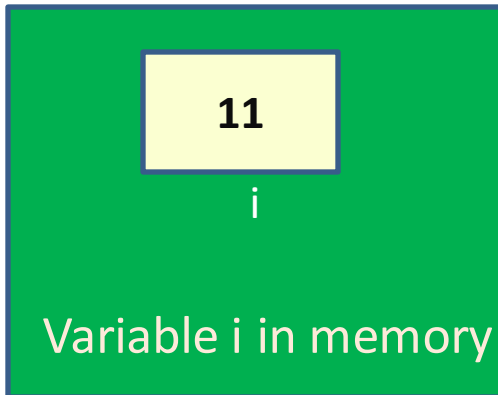
Condition is checked again



# Java – Control Statements

## while Loop

### Functionality



### Output

```
1
2
3
4
5
6
7
8
9
10
```

```
i=1;
while(i<=10)
{
    System.out.println(i);
    i=i+1;
}
```

Condition  
is FALSE

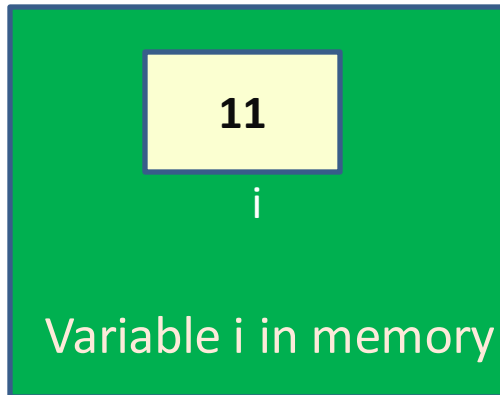




# Java – Control Statements

## while Loop

### Functionality



### Output

1  
2  
3  
4  
5  
6  
7  
8  
9  
10

```
i=1;  
while(i<=10)  
{  
  
    System.out.println(i);  
  
    i=i+1;  
}
```

Skip the body of the loop and executes the statement just after the loop

## while Loop

Functional

Final value of *i*  
after the  
complete loop

Output

11

*i*

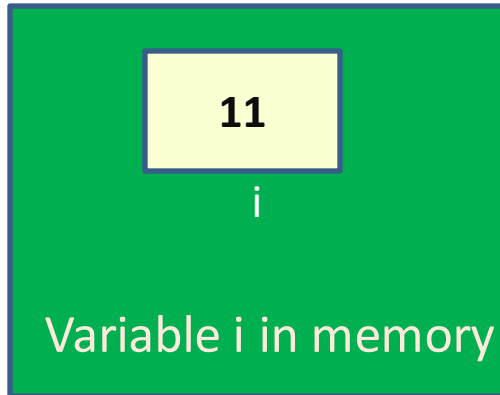
Variable *i* in memory

1  
2  
3  
4  
5  
6  
7  
8  
9  
10

```
i=1;  
  
while(i<=10)  
{  
  
    System.out.println(i);  
  
    i=i+1;  
  
}
```

## while Loop

### Functionality



Output

1  
2  
3  
4  
5  
6  
7  
8  
9  
10

Final output of the loop

```
while(i<=10)
```

```
{
```

```
System.out.println(i);
```

```
i=i+1;
```

```
}
```



# Java – Control Statements

## for Loop

Syntax

**for( initialization ; condition ; increment )**

{

True

**Statements**

}

False



# Java – Control Statements

## *for Loop*

**Example: print values from 1 to ten**

```
File Edit Format View Help
import java.util.Scanner;
class Eg_while
{
    public static void main(String[] args)
    {
        int i;
        Scanner in=new Scanner(System.in);
        for(i=1;i<=10;i=i+1)
        {
            System.out.println(i);
        }
    }
}
```



# Java – Control Statements

## *for Loop*

**Example: print values from 1 to ten**

```
ca. Command Prompt
D:\Java Programs>javac Eg_for.java
D:\Java Programs>java Eg_for
1
2
3
4
5
6
7
8
9
10
D:\Java Programs>
```

The screenshot shows a Windows Command Prompt window titled "ca. Command Prompt". The window has standard Windows window controls (minimize, maximize, close) in the top right corner. The command prompt shows the following sequence of commands and output:  
1. The user enters the command `D:\Java Programs>javac Eg_for.java`.  
2. The user enters the command `D:\Java Programs>java Eg_for`.  
3. The program outputs the numbers 1 through 10, one per line.  
4. The prompt returns to `D:\Java Programs>`.



# Java – Control Statements

## for Loop

### Functionality

Initialization Statement is used to initialize a variable/ counter.



```
for(i=1; i<=10; i=i+1)
```

```
{
```

```
    System.out.println(i);
```

```
}
```



# Java – Control Statements

## For Loop

### Functionality

The condition statement controls the execution of loop

The loop executes till the condition statement is true



```
for(i=1; i<=10; i=i+1)
```

```
{
```

```
    System.out.println(i);
```

```
}
```





# Java – Control Statements

## for Loop

### Functionality

This section is used to increment or decrement the variable value



```
for(i=1; i<=10; i=i+1)
{
    System.out.println(i);
}
```



# Java – Control Statements

## for Loop

### Functionality

The execution statements are the main body of a loop  
All action statements of loop are written here

```
for(i=1; i<=10; i=i+1)
{
    System.out.println(i);
}
```

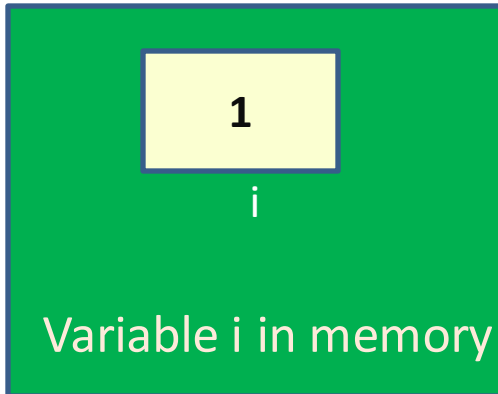




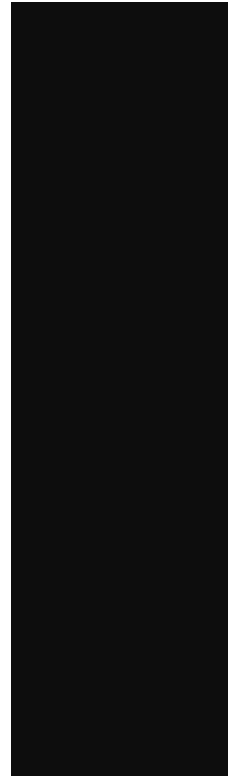
# Java – Control Statements

## for Loop

### Functionality



### Output



```
for(i=1; i<=10; i=i+1)
{
    System.out.println(i);
}
```

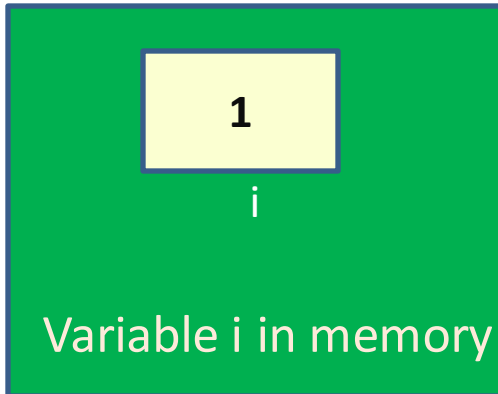
A yellow thought bubble with the word 'Initialization' in red text points to the 'i=1' part of the for loop condition, which is circled in red.



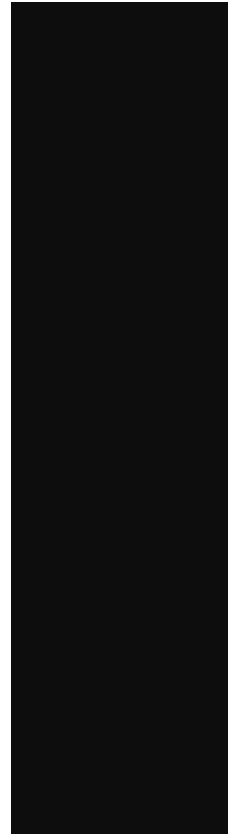
# Java – Control Statements

## for Loop

### Functionality



### Output



```
for(i=1; i<=10; i=i+1)
{
    System.out.println(i);
}
```





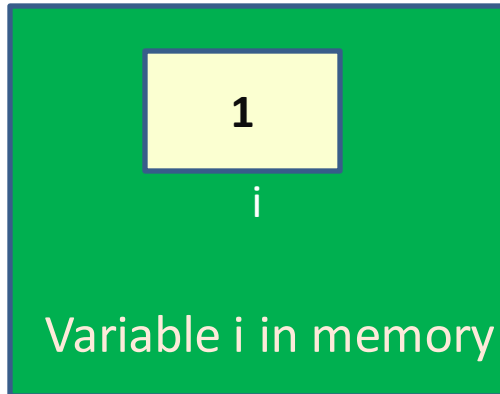
# Java – Control Statements

## for Loop

### Functionality



Output



1

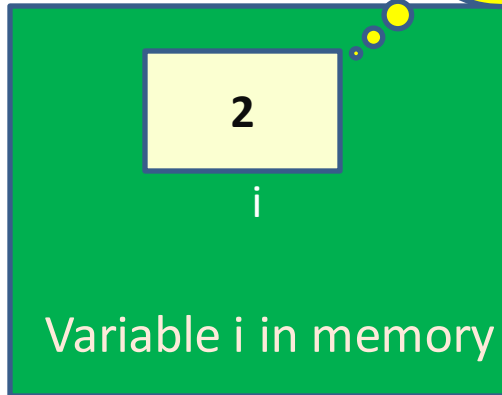
```
for(i=1; i<=10; i=i+1)
{
    System.out.println(i);
}
```



# Java – Control Statements

## for Loop

### Functionality



Output

1



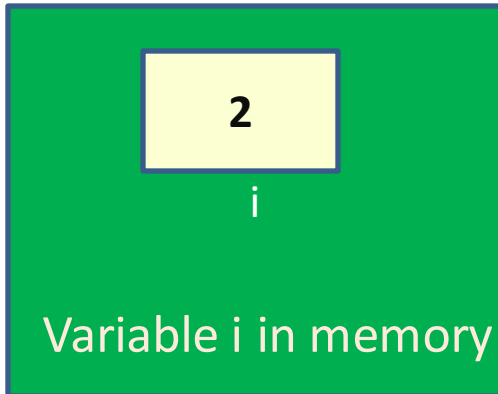
```
for(i=1; i<=10; i=i+1)
{
    System.out.println(i);
}
```



# Java – Control Statements

## for Loop

### Functionality



### Output

1

```
for(i=1; i<=10; i=i+1)
{
    System.out.println(i);
}
```

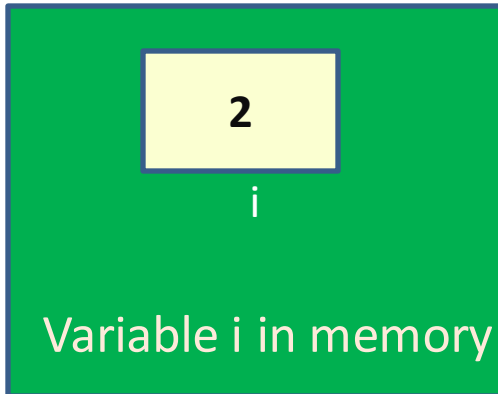
Condition is checked again



# Java – Control Statements

## for Loop

### Functionality



### Output

1

```
for(i=1; i<=10; i=i+1)
{
    System.out.println(i);
}
```

Condition  
is True

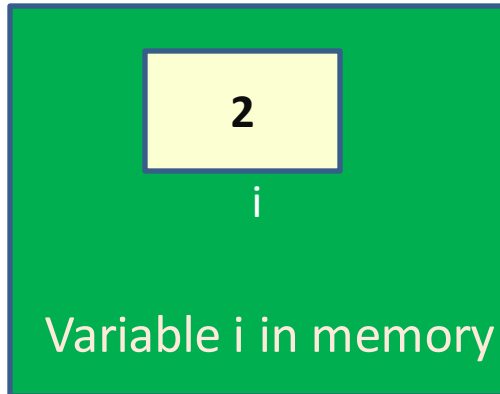




# Java – Control Statements

## for Loop

### Functionality



Output

1  
2



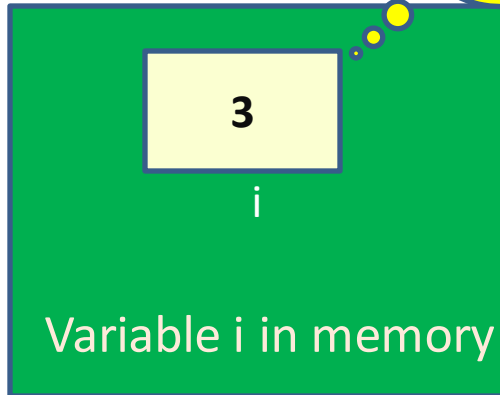
```
for(i=1; i<=10; i=i+1)
{
    System.out.println(i);
}
```



# Java – Control Statements

## for Loop

### Functionality



### Output

1  
2

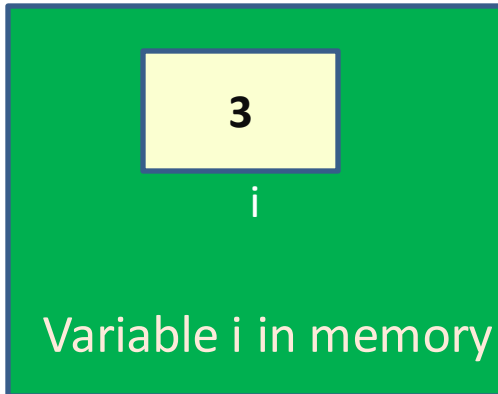
```
for(i=1; i<=10; i=i+1)  
{  
    System.out.println(i);  
}
```



# Java – Control Statements

## for Loop

### Functionality



### Output

1  
2

```
for(i=1; i<=10; i=i+1)
{
    System.out.println(i);
}
```

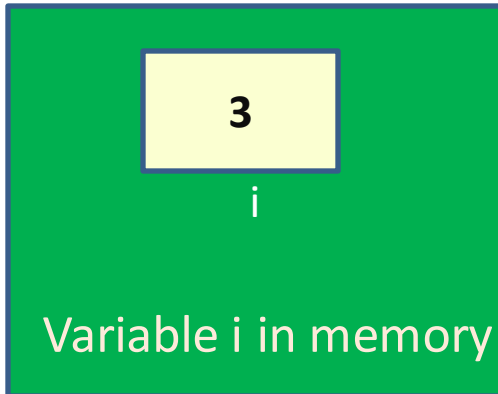
Condition is checked again



# Java – Control Statements

## for Loop

### Functionality



### Output

1  
2

```
for(i=1; i<=10; i=i+1)
{
    System.out.println(i);
}
```

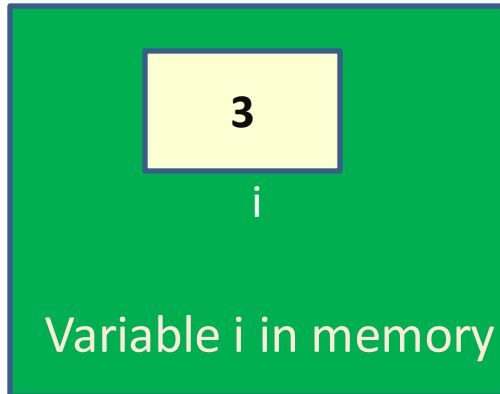
Condition  
is True



# Java – Control Statements

## for Loop

### Functionality



Output

1  
2  
3



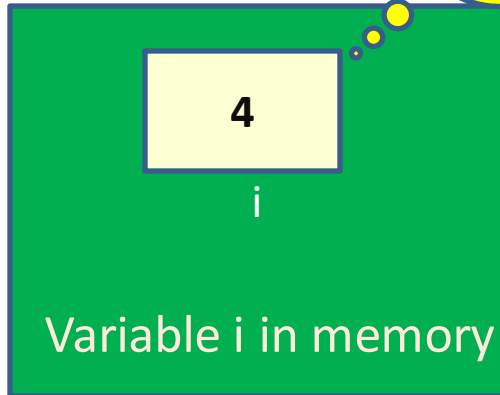
```
for(i=1; i<=10; i=i+1)
{
    System.out.println(i);
}
```



# Java – Control Statements

## for Loop

### Functionality



### Output

```
1  
2  
3
```

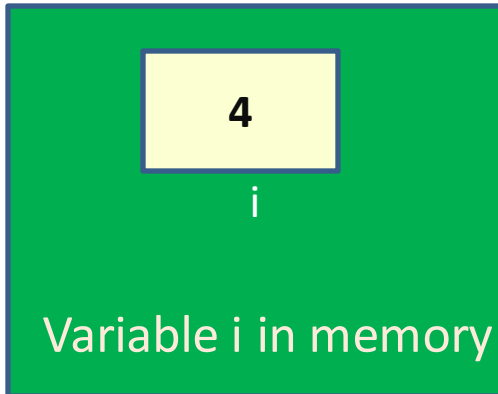
```
for(i=1; i<=10; i=i+1)  
{  
    System.out.println(i);  
}
```



# Java – Control Statements

## for Loop

### Functionality



### Output

1  
2  
3

```
for(i=1; i<=10; i=i+1)
{
    System.out.println(i);
}
```

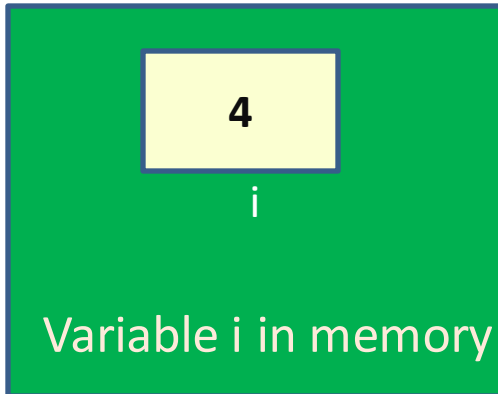
Condition is checked again



# Java – Control Statements

## for Loop

### Functionality



### Output

1  
2  
3

```
for(i=1; i<=10; i=i+1)
{
    System.out.println(i);
}
```

A yellow thought bubble with a blue outline points to the condition `i<=10` in the code. The text inside the bubble reads "Condition is True".

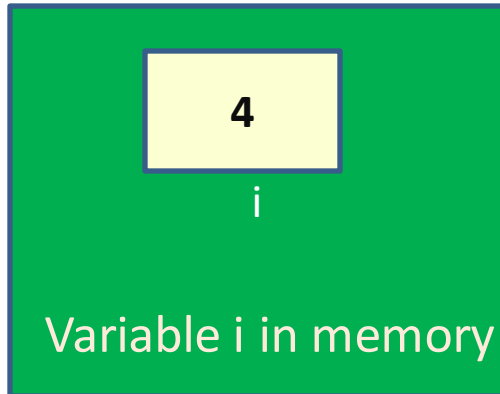




# Java – Control Statements

## for Loop

### Functionality



Output

1  
2  
3  
4



```
for(i=1; i<=10; i=i+1)
{
    System.out.println(i);
}
```



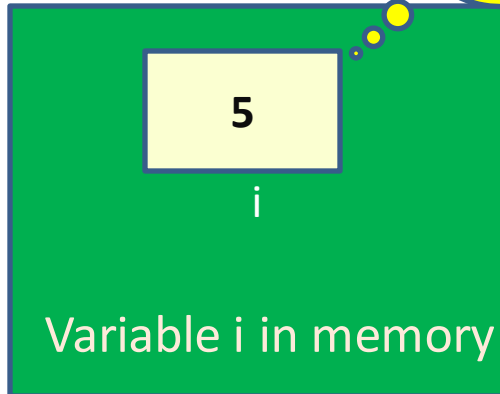
# Java – Control Statements

## for Loop

### Functionality

Check it  
out here

### Output



1  
2  
3  
4

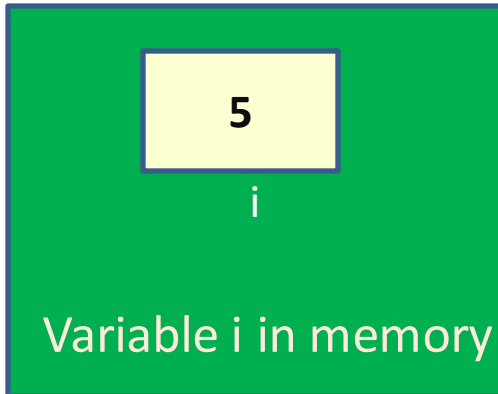
```
for(i=1; i<=10; i=i+1)  
{  
    System.out.println(i);  
}
```



# Java – Control Statements

## for Loop

### Functionality



### Output

1  
2  
3  
4

```
for(i=1; i<=10; i=i+1)  
{  
    System.out.println(i);  
}
```

Condition is checked again



# Java – Control Statements

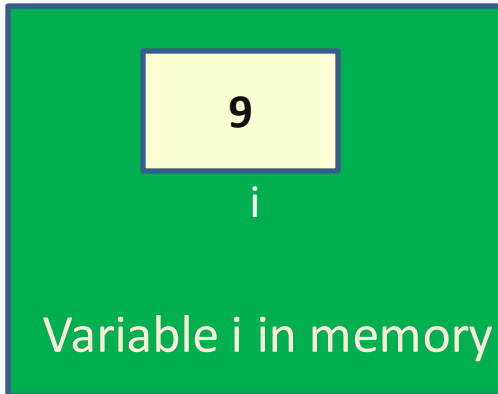
## *for Loop*

This process will continue till the condition become false

Suppose value of i is 9 now

## for Loop

### Functionality



### Output

1  
2  
3  
4  
5  
6  
7  
8

```
for(i=1; i<=10; i=i+1)
{
    System.out.println(i);
}
```

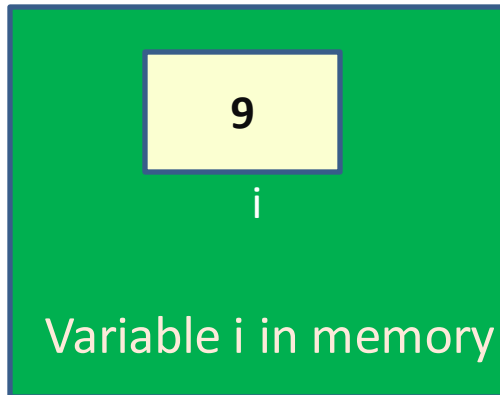
Condition  
is True



# Java – Control Statements

## for Loop

### Functionality



Output

1  
2  
3  
4  
5  
6  
7  
8  
9



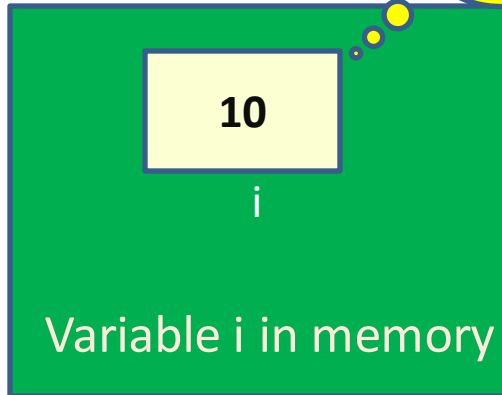
```
for(i=1; i<=10; i=i+1)
{
    System.out.println(i);
}
```



# Java – Control Statements

## for Loop

### Functionality



### Output

1  
2  
3  
4  
5  
6  
7  
8  
9

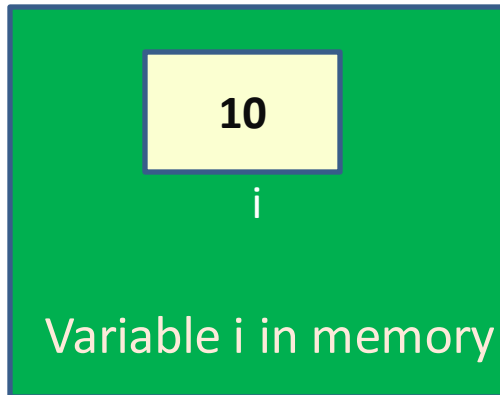
```
for(i=1; i<=10; i=i+1)  
{  
    System.out.println(i);  
}
```



# Java – Control Statements

## for Loop

### Functionality



### Output

1  
2  
3  
4  
5  
6  
7  
8  
9

```
for(i=1; i<=10; i=i+1)
{
    System.out.println(i);
}
```

Condition is checked again

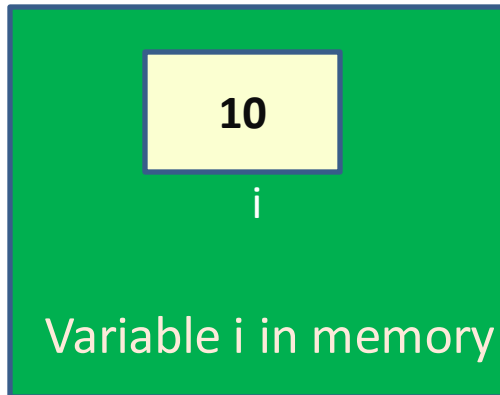




# Java – Control Statements

## for Loop

### Functionality

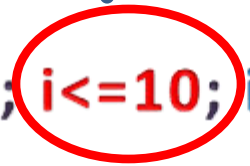


### Output

```
1  
2  
3  
4  
5  
6  
7  
8  
9
```

```
for(i=1; i<=10; i=i+1)  
{  
    System.out.println(i);  
}
```

Condition  
is Still True

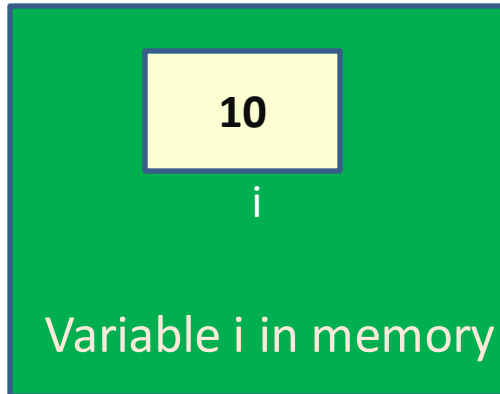




# Java – Control Statements

## for Loop

### Functionality



Output

1  
2  
3  
4  
5  
6  
7  
8  
9  
10



```
for(i=1; i<=10; i=i+1)
{
    System.out.println(i);
}
```



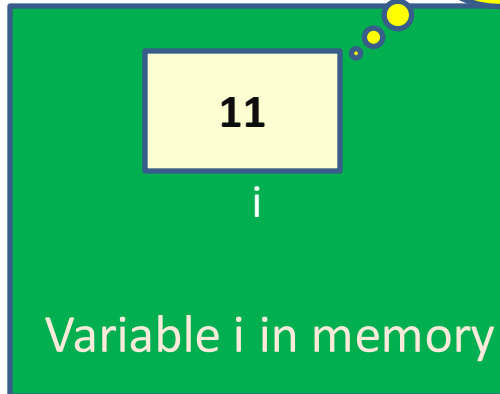
# Java – Control Statements

## for Loop

### Functionality

Check it  
out here

### Output



1  
2  
3  
4  
5  
6  
7  
8  
9  
10

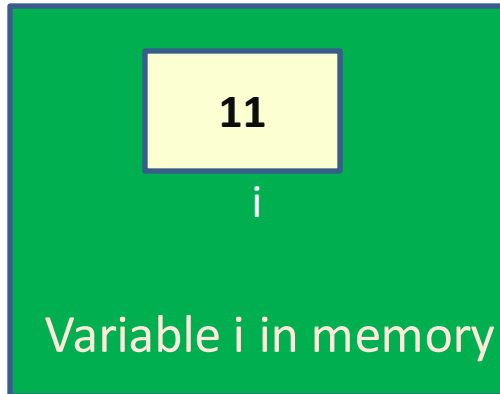
```
for(i=1; i<=10; i=i+1)  
{  
    System.out.println(i);  
}
```



# Java – Control Statements

## for Loop

### Functionality



### Output

1  
2  
3  
4  
5  
6  
7  
8  
9  
10

```
for(i=1; i<=10; i=i+1)
{
    System.out.println(i);
}
```

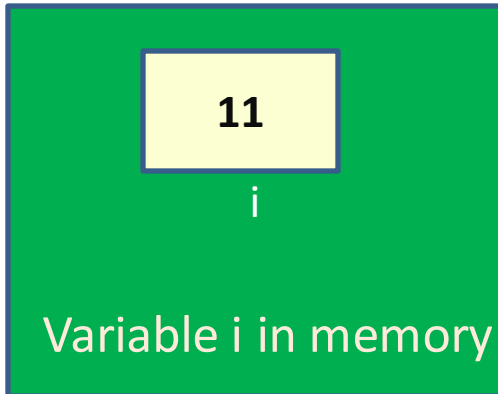
Condition is checked again



# Java – Control Statements

## for Loop

### Functionality



### Output

1  
2  
3  
4  
5  
6  
7  
8  
9  
10

```
for(i=1; i<=10; i=i+1)
{
    System.out.println(i);
}
```

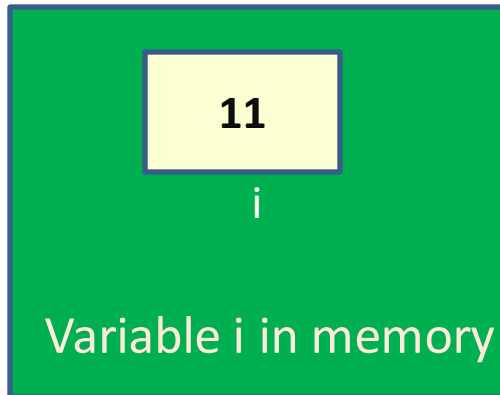
Condition  
is FALSE



# Java – Control Statements

## for Loop

### Functionality



### Output

1  
2  
3  
4  
5  
6  
7  
8  
9  
10

```
for(i=1; i<=10; i=i+1)  
{  
    System.out.println(i);  
}
```

Skip the body of the loop and executes the statement just after the loop

## for Loop

Functional

Final value of i  
after the  
complete loop

11

i

Variable i in memory

Output  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10

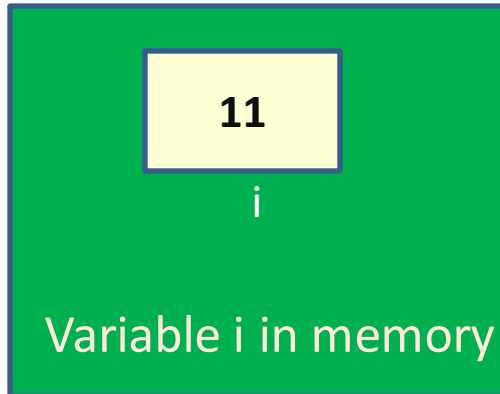
```
for(i=1; i<=10; i=i+1)
{
    System.out.println(i);
}
```



# Java – Control Statements

## for Loop

### Functionality



Output

1  
2  
3  
4  
5  
6  
7  
8  
9  
10

Final output of the loop

```
for (int i = 1; i <= 10; i = i + 1)
```

```
System.out.println(i);
```

```
}
```

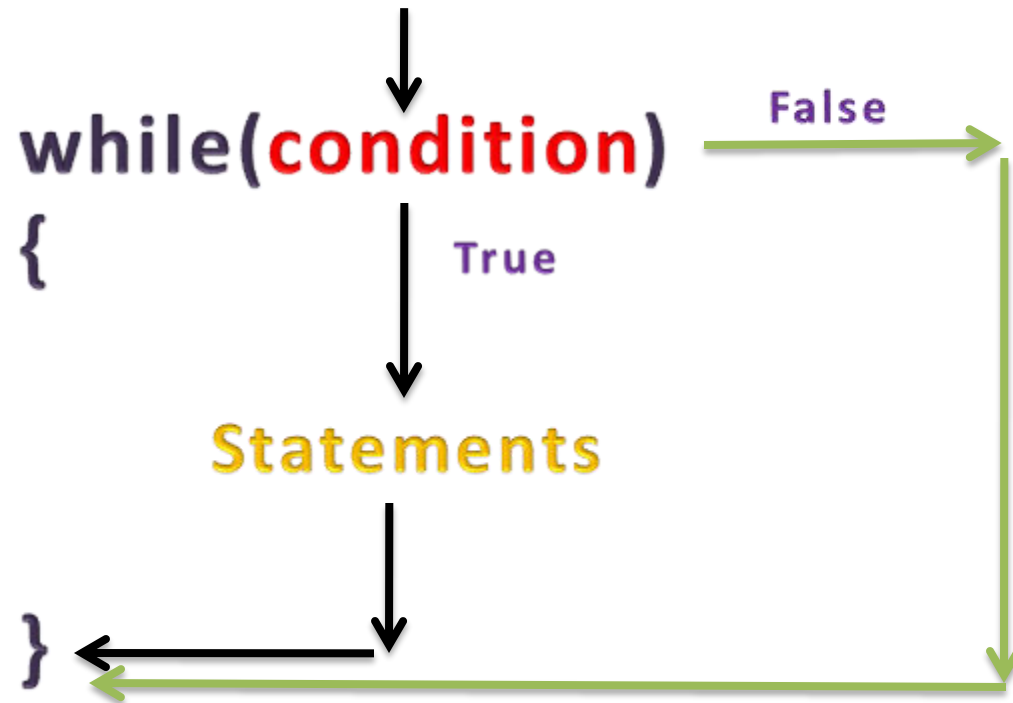




# Java – Control Statements

## do-while Loop

Syntax





# Java – Control Statements

## *do-while Loop*

**Example: print values from 1 to ten**


```
Eg_dowhile - Notepad
File Edit Format View Help
import java.util.Scanner;
class Eg_dowhile
{
    public static void main(String[] args)
    {
        int i;
        Scanner in=new Scanner(System.in);
        i=1;
        do
        {
            System.out.println(i);
            i=i+1;
        }while(i<=10);
    }
}
```



# Java – Control Statements

## *do-while Loop*

**Example: print values from 1 to ten**



```
CA. Command Prompt
D:\Java Programs>javac Eg_dowhile.java
D:\Java Programs>java Eg_dowhile
1
2
3
4
5
6
7
8
9
10
D:\Java Programs>
```

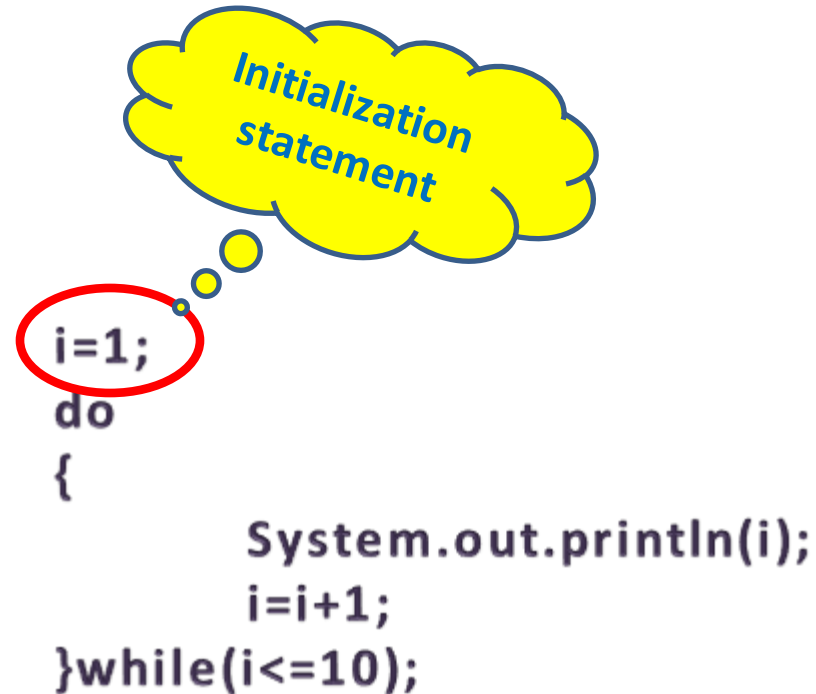


# Java – Control Statements

## do-while Loop

### Functionality

Initialization Statement is used to initialize a variable/ counter.





# Java – Control Statements

## do-while Loop

### Functionality

The condition statement controls the execution of loop

The loop executes till the condition statement is true

```
i=1;  
do  
{  
    System.out.println(i);  
    i=i+1;  
}while(i<=10);
```

Condition Statement



# Java – Control Statements


## do-while Loop

### Functionality

The execution statements are the main body of a loop

All action statements of loop are written here

```
i=1;  
do  
{  
    System.out.println(i);  
    i=i+1;  
}while(i<=10);
```






# Java – Control Statements

## do-while Loop

### Functionality

This section is used to increment or decrement the variable value

```
i=1;  
do  
{  
    System.out.println(i);  
    i=i+1;  
}while(i<=10);
```



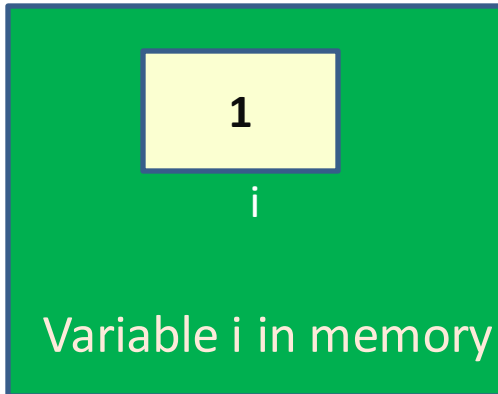
Increment/  
Decrement



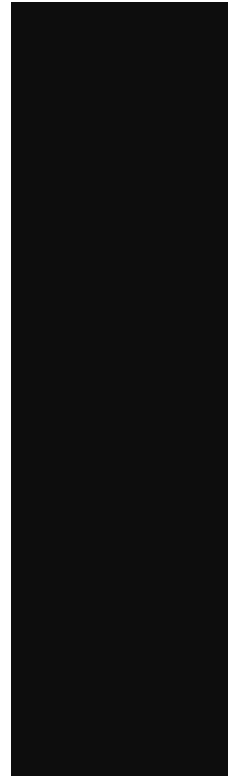
# Java – Control Statements

## do-while Loop

### Functionality



### Output



```
i=1;  
do  
{  
    System.out.println(i);  
    i=i+1;  
}while(i<=10);
```

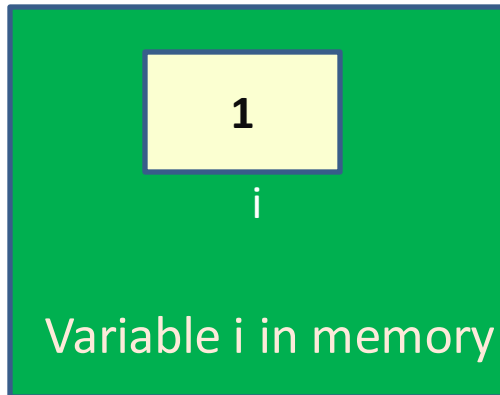




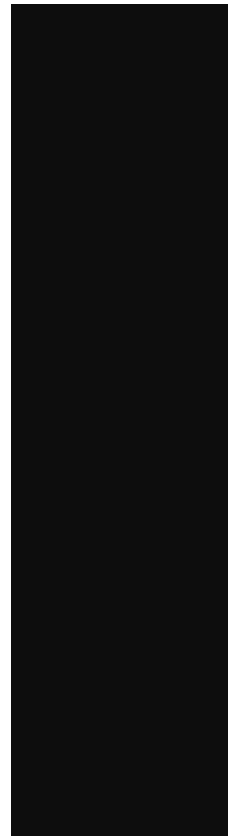
# Java – Control Statements

## do-while Loop

### Functionality



### Output



```
i=1;  
do  
{  
    System.out.println(i);  
    i=i+1;  
}while(i<=10);
```

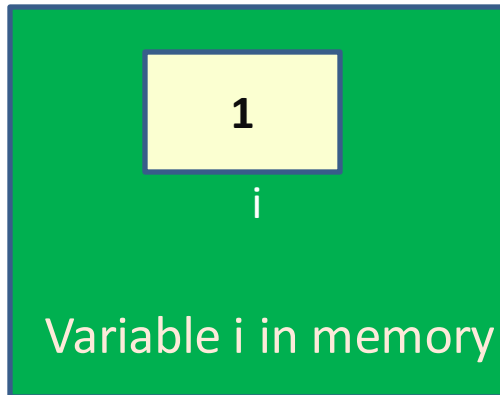
No condition checking at entry of the loop



# Java – Control Statements

## do-while Loop

### Functionality



Output

1



```
i=1;
```

```
do
```

```
{
```

```
    System.out.println(i);
```

```
    i=i+1;
```

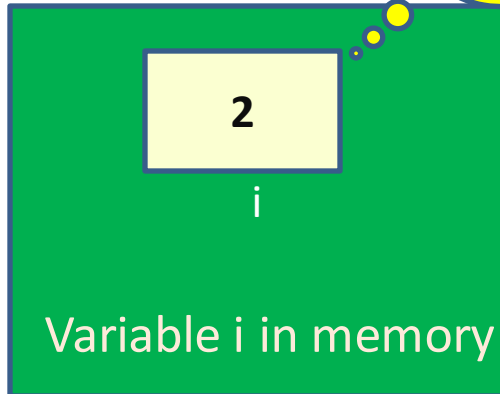
```
}while(i<=10);
```



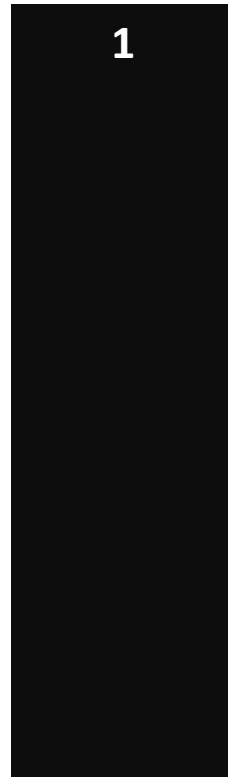
# Java – Control Statements

## do-while Loop

### Functionality



### Output



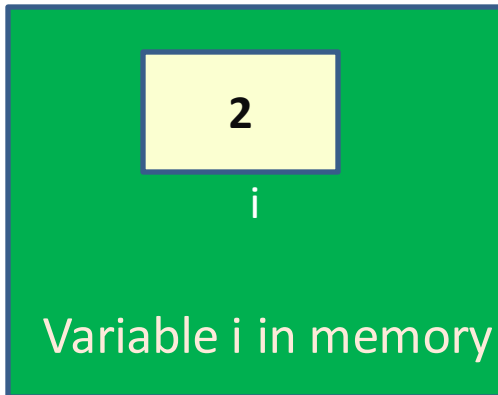
```
i=1;  
do  
{  
    System.out.println(i);  
    i=i+1;  
}while(i<=10);
```



# Java – Control Statements

## do-while Loop

### Functionality



### Output

1

```
i=1;  
do  
{  
    System.out.println(i);  
    i=i+1;  
}while(i<=10);
```

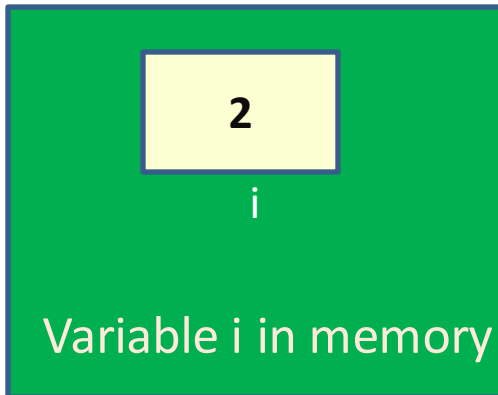
Condition is checked again



# Java – Control Statements

## do-while Loop

### Functionality



### Output

1

```
i=1;  
do  
{  
    system.out.println(i);  
    i=i+1;  
}while(i<=10);
```

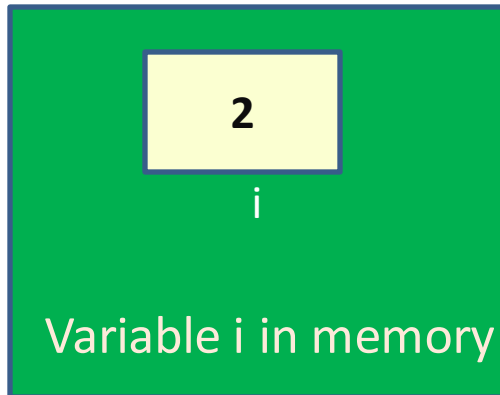
A yellow thought bubble with a blue outline contains the text "Condition is True" in red. A red oval highlights the closing brace and the while condition of the code block.



# Java – Control Statements

## do-while Loop

### Functionality



Output

1  
2



```
i=1;  
do  
{
```

**System.out.println(i);**

```
i=i+1;
```

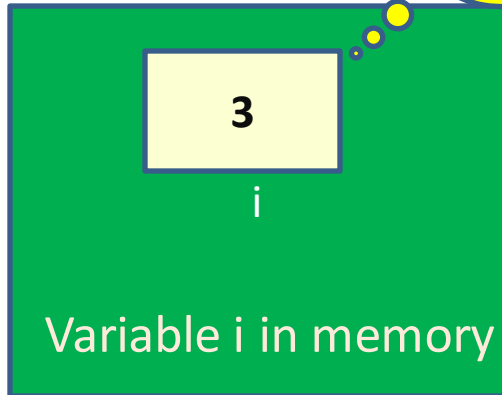
```
}while(i<=10);
```



# Java – Control Statements

## do-while Loop

### Functionality



### Output

```
1  
2
```

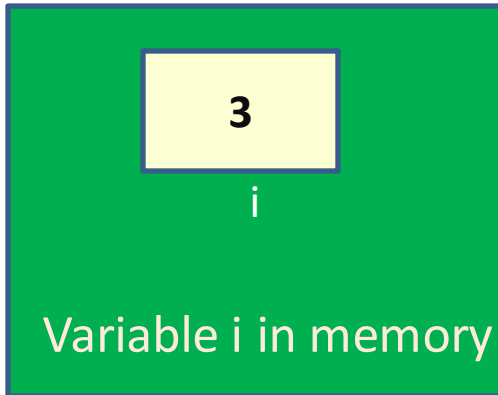
```
i=1;  
do  
{  
    System.out.println(i);  
    i=i+1;  
}while(i<=10);
```



# Java – Control Statements

## do-while Loop

### Functionality



### Output

```
1
2
```

```
i=1;
do
{
    System.out.println(i);
    i=i+1;
}while(i<=10);
```

Condition is checked again

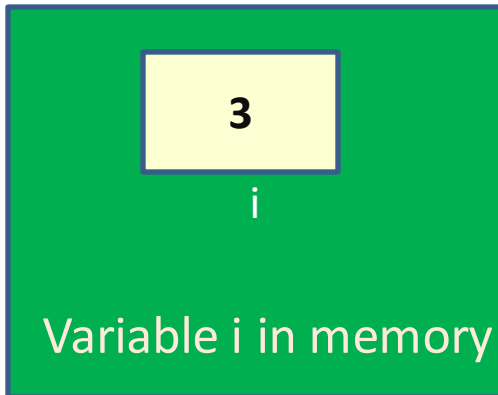




# Java – Control Statements

## do-while Loop

### Functionality



### Output

```
1
2
```

```
i=1;
do
{
    System.out.println(i);
    i=i+1;
}while(i<=10);
```

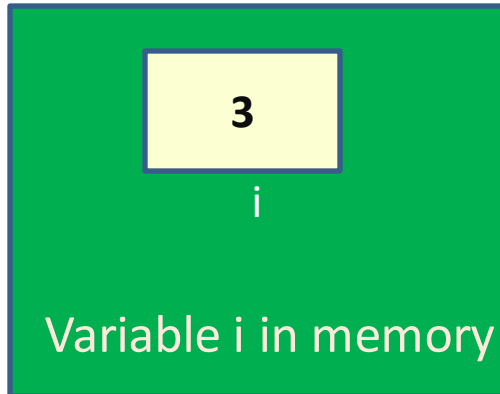
A yellow thought bubble with a blue border contains the text "Condition is True" in red. A red oval highlights the closing brace and while condition of the code block.



# Java – Control Statements

## do-while Loop

### Functionality



Output

1  
2  
3

Check it  
out here

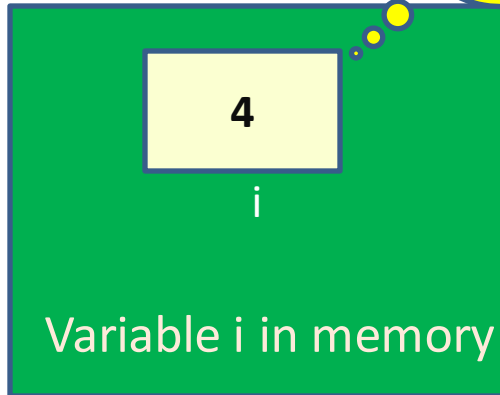
```
i=1;  
do  
{  
    System.out.println(i);  
    i=i+1;  
}while(i<=10);
```



# Java – Control Statements

## do-while Loop

### Functionality



### Output

```
1  
2  
3
```

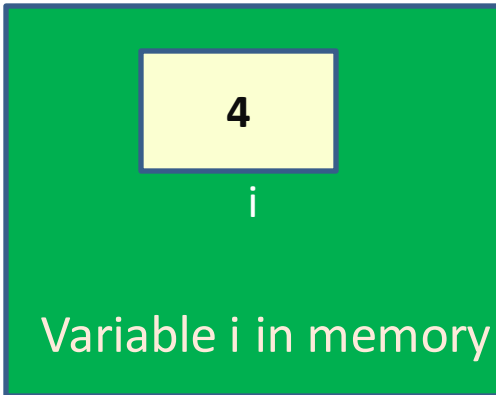
```
i=1;  
do  
{  
    System.out.println(i);  
    i=i+1;  
}while(i<=10);
```



# Java – Control Statements

## do-while Loop

### Functionality



### Output

```
1  
2  
3
```

```
i=1;  
do  
{  
    System.out.println(i);  
    i=i+1;  
}while(i<=10);
```

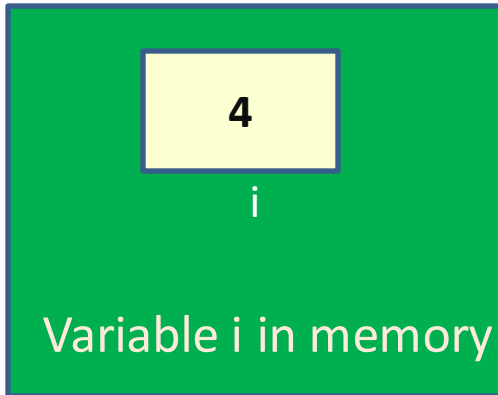
Condition is checked again



# Java – Control Statements

## do-while Loop

### Functionality



### Output

```
1  
2  
3
```

```
i=1;  
do  
{
```

```
System.out.println(i);  
i=i+1;  
}while(i<=10);
```

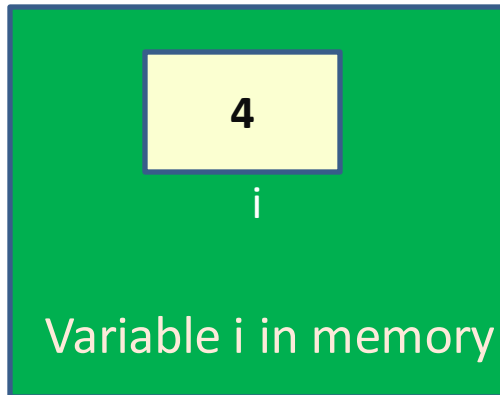




# Java – Control Statements

## do-while Loop

### Functionality



Output

1  
2  
3  
4



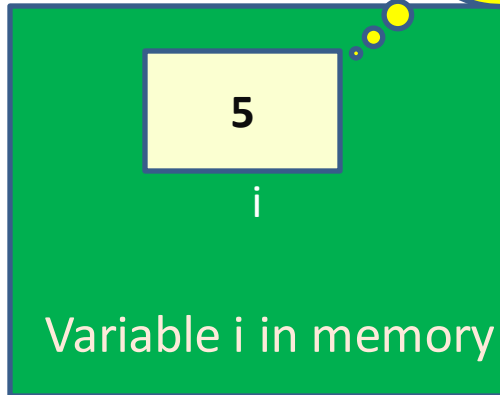
```
i=1;  
do  
{  
    System.out.println(i);  
    i=i+1;  
}while(i<=10);
```



# Java – Control Statements

## do-while Loop

### Functionality



### Output

```
1  
2  
3  
4
```

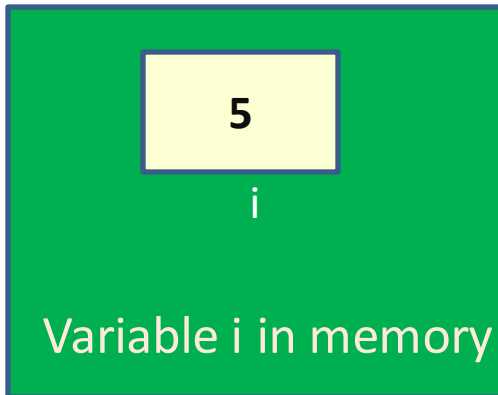
```
i=1;  
do  
{  
    System.out.println(i);  
    i=i+1;  
}while(i<=10);
```



# Java – Control Statements

## do-while Loop

### Functionality



### Output

```
1  
2  
3  
4
```

```
i=1;  
do  
{  
    System.out.println(i);  
    i=i+1;  
}while(i<=10);
```

Condition is checked again





# Java – Control Statements

## *do-while Loop*

This process will continue till the condition become false

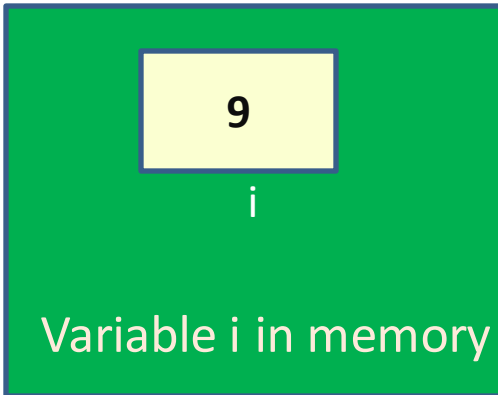
Suppose value of i is 9 now



# Java – Control Statements

## do-while Loop

### Functionality



### Output

```
1  
2  
3  
4  
5  
6  
7  
8
```

```
i=1;  
do  
{  
    System.out.println(i);  
    i=i+1;  
}while(i<=10);
```

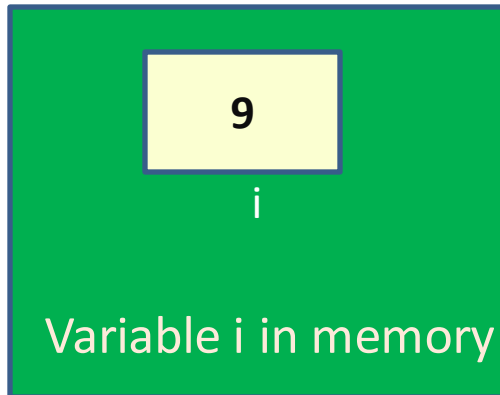
A yellow thought bubble with a blue outline contains the text "Condition is True" in red. A red oval highlights the `while(i<=10);` line in the code block.



# Java – Control Statements

## do-while Loop

### Functionality



Output

1  
2  
3  
4  
5  
6  
7  
8  
9

Check it  
out here

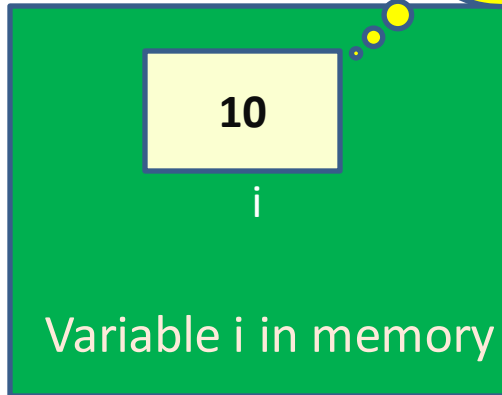
```
i=1;  
do  
{  
    System.out.println(i);  
    i=i+1;  
}while(i<=10);
```



# Java – Control Statements

## do-while Loop

### Functionality



### Output

```
1  
2  
3  
4  
5  
6  
7  
8  
9
```

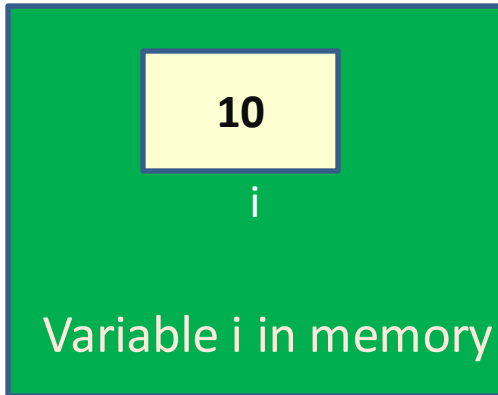
```
i=1;  
do  
{  
    System.out.println(i);  
    i=i+1;  
}while(i<=10);
```



# Java – Control Statements

## do-while Loop

### Functionality



### Output

```
1
2
3
4
5
6
7
8
9
```

```
i=1;
do
{
    System.out.println(i);
    i=i+1;
}while(i<=10);
```

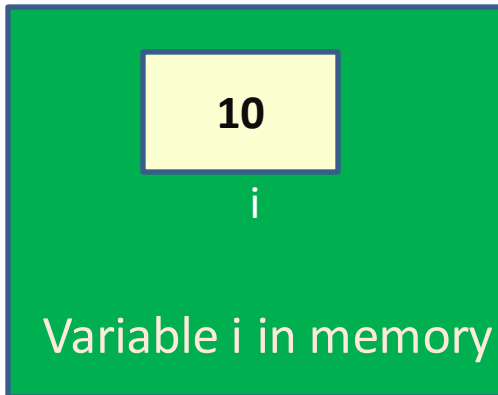




# Java – Control Statements

## do-while Loop

### Functionality



### Output

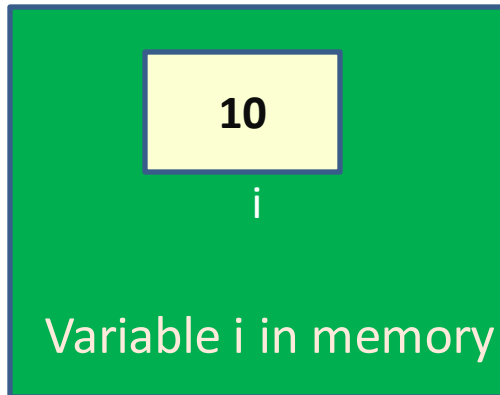
```
1
2
3
4
5
6
7
8
9
```

```
i=1;
do
{
    System.out.println(i);
    i=i+1;
}while(i<=10);
```

Condition is Still True

## do-while Loop

### Functionality



Output

1  
2  
3  
4  
5  
6  
7  
8  
9  
10

Check it  
out here

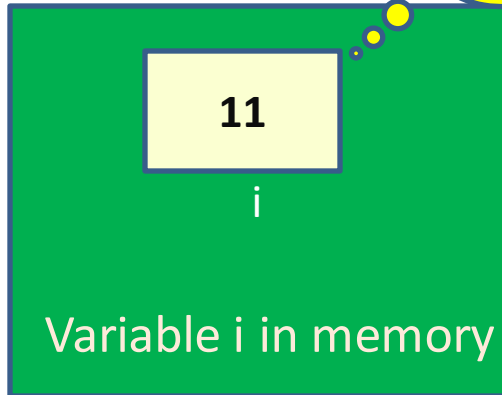
```
i=1;  
do  
{  
    System.out.println(i);  
    i=i+1;  
}while(i<=10);
```



# Java – Control Statements

## do-while Loop

### Functionality



### Output

1  
2  
3  
4  
5  
6  
7  
8  
9  
10

```
i=1;  
do  
{  
    System.out.println(i);  
    i=i+1;  
}while(i<=10);
```

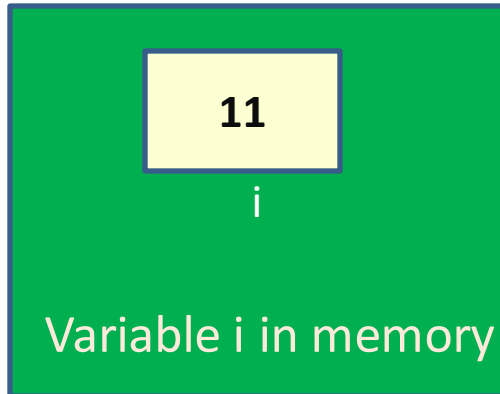




# Java – Control Statements

## do-while Loop

### Functionality



### Output

```
1
2
3
4
5
6
7
8
9
10
```

```
i=1;
do
{
    System.out.println(i);
    i=i+1;
}while(i<=10);
```

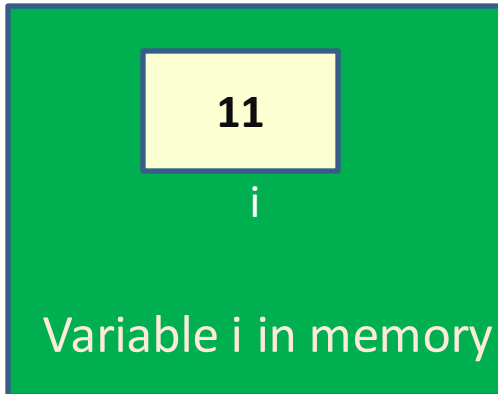
Condition is checked again



# Java – Control Statements

## do-while Loop

### Functionality



### Output

```
1
2
3
4
5
6
7
8
9
10
```

```
i=1;
do
{
    System.out.println(i);
    i=i+1;
}while(i<=10);
```

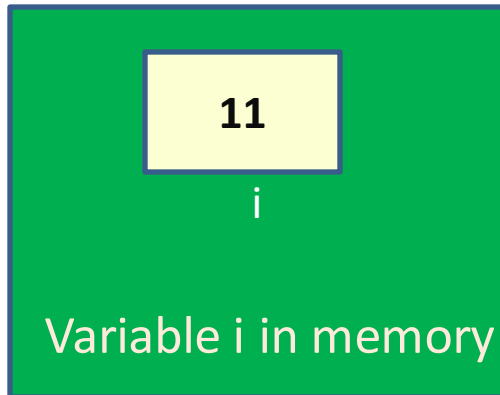
A yellow thought bubble with a blue outline contains the text "Condition is FALSE" in red. A red oval highlights the closing brace and the while condition of the code block.



# Java – Control Statements

## do-while Loop

### Functionality



### Output

1  
2  
3  
4  
5  
6  
7

```
i=1;  
do  
{  
    System.out.println(i);  
    i=i+1;  
}while(i<=10);
```

Stop the execution of the loop and executes the statement just after the loop



# Java – Control Statements

## do-while Loop

Functional

Final value of *i*  
after the  
complete loop

Output

11

*i*

Variable *i* in memory

1  
2  
3  
4  
5  
6  
7  
8  
9  
10

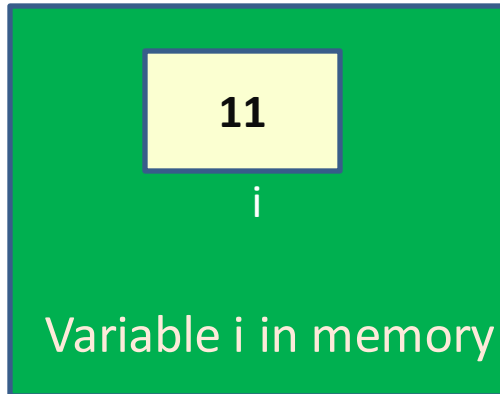
```
i=1;  
do  
{  
    System.out.println(i);  
    i=i+1;  
}while(i<=10);
```



# Java – Control Statements

## do-while Loop

### Functionality



Output

1  
2  
3  
4  
5  
6  
7  
8  
9  
10

Final output of  
the loop

```
System.out.println(i);  
i=i+1;  
}while(i<=10);
```