Java If --- else

Use of If Condition.

- Use If to specify a block of code to be executed, if a specified condition is true.
- Use Else to specified a block of code to be executed, if the same condition is false.
- Use else-if to specify a new condition to test, if the first condition is false.
- Use switch to specify many alternative blocks of codes to be executed.

(1) The if statement :

Use the if statement to specify a block of Java code to be executed if a condition is true.

Syntax :

```
if (Condition) {
    // Comment
    Statement or code to be execute
}
```

eg

```
if (20 > 18){
    SOUT ("20 is greater than 18");
}
```

" System.Out.print" ↳ SOUT

② else statement :

The use of else statement when if statement condition is false. then else statement check the false condition if false condition is true. then else statement given false output ~~or print~~.

```
for   else {
              //(statement or
               code to be executed);
      }
```

```
eg   ~~int age = 20;~~
     ~~if ( age < 20 )~~
     if (20
```



```
eg   int age = 20;
     if (age < 22) {
                 (statement);
     } else {
             // (statement);
     }.
```

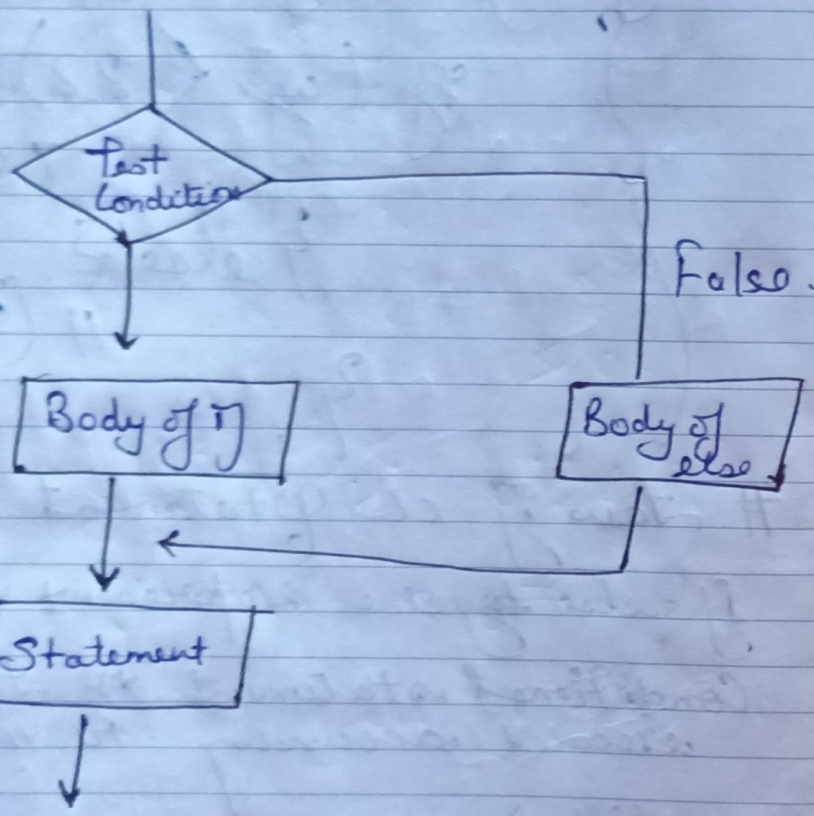
Java if - else statement

if -else together represent the set of conditional statement in Java that are executed according to the condition which is true.

Syntax

```
if (condition) {  // Executes this block if
                  // condition is true
} else {
   // Execute this block
   // condition is false
}
```

Flow Chart



① Program starts.
② 'i' is initialized to 20.
③ If - condition is checked. 20<15, yields false.
④ Flow enters the else block.

4 (a) "i is greater than 15" is printed.

(5) "Outside of if-else block" is printed.

```
Import java.util.Scanner
Class IfElseDemo{
    public static void main (String
                [] args)
    {
    Scanner sc = new Scanner (
                System.in)
    Sout ("Enter the age : ");
    Int age = sc.nextInt();
    if (age>18){
        Sout ("Eligible to drive");
    }else{
    Sout ("Not Eligible to drive");
    }
    sc.close();
}
```

Note

SOUT
{System.out.println}

#DoWhatFloatsYourBoat

Relational operatose

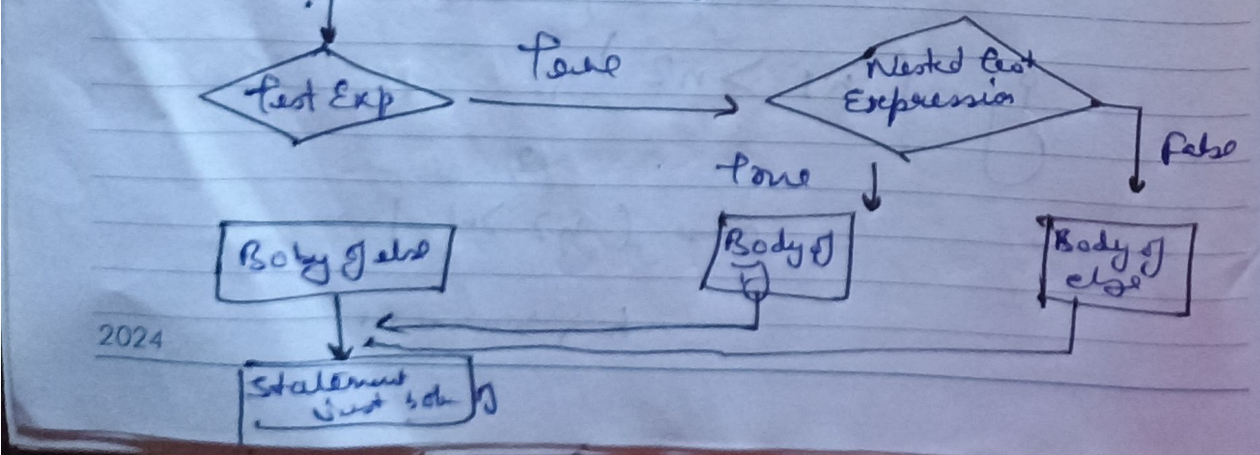
Relational operators are used to evaluate condition (true or false) inside the if statement some examples of relational operators are.

== , >= , > , < , <= , !=

equals — 0 — 1 — Not equals

Greater than equal — Smaller than equal

Flow Chart of Nested if



Java's selection statement.

> Nested if statement

Nested if statement involve placing one 'if' statement inside another if statement this allow for more complex decision making processes.

```
int num = 10;
if (num > 0) {
    if (num % 2 == 0) {
        SOUT("the number is positive and even");
    } else {
        System.println("the number is negative and odd");
    }
} else {
    if (num % 2 == 0) {
        SOUT("the ---");
    } else {
        Sout("the ---");
    }
}
```

28 MARCH THURSDAY

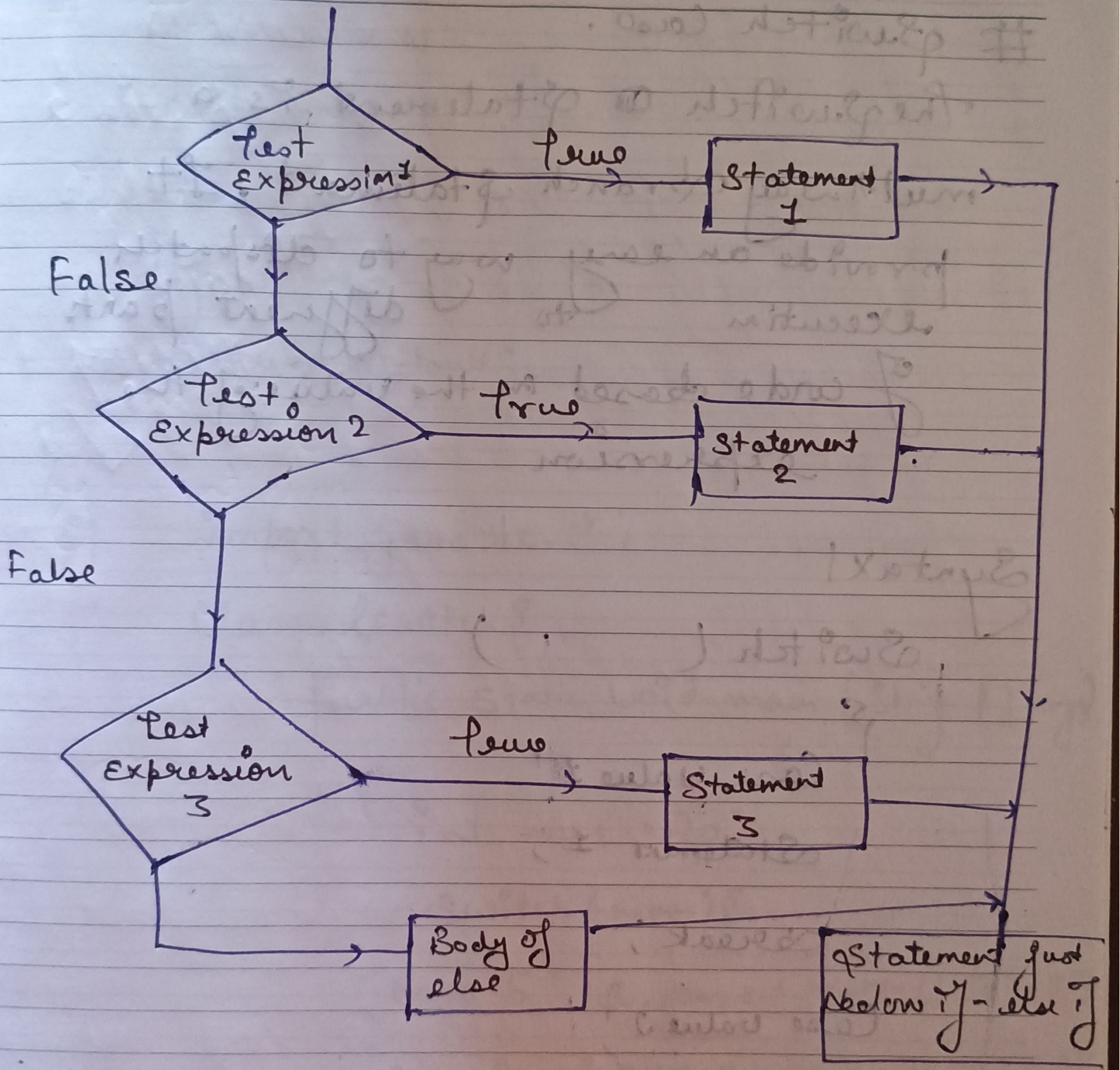
* if - else if Ladder.

The if - else if ladder is used to test ~~multiple~~ multiple conditions. If one the condition is true, the corresponding block of code is executed

eg int marks = 95;

```
if (marks >= 90){
        SOUT("Grade:A");
} else if (marks >= 80){
        SOUT("Grade:B");
} else if (marks >= 70){
        SOUT("Grade:C");
} else if (marks >= 5.
        SOUT("Grade:D");
}
```

2024

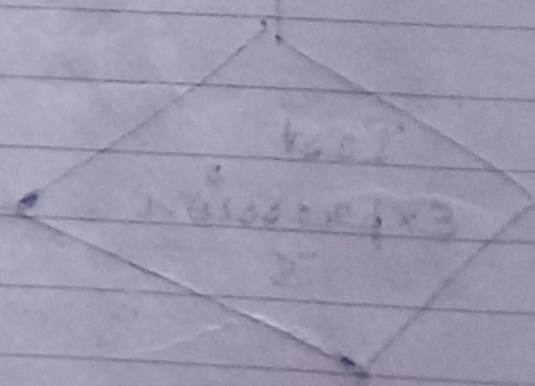

Test Expression 1
True
Statement 1
False
Test Expression 2
True
Statement 2
False
Test Expression 3
True
Statement 3
Body of else
Statement just below if-else if

Switch Case.

The Switch Case statement is a multiway branch statement. It provide an easy way to dispatch execution to different parts of code based on the value of the expression

Syntax!

```
Switch ( . )
{
  Case value 1 :
    Statement 1;
    break;
  Case value 2 :
    Statement 2;
    break 2 }
  break }
```



```
Case Value N:
    Statement N;
    break;
    default:
    StatementDefault;
}
eg  import java.io.*;
Class Lswitch{
    public static void main (String [] args)
    {
        int num =20;
        switch (num){
        Case 5: System.out.println ("It is 5");
                break;
        Case 10; System.out.println ("It is 10");
                break;
        Case 15: SOUT ("It is 15");
                break;
        Case 20: SOUT ("It is 20");
                break;
        default: SOUT ("NOT present");
    }
}
```

02 APRIL TUESDAY
DAYS 093-273

S	M	T	W	T	F	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

Loops in Java.

- While loop :- A while loop is a control flow statement that allows code to be executed repeatedly based on a given Boolean condition. The while loop can be thought of as a repeating if statement

Syntax of while loop

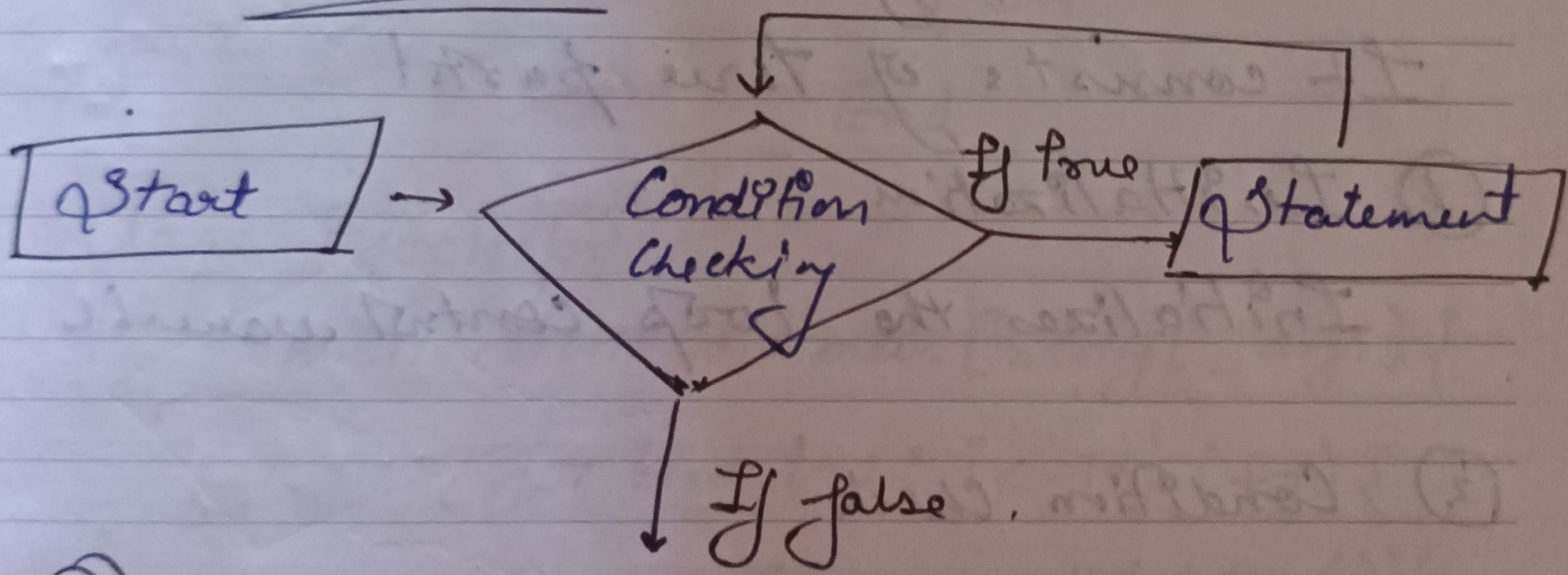
```
while (boolean condition)
{
    loop statement;
}
```

```
import java.io.*;
class while{
    public static void main (String [] arg
    {
        int i=0;
```

2024

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

Flow chart



① Condition Check :- The loop starts by checking a boolean condition

② loop Condition : If condition is true then the loop body is executed

③ Update :- The loop body usually contains an update to the variable that affects the condition

(4) Termination : When the Condition evaluates to 'False' the loop stops and the program continues with the statement after the loop

For loop : A 'For' loop in Java is used to execute a block of statement repeatedly until a specified condition is met.

It consists of three parts!

(1) Initialization

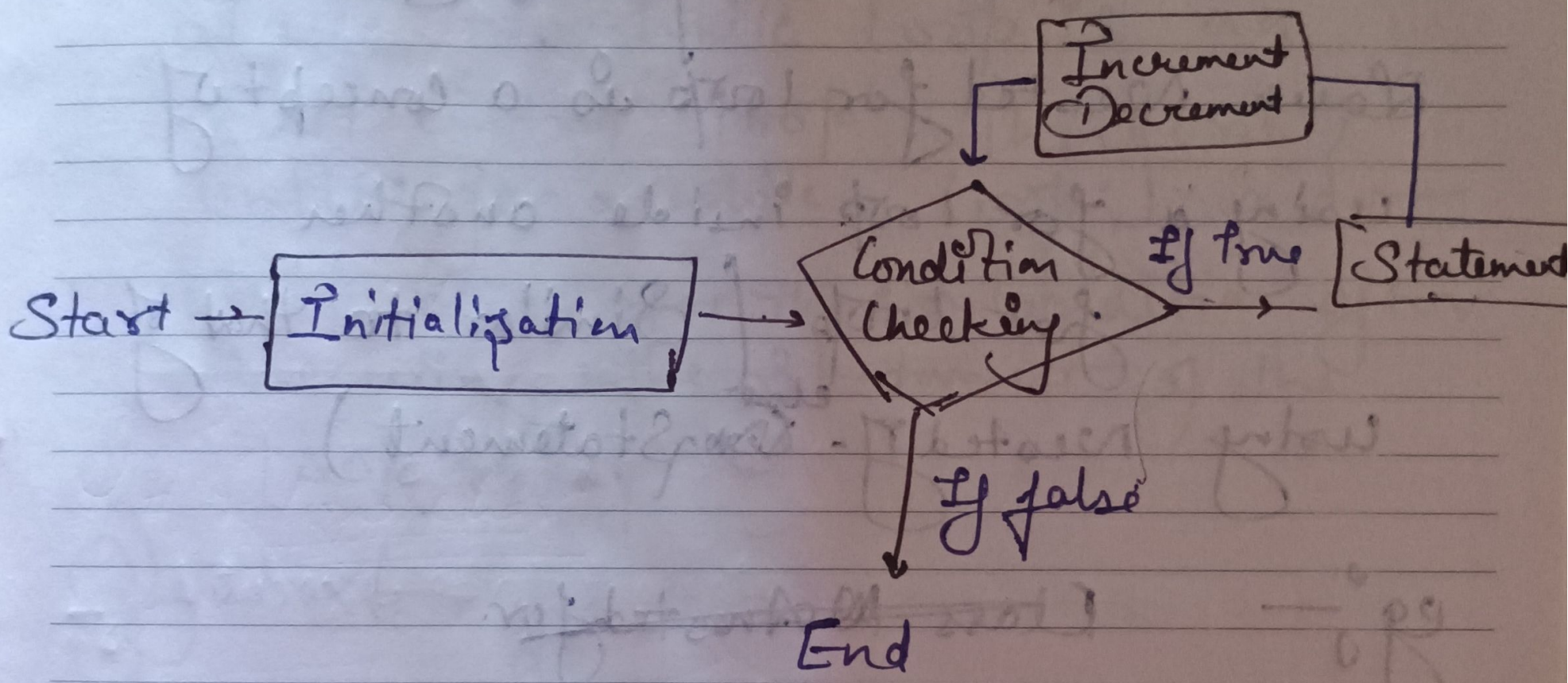
Initializes the loop control variable

(2) Condition Check

Evaluates the condition before each iteration

(3) Update : Updates the loop variable after each iteration

Flow Chart



Class F{for loop}{

public static void main (String[] args){

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

System.out.println(i);}

}

}

}

Nested for Loop in Java

Java Nested for loop is a concept of using a for loop inside another for loop (Similar to that of using Nested if-else statement)

eg:-

~~Class Nested for~~

```
Class Anested_for_loop{
public static void main(String [] args)
{
For (i=1; i<=5; i++)
{
for(j=1; j<=5; j++)
{
System.out.println
(j + " ");
}System.out.println();}
}
}
```


Java Do-while loop:-

Java Do-while loop is an Exit control loop. Therefore, unlike for a while loop, a do-while check for the test condition after executing the statement of the loop body

Syntax:-

```
do
{ // Loop Body
  Update expression
}
// condition check
while (test_Expression);
```

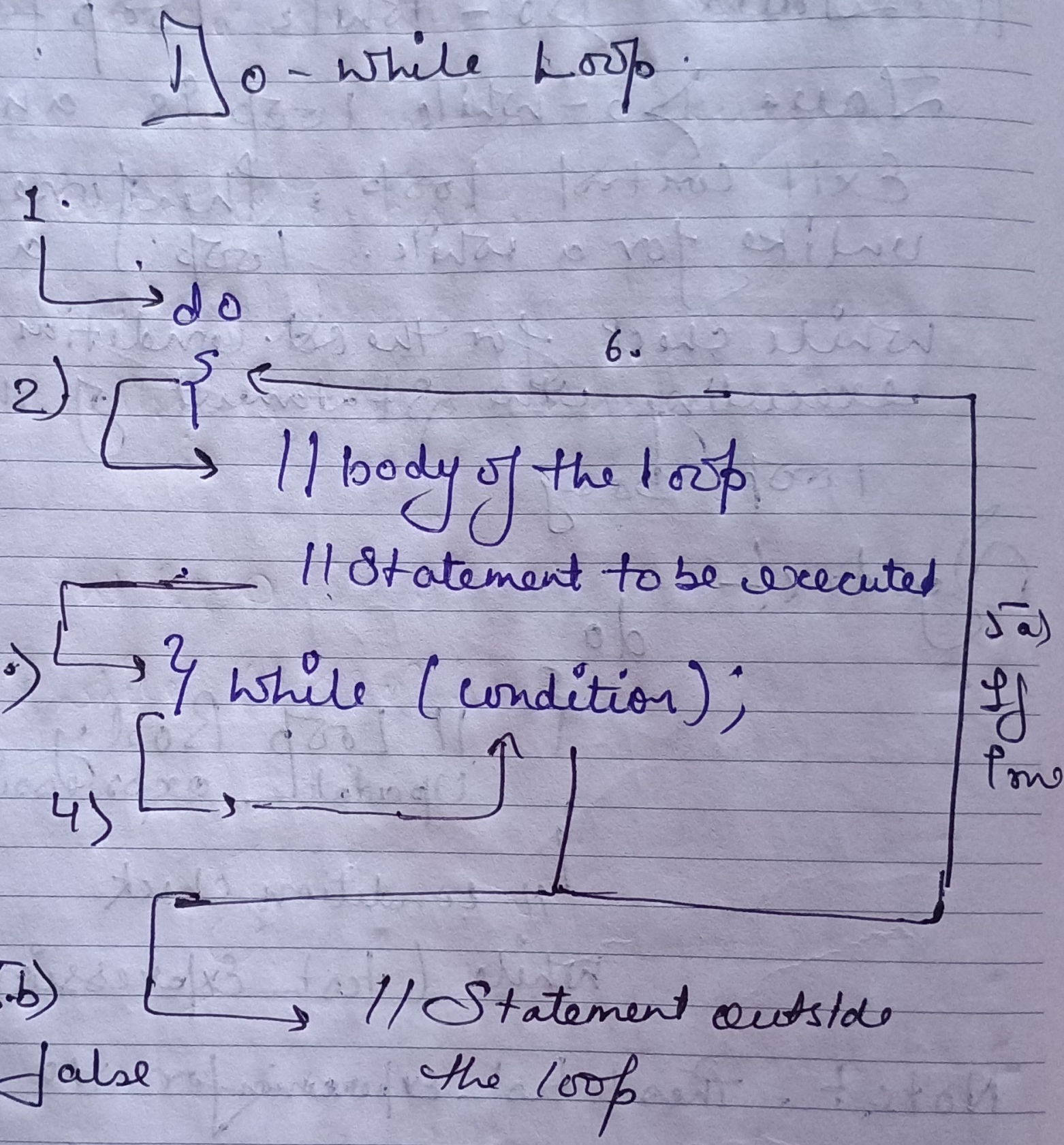
Note:- Here test_Expression for the do-while loop must return a boolean value else we would get Compile-time error

09 APRIL TUESDAY
DAYS 100-266

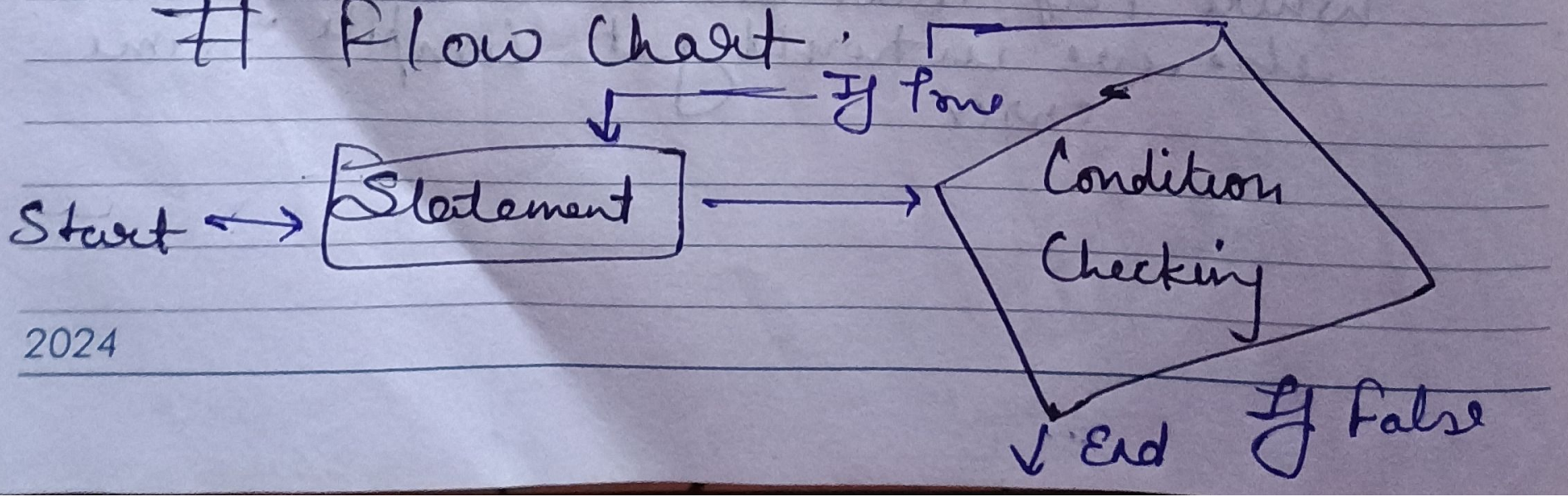
S	M	T	W	T	F	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

APRIL '24

Do-while Loop.



Flow Chart:



For-each loop in Java

For-each loop is another array traversing technique like for loop, while loop, do-while loop introduced in Java 5

Syntax

```
For ( type var : array)
    {
      statement using var;
    }
```

```
class Sfor{
    public static void main(String[] args)
        {
          int ar[]={10, 20, 30, 40, 50};
          For ( int element : ar)
            System.out.println
                (element + " ");
        }
}
```

11 APRIL THURSDAY
DAYS 102-264

S	M	T	W	T	F	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

APRIL 24

Continue statement is often used inside in programming languages inside loops control structure.

- Continue statement used in Such as 'For', while, and do-while

eg
```
int i=0;
while (i<20) {
    i++;
    if (i%2==0) {
        continue;
    }
    Sout(i);
}
```

Break Statement :

The break statement in Java is used to terminate the current loop or switch statement.

When 'break' statement is encountered, the control exits the loop or switch statement immediately & resumes execution at the next statement following the loop or switch

Usage of Break statement

(1) In loops

Break Statement can be used in all type of loops (for, while, and do while) to exit the loop prematurely based on a specific Condition

(2) In Switch Statement

Switch Statement to terminate a case & prevent fall-through to subsequent case

```
for (int i = 0; i < 10; i++)
{
    if (i == 5){
        break;
    }
    System.out.println(i);
}
```