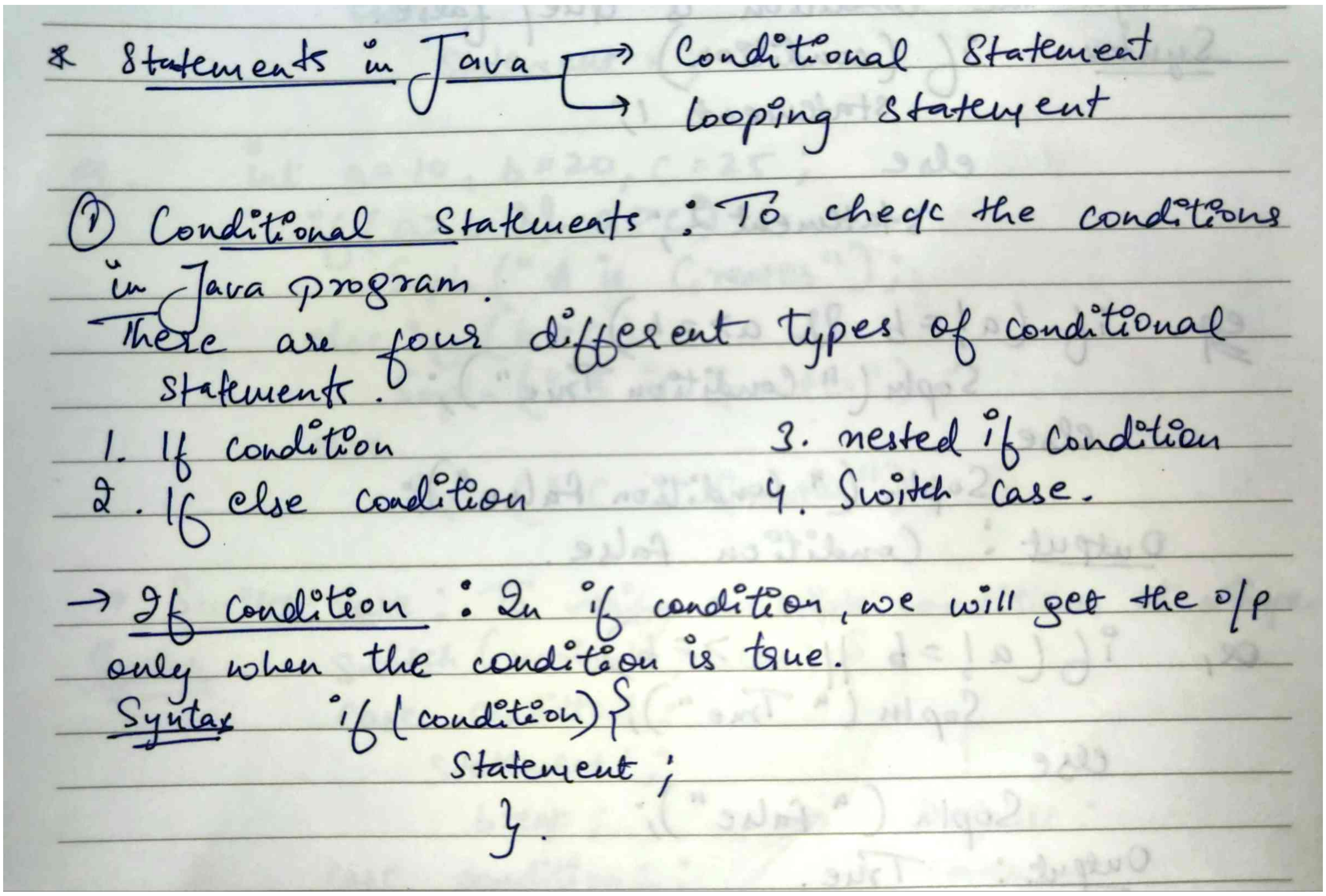

$$\begin{tikzcd}[row sep=small] & \text{Conditional Statement} \\ \text{* Statements in Java} \arrow[ur] \arrow[r] & \text{looping statement} \end{tikzcd}$$

① Conditional Statements : To check the conditions in Java program.

There are four different types of conditional statements.

1. If condition
2. If else condition
3. nested if condition
4. Switch case.

→ If condition : In if condition, we will get the o/p only when the condition is true.

Syntax

```
if (condition){
        Statement ;
    }.
```

eg.
```
int a = 10;
int b = 15;
  if (a<b) {
        Sopln ("Condition True");
  }
```
Output : Condition True

or,
```
if (a != b && a <= b) {
      Sopln ( " Condition True " );
   }
```
Output : Condition True.

→ If Else Condition : We will get the output even though the condition is true/ false.

Syntax
```
if (condition)
        statement 1;
    else
        statement 2;
```

eg.
```
if (a != b && a >= b)
        Sopln (" Condition True ");
   else
        Sopln (" Condition false ");
```
Output : Condition false.

or,
```
if (a != b || a >= b)
        Sopln (" True ");
   else
        Sopln (" false ");
```
Output : True.

→ Nested If Condition: It will print the output if condition is true or it will check the another condition if the condition is false.
Also called if else if condition.

Syntax

```
if (condition 1)
    statement 1;
else if (condition 2)
    Statement 2;
else if (condition 3)
    statement 3;
        ;
        '
else
    Statement n;
```

eg.

```
int a=10, b=20, c=25;
if (a>b && a>c)
    Sopln ("A is Greater");
else if (b>c)
    Sopln ("B is Greater");
else
    Sopln (" C is Greater");
```

→ Switch Case: To verify multiple conditions at a time.

Syntax

```
switch (variable/value) {
Case condition1 :
    Statement 1;
        break;
Case condition2 :
    Statement 2;
      break;
                        |
                        |
                   Default :
                    statement ;
                    break;
```

eg.
```
int a= 10 ;
int b= 7 ; int c= a-b ;
switch (c) {
  case 1:
       Sopln (" C value is 1");
       break;
  case 2 :
       Sopln (" C value is 2") ;
       break;
  case 3 :
       Sopln (" C value is 3") ;
       break ;
  case 4:
       Sopln ("C value is 4");
       break;
  default :
       Sopln (" C value is Out of the range");
       break ;
}.
```
Output : C value is 3 .

Or,
```
int a=10;
int b= 3 ;
int c= a -b ;
```
Now, for the same code.
The output will be

C value is Out of the range.

because c = a-b
c = 10-3
c = 7

as, there is no value of c in the code, therefore prints the default statement.

NOTE : break is used to break the condition & come directly out of the loop.

* Looping Statements

We are ~~executing~~ using looping statements to execute same line of code for multiple times. There are 3 different looping statements in Java

1. for loop 2. Nested for loop 3. while loop.

→ for loop : To execute same lines for specific number of times.

Syntax
```
for (initialization ; condition ; inc/dec)
{ statements ; }.
```

eg.
```
for (int i = 1 ; i<=5 ; i++)
{
    System.out.println(i) ;
}.
```

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

→ Nested for loop : Inserting one for loop inside ~~the~~ another for loop. Execution starts always from main loop, then it goes to sub loop, complete the sub loop. all the iterations then it comes back to the main loop.

<u>Syntax</u> for (initialization; condition; inc/dec.)

```
for (initialization; condition; inc/dec.)
{
    for (init.; condition; inc/dec.)
    {
        ;
        {
            statements ;
        }
    ; }
}
```

<u>eg.</u>

```
for (int i = 1; i <= 5; i++)
{
    for (int j = 1; j <= 5; j++)
    {
        System.out.println(i + " - " + j);
    }
}
```

<u>Output</u> :

```
1 - 1          3 - 3
1 - 2          3 - 4
1 - 3          3 - 5
1 - 4            |
1 - 5            |
2 - 1            |
2 - 2          5 - 5
2 - 3
2 - 4
2 - 5
3 - 1
3 - 2
```

→ While loop: To execute the same lines of code for multiple times until the condition is satisfied.

Syntax

```
while (condition) {
        Statements;
        increment / decrement
    }
```

eg.

```
int i=1;
While (i<5) {
    System.out.println("i value is:" +i);
    i++;
}.
```

Output

```
:i value is: 1
i value is: 2
i value is: 3
i value is: 4
```

* String functions / Methods
we are using string functions to perform the string related validations in java.

1. length() : To check the length of the string.
Syntax variable.length();

2. startsWith : To check whether the starting with specific character or not.
Syntax variable.startsWith("expected text");

3. endsWith : To check whether the variable is ending with specific character or not.
Syntax variable. endsWith ("expected text");

4. toLowerCase : To convert string from uppercase into lowercase.
Syntax variable. toLowerCase();

5. toUpperCase : To convert string into uppercase.
Syntax variable. toUpperCase();

6. equals : To verify whether one string is equal to another string or not.
Syntax variable. equals (second/another variable);

7. equalsIgnoreCase : To verify whether variable is same as another variable or not irrespective of case.
Syntax variable. equalsIgnoreCase (another variable);

8. contains : To verify whether the string contains specific value or string or not.
Syntax variable. contains ("value");

eg. String a = "Bhawna";
Sopln (a. contains ("x")); // False
Sopln (a. contains ("na")); // True.

9. concat : To concatenate one string to another
Syntax variable. concat ("Second String/var");

10. charAt : To verify which character is available in which index of the string.
<u>Syntax</u> variable. charAt (index no);

11. indexOf : To verify in which index which character is available.
<u>Syntax</u> variable.indexOf (char);

12. trim : To eliminate starting & ending spaces Of the string.
<u>Syntax</u> variable. trim();

<u>eg</u>. String a = "   Bhawna   Manral   ";
System. out. println(a.trim());
<u>Output</u> : Bhawna Manral

13. replace : To replace old characters or strings with new characters or strings in a var.
<u>Syntax</u> var. replace (" Old Char | String", "New Char/string");

<u>eg</u>. String a = " Bhawna Thareja";
Sopln (a. replace ("n", "z"));
Sopln (a.replace (" ", ""));

<u>Output</u> : Bhawza Thareja
BhawnaThareja

14. split : To split variable / String into multiple substring.
<u>Syntax</u> variable. Split ("separator");

eg. String a = "Manral Java and Selenium";
String[] arr = a.split(" ");

Manral	Java	and	Selenium
arr(0]	arr [1]	arr (2]	arr [3]

for (int i=0 ; i< arr.length ; i++)
System.out.print ~~Sopln~~ (arr [i]);

Output : ManralJavaandSelenium

eg. String a = "Bhawna#Java";
String[] arr = a.split ("#");

for (int i= arr.length -1 ; i>=0 ; i--)
System.out.print (arr [i]);

Output : BhawnaJava

Example : Prepare program to split variable into multiple substrings & print strings which length is only 3.

```
psvm ( string[] args) {
    String a = "abcd xyz 1111 2222 234";
String[] b = a.split(" ");
for ( int i=0 ; i<b.length ; i++) {
    if(b[i].length () == 3)
        Sopln(b[i]); }
  } }
```

O/P. xyz
234