Conditional Statements

Change the flow of program execution

3 Types:
- Decision Making
- Loop Statements
- Jump Statements

Decision Making:

Program execution of statements based on certain conditions

→ if
→ if else
→ if else if else
→ nested if
→ switch

if: Execute the specific block of code

Syn:
```
if (condition)
{
    Block;
}
```

Eg:
```
int a=5, b=4;
if (a>b)  → Condition is True
{
System.out.println("Greater"+a);   ← o/p displayed
}
```

if else: Execute the block if condition is true otherwise else executed

Syn:
```
if(condition)
{
    Block;
}
else
{
    Block;
}
```

Eg:
```
int n=8;
if (n>10)  → Condition is false   (moved to else part)
{
System.out.print("True")
}
else
{
System.out.print("False")   ← o/p will be displayed
}
```

if else if else: If condition is true execute otherwise else if executed, both condition are false else part will be executed

Syn:
```
if (condition) {
    Code;
}
else if (condition) {
    Code;
}
else {
    Code;
}
```

Eg:
```
int a = 5, b = 7;
if (a == b) {          // False
    System.out.print("Equal");
}
else if (a > b) {      // check, True
    System.out.print("a is greater");   // Executed
}
else {
    System.out.print("b is greater");
}
```
o/p

nested if: if within if

Syn:
```
if (condition1)
{
    if (condition2) {
        Block;
    }
}
```

Eg:
```
int a = 5, b = 10;
if (a > 0)
{
    if (b != 0) {          // condition is True
        System.out.print("b is greater than a");
    }
}
```

Switch: It is a multi-branch statement.

Break: Save lot of execution

Default: If there is no case match, then default will be executed.

> **NOTE**:
> → Cases cannot be duplicate
> → Case variables can be byte, int, short, long, enumeration, strings.

Syn:
```
switch (expression) {
    case value1:
        code;
    break;
    .
    .
    .
    default:
        code;
}
```

Eg:
```
switch (String) {
    case "hi": System.out.print("Hi");
    break;
    case "hello": System.out.print("Hello");
    break;
    default: System.out.print("Not found");
}
```

Jump Statements:

Break:

Used within a loop, Terminate the seq in switch statement

Eg:
```
for (int i = 1; i <= 10; i++) {
    if (i == 5)        → Condition satisfy
        break;         ← break the loop
    System.out.print(i + " ");
}
```
1 2 3 4

Continue:

Used to jump to the next iteration of the loop.

Eg:
```
for (int i = 1; i <= 5; i++) {
    if (i == 3)        → Skip this
        continue;      → & Continue upto end
    System.out.print(i + " ");
}
```
1 2 4 5

Return:

Immediately terminate the method in which code is executed.

Declare a method return type in method declaration

If attempt to return value from the method, then it is Compile-time error.

Eg:
```
public static int findsum(int a, int b) {
    int sum = a + b;
    return sum;
}
public static void main(String args[]) {
    Return obj = new Return();
    System.out.print(obj.findsum(10, 20));
}
```
Return Giving value to findsum to main

Loop Statement:

Used to repeat a block of code

for: Execute the block of code

Syn:
```
for ( initializing ; condition ; inc/dec)
{
    Block of code;
}
```

Eg:
```
for (int i=1 ; i<=10 ; i++) {
    System.out.print ("*");
}
```
o/p: * * * * * * * * * *

for each: Used on an array or collection type.

Works as a Iterator

No inc/dec need to update the loop variable

Syn:
```
for ( datatype var : arr_name)
{
    Block of code;
}
```

Adv: Use for array, array-list, linked-list

Dis: Backtracking is impossible

Eg:
```
int arr [] = {4, 5, 1, 3};
for(int n : arr) {
    System.out.print (n + " ");
}
```

nested for: Loop inside loop

Syn:
```
for ( init ; cond ; inc/dec){
    for ( init ; cond ; inc/dec){
        Inner loop ;
    }
    outer loop;
}
```

Eg:
```
int n = 5;
for (int i=1 ; i<=n ; i++){
    for(int j=1 ; j<=n ; j++){
        System.out.print(" *");
    }
    System.out.println( );
}
```

while loop: Entry controlled loop

Iterate no of. statement multiple times

Syn:
```
while (condition)
{
    block of code;
}
```

Eg:
```
int n, i=1 ;
while (i<=n){
    System.out.print(i+" ");
    i+=2;
}
```

do while loop — Exit controlled loop

do block executes atleast once, even if condition is false

Syn:
```
do
{
  // block code;
} while (condition);
```

Eg:
```
int n = 5, i = 1;
do {
  System.out.print(i + " ");
  i++;
} while (i < n);
```

PROGRAM

Reverse String without built-in functions

```
public class ReverseString {
  public static String reverse(String str) {
    char[] charArray = str.toCharArray();
    String reversed = "";
    for (int i = charArray.length - 1; i >= 0; i--) {
      reversed += charArray[i];
    }
    return reversed;
  }
  public static void main(String args[]) {
    String original = "Helloworld";
    String result = reverse(original);
    System.out.println("Original: " + original);
    System.out.println("Reversed: " + result);
  }
}
```

O/p:
Original: Helloworld
Reversed: dlrowolleH

Reverse Number:

```
int rem, res = 0;
while (n != 0) {
  rem = n % 10;
  res = (res * 10) + rem;
  n = n/10;
}
System.out.print(res);
```

I/p: 432
o/p: 234

Prime or Not:

```
int n = SC.nextInt();
for(int i=2; i*i<=n; i++){
  if (n % i == 0) {
    System.out.print("Not ");
  }
}
System.out.print("Prime");
```

I/p: 5
Prime

Swap two numbers:

I/p : 4 5 o/p : 5 4

```
Int a=5;
Int b=b;
Int c;
c=a;
a=b;
b=c;
System.out.println(a);
System.out.println(b);
```

Swap without 3rd variable

```
a= a+b;   a= 4+5=9
b=a-b;    b= 9-5=4
a=a-b;    a= 9-4=5  → a=5, b=4
```

```
a = a^b;
b = a^b;
a = a^b;
```

Fibonacci

0 1 1 2 3 5

```
Int n=5;
Int a=0, b=1, c=1;
System.out.print(a+" ");
for (Int i=0; i<n; i++)
{
    a=b;
    b=c;
    c=a+b;
    System.out.print(a+" ");
}
```

Add the prev value & give to next value.

Odd or Even:

User I/p

```
int n=Sc.nextInt();
if (n%2==0) {
    System.out.print("Even");
}
else {
    System.out.print("Odd");
}
```

o/p: 2 → Even 5 → Odd

Armstrong Number 153.

$1^3 + 5^3 + 3^3 = 3 + 125 + 27 = 153$

```
public static boolean
    isArmstrong(int num) {
int a=num, sum=0;
int digit = String.valueOf(num).length();
while (num!=0) {
    int digit2= num%10;
    sum+= Math.pow(digit2, digit);
    num/=10;
}
return sum==a;
}
```

Factorial Number

$n! = n*(n-1)*(n-2)*\ldots 1$

$5! = 5*4*3*2*1 = 120$

```
int n=Sc.nextInt();
int fact=1;
for (int i=1; i<=n; i++){
    fact = fact*i;
}
System.out.print(fact);
```

Sum of digits:

Add all i/p values

3 + 4 + 5 = 12

```
int n = Sc.nextInt();
int res = 0, rem;
while (n != 0) {
    rem = n % 10;
    res = res + rem;
    n = n / 10;
}
System.out.println(res);
```

O/p: I/p 1 2 3
→ 6

Palindrome

If number is same as rev value then it is palindrome

121 → Palindrome 153 → Not

```
int n = sc.nextInt();
int res = 0, rem, temp = n;
while (n != 0) {
    rem = n % 10;
    res = (res * 10) + rem;
    n = n / 10;
}
if (temp == res) {
    System.out.print("Palindrome");
}
else {
    System.out.print(" Not Palindrome");
}
```