Video 5 - Java Control Flow, String Handling and Arrays

Java Programming for Selenium

1) Conditional Statements / Decision Making Statements
2) Loop Statements
3) Branching Statements
4) String Handling in Java
5) Java Arrays

1) Conditional Statements / Decision Making Statements

a) Two types of Conditional statements

1) if statement
2) switch statement

b) Three types of conditions in Computer programming

1) Single Condition (Positive and Negative conditions)
Syntax:
```
if (condition){
.
.
}
```
Example:
Positive Condition
```
if (a>b) {
.
.
}
```

Negative condition

```
if (b<a){
.
.
}

if (!(b>a)){
.
```

.
}

2) Compound Condition (Positive and Negative conditions)

Positive Condition:

```
if ((a>b) && or || (a>c)){
.
.
}
```

Negative Condition

```
if (!(b>a) && or || (c>a)){
.
.
}
```

3) Nested condition (Positive and Negative conditions)

```
if (a>b){
if (a>c){
if (a>d){
……
……
}
}
}
```

c) Usage of Conditional Statements

1) Execute a block of statements when a condition is true

Syntax:

```
if (condition){
Statements
……
……
……
}
```

Example:
```
int a=100, b=500;

if (a>b){
System.out.println("A is a Big Number");
}

int a=100, b=50;

if (!(b>a)){
System.out.println("A is a Big Number");
}
```

2) Execute a block of statements when a condition is true, otherwise execute another block of statements

Syntax:
```
if (condition) {
Statements
……
……
……..
}
else
{
Statements
……
……
……..
}
```

Example:
```
int a=100, b=100;

if (a>b){
System.out.println("A is a Big Number");
}
else
{
System.out.println("B is a Big Number");
}
```

3) Execute a block of statements when a compound condition is true

Syntax:
```
if ((condition1) && or ||(condition2) && or || (condition3)){
statements
……
……
……..
}
```

Example:
```
int x=100, y=90, z=700;

if ((x>y) || (x>z)){
System.out.println("A is a Big Number");
}
```

4) Execute a block of statements when a compound condition is true, otherwise execute another block of statements

Syntax:
```
if ((condition1) && or ||(condition2) && or || (condition3)){
statements
……
……
……..
}
else {
……
……
……..
}
```

Example:
```
int x=100, y=900, z=70;

if ((x>y) && (x>z)){
System.out.println("X is a Big Number");
}
else {
System.out.println("X is not a Big Number");
}
```

5) Decide among several alternates (else if)

Syntax:

```
if (condition){
Statements
……
……
else if (condition){
Statements
……
……
else if (condition){
Statements
……
……
else if (condition){
Statements
……
……
else {
Statements
……
……..
}
```

Problem: Initialize an Integer variable and verify the range

if the number is in between 1 and 100 then display "Number is a Small Number"
if the number is in between 101 and 1000 then display "Number is a Medium Number"
if the number is in between 1001 and 10000 then display "Number is a Big Number"
if the number is more than 10000 then display "Number is a High Number"
Otherwise display "Number is either Zero or Negative number"

Solution:

Input:
1) 50
2) 150
3) 1500
4) 15000

5) 0
6) -100

```
int val =-100;

if ((val>0) && (val <=100)){ System.out.println("Val is a Small Number"); }
else if ((val>100) && (val<=1000)){
System.out.println("Val is a Medium Number");
}

else if ((val>1000) && (val<=10000)){
System.out.println("Val is a Big Number");
}

else if (val >10000){
System.out.println("Val is High Number");
}

else {
System.out.println("val is either Zero or Negative Number");
}
```

6) Execute a block of statements when more than one condition is true (Nested if)

Syntax:

```
if (condition1){
if (condition2){
if (condition3){
Statements
……
……
}
}
}
```

Example:

```
int a=100, b=90, c=80, d=70;

if (a>b){
if (a>c){
if (a>d){
System.out.println("A is a Big Number");
```

```
}
}
}
```

7) Execute a block of statements when more than one condition is true (Nested if), otherwise execute another block of statements

Syntax:

```
if (condition1){
if (condition2){
if (condition3){
Statements
……
……
}
else {
Statements
……
……
}
}
else {
Statements
……
……
}
}
else {
Statements
……
……
}
```

```
int a=100, b=90, c=80, d=70;

if (a>b){
if (a>c){
if (a>d){
System.out.println("A is a Big Number");
}
else {
System.out.println("A is Not a Big Number - 3rd Condition is False");
}
```

```
}
else {
System.out.println("A is Not a Big Number - 2nd Condition is False");
}
}
else {
System.out.println("A is Not a Big Number - 1st Condition is False");
}
```

Using Compound Condition

```
int a=100, b=900, c=80, d=70;

if ((a>b) && (a>c) && (a>d)){
System.out.println("A is a Big Number");
}
else {
System.out.println("A is Not a Big Number");
}
```

Nested Condition versus Compound Condition

In Nested Condition we can write multiple else parts, where as in Compound condition
single else part only…

Problem: Find the biggest value (Integer values) among four values

Hint: use else if and compound condition

Java Program:

```
int a=100, b=100, c=100, d=100;

if ((a>b) && (a>c) && (a>d)){
System.out.println("A is a Big Number");
}
else if ((b>a) && (b>c) && (b>d)){
System.out.println("B is a Big Number");
}
else if ((c>a) && (c>b) && (c>d)){
System.out.println("C is a Big Number");
}
else {
System.out.println("D is a Big Number");
```

```
}
}
}
```

8) Decide among several alternates (using switch statement)

Syntax:

```
switch (expression){
case1 value:
statements
……
……
break;

case2 value:
statements
……
……
break;

case3 value:
statements
……
……
break;
case4 value:
statements
……
……
break;

default:
statements
……
……
}

char grade ='1';

switch (grade){
```

```
case 'A':
System.out.println("Excellent");
break;

case 'B':
System.out.println("Good");
break;

case 'C':
System.out.println("Better");
break;

default:
System.out.println("Invalid Grade");
}
```

Note:
You can use Condition/s only in your program,
or Loop/s only in your Program,
You can insert Loop/s in Condition/s and vice versa.

Java Programming for Selenium

2) Java Control Flow - Loop Statements

1) for loop
2) while loop
3) do while loop
4) enhanced for loop

1) for loop

Description: It repeats a block of statements for a specified number of times

Syntax:

```
for (StratValue; EndValue; Increment / Decrement){
Statements
……
……
}
```

Examples:

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

Print 1 to 5 Number except 4th Number

```
for (int i=1; i<=5; i++){
if (i !=4){
System.out.println(i);
}
```

Print 1 to 10 Numbers in reverse Order
```
for (int i=10; i>=1; i--){
System.out.println(i);
}
```

2) while loop

Description: It repeats a block of statements while condition is true

Syntax:
```
Initialization
while (condition){
Statements
……
……
Increment/Decrement
}
```

Example/s:
Print 1 to 10 Numbers
```
int i=1;

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

Print 1 to 5 Number except 4th Number
```
while (i<=5){
if (i !=4) {
```

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

Print 1 to 10 Numbers in reverse Order

```
int i=10;

while (i>=1){
System.out.println(i);
i--;
}
```

3) do while

It repeats a block of statements while condition is true
It executes statements at least once irrespective of the condition

Syntax:

```
Initialization
do
{
Statements
……
……
Increment / Decrement
} while (condition);
```

Example:
Print 1 to 5 Numbers:

```
int i=1;

do
{
System.out.println(i);
i++;
} while (i<=5);

int i=10;

do
{
```

```
System.out.println(i);
i++;
} while (i<=5);
```

Print 1 to 5 Numbers except 3rd Number

```
int i=1;

do
{
if (i!=3){
System.out.println(i);
}
i++;
} while (i<=5);
```

4) Enhanced for loop

It executes all elements in an Array

Syntax:

```
Array Declaration;
for (declaration: Expression / Array){
Statements
……
……
}
```

Example:

```
String [] languages = {"COBOL", "C", "Java", "VBScript"};

for (String lang: languages){
System.out.println(lang);
}

int a=10, b=5;
int [] mathOperations = new int[3];

mathOperations[0] = a+b;
mathOperations[1] = a-b;
mathOperations[2] = a*b;
```

```
for (int operation: mathOperations){
System.out.println(operation);
}
```

3) Java Control Flow - Branching Statements

- Branching statements are used to transfer control from one point to another in the code
- Branching statements are defined by three keywords - break, continue and return

1) break

- break statement is used to stop the execution and comes out of loop
- Mostly break statements are used in switch and in loops….

Example:

```
for (int i=1; i<=10; i++){
System.out.println(i);
if (i==4){
break;
}
}
```

2) continue

continue statement is also same as break statement, the only difference is when break statement executes it comes out from the loop, whereas continue statement executes comes out from the loop temporarily.

Example:

```
for (int i=1; i<=10; i++){
if (i%2 != 0){
continue;
}
System.out.println(i);
}
```

3) return

return statement is used in User defined methods (methods with return value), return statement must be always last statement in the method

Example:

```
public class ControFlow {

public static void main(String[] args) {
ControFlow obj = new ControFlow();
int res= obj.add(100, 50);
System.out.println(res);//150

System.out.println(obj.add(100, 200));//300

}
public int add(int a, int b){
int result=a+b;
return result;
}
}
```

//Create Object

```
(ClassName objectName = new ClassName();
ControFlow obj = new ControFlow();

DataType variableName = Object.Method(Values for Arguments)
int res= obj.add(100, 50);
```

4) String Handling in Java

What is String?

String is a sequence of characters written in double quotes

Syntax:

```
String stringName= "value"
```

String may have Alphabets, Numbers and Special characters

```
String a="India";
String b="100";
int c=100;
String d="India123";
string e="$%^";
String f= "India123#$";
```

Operations on Strings:

1) Concatenating Strings

String + String - String
String + Number - String
Number + Number - Addition

Example:

```
String str1= "Selenium";
String str2 = " Testing";
//String concatenation using + operator
System.out.println(str1+str2);//Selenium Testing

//String concatenation using "concat" Method
System.out.println(str1.concat(str2));//Selenium Testing

System.out.println("Selenium"+"Testing");//SeleniumTesting
System.out.println("Selenium"+1+1);//Selenium11
System.out.println(1+1+"Selenium");//2Selenmium
System.out.println(1+11);//12
```

2) String Comparison

In Computer programming we have two types of comparisons,
1) 2-way Comparison (Logical/ true or false)
2) 3-way Comparison (Zero, Greater than Zero, Less than Zero)

Ways of String Comparison

a) String Comparison using Relational Operator (==)
It supports 2-way comparison

b) String Comparison using equals() method
It supports 2-way comparison

c) String Comparison using compareTo() method
It supports 3-way Comparison

Result criteria for 3-way comparison

if string1 == string2 then 0
if string1 > string 2 then Positive Value
if string1 < string 2 then Negative value

Comparing two number - based on their values (3>2)
Comparing two strings - based on ANSI values

ANSI character codes

A to Z (65 to 90)
a to z (97 to 122)
0- to 9 (48 to 57)

Example:

```
String str1 = "SELENIUM";
String str2 = "selenium";
String str3 = "SELENIUM";
String str4 = "zseleniu";

//String Comparison using Relational (==) Operator
System.out.println(str1 == str2);//false
System.out.println(str1 == str3);//true

//String Comparison using equals() Method
System.out.println(str1.equals(str2));//false
System.out.println(str1.equals(str3));//true

//String comparison using compareTo()
System.out.println(str1.compareTo(str2));//Negative value
System.out.println(str1.compareTo(str3));//0
System.out.println(str4.compareTo(str1));//Positive value
```

5) Java Arrays

- In Java, Array is an Object that holds a fixed number of values of a single data type
- The length of Array is established when the Array is created.
- Array length is fixed and index starts from zero to n-1,

Declaration of Arrays

1st Method

Syntax:
dataType arrayName[]; Declare an Array/ Create an Array
arrayName = new dataType[size]; //Define size
arrayName[index] = value; //Assign a value
.
.
.

Example:
```
int a[];
a=new int[3];
a[0] = 10;
a[1] = 20;
a[2] = 30;

System.out.println(a[1] + a[2]);//50
```

Assign values to Array elements that more than the length of Array (Run-time Error)

```
int a[];
a=new int[3];
a[0] = 10;
a[1] = 20;
a[2] = 30;
a[10] =40;

System.out.println(a[1] + a[2]);//50
```

Assign values to some Array Elements only

```
int a[];
a=new int[3];
a[0] = 10;
a[1] = 20;

System.out.println(a[0] + a[1]);//30
```

Assign different type of data to Array Elements (Syntax Error)

```
int a[];
a=new int[3];
a[0] = 10;
a[1] = 20;
//a[2] =10.23;

System.out.println(a[0] + a[1]);//30
```

2nd Method

```
dataType [] arrayName = new dataType [size]; //Create an Array with length
arrayName[index] = value; //Assign value
.
.
```

Example:

```
int [] a = new int [3];
a[0] =10;
a[1] =20;
a[2] =30;
System.out.println(a[1] + a[2]);//50
```

3rd Method

```
dataType [] arrayName = {value1, value2, value3, value4};//Create an Array with Initialization
```

Example:

```
int [] a= {10, 20, 30, 40};
System.out.println(a[1] +a[3]);//60
```

Creating different type Arrays

```
int [] a= {10, 20, 30, 40, 50}; //Array of Integers
char [] b= {'A', 'S', '1', '*'};//Array of Characters
String [] c = {"UFT", "Selenium", "RFT", "SilkTest"};//Array of Strings
double [] d ={1.234, 3.456, 6.45, 7.890}; // Array of decimal ponit values
boolean [] e = {true, true, false, true, false}; //Array og Boolean Values / Logical Values

System.out.println(a[2]);//30
System.out.println(d[3]);//7.890
System.out.println(e[0]);//true
```