Selenium Class 16 - Java User Defined Methods

i) Introduction to Java Methods

ii) Types of Methods

iii) User defined Methods

i) Introduction to Java Methods

What is Method?

Java Method is a set of statements/steps that are grouped together to perform an operation

Methods are equivalent to Functions in Structure Programming

> In Structured Programming (ex: C Language) we use Functions (Built-in and User defined)

> In Object Oriented programming (ex: Java) we use methods (Built-in and User defined)

When we choose Methods?

Whenever we want to perform any operation multiple times then we choose Methods

Advantages of Methods

i) Code reusability, so that we can reduce the code size

ii) Code maintenance is easy

ii) Types of Methods

Two types of Methods in Java

1) Built-in Methods

2) User defined methods

Categories of Built-in Methods

1) String Methods

2) Array Methods

3) Number Methods

4) Character Methods

Etc...

iii) User defined Methods

1) Method with return a value

(Perform operation/s and return a value)

a) Call methods by invoking object (Non Static Method)

b) Call methods without invoking Object (Static Method)

2) Method without return any value

(perform operations)

a) Call methods by invoking object (Non Static Method)

b) Call methods without invoking Object (Static Method)

Note: We Write Methods outside of main method, but we call methods inside of method.

Examples:

1) Method with return a value

a) Call methods by invoking object (Non Static Method)

Syntax for Creating Method:

accessModifier retrunType methodName (Argumnets...){

Method Body

Statements

.....

.....

.....

return Statement

}

Note: Methods use Arguments but Arguments are optional

(String Username, String password)

(int a, int b, int c)

()

Call Method

We call Non Static Methods using Object

Create Object

ClassName objectName= new ClassName();

Example:

```
public class JavaMethods {
//Non Static method with return a value
public int add(int a, int b){
int result = a+b;
return result;
}
public static void main(String[] args) {
//Create Object
JavaMethods       obj = new JavaMethods();
int res= obj.add(100, 200);
System.out.println(res);//300
//Or
System.out.println(obj.add(10, 20));//30
}
}
```

1) Method with return a value

b) Call methods without invoking Object (Static Method)

Create Static Syntax:

accessModifier NonAccessModifier retrunType methodname (Arguments){

Statements

.....

.....

.....

return Statement

}

Call Static method

We call Static methods using class name or directly (without class name)

```
public class JavaMethods {
//Static Method with returns a value and Arguments
public static int multiply(int a, int b){
int result = a*b;
return result;
}
//Static method with returns a value and no arguments
public static int multiply2(){
int a=100, b=20;
int result = a*b;
return result;
```

```
}
public static void main(String[] args) {
//Call Static Methods
int res1 = JavaMethods.multiply(10, 70);//700
int res2 = JavaMethods.multiply2(); //2000

int res3= multiply(10, 20);//200
System.out.println(res1);
System.out.println(res2);
System.out.println(res3);
}
}
```

2) Method without return any value

a) Call methods by invoking object (Non Static Method)

Syntax:

```
accessModifier returnsNothing methodname (Arguments){
Statements
.....
.....
.....
}
```

Example:

```
//Create a Non Static Method with returns Nothing
public void comparison(int a, int b){
```

```
if (a>b){
System.out.println("A is a Big Number");
}
else {
System.out.println("B is a Big Number");
}
}
public static void main(String[] args) {
//Create Object
JavaMethods obj= new JavaMethods();
//Call Non Static Method
obj.comparison(100, 102); //B is a Big Number
}
}
```

2) Method without return any value

b) Call methods without invoking Object (Static Method)

Syntax:

```
accessModifier NonAccessModifier returnsNothing methodName(Arguments){
Statements
.....
.....
.....
}
```

Example:

```
public class JavaMethods {
//Create Static Method with returns Nothing
public static void comparison(){
int a=100, b=40;

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

public static void main(String[] args) {
//Call Static Method with returns Nothing
JavaMethods.comparison();//A is a Big Number
comparison();//A is a Big Number
}
}
```

Usage of User defined Methods

1) Internal Use (Create and Call Methods within the same Class)

2) External Use (Call Methods from another Class)

1) Internal Use (Create and Call Methods within the same Class)

```
public class JavaMethods {
public int add(int a, int b){
int res = a+b;
return res;
}

public static void main(String[] args) {
JavaMethods obj = new JavaMethods();
int x= obj.add(100, 30);
System.out.println(x);
}
}
```

2) External Use (Call Methods from another Class)

class1:

```
public class JavaMethods {
public int add(int a, int b){
int res = a+b;
return res;
}

public static void main(String[] args) {
JavaMethods obj = new JavaMethods();
```

```
int x= obj.add(100, 30);
System.out.println(x);
}
}
```

Class 2:

```
public class Class2 {

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