Section 17: Learn Java Object Oriented Principles needed for Framework development

146. How TestNG Annotations help with Inheritance to remove Boilerplate code in test

Lets discuss some core Java topics which are related to framework development. we need to understand some object oriented principles

Create one class treat it like a parent class (ps)

ll’y Create one class treat it like a (ps1)

1. concept of inheritance ?

what is inheritance ? - means you can acquire the properties of your parent class.

let's say you have declared some methods and some variables in the PS class if you want to access all the methods, variables, what

all you declared in PSclass into PS1 class. Now PS1 is the another class which is trying to access all the properties and methods of the PS class.

In ps1 class - Public class ps1 extends PS {

So when you define like this, then PS becomes your parent class.

PS one become child class. So by giving this extends key words, we are telling that this child class is trying to acquire the properties of its parent.

So this parent child relation will be established if you use a keyword called extents.

I declared 1 method in the parent PS class.

Public void doThis()

{

Println(“I am here”)

}

Now, in the child class(PS1), I want to access that method and I want to execute in this PS1 class.

@Test

Public void Testrun()

{

doThis();

}

Note : If you declare @before method & @Aftermethod in parent class and if you run only the child class is these methods run ? – yes if you add inheritance in the child class it should run.

Now, in selenium web browser you need to open chrome browser every time you need to maximize it.

All that you can write in before method and you can push it in parent class.

So your actual test case only focus on the actual functionality remaining thing will all happen in the backend and by default it will execute before your test run by having connections with the testNG plus inheritance.

147. How to pass values from test through Parameterized Constructor & this keyword

In child class ps1 just add ( int a = 3; )

Create one new class as PS2

to call the method present in another class the one way is create object of that class

add this in ps1 class

PS2 ps2 = new PS2()

ps2 is the object with that object we can call the methods present in that class (ps2 dot)

Note : Two types of creating and calling the methods from other classes.

One is inheritance and one is with object.

When you create an object for any class, a constructor will be called.

So if you don't send any parameters, if you just create an object without sending any parameters, there is a default constructor which will be called in the back end, which you need not be aware because that's all handled in the back end.

PS2 ps2 = new PS2(3) // 3 is the parameter

you are calling and creating an object with an parameter. That means you need to define one constructor.

It's saying either remove this argument so that default constructor will be called.

If not, you need to create explicitly one constructor with this argument.

A constructor name will always be the class name and constructor will also reflect how many number of parameters we need to create

There are 2 variables here

1 is int a; - class variable // global varaible

2 is public PS2 (int a) -instance variable

You can call your own class variables with this. The dot keyword, this dot.

When you say that refers to the current class variables.

And here this dot refers to the your class global variable.

So from this lecture you understood what is instance variable?

What is class variable? What is parameterized constructor and how to make your class independent of test data.

This keyword refers to the current class variable.

If there are instance variable and current class variable, you can differentiate by pointing out this dot.

PS1.JAVA – new class

Public class PS1 extends PS {

@Test

Public void testRun()

{

PS2 ps2 = new PS2(3); // parameterized constructor

Int a = 3;

doThis(); // parent class

println(ps2.increment());

println(ps2.decrement());

}

PS2.JAVA – class name

Public class PS2 {

Int a;

//default constructor

Public PS2(int a)

{

this.a=a;

}

public int increment()

{

a= a+1;

return a;

}

Public int decrement()

{

a=a-1;

return a;

}

148. Usage of Super keyword in the Constructor to pass values to Parent class

Now for multiplication, you got an requirement that you need to create another utility class to handle only multiplication related to operations.

Create one more class as ps3

So to now access the methods present in PS3 class, what will you do?

You will just create object for that PS3 class and then you access those methods,

Can't you do like this instead of having crating new object ps3

I just want you to access this multiply three method directly with this utility object for PS2

inheritance concept where if you inherit your parent class, then all your parent class methods are loaded in your child class memory.

So when you create object for child class memory and try to access methods, you can also access methods of your parent class if you form this inheritance relation.

In ps2 add extends ps3

Super(a); - Basically, when you call this super keyword, that means your parent class constructor is invoked.

So super, whenever it sees that, it will go back to your parent class and wherever youare, constructor is there in your parent class.This is the constructor. This will be executed.

Note : super keyword is only used when there is an inheritance in place.

When you don't have any parent class, then this super keyword will go no where because the duty of this is to invoke parent class constructor when there is no inheritance itself where it will invoke.

…whenever you are writing super keyword, make sure that should be the first line in your child constructor.

PS3.JAVA - Class name

Public class PS3 {

Int a;

**public** PS3(**int** a) {

// this is constructor automatically create due to we are added super keyword in Ps2

// **TODO** Auto-generated constructor stub

**this**.a=a;

}

Public int multiplyTwo()

{

a = a \* 2 ;

return a;

}

Public int multiplyThree()

{

a = a \* 2 ;

return a;

}

Open PS1.JAVA – class name

Add this line in PS1 java class

Println (ps2. multiplyTwo());

Println (ps2.multiplyThree());

Console – 4,3,6,12