Selenium 4 Beginners

Rex Jones II

Learn Java
Classes, Objects & Methods

Table of Contents

Introduction 2
Classes 2
Objects 2
Methods 2
Concept of Classes, Objects, and Methods 2
Classes 3
Objects 3
Methods 3
Demo 3
Classes - Objects 3
Method - Return Vales 4
Method Pass Arguments To Parameters 5

Selenium 4 Beginners

Rex Jones II

Introduction

Hello Everybody, Welcome To Selenium 4 Beginners. My name is Rex Allen Jones II. We are going to discuss the Relationship Between Java's Classes, Objects, and Methods.

In this video's Tutorial Plan, I will Introduce Classes, Objects, and Methods, discuss the Concepts of Classes, Objects, and Methods, followed by a demo of Classes, Objects, and Methods.

If you are interested, you can download this presentation and code that I will demo in this video at https://tinyurl.com/Java-Classes-Objects-Methods

First, I am going to Introduce Classes, Objects, and Methods

Classes

Starting with a class. Class is created using the reserved word class. In this syntax, our reserved word is written in purple. ClassName written in white identifies the name of the class. Java is built upon classes because a class defines the nature of an object. Therefore, a class forms the basis of Object-Oriented Programming which is known as OOP.

Within a class, we can create objects, declare variables, and we can also declare methods. A Method's Body can include objects, variables, and all kind of statements that perform an action.

Objects

There's 2 ways to create an object. Both ways require the Class Name. The first syntax shows ClassName and an Object Reference Variable on one line then the Object Reference Variable equal new ClassName on the second line.

A shortcut of this same syntax is placing both lines on one line. We see the ClassName followed by the Object Reference Variable equal to new ClassName. Notice there are 2 ClassNames in both syntaxes. The first ClassName help declares a reference to the object while the second ClassName assist with allocating the new object.

Methods

In a good written Java program, a method performs only one task. The return type determines the type of data returned from the method. If no data is returned from the method then we use void. However, if data is returned from the method then the Return Type will be a Non-Object or Object Data Type.

All methods have a name and the name is used to call the method. The parameter list is 1 or more parameters that receive arguments. An argument is a value passed to the method. The parameters are declared just like a variable. 2 or more parameters are separated by a comma. Last but not least, the Method's Body is where we write our code.

Concept of Classes, Objects, and Methods

Now, let's discuss the concept of Classes, Objects, and Methods.

Classes

With this illustration, Car is our class. By definition, according to dictionary.com, a class is a number of persons or things regarded as forming a group by reason of common attributes. Therefore, a class contains a group of common attributes. Our car class will form a group of new cars.

Objects

That group of new cars with common attributes are called objects. We have 3 new car objects. Each of these new cars is a template of the Car class. That means, objects have the same attributes as their class. In Java, we call those attributes; members. A member can be a variable or method.

There is a difference between an object and an object reference variable. New Car, New Car, and New Car are the objects while coupe is an object reference variable. We can say object reference for short. Coupe is an object reference variable because it refers to the first New Car object. The same goes for minivan and sedan which refers to their object. In our industry, an object reference variable is mistakenly sometimes called an object. I can admit that I am guilty of calling an object reference; an object although they are different.

Methods

If you wanted to buy a new car, what's one of the main things you want to know. How much does it cost? We would use a method to calculate the cost. To calculate cost, we need to add the price and tax. Price and tax are data that's manipulated by the method.

Let's take another look at the same concept of Classes, Objects, and Methods but in a different way. On the left side, we have a Class and Method. On the right, we have Objects. Know what, I am going to bypass discussing Classes, Objects and Methods in this presentation slide but discuss them later in our demo.

Demo

Next, we have demo Classes, Objects, and Methods.

First, we will see how to create and use objects, followed by returning values to a method, then we will finish with arguments and parameters.

Let's start with how to create and use objects.

Classes - Objects

In Object-Oriented Programming, most classes have data and methods. The class is Car. Data will be instance variables int price and int tax. Our method will be calculateCost (public void calculateCost), which has a task of adding the price and tax. (int cost = price + tax) After assigning the value to cost, we print the cost by writing sysout "What Is The Car's Cost? $" + cost

In a different class, we have a main method to create our objects. Car coupe, minivan, sedan. coupe, minivan, and sedan are the type of Cars. The Data Type is Car while coupe, minivan, and sedan refer to the Car type.

Do you remember from the previous video “Understanding Java Variables & Operators”? We declared a double Non-Object Data Type and amount as the variable then initialized the variable with 23.45. We can do the same with Object Data Types.

I’m going to take this, one at a time and erase minivan and sedan then write coupe = new Car (). The 1st line declares coupe as a reference to an object of type Car. That’s why we call coupe an object reference variable because it is a variable that refers to an object. The 2nd line creates a new Car object and assigns a reference to coupe. These 2 lines can be written as Car coupe = new Car (); Let’s create one more for minivan. Car minivan = new Car.

Coupe and minivan refer to the Car object. As a result, the Car Object is a template of the Car class. So, check this out, since the Car class has price, tax, and the ability to calculate the cost. Coupe will have its own price, coupe will have its own tax and coupe will have calculateCost. The same with minivan. Minivan will have its own price, tax, and calculateCost. Objects are copies of a class which allows each object reference to have its own copy.

Therefore, we can write coupe dot. The dot operator provides access to its copy of variables and methods. Notice the intellisense shows price, tax, and calculateCost. This shows, that we can assign values to price and tax. coupe.price = 14000; coupe.tax = 1000; then call the calculateCost method: coupe.calculateCost;

Here’s what happen when coupe calls the calculateCost method. Our code will assign the price and tax to coupe, call calculateCost method, add both values within the method, assign the value to cost then print the cost of coupe. Let me show you by debugging.

First, we add a breakpoint by double clicking the line or shortcut CTRL + SHIFT + B. A breakpoint is the blue dot in front of the line number. Let’s Debug. Debug has stopped at price. The coupe’s price and tax are set to 0. F5 steps into our code. 14000 is assigned to price. F5 again then tax is assigned 1000. Coupe is getting ready to call the calcuateCost method. F5 – Debug moves to the calculateCost method to add our price and tax. F5 and now we see cost is assigned 15000. Next, we print the cost “What Is The Car’s Cost? 15000”

The same happens for minivan. Let’s assign 9000 to the price of minivan: minivan.price = 9000; minivan.tax = 1000 then we calculate cost for minivan. minivan.calculateCost. Remove the Breakpoint and Run again. coupe is 15000 and minivan is 10000

Next is returning values to a method.

Method - Return Vales

The key to returning values is Return Type. Return Type determines the type of data returned by the method. Currently, void is the return type which means our method calculateCost will not return a value. The Return Type can be any Object or Non Object Data Type.

Let’s go back to coupe and minivan. We see coupe and minivan each calls the calculatetCost method but what if, we wanted to see the cost difference between both cars? How would we calculate the difference coupe and minivan. We need to access the cost but the cost is calculated in a separate class.

We write return cost, to access cost and return it back to both callers: coupe.calculateCost and minivan.calculateCost. Notice the error, Void methods cannot return a value. We must change void to int because cost has an int Data Type. After returning cost, we need to assign the value. Let's assign coupe.calculateCost to int coupeCost. So when coupe calls calculateCost, the price and taxes are calculated, printed, returned to the method, then stored in variable coupeCost.

The same for minivan. We will store the cost into variable: int minivanCost. Now, we have access to cost for coupe and minivan. Let's calculate the difference between both cars. sysout "Difference Between Car's Cost = ?" + (coupeCost - minivanCost). This print statement subtracts the cost of minivan from the cost of coupe. Let's run. What's the Car's Cost? $15,000 and $10,000 and the difference is $5000

Next is How To Use Arguments and Parameters

Method Pass Arguments To Parameters

It's possible to pass one or more values to a method when calling the method. The values passed to a method are called arguments. To pass an argument, we place a value **inside** of the caller's parenthesis. For example, let's say registration is also a cost to a car. We write the registration cost 1500 inside coupe.calculateCost.

The variable that receives an argument is called a parameter. A parameter is declared inside the parenthesis following the method's name using a Data Type and variable name: int registration. Notice the error was removed from coupe.calculateCost to minivan.calculateCost. The method calculateCost (int) in the type Car is not applicable for the arguments. It's not applicable because the caller has no argument while the method is expecting a registration value. Let's add 500 as the registration cost for minivan. Next we add registration to the calculation with price and tax. When coupe and minivan calls calculateCost it will pass 1500 and 500 to registration inside the parenthesis, so registration is included with the cost calculation. Let's Run. The cost difference changed from $5000 to $6000.

This is how we pass arguments to parameters. We pass more than one argument to a parameter using a comma. Add int price comma int tax comma as parameters. Add price and tax as arguments: 14000 and 1000 for coupe, 9000 and 1000 for minivan. Remove coupe.price, coupe.tax, minivan.price, and minivan.tax

Run. We have the same results. Coupe is $16500, Minivan is $10500 with a difference of $6000.

That's it for the Relationship Between Classes, Objects, and Methods. If you are interested, download the presentation and code at https://tinyurl.com/Java-Classes-Objects-Methods