|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Section** | **Lecture** | **Slide** | **Slide Content** | **Transcript** |
| Introduction | Introduction to Building Automation Frameworks using Selenium and Java | About Me |  | Hi All! My Name is Rajesh Iyer. I am working in the Software Testing industry for over 10 years. I have architected and developed some sophisticated automation frameworks for some renowned clients in my organization. During my tenure I have majorly worked with Selenium RC and Selenium Webdriver, REST API testing and also Performance Testing. Though my educational background does not qualify me to be in the Software Testing industry, I consider myself lucky to get a start. And that was the only luck I had. After that it was all about continuous learning that has lead me to get into more challenging positions and also help some of my junior colleagues in their careers. |
| Why I created this course |  | Right since my school days, I have the passion for teaching. I have conducted trainings for my junior colleagues and also have provided mentorship to a handful of my colleagues. Now I want to widen my horizons and reach out to a lot of people. That is why I have created this course.  The other aspect is, I believe that the Automation is not just about knowing how to use Selenium Webdriver, but there is a science to developing a robust and sophisticated framework, that can be reused over and over to provide:   1. Better ROI for automation 2. Maintainability of the tests 3. Quick Turnaround Time in developing the tests   If you have these goals while developing a framework, your approach will be driven in this direction. The objective of this course is to help you achieve the above goals. |
| What to expect from this course |  | The first thing you can expect from this course is that there won’t be those boring theories about Java, Selenium or any of the different aspect of building frameworks. I don’t want the students of this course get all the information and when they come out of this course wonder “Aww! How do I apply this information now?”  You can expect a lot of interesting stories and real-world examples, that will help you understand every concept in detail and a few recommendations on how to apply the principles you will be learning throughout the course. So are you ready to get started? Let’s first learn Java or for some it will be brush up on Java. |
| Java Programming | Basic Java | Introduction |  | At the beginning of the course, I said I will not be giving the boring theories of Java or Selenium, but I thought it worthwhile to share the concept of how Java actually runs your code.  The program that you write is a .java file.  There is a java compiler or javac program which converts the .java file to a .class file which is a bytecode data.  The core of the Java is the JVM i.e Java Virtual Machine. This JVM is responsible for running your application. The JVM contains a Byte Code Verifier and Class Loader, which will check the byte codes of the class file. Once the check passes, there is another section called the Java Intrepreter and JIT compiler.  Java Interpreter reads the bytecode and performs the functions accordingly. The disadvantage of the Java Interpreter is that it slow as it has to look up the meaning of each byte during execution.  This is overcome by the Just-In-Time Compiler, which compiles the bytecode to the native code just before the execution.  Java can be installed with Java Development Kit (JDK) or Java Runtime Environment (JRE). Both come with JVM packaged into it. However, JDK has some additional Java APIs which will assist the developers in |
| Installation |  |  |
| Creating a class and object |  | Let’s think of a car. What are the different parts of a car? A car will have an engine, steering system, clutching system, braking system, accelerating system, mirrors, tyres, seats, body etc. All the cars have the same system, isn’t it? So how do you find same kinds of car on road. It is because they have been developed using the same blue print. This blue print is called a class. Once the class is created you can use it to create an object. Let’s say I want to create a Mercedes AMG 63 car. How do I do that. Here’s an example. I have created a car with different systems. If you look at this part, these are termed as fields of the class. They can also be considered the parameters of the class. This is called the methods in the class. Methods are majorly responsible to set or get the value of the fields or perform certain operations in the class like printing or converting. For example, if you look at getName method. It is responsible to give you the value of the name field. And the setName method is responsible to set the value of the name field. So our car blue print is ready. Now let’s create a Mercedes AMG 63 car. I have created different classes for different systems in the car and then assembled them in the car class. You can define each of the specification thoroughly and define your dream car. So here’s how we create a class and an object of that class. |
| Constructor |  | A class can have a method which has a same name as that of the class. This is called a constructor. So in case of a Car, the constructor method will have a name “Car”. You may notice that the class that I created does not have a constructor. In java, when you create a class, the java interpreter understands that there is a method Car() within that class which does nothing. OK, so now let’s create a Car constructor which will take the name of the car as a parameter. We will talk about this keyword later. So what this constructor does is, it stores the name that you will pass while creating an object of class Car. So, well modify our code with this new constructor. The output will remain the same. You can have multiple constructors for a class. This is called Class Overloading, which we will discuss in a later chapter. |
| static keyword |  | Now let’s say, we want to produce the car in our Car Factory and count the number of cars that were produced using the Car Class. So, this is where static variable is used. A static variable maintains its state irrespective of the instance of the object. So we need to make sure that every time a Car is created, it should be counted. So we will introduce a static variable noOfCars and every time a Car object is created, this variable will add by one. So let’s change the Car class as follows:   1. Add a static variable noOfCars 2. In the constructor, increment the noOfCars by 1 3. Add a static method getCarCount, to fetch the noOfCars value.   You may notice that I have created all the fields as private and have provided methods to access them. This is part of the encapsulation principle which we will learn in the next Chapter.  So let’s make these changes to our code.  Ok now let’s test it. To check if two cars were created, let’s create another car a Porsche. So we have two cars. Now let’s run the test  Fair enough we have two cars created and the noOfCars shows 2. |
| this keyword |  | You must have noticed the use of this keyword in the process of adding methods for the Car class. So what does “this” keyword mean?  The keyword “this” means whichever class instance you are in right now, access its method or fields and change its value. The operative word being “class instance” which means that it can be used for an instance. Meaning, you cannot access the static fields or methods using the “this” keyword.  The other usages of this is to invoke the current class constructor or a current class method or pass the current class instance as a parameter in a function or it can even return the current class instance from a method.  Some of the usages will be covered during the framework design. So for now just remember the different usages. |
| final keyword |  | In your Car Factory, you only manufacture cars which have a maximum speed limit of 100 as an example. And you do not want to change this maximum speed limit. This is when you can use the final keyword. Here’s how you achieve it. You will add a,   1. private final int maxSpeed and set it to 100.   Now, you will not be able to change anywhere in the methods where the Car class is invoked. Now you can only create a getMaxSpeed method, but you won’t be able to create a setMaxSpeed method. If you try to do it, it will give you a compile time error “cannot assign value to a final variable maxSpeed”. |
| Working with Conditional, Looping and Controlling statements and arrays |  | Let’s now work with Conditions and Looping and Controlling statements and Arrays. I will first define an array; so that we can build on them as well go along.  An array is a collection of objects of the same type. So it could be a collection of text or numbers or objects like car in our case. So how to instantiate an array of Cars? Here’s how we do it  Now, let’s work with the conditional statements if, if-else, if-else if.  In your Car Factory, you will provide texture design and black color to the car body if your customer is a Special Customer then this is how you will be doing it. Let’s introduce a variable customerType in the Car Factory class. So here’s how the code will be. Here we are accessing the Body of the Car and setting the color for it.  Next, let’s assume that if the customerType is an ordinary customer you will provide only White Color body car. Then here’s how you code will look like.  We will go one step further and introduce another customerType let’s say “Privileged” customer. For a privileged customer you are going to provide texture design for the car as well as give them a choice of color. So in this case, let’s introduce another variable, say “myColor”, which the user will specify. So assuming he specifies Black, we will create a Black Mercedes Car with texture design for that car. So we will use else-if between the if and else statements to achieve this as follows.  Now let’s work with loops- for and while  As a worker, now you are given a register containing 5 customers and you need to create a Mercedes car for them. Each of those customers are different: 1 is a privileged customer, 1 is a special customer and remaining 3 are ordinary customers. Now you need to roll out 5 cars based on the specifications. The privileged customer wants an Ivory Silver car. So what do you do here?  You will add 2 Arrays one to store the customer type, the other to store the created cars. Let’s first achieve it using a simple for loop. |
| Creating an Enum class |  |  |
| Creating an abstract class |  |  |
| Creating an interface and implementation |  |  |
| File Handling |  |  |
| String and String Manipulations |  |  |
| Regular Expression handling |  |  |
| OOPS Concepts | Inheritance |  |  |
| Polymorphism |  |  |
| Abstraction |  |  |
| Encapsulation |  |  |
| Advanced Java | Collections framework |  |  |
| Generics |  |  |
| Reflections |  |  |
| SOLID principles | Introduction |  |  |
| Single Responsibility Principle |  |  |
| Open for extension, Closed for modification Principle |  |  |
| Liskov Substitution Principle |  |  |
| Interface Segregation Principle |  |  |
| Dependency Inversion Principle |  |  |
| Design Patterns |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| Frameworks | Types of Automation Frameworks | Data driven Framework |  |  |
| Keyword driven Framework |  |  |
| Behaviour Driven Framework |  |  |
| Hybrid Framework |  |  |
| Different components of an Automation Framework | Selenium Webdriver |  |  |
| TestNG Framework |  |  |
| Maven |  |  |
| Dataproviders |  |  |
| Custom Page Creation |  |  |
| Object Repository Management |  |  |
| Reporting |  |  |
| Continuous Integration with Jenkins |  |  |
| Selenium Webdriver | Introduction |  |  |  |
| How does Selenium Work? |  |  |  |
| Different implementations of Webdriver |  |  |  |
| Working with UI Components Using Webdriver |  |  |  |
| Handling Waits in Selenium Webdriver |  |  |  |
| Taking Screenshots using Remote Webdriver |  |  |  |
| Working with Actions API |  |  |  |
| Using JavascriptExecutor |  |  |  |
| Conclusion |  |  |  |
| Maven | Introduction |  |  |  |
| What is Maven? |  |  |  |
| What is pom.xml? |  |  |  |
| How to add dependencies to your project |  |  |  |
| How to add plugins to your project |  |  |  |
| Conclusion |  |  |  |
| TestNG Framework | Introduction |  |  |  |
| What is TestNG |  |  |  |
| Difference between TestNG and Junit |  |  |  |
| Different annotations in TestNG |  |  |  |
| Working with Dataprovider annotation |  |  |  |
| Dependent tests using TestNG |  |  |  |
| Grouping of tests in TestNG |  |  |  |
| Running tests in parallel |  |  |  |
| Writing Sample scripts with TestNG |  |  |  |
| Dataproviders | How to manage data in your framework? |  |  |  |
| Understanding the Apache POI API |  |  |  |
| Creating an utility to read an excel file using Apache POI API |  |  |  |
| How to pass data to the TestNG test |  |  |  |
| Object Repository Management | Identifying Web Elements |  |  |  |
| Advanced CSS and XPATH identifiers |  |  |  |
| Using Firebug and Firepath to identify elements |  |  |  |
| Creating Custom Page Methods | What is a Page Factory Pattern? |  |  |  |
| Creating a Page using Page Factory pattern |  |  |  |
| Creating Object Repository for a Page |  |  |  |
| Reporting and Logging | How to report the execution status |  |  |  |
| Adding logs to your tests |  |  |  |
| Integrating the reports using ATU reports |  |  |  |
| Conclusion | Final Thoughts |  |  |  |