15. Demonstrate using listeners in Selenium.

Step 1: Using iTestListener:

- Listener is an interface that modifies the TestNG behavior.
- Listener listens to the event defined in the Selenium and behaves accordingly.
- It is used in Selenium by implementing Listeners Interface.
- iTestListener is used in Selenium to generate logs or customize the TestNG reports.
- iTestListener has the following methods:
    a. **onStart:** onStart method is called when any Test starts.
    b. **onTestSuccess:** onTestSuccess method is called on the success of any Test.
    c. **onTestFailure:** onTestFailure method is called on the failure of any test.
    d. **onTestSkipped:** onTestSkipped method is called when any test gets skipped.
    e. **onTestFailedButWithinSuccessPercentage:** onTestFailedButWithinSuccessPercentage method is called each time Test fails but within the success percentage.
    f. **onFinish:** onFinish method is called after all the tests are executed.
- Open Eclipse and create a Project.
- Create a Listener class that will implement iTestListener.
- The code in Eclipse will look like:

```
package test.testing;

import org.testng.ITestContext;
import org.testng.ITestListener;
import org.testng.ITestResult;

public class ListenersTest implements ITestListener {

	public void onFinish(ITestContext Result) {
	   System.out.println(Result.getName()+"case finished");

	}

	public void onStart(ITestContext Result) {
	   // TODO Auto-generated method stub

	}

	public void onTestFailedButWithinSuccessPercentage(ITestResult Result) {
	   // TODO Auto-generated method stub

	}

	public void onTestFailure(ITestResult Result) {
	   // TODO Auto-generated method stub
	   System.out.println("The name of the testcase failed is :"+Result.getName());
	}
```

```
        public void onTestSkipped(ITestResult Result) {
            // TODO Auto-generated method stub
            System.out.println("The name of the testcase Skipped is :"+Result.getName());
        }

        public void onTestStart(ITestResult Result) {
            // TODO Auto-generated method stub
            System.out.println(Result.getName()+" test case started");
        }

        public void onTestSuccess(ITestResult Result) {
            // TODO Auto-generated method stub
            System.out.println("The name of the testcase passed is :"+Result.getName());
        }

}
```

- Create a Java class and implement a Listener class in this.
- The code in Eclipse will look like:

```
package test.testing;

import org.openqa.selenium.By;
import org.testng.Assert;
import org.testng.annotations.Listeners;
import org.testng.annotations.Test;

@Listeners(test.testing.ListenersTest.class)

    public class Home extends Baseclass {

          @Test
          public void clickOnCategory()
          {
           driver.findElement(By.xpath("//div[@id='navigation']/a[1]")).click();
           System.out.println("Clicked on links");
          }

}
```

Step 2: Running the code

- Run the code through Eclipse.

Step 3: Pushing the code to your GitHub repositories

Open your command prompt and navigate to the folder where you have created your files.

cd <folder path>

Initialize your repository using the following command:

git init

Add all the files to your git repository using the following command:

git add .

Commit the changes using the following command:

git commit . -m “Changes have been committed.”

Push the files to the folder you initially created using the following command:

git push -u origin master