1.

import java.util.Scanner;

public class IT\_Calculator {  
    private double taxableIncome;  
    private double taxPaidPerYear;  
    private double taxPayablePerMonth

    public IT\_Calculator() {  
        this.taxableIncome = 0;  
        this.taxPaidPerYear = 0;  
        this.taxPayablePerMonth = 0;  
    }

    public void calculateTax() {  
        if (taxableIncome <= 25000) {  
            taxPaidPerYear = 0;  
        } else if (taxableIncome <= 50000) {  
            taxPaidPerYear = taxableIncome \* 0.10;  
        } else if (taxableIncome <= 75000) {  
            taxPaidPerYear = taxableIncome \* 0.20;  
        } else {  
            taxPaidPerYear = taxableIncome \* 0.30;  
        }

        taxPayablePerMonth = taxPaidPerYear / 12;  
    }  
    public void readTaxableIncome() {  
        Scanner scanner = new Scanner(System.in);  
        System.out.print("Enter Taxable Income: $");  
        taxableIncome = scanner.nextDouble();  
    }  
    public void displayResults() {  
        System.out.println("Tax Paid in a Year: $" + taxPaidPerYear);  
        System.out.println("Tax Payable in a Month: $" + taxPayablePerMonth);  
    }

    public static void main(String[] args) {  
        IT\_Calculator calculator = new IT\_Calculator();  
        calculator.readTaxableIncome();  
        calculator.calculateTax();  
        calculator.displayResults();  
    }  
}

2.

import math

class CircleComp:  
    def \_\_init\_\_(self):  
        self.radius = 0  
        self.diameter = 0  
        self.circumference = 0  
        self.area = 0

    def calculate(self):  
        self.diameter = self.radius \* 2.0  
        self.circumference = 2 \* math.pi \* self.radius  
        self.area = math.pi \* self.radius \* self.diameter

    def read\_radius(self):  
        self.radius = float(input("Enter the radius of the circle: "))

    def display\_results(self):  
        print(f"Diameter: {self.diameter}")  
        print(f"Circumference: {self.circumference}")  
        print(f"Area: {self.area}")

if \_\_name\_\_ == "\_\_main\_\_":  
    circle = CircleComp()  
    circle.read\_radius()  
    circle.calculate()  
    circle.display\_results()

3a.

//Launch a browser

WebDriver driver = new ChromeDriver();

driver.navigate().to("https://www.automationanywhere.com/");

WebElement element = driver.findElement(By.xpath(“ //a[text()="Products"]”));

Actions actions = new Actions(driver); actions.moveToElement(element).perform(); actions.click(element).perform();

3b.

driver.findElement(By.xpath("(//a[text()="

Process Discovery "]//parent::li[@class="

coh - menu - list - item js - coh - menu - item "])[1]")).click();

3c.

String strUrl = driver.getCurrentUrl();

System.out.println(“Page Is Navigated To: ”+strUrl);

4a.

//Launch a browser

WebDriver driver = new ChromeDriver();

driver.navigate().to("https://www.automationanywhere.com/");

driver.findElement(By.xpath("//a[@title="

Request demo "]")).click();

4b.

String strUrl = driver.getCurrentUrl();

System.out.println(“Page Is Navigated To: ”+strUrl);

4c.

// return first label name

String firstname = driver.findElement(By.xpath("//label[@id="

LblFirstName "]").getText(); String lastname = driver.findElement(By.xpath("//label[@id="

LblFirstName "]").getText();

if (firstname.equals( \* First Name) && lastname.equals( \* Last Name)) {

System.out.println("\*\*\*Label names are present as expected\*\*\*");

}

5.

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.testng.annotations.AfterMethod;

import org.testng.annotations.BeforeMethod;

import org.testng.annotations.Test;

public class program3 {

WebDriver driver;

@BeforeClass

public void setUp() {

System.setProperty("webdriver.chrome.driver", "path/to/chromedriver.exe");

driver = new ChromeDriver();

driver.manage().window().maximize();

}

@Test

public void launchBrowserTest() {

driver.get("https://www.automationanywhere.com");

WebElement element = driver.findElement(By.xpath(“ //a[text()="Products"]”));

Actions actions = new Actions(driver);

actions.moveToElement(element).perform();

actions.click(element).perform(); driver.findElement(By.xpath("(//a[text()=" Process Discovery "]//parent::li[@class="coh - menu - list - item js - coh - menu - item "])[1]")).click();

String strUrl = driver.getCurrentUrl();

System.out.println(“Page Is Navigated To: ”+strUrl);

}

@AfterClass

public void tearDown() {

driver.quit();

}

}

6.

My-maven-project  
|-- src  
|   |-- main  
|   |   `-- java  
|   |       `-- com  
|   |           `-- example  
|   |               `-- IT\_Calculator.java  
|   |               `-- CircleComp.java  
|   `-- test  
|       `-- java  
|           `-- com  
|               `-- example  
|                   `-- IT\_CalculatorTest.java  
|                   `-- CircleCompTest.java  
|-- pom.xml

2. Created Java Classes

Implement the IT\_Calculator and CircleComp classes in the src/main/java directory.

3.Created Test Classes

Implement the test classes (IT\_CalculatorTest and CircleCompTest) in the src/test/java directory.

4.Created pom.xml

Create a pom.xml file in the root of your project with the following content:

<project xmlns="[http://maven.apache.org/POM/4.0.0"](http://maven.apache.org/POM/4.0.0%22)  
         xmlns:xsi="[http://www.w3.org/2001/XMLSchema-instance"](http://www.w3.org/2001/XMLSchema-instance%22)  
         xsi:schemaLocation="<http://maven.apache.org/POM/4.0.0> [https://maven.apache.org/xsd/maven-4.0.0.xsd">](https://maven.apache.org/xsd/maven-4.0.0.xsd%22%3e)  
<modelVersion>4.0.0</modelVersion>

    <groupId>com.example</groupId>  
<artifactId>your-maven-project</artifactId>  
<version>1.0-SNAPSHOT</version>

    <properties>  
<maven.compiler.source>1.8</maven.compiler.source>  
<maven.compiler.target>1.8</maven.compiler.target>  
</properties>

    <dependencies>  
<!-- Add your dependencies here -->  
<!-- For example, JUnit -->  
<dependency>  
<groupId>junit</groupId>  
<artifactId>junit</artifactId>  
<version>4.12</version>  
<scope>test</scope>  
</dependency>  
<!-- Add other dependencies as needed -->  
</dependencies>

    <build>  
<plugins>  
<plugin>  
<groupId>org.apache.maven.plugins</groupId>  
<artifactId>maven-surefire-plugin</artifactId>  
<version>3.0.0-M5</version>  
<configuration>  
<includes>  
<include>\*\*/\*Test.java</include>  
</includes>  
</configuration>  
</plugin>  
</plugins>  
</build>  
</project>

5. Run Tests

Run the following command in the terminal to build and run tests using Maven:

mvn clean test

7.

Mkdir C:\Users\VAMSHIM\Documents\Techademy\ comprehensivebasic3

Cd C:\Users\VAMSHIM\Documents\Techademy\ comprehensivebasic3

Git init

Git branch comprehensive\_basic\_3

Git checkout comprehensive\_basic\_3

<Code>

Git commit

8.

{

"info": {

"\_postman\_id": "10edc0f1-7816-4bf5-9415-da3e5d217841",

"name": "API",

"schema": "https://schema.getpostman.com/json/collection/v2.1.0/collection.json",

"\_exporter\_id": "23293308"

},

"item": [

{

"name": "positiveScenario",

"event": [

{

"listen": "test",

"script": {

"exec": [

"pm.test(\"Status code is 200\", function () {\r",

" pm.response.to.have.status(200);\r",

"});\r",

"\r",

""

],

"type": "text/javascript"

}

}

],

"protocolProfileBehavior": {

"disableBodyPruning": true

},

"request": {

"method": "GET",

"header": [],

"body": {

"mode": "raw",

"raw": "",

"options": {

"raw": {

"language": "json"

}

}

},

"url": {

"raw": "https://restcountries.com/v2/name/india",

"protocol": "https",

"host": [

"restcountries",

"com"

],

"path": [

"v2",

"name",

"india"

]

}

},

"response": []

},

{

"name": "negativeScenario",

"event": [

{

"listen": "test",

"script": {

"exec": [

"pm.test(\"Status code is 404\", function () {\r",

" pm.response.to.have.status(404);\r",

"});"

],

"type": "text/javascript"

}

}

],

"request": {

"method": "GET",

"header": [],

"url": {

"raw": "https://restcountries.com/v2/name/delhi",

"protocol": "https",

"host": [

"restcountries",

"com"

],

"path": [

"v2",

"name",

"delhi"

]

}

},

"response": []

}

]

}

package com.aspireapiintegration;

import io.restassured.\*;

import io.restassured.matcher.RestAssuredMatchers;

import io.restassured.response.Response;

import org.hamcrest.Matcher;

import org.testng.annotations.Test;

public class APIautomation {

@Test

public static void positiveScenario() {

RestAssured.baseURI = "";

String statusBody = "";

Response response =RestAssured.given()

.header("Content-type", "application/json")

.when()

.get("https://restcountries.com/v2/name/india")

.then().assertThat().statusCode(200)

.extract().response();

System.out.println(response.asString());

}

@Test

public static void negitiveSCenario() {

String statusBody = "";

Response response =RestAssured.given()

.header("Content-type", "application/json")

.when()

.get("https://restcountries.com/v2/name/delhi")

.then().assertThat().statusCode(400)

.extract().response();

System.out.println(response.asString());

}

}

9.

10.

import pytest  
from selenium import webdriver  
from selenium.webdriver.common.by import By  
from selenium.webdriver.common.action\_chains import ActionChains

@pytest.fixture  
def browser():  
    driver = webdriver.Chrome()  
    yield driver  
    driver.quit()

def test\_process\_discovery\_navigation(browser):  
    # Open the website  
    browser.get("[https://www.automationanywhere.com/")](https://www.automationanywhere.com/%22))

    # Locate "Products" in the navigation bar  
    products\_menu = browser.find\_element(By.XPATH, "//a[text()='Products']")

    # Hover over "Products" using ActionChains  
    action\_chains = ActionChains(browser)  
    action\_chains.move\_to\_element(products\_menu).perform()

    # Locate and click on "Process Discovery"  
    process\_discovery\_link = browser.find\_element(By.XPATH, "//a[text()='Process Discovery']")  
    process\_discovery\_link.click()

    # Verify the current URL matches the expected URL  
    expected\_url = "[https://www.automationanywhere.com/products/process-discovery"](https://www.automationanywhere.com/products/process-discovery%22)  
    assert browser.current\_url == expected\_url, f"Expected URL: {expected\_url}, Actual URL: {browser.current\_url}"

if \_\_name\_\_ == "\_\_main\_\_":  
    pytest.main()

GITHUB URL: https://github.com/vamshinani2012/comprehensivebasic3/upload/comprehensive\_basic\_3