Task 1: Form Element Interaction and Data Extraction

Topic: Advanced Element Interaction - Handling Tables and Form Elements

Description: Write a Selenium script to extract data from a table and interact with form elements like checkboxes and radio buttons on a sample web page.

Steps:

Launch the web browser and navigate to a predefined web page with a table and form elements.

Use Selenium's WebDriver to locate the table element and retrieve the text from a specific cell.

Locate a checkbox and a radio button on the form and use Selenium commands to select and unselect the checkbox, and select the radio button.

Print out the extracted data and the status of checkbox and radio button selections to the console.

```
package Day3;


import org.openqa.selenium.By;
import org.openqa.selenium.WebElement;


import selenium_programs.BrowserOperations;


public class Task1 extends BrowserOperations {


	public static void main(String[] args) throws InterruptedException {
		selectBrowser("chrome");
		driver.get("https://rahulshettyacademy.com/AutomationPractice/");


		WebElement table=driver.findElement(By.cssSelector(".tableFixHead"));
		WebElement cell=table.findElement(By.xpath(".//tr[4]/td[3]"));
		String Text=cell.getText();
		System.out.println(Text);
		WebElement checkbox=driver.findElement(By.id("checkBoxOption2"));
		if(!checkbox.isSelected()) {
			checkbox.click();
			Thread.sleep(2000);
		}
		System.out.println("Checkbox selected"+ checkbox.isSelected());
```

```
		WebElement radioButton=driver.findElement(By.xpath("//input[@value='radio2']"));
		if(!radioButton.isSelected()) {
			radioButton.click();
			Thread.sleep(2000);
		}
		System.out.println("Radio button selected: "+radioButton.isSelected());
		Thread.sleep(2000);
		driver.close();
	}
}
```

Task 2: Synchronization and Alert Handling

Topic: Selenium Synchronization Strategies and Handling Alerts

Description: Implement an explicit wait to synchronize a web element's presence and handle a simple JavaScript alert on the page.

Steps:

Open a web browser and navigate to a predefined web page that triggers a JavaScript alert after a button click.

Implement an explicit wait using WebDriverWait to wait for the button to become clickable.

Click the button to trigger the alert and then implement another explicit wait to ensure the alert is present.

Use Selenium's Alert interface to accept the alert and then print a confirmation to the console.

```
package Day3;


import java.time.Duration;


import org.openqa.selenium.Alert;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;


public class Task2 {


	public static void main(String[] args)  {
		selectBrowser("chrome");
		driver.get("https://rahulshettyacademy.com/AutomationPractice/");


		driver.manage().window().maximize();
		driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));
		tr
				WebDriverWait wait=new WebDriverWait(driver,Duration.ofSeconds(40));
				WebElement
button=wait.until(ExpectedConditions.elementToBeClickable(By.id("alertbtn")));
				button.click();
				wait.until(ExpectedConditions.alertIsPresent());
```

```
                    Alert alert=driver.switchTo().alert();
                    Thread.sleep(2000);
                alert.accept();



                    System.out.println("Alert successfully");
            }catch(Exception e) {
                e.printStackTrace();
            }finally {
                Driver.cloase();
        }
        }
}
```