WHITE DATA SYSTEMS INTERVIEW QUESTIONS

1ST ROUND:

1. **How to read input data's from excel?**

   To read data from Excel:

```
public class HotelSearchTest {
        public static void main(String[] args) {


                List<HashMap<String, String>> mapDatasList = new ArrayList();
                try {
                        File excelLocaltion = new File("./Excel/Adactin.xlsx");

                        String sheetName = "Adact";

                        FileInputStream f = new FileInputStream(
                                        excelLocaltion.getAbsolutePath());
                        Workbook w = new XSSFWorkbook(f);
                        Sheet sheet = w.getSheet(sheetName);
                        Row headerRow = sheet.getRow(0);
                        for (int i = 0; i < sheet.getPhysicalNumberOfRows(); i++) {
                                Row currentRow = sheet.getRow(i);
                        HashMap<String, String> mapDatas = new HashMap<String,
String>();
                        for (int j = 0; j < headerRow.getPhysicalNumberOfCells(); j++) {
                                        Cell currentCell = currentRow.getCell(j);

                                        switch (currentCell.getCellType()) {
                                        case Cell.CELL_TYPE_STRING:

        mapDatas.put(headerRow.getCell(j).getStringCellValue(),

        currentCell.getStringCellValue());
                                                        break;
                                        case Cell.CELL_TYPE_NUMERIC:

        mapDatas.put(headerRow.getCell(j).getStringCellValue(),
                                                                        String.valueOf(currentCell

        .getNumericCellValue()));

                                                        break;

                                        }
```

```
                }

                mapDatasList.add(mapDatas);
            }

            // System.out.println(mapDatasList);
            String s = mapDatasList.get(1).get("Username");
            String s1 = mapDatasList.get(1).get("Password");
            System.out.println(s);
            System.out.println(s1);

        } catch (Throwable e) {
            e.printStackTrace();
        }

    }

}
```

2. **Explain about implicity wait and explicity wait?**

Implicit wait:

- It applicable for all elements
- The implicit wait will tell to the web driver to wait for certain amount of time before it throws a "No Such Element Exception"
- Once we set the time, web driver will wait for that time before throwing an exception.

Syntax:

driver.manage().timeouts().implicitlyWait(time, TimeUnit.SECONDS);

Ex:

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

Explicit wait:

- The explicit wait is used to tell the Web Driver to wait for particular element to certain conditions (Expected Conditions) or the maximum time exceeded before throwing an exception.
- The explicit wait is an intelligent kind of wait, but it can be applied only for specified elements.

- Explicit wait gives better options than an implicit.
- Once we declare explicit wait we have to use "ExpectedCondtions" or we can configure how frequently we want to check the condition using Fluent Wait.

Syntax:

WebDriverWait wait = new WebDriverWait(driver,time);

3. **How to switch one window to another window?**

- Using window handling method we can switch

```
public class Dummy9{

	public static void main(String[] args) throws InterruptedException {
		System.setProperty("webdriver.gecko.driver",

"C:/Users/siva/workspace/Selenium/driver/geckodriver.exe");
		WebDriver driver = new FirefoxDriver();
		driver.get("https://www.hdfcbank.com/");
		driver.findElement(By.xpath(".//*[@id='cee_closeBtn']/img")).click();
		String parentWindowId = driver.getWindowHandle();
		System.out.println("Parent Window ID:" + parentWindowId);
		driver.findElement(By.id("loginsubmit")).click();
		Set<String> allWindowId = driver.getWindowHandles();
		for (String x : allWindowId) {


			if (!parentWindowId.equals(x)) {
				System.out.println("Child Window ID:" + x);
				driver.switchTo().window(x);
				Thread.sleep(3000);

driver.findElement(By.xpath("html/body/div[4]/div[2]/div[1]/a"))
                                                .click();
				driver.manage().window().maximize();
				Thread.sleep(2000);
				driver.switchTo().defaultContent();
			}
		}
		Thread.sleep(3000);
		driver.quit();

	}
```

4. **How will you take screenshot?**

```
public class Dummy {
        public static void main(String[] args) throws IOException {
        System.setProperty("webdriver.gecko.driver",
        "C:/Users/siva/workspace/Selenium/driver/geckodriver.exe");
        WebDriver driver = new FirefoxDriver();
        driver.get("https://www.facebook.com/");
        driver.findElement(By.id("email")).sendKeys("Hello");
        //screenshot declaration
        TakesScreenshot tk=(TakesScreenshot) driver;
        File f = tk.getScreenshotAs(OutputType.FILE);
        File f1=new File("F:/facebook.png");
        FileUtils.copyFile(f,f1 );
        }

        }
```

5. **Explain about JDBC connection?**

   JDBC Connection Steps:

   1. Load driver
   2. Connect to database
   3. Write sql query
   4. Prepare the statement
   5. Set the values
   6. Execute query

2nd round (Manager round)

1. **What is the use of maven?**

   - Project buid tool
   - If one person (D) download jar file, it will automatically updated in maven repository. Then anyone can use

   - Maven is used to define project structure, dependencies, build, and test management.
   - Using pom.xml(Maven) you can configure dependencies needed for building testing and running code.

2. **Explain about Jenkins?**

   - Jenkins is an leading open source continuous integration server built with Java. It is used to build and test software projects continuously making it easier to integrate changes to the project.
   - Support for scheduled builds & automation test execution.

3. Explain about Junit and their annotations?

- **JUnit** is an open source Unit Testing Framework for JAVA.
- It is useful for Java Developers/testers to write and run repeatable tests

Annotations:

@Test: Annotation lets the system know that the method annotated as @Test is a test method. There can be multiple test methods in a single test script.

2. **@Before:** Method annotated as @Before lets the system know that this method shall be executed every time before each of the test method.

3. **@After:** Method annotated as @After lets the system know that this method shall be executed every time after each of the test method.

4. **@BeforeClass:** Method annotated as @BeforeClass lets the system know that this method shall be executed once before any of the test method.

5. **@AfterClass:** Method annotated as @AfterClass lets the system know that this method shall be executed once after any of the test method.

6. **@Ignore:** Method annotated as @Ignore lets the system know that this method shall not be executed.