Robot Class in Java

Why Robot Class?

In certain Selenium Automation Tests, there is a need to control keyboard or mouse action to interact with OS windows like ***Download pop-up***, ***Alerts***, ***Print Pop-ups***, etc. or native Operation System applications like ***Notepad, Skype, Calculator***, etc.

Selenium WebDriver cannot handle these OS pop-ups / applications. In Java version 1.3 Robot Class was introduced. Robot Class can handle OS pop-ups/applications. Robot class is present in AWT package of JDK.

Advantages -

1. Robot Class can simulate Keyboard and Mouse Event
2. Robot Class can help in upload/download of files when using selenium web driver
3. Robot Class can easily be integrated with current automation framework (keyword, data-driven or hybrid)

Robot Class internal methods and usage

Robot Class methods can be used to interact with keyboard / mouse events while doing browser automation. Alternatively **AutoIT** can be used, but its drawback is that it generates an executable file (exe) which will only work on windows, so it is not a good option to use.

Some usual methods of Robot Class during web automation -

Sr. No	Method	Description
1	***keyPress***()	press down arrow key of Keyboard
2	***mousePress***()	press the right click of your mouse
3	***mouseMove***()	move mouse pointer to the specified X & Y coordinates
4	***keyRelease***()	release down arrow key of Keyboard
5	***mouseRelease***()	release the right click of your mouse

Sample code to automate common use cases using Robot Class –

1. Creating a Robot class Object :

```
Robot robo = new Robot();
```

2. To press **Down ⬇ Arrow Key** of Keyboard :

```
robo.keyPress(KeyEvent.VK_DOWN);
```

3. To press **TAB key** of Keyboard

```
robo.keyPress(KeyEvent.VK_TAB);
```

4. To press **Enter key** of keyboard

```
robo.keyPress(KeyEvent.VK_ENTER);
```

5. To move **Mouse Pointer** of mouse

```
robo.mouseMove(630,420) ; // x & y offsets (Co-ordinates)
```

6. To press **left Mouse Button** of mouse

```
robo.mousePress(InputEvent.BUTTON1_DOWN_MASK);
```

7. To release **left Mouse Button** of mouse

```
robo.mouseRelease(InputEvent.BUTTON1_DOWN_MASK);
```

Disadvantages of Robot class –

1. Keyword / mouse event will only ***works on current instance of Window***. E.g. suppose a code is performing any robot class event, and during the code execution user has moved to some other screen then keyword / mouse event will occur on that screen.
2. Most of the methods like **mouseMove** is screen resolution dependent so there might be a chance that code working on one machine might not work on other.

Example Code –

```
public class demo {
        public static void main(String[] args) throws InterruptedException, AWTException {

        System.setProperty("webdriver.chrome.driver","F:\\chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
        driver.manage().window().maximize();
        driver.get("https://www.facebook.com/");

        WebElement emailOrPhone = driver.findElement(By.xpath("//input[@type='email']"));
        WebElement password  = driver.findElement(By.xpath("//input[@name='pass']"));
        Actions act = new Actions(driver) ;
        act.sendKeys(emailOrPhone, "pqrxyz").build().perform();
        act.sendKeys(password, "abcde").build().perform();

        Robot robo = new Robot();
        robo.keyPress(KeyEvent.VK_ENTER); // pressing Enter Key
        robo.keyRelease(KeyEvent.VK_ENTER); // releasing  Enter Key
    }
}
```

Selenium Actions or Java AWT Robot?

Selenium uses WebDriver API and sends commands to browser driver to perform actions (through the "**JSON wire protocol**"). However, Java AWT Robot uses native system events to control ***keyboard*** and ***mouse*** operations. If your aim is, browser automation, then technically you don't need to use Robot framework because the ***Actions***() functionality provided by Selenium is more than enough. But of course, there are cases when browser or native ***OS popup*** comes up like uploading/ downloading a file. This can also be solved using Robot framework- though generally there are selenium-specific solutions/workarounds that can help avoiding using Robot. The key idea of these workarounds is "since we cannot control the popups, just don't let them open".

For instance, while downloading a file in Firefox, you will get a popup asking you to choose a location and filename where it should get saved. This kind a situation cannot be manipulated using selenium. But, however what you can do is, let Firefox know for certain file types, where to save the downloads automatically, without handling the popup.

Robot class is a **java based** utility which emulates the keyboard and mouse actions. The ***Actions class*** is ***selenium based*** utility, user-facing API for emulating complex user **action** events

Simply we can say that **Java AWT Robot class is useful to handle OS windows like AutoIT whereas Action class is not meant to Automate OS windows**.