# Robot Class in Java - An Introduction

we will understand an interesting  Java class called the ***Robot class***. The Selenium-based test automation frameworks widely use it for simulating the keyboard and mouse events. In this article, we are going to learn:-

* What is a Robot class, its methods and how to use them?
* Its Advantages and Limitations

## What is an exciting thing about Robot class?

Selenium scripts use Robot class for automating the browser and desktop pop-ups, but the exciting thing is this class is not part of ***org.openqa.selenium*** package of ***Web Driver API***.

Then from where does this class come?

It doesn’t reside in Web Driver API; it is part of the [***Java API awt***](https://docs.oracle.com/javase/10/docs/api/java/awt/Robot.html) package.

### ***Need for Robot class***

In the [***Actions class***](https://docs.oracle.com/javase/10/docs/api/java/awt/Robot.html) tutorials series, we have already seen various methods for handling keyboard and mouse events. Now, Actions class handles cases that web driver commands can't handle. Therefore, one may think, why do we need this package which is not even a part of WebDriver API? Answer to lies in the scenarios like below:

* When the user needs to handle alert pop-ups on a webpage, or
* User needs to enter text on the pop-ups with a combination of modifier keys such as Alt, Shift, etc.

***Here, the pop-ups/alerts are Windows pop-ups instead of Webpage pop-ups.***

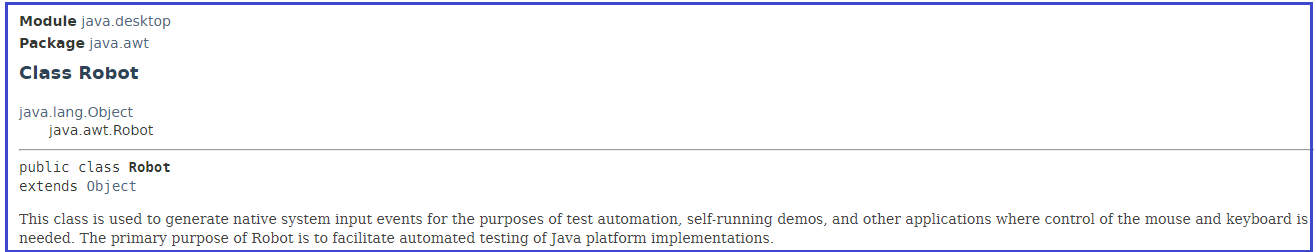
We know that to perform any action on a web element, we need a locator for the element. But Windows pop-ups don't have any locators, as they are not part of the webpage, they are native OS pop-ups. To handle such pop-ups we need the Robot class.

For instance, if you are trying to download Email Attachment, Windows pop-up, 'Save Attachment' prompts to specify Download Location, appears. It is nothing but a native OS pop-up.

One can't use Action class methods to handle keyboard/mouse events on Desktop windows pop-up. The reason being, Actions class methods need WebElement objects to perform actions. Whereas for Desktop windows pop-up, no locator exists, and the same can be verified using browser developer tools. Therefore, to handle such scenarios, the Robot class is used.

## What is a Robot class?

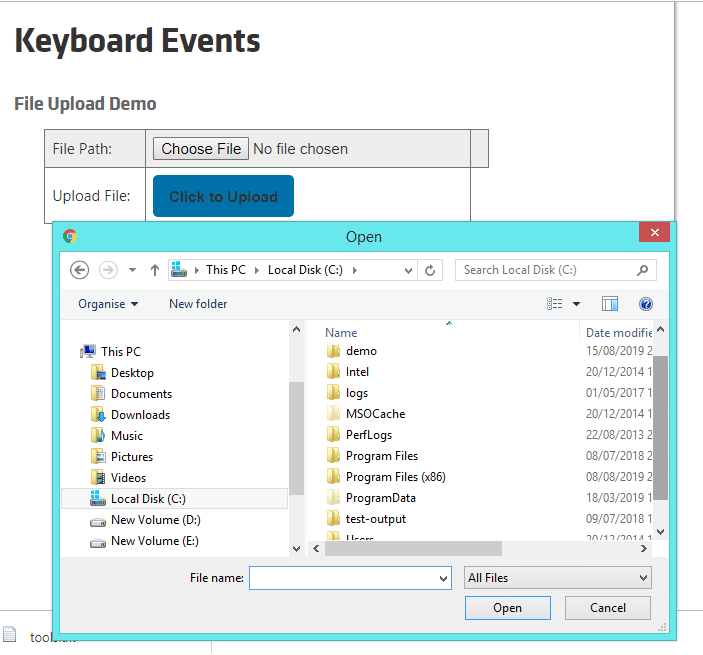
As per the class description, this class is used to generate native system input events. This class uses native system events to control the mouse and keyboard.



It differs from Selenium which uses the WebDriver API and invokes commands to a browser to perform actions.

### ***How to use Robot class methods?***

Let us understand the use of Robot class methods with the help of an example on ToolsQa's demo site [***https://demoqa.com/keyboard-events/***](https://demoqa.com/keyboard-events/). Consider the scenario where a user wants to upload a file.



So, the user clicks on the ***Choose File*** button first to enter the file path to be uploaded. Here, pop-up to select file is Desktop Windows appears. Let's use the Robot class methods to enter the file path.

***1.Import package:*** Robot class has to import first, to use.

import java.awt.Robot;

***2. Instantiate***: A robot class object is needed to invoke its methods. So, let’s instantiate the Robot class.

Robot robot = new Robot();

***3. Invoke method:*** Now invoke the required method on robot object.

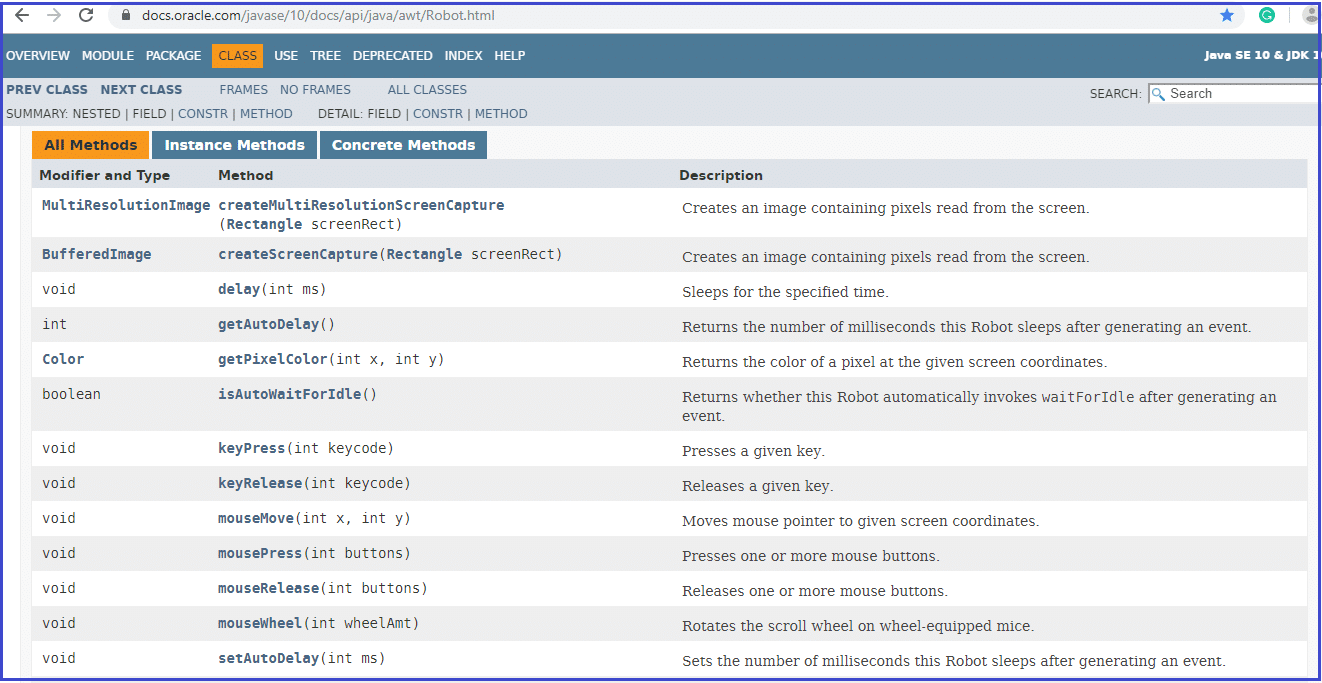
robot.<required\_method>();

The Robot class provides various methods for handling mouse and keyboard events. For entering the file path, we would need a method to enter text. So, a method to be used here is keyPress(int keycode) which presses a given key keycode.

robot.keyPress(keycode);

### ***Methods in Robot class:***

As you can see ***[java.awt.Robot](https://docs.oracle.com/javase/10/docs/api/java/awt/Robot.html)*** class provides various methods needed for controlling mouse and keyboard.



But, we will only cover a few of the commonly used methods for browser test automation.

Following are some of the methods commonly used in browser test automation:

***Keyboard methods:***

* keyPress(int keycode): This method presses a given key. For Example, keyPress(KeyEvent.VK\_SHIFT) method presses ''SHIFT' key
* keyRelease(int keycode): This method releases a given key. For Example, keyRelease(KeyEvent.VK\_SHIFT) method releases ''***SHIFT***" key

***Mouse Methods:***

* mousePress(int buttons): This method presses one or more mouse buttons.For Example, mousePress(InputEvent.BUTTON1\_DOWN\_MASK) method is used left click mouse button
* mouseRelease(int buttons): This method releases one or more mouse buttons. For Example, mouseRelease(InputEvent.BUTTON1\_DOWN\_MASK) method is used to release the left mouse button click
* mouseMove(int x, int y): This method moves the mouse pointer to given screen coordinates specified by x and y values. For Example, mouseMove(100, 50) will move the mouse pointer to the x coordinate 100 and y coordinate 50 on the screen.

You can refer to [***Robot***](https://docs.oracle.com/javase/10/docs/api/java/awt/Robot.html) class java documents to explore more methods.

### ***Advantages:***

Here are some of the benefits:

* It provides control over the Keyboard as well as Mouse events.
* It offers a way to handle an interaction with Operating system pop-ups support of which is not possible with Selenium Web Driver API.
* Robot class is especially useful in managing file upload/download actions by interacting with OS pop-ups.
* It is easy to consume in the java Selenium scripts as this class is part of Java package.

### ***Limitations:***

But methods mentioned above to control Keyboard and Mouse have some limitations also. Consider some of those following limitations while writing automation scripts:

* Most of the methods like mouseMove are dependent on screen resolution, so, the method may perform differently on different screens.
* This class acts only on the window in focus, so the behavior may differ when multiple windows open.
* Switching between different frames or windows is difficult with Robot methods.

# Robot Class Keyboard Events

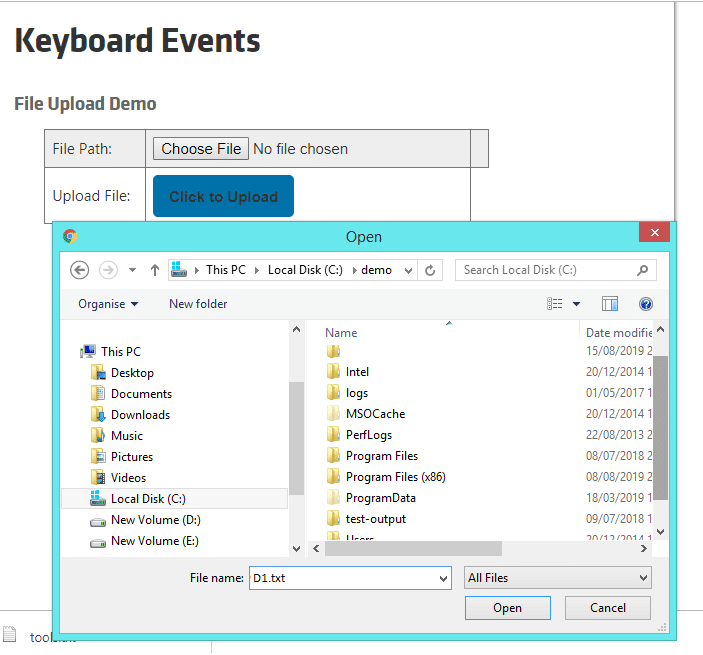
To simulate keyboard actions, following Robot class methods should be used :

**Keyboard methods:**

* keyPress(int keycode): This method presses a given key. The parameter keycode is an integer value for the key pressed. For example, to press a key for alphabet A, the value that has to pass is ***KeyEvent.VK\_A*** i.e., keyPress(KeyEvent.VK\_A).
  + ***KeyEvent*** is generally a low-level event. In Java AWT, low-level events are events that indicate direct communication from the user like a keypress, key release or a mouse click, drag, move, or release, etc. KeyEvent indicates an event that occurs on pressing, releasing, or typing a key on the component object like a text field.
  + This KeyEvent class has various constant fields like VK\_0, VK\_1 till VK\_9  of type integer. These values are the same as ASCII code for numbers '0' till '9'. Similarly, for alphabets, this class has constant fields like VK\_A, VK\_B till VK\_Z. It also has constant fields for representing special characters like ***VK\_DOLLAR*** for  "***$***" key, modifier keys like VK\_SHIFT for Shift key, etc.
* keyRelease(int keycode): This method releases a given key. For Example, the Shift key pressed using the \*keyPress(KeyEvent.VK\_SHIFT \*) method needs to release using the keyRelease (KeyEvent.VK\_SHIFT ) method.

### ***Practice Exercise to Perform Keyboard events using java Robot Class in Selenium***

Let’s discuss an example from an already available demo page on Toolsqa as “[***https://demoqa.com/keyboard-events/***](https://demoqa.com/keyboard-events/)“.



Here, to Upload any file, select the file first from Open popup which is Desktop Windows popup. Which, in turn, requires the filename to enter, e.g., D1.txt in this example.

Let's understand how to use the Robot class method to enter a filename.

* ***Instantiate Robot Class***

Robot robot = new Robot();

* ***Invoke the keyPress method to enter the text***

robot.keyPress(<keycode integer value>);

If a user needs to enter "***D1.txt***" in a textbox on a Webpage, he/she can use the sendKeys method, and just a single method call sendKeys("***D1.txt***") will suffice the purpose. But, to enter text in Desktop Windows popup using keyPress() method, the method needs to be invoked for each character of the input string, like keyPress(KeyEvent.VK\_D), keyPress(KeyEvent.VK\_1), etc.

***Let us automate the following scenario :***

1. Launch the web browser and launch our practice page [***https://demoqa.com/keyboard-events/***](https://demoqa.com/keyboard-events/)
2. Click on ‘Click here to browse’ button
3. Press Shift Key
4. Enter d to type it as D as the modifier Shift key press
5. Release Shift Key
6. Enter remaining part of the file name, i.e., 1.txt to display it as D1.txt
7. Press Enter key
8. Click on the Upload button and close the alert
9. Close the browser to end the program

***Prerequisite***: Create a file C:\demo\D1.txt

***Following is the sample code:***

package com.toolsqa.tutorials.actions;

import java.awt.AWTException;

import java.awt.Robot;

import java.awt.event.KeyEvent;

import java.io.IOException;

import org.openqa.selenium.Alert;

import org.openqa.selenium.By;

import org.openqa.selenium.Keys;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.openqa.selenium.support.ui.ExpectedConditions;

import org.openqa.selenium.support.ui.WebDriverWait;

public class RobotKeyboardDemo {

public static void main(String[] args) throws InterruptedException, AWTException, IOException {

System.setProperty("webdriver.gecko.driver","C:\\Selenium\\lib\\geckodriver-v0.24.0-win64\\geckodriver.exe");

// Create a new instance of the Firefox driver

WebDriver driver = new FirefoxDriver();

String URL = "https://demoqa.com/keyboard-events/";

//Start Browser

driver.get(URL);

//maximize browser

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

Thread.sleep(2000);

// This will click on Browse button

WebElement webElement = driver.findElement(By.id("browseFile"));

//click Browse button

webElement.sendKeys(Keys.ENTER);

//Create object of Robot class

Robot robot = new Robot();

//Code to Enter D1.txt

//Press Shify key

robot.keyPress(KeyEvent.VK\_SHIFT);

//Press d , it gets typed as upper case D as Shift key is pressed

robot.keyPress(KeyEvent.VK\_D);

//Release SHIFT key to release upper case effect

robot.keyRelease(KeyEvent.VK\_SHIFT);

robot.keyPress(KeyEvent.VK\_1);

robot.keyPress(KeyEvent.VK\_PERIOD);

robot.keyPress(KeyEvent.VK\_T);

robot.keyPress(KeyEvent.VK\_X);

robot.keyPress(KeyEvent.VK\_T);

//Press ENTER to close the popup

robot.keyPress(KeyEvent.VK\_ENTER);

//Wait for 1 sec

Thread.sleep(1000);

//This is just a verification part, accept alert

webElement = driver.findElement(By.id("uploadButton"));

webElement.click();

WebDriverWait wait = new WebDriverWait(driver, 10);

Alert myAlert = wait.until(ExpectedConditions.alertIsPresent());

//Accept the Alert

myAlert.accept();

//Close the main window

driver.close();

}

}

***Note***: Even though Robot class specifies to follow keyRelease for each keyPress event, Alphabets and numbers don't have any side effects on the next statements. Therefore, generally, users skip the keyRelease event for Alphabets and numbers. On the other hand, all the modifier keys such as SHIFT, ALT, etc. will always have a side effect on the next statements. As a result, it is still mandatory to specify keyRelease for each keyPress event of the modifier keys.

Just Try commenting out ***robot.keyRelease***(KeyEvent.VK\_SHIFT); in the above sample code and run the script. You will notice the file name types as, 'D!.TXT'. It is because the key pressed effect for the SHIFT key gets carried forward to the next typed text and types it in the uppercase.

package seleniumDemoPackage;

import java.awt.AWTException;

import java.awt.Robot;

import java.awt.event.KeyEvent;

import org.openqa.selenium.Alert;

import org.openqa.selenium.By;

import org.openqa.selenium.Keys;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.support.ui.ExpectedConditions;

import org.openqa.selenium.support.ui.WebDriverWait;

public class KeyBoardMethosInRobotClass {

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

/\*1. open : //https://demoqa.com/keyboard-events/

2. Click on browse button

3. Press shift key

4. Enter d to type it as D as the modifier shift key press

5. Release shift key

6. Enter remaining part of the file name that is: 1.txt to display it as "D1.txt"

7. Press enter key

8. click on the upload button and close the alert

9. close the browser to end the program\*/

System.setProperty("webdriver.chrome.driver", System.getProperty("user.dir") + "//externalLibraries//chromedriver.exe");

WebDriver driver = new ChromeDriver();

driver.get("http://www.demoqa.com/keyboard-events/");

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

Thread.sleep(2000);

//Instantiate robot class

Robot robot = new Robot();

//Invoke the keyPress method to enter text

//robot.keyPress(<keycode integerValue>);

//This will click on browse button

WebElement element = driver.findElement(By.id(""));

element.sendKeys(Keys.ENTER);

//press shift key

robot.keyPress(KeyEvent.VK\_SHIFT);

//Press d, it gets typed as upper case D as shift key is pressed

robot.keyPress(KeyEvent.VK\_D);

//Release SHIFT key to release upper case effect.

robot.keyRelease(KeyEvent.VK\_SHIFT);

robot.keyPress(KeyEvent.VK\_1);

robot.keyPress(KeyEvent.VK\_PERIOD);

robot.keyPress(KeyEvent.VK\_T);

robot.keyPress(KeyEvent.VK\_X);

robot.keyPress(KeyEvent.VK\_T);

//We want to ENTER to close the popup

robot.keyPress(KeyEvent.VK\_ENTER);

Thread.sleep(2000);

//This is just a verification part, for the accept alert

WebElement ele = driver.findElement(By.id(""));

ele.click();

WebDriverWait wait = new WebDriverWait(driver, 10);

Alert myAlert = wait.until(ExpectedConditions.alertIsPresent());

myAlert.accept();

driver.close();

}

}