POI API - Excel Driven Data Testing

This is the next tutorial in selenium-java series. Please go through the previous tutorials before you start this one. In the last tutorial, we learned about combo boxes. In this tutorial we will learn about excel driven data testing!

What you will Learn:
1. Download poi jars
2. Strategy to access excel data
3. Execute strategy
4. Source code

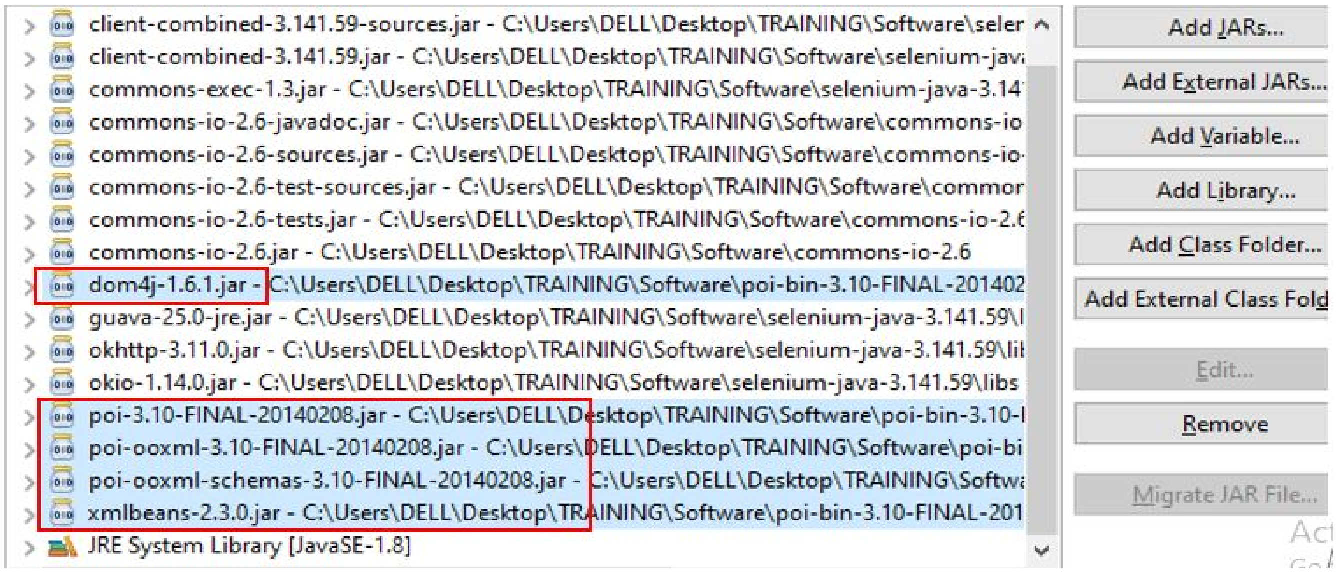
Download poi jars

If you have test data in excel, how will you pull that test data from excel into your java test cases? Apache's poi is the answer! Apache poi api is used to connect excel to java test cases.

You can download apache poi zip from the below location:
https://archive.apache.org/dist/poi/release/bin/poi-bin-3.10-FINAL-20140208.zip

After unzipping the jar file, add the following 5 external jars to your java project, see figure A:

dom4j-1.6.1.jar
poi-3.10-FINAL-20140208.jar
poi-ooxml-3.10-FINAL-20140208.jar
poi-ooxml-schemas-3.10-FINAL-20140208.jar
xmlbeans-2.3.0.jar



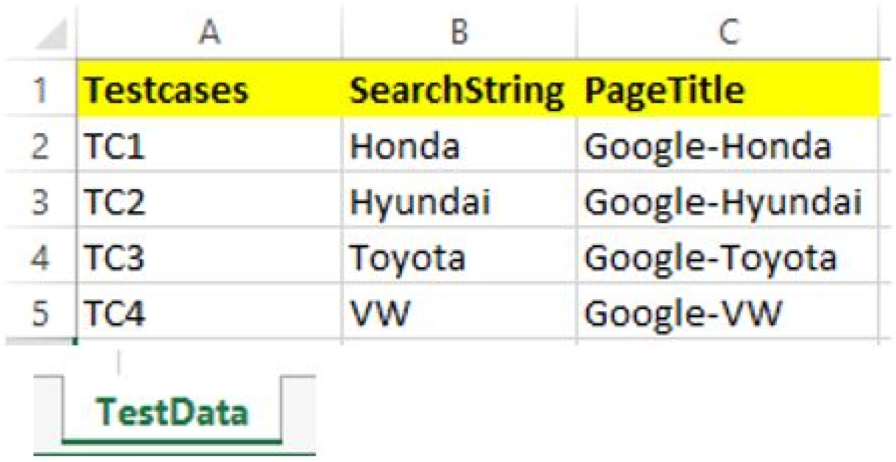
```
client-combined-3.141.59-sources.jar - C:\Users\DELL\Desktop\TRAINING\Software\seler
client-combined-3.141.59.jar - C:\Users\DELL\Desktop\TRAINING\Software\selenium-java
commons-exec-1.3.jar - C:\Users\DELL\Desktop\TRAINING\Software\selenium-java-3.14
commons-io-2.6-javadoc.jar - C:\Users\DELL\Desktop\TRAINING\Software\commons-io
commons-io-2.6-sources.jar - C:\Users\DELL\Desktop\TRAINING\Software\commons-io-
commons-io-2.6-test-sources.jar - C:\Users\DELL\Desktop\TRAINING\Software\commor
commons-io-2.6-tests.jar - C:\Users\DELL\Desktop\TRAINING\Software\commons-io-2.6
commons-io-2.6.jar - C:\Users\DELL\Desktop\TRAINING\Software\commons-io-2.6
dom4j-1.6.1.jar - C:\Users\DELL\Desktop\TRAINING\Software\poi-bin-3.10-FINAL-201402
guava-25.0-jre.jar - C:\Users\DELL\Desktop\TRAINING\Software\selenium-java-3.141.59\l
okhttp-3.11.0.jar - C:\Users\DELL\Desktop\TRAINING\Software\selenium-java-3.141.59\lil
okio-1.14.0.jar - C:\Users\DELL\Desktop\TRAINING\Software\selenium-java-3.141.59\libs
poi-3.10-FINAL-20140208.jar - C:\Users\DELL\Desktop\TRAINING\Software\poi-bin-3.10-l
poi-ooxml-3.10-FINAL-20140208.jar - C:\Users\DELL\Desktop\TRAINING\Software\poi-bi
poi-ooxml-schemas-3.10-FINAL-20140208.jar - C:\Users\DELL\Desktop\TRAINING\Softwa
xmlbeans-2.3.0.jar - C:\Users\DELL\Desktop\TRAINING\Software\poi-bin-3.10-FINAL-201
JRE System Library [JavaSE-1.8]
```

Figure A

Strategy to access excel data
We will be using below strategy to access data from excel:
1. Create object of XSSFWorkbook class
2. Get access to Sheet
3. Get access to all rows of Sheet
4. Get access to specific row from all rows
5. Get access to all cells of specific row

<u>Execute strategy</u>

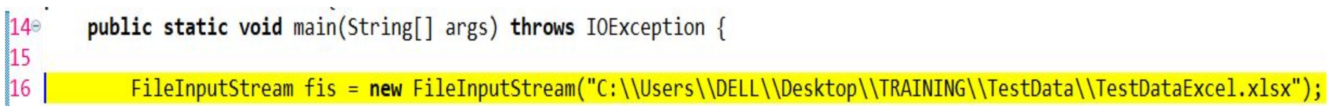
Let us create a new excel and save it as “TestDataExcel”. Create test cases with their corresponding sample test data in it. Rename the sheet as ‘TestData’, see below



	A	B	C
1	Testcases	SearchString	PageTitle
2	TC1	Honda	Google-Honda
3	TC2	Hyundai	Google-Hyundai
4	TC3	Toyota	Google-Toyota
5	TC4	VW	Google-VW

Figure B

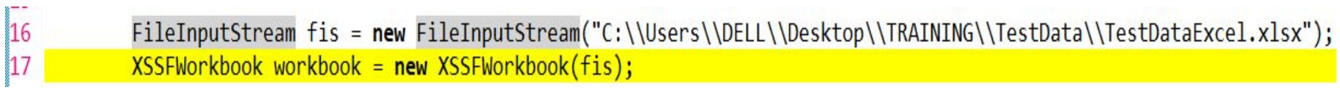
Step 1: Create new java class. Create FileInputStream object pointing towards the excel (make sure that the filepath is within double quotes)



```
    public static void main(String[] args) throws IOException {

        FileInputStream fis = new FileInputStream("C:\\Users\\DELL\\Desktop\\TRAINING\\TestData\\TestDataExcel.xlsx");
```

Figure 1

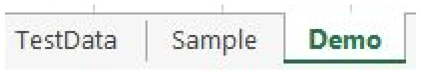
Step 2: Create an object of XSSFWorkbook class to take control of the entire excel application. This class has all the methods to pull the data from excel. Pass the reference of FileInputStream in its constructor:



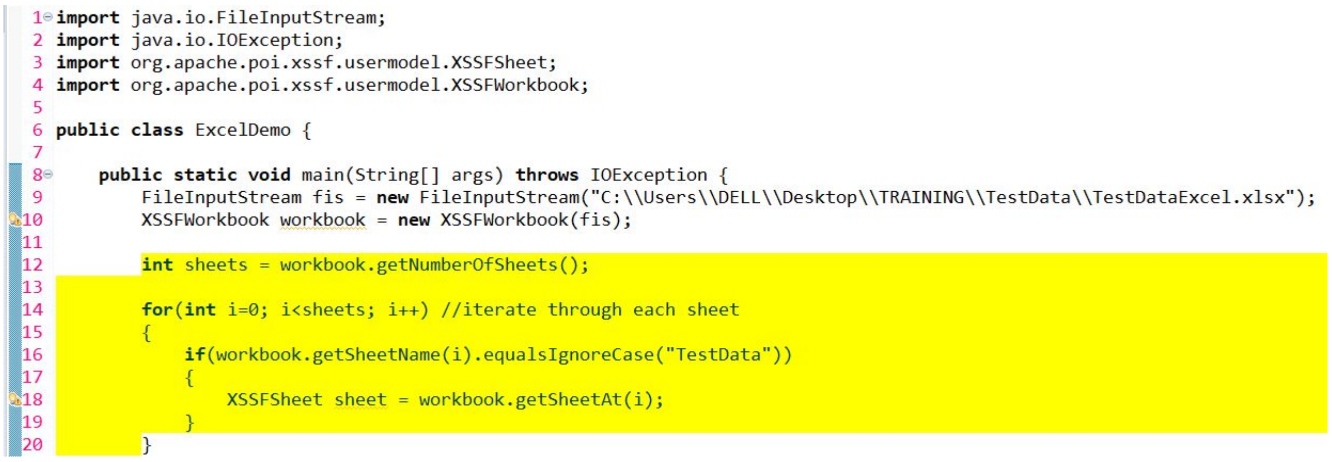
```
        FileInputStream fis = new FileInputStream("C:\\Users\\DELL\\Desktop\\TRAINING\\TestData\\TestDataExcel.xlsx");
        XSSFWorkbook workbook = new XSSFWorkbook(fis);
```

Figure 2

Step 3: Recall that in figure ‘B’, we had renamed the sheet to ‘TestData’. Now, how do we tell workbook to read only ‘TestData**’** sheet? To do that, we will first get the count of total number of sheets present in the workbook ‘TestDataExcel’. If we get the result that there are 3 sheets, then we will loop each and every sheet and compare whether the sheet name is ‘TestData**’** sheet. Once it matches, than we can access the sheet. Let’s suppose that the 3 sheets are: TestData, Sample, Demo



Write below logic to fetch “TestData” sheet (self-explanatory)

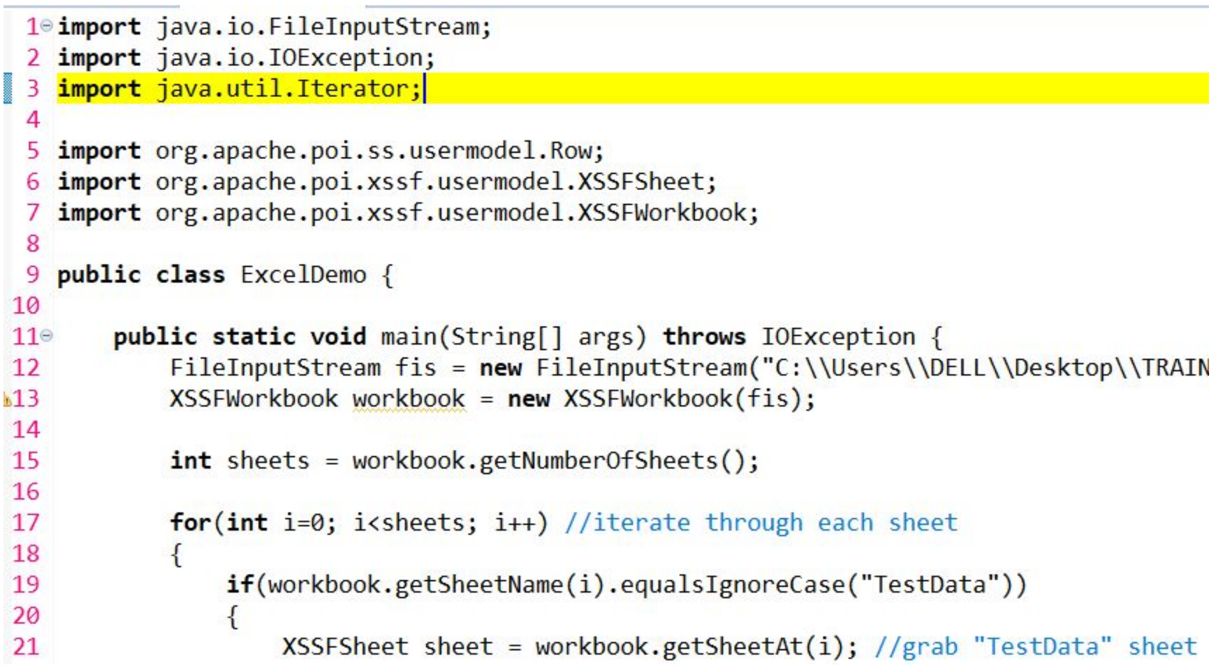


```
import java.io.FileInputStream;
import java.io.IOException;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;

public class ExcelDemo {

    public static void main(String[] args) throws IOException {
        FileInputStream fis = new FileInputStream("C:\\Users\\DELL\\Desktop\\TRAINING\\TestData\\TestDataExcel.xlsx");
        XSSFWorkbook workbook = new XSSFWorkbook(fis);

        int sheets = workbook.getNumberOfSheets();

        for(int i=0; i<sheets; i++) //iterate through each sheet
        {
            if(workbook.getSheetName(i).equalsIgnoreCase("TestData"))
            {
                XSSFSheet sheet = workbook.getSheetAt(i);
            }
        }
```

Figure 3

So we now have control over "TestData" sheet that has real test data (see figure B).

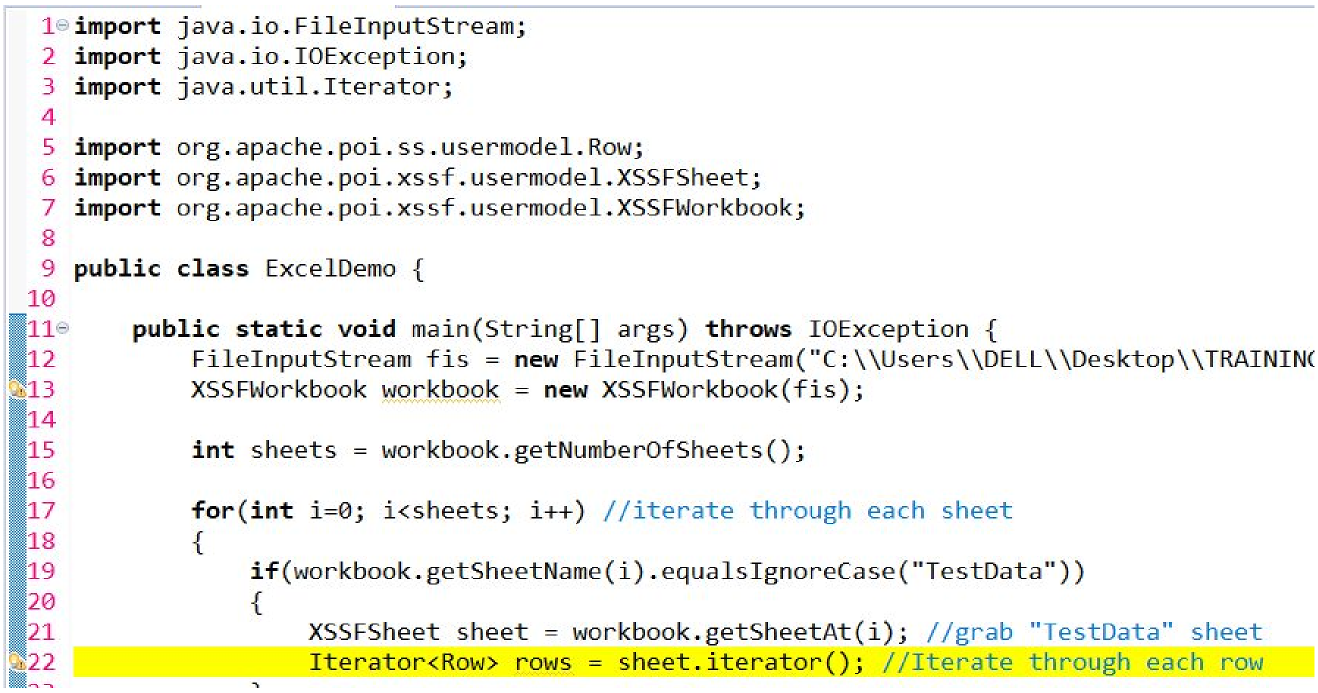
Step 4: We will now scan entire 1st row of "TestData" sheet and check for the presence of '**Testcases'** column (see figure B). To do that, we will first import java.util.Iterator;



```
import java.io.FileInputStream;
import java.io.IOException;
import java.util.Iterator;

import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;

public class ExcelDemo {

    public static void main(String[] args) throws IOException {
        FileInputStream fis = new FileInputStream("C:\\Users\\DELL\\Desktop\\TRAIN
        XSSFWorkbook workbook = new XSSFWorkbook(fis);

        int sheets = workbook.getNumberOfSheets();

        for(int i=0; i<sheets; i++) //iterate through each sheet
        {
            if(workbook.getSheetName(i).equalsIgnoreCase("TestData"))
            {
                XSSFSheet sheet = workbook.getSheetAt(i); //grab "TestData" sheet
```

Figure 4

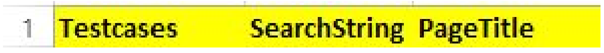
Step 5: Next, we will create a reference variable 'rows' that will iterate through each row of the sheet



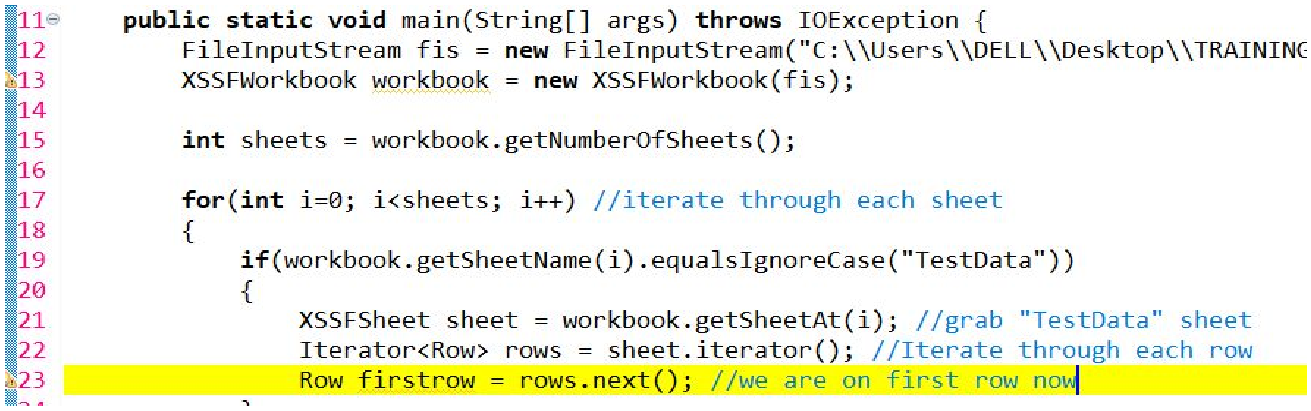
```
import java.io.FileInputStream;
import java.io.IOException;
import java.util.Iterator;

import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;

public class ExcelDemo {

    public static void main(String[] args) throws IOException {
        FileInputStream fis = new FileInputStream("C:\\Users\\DELL\\Desktop\\TRAININ
        XSSFWorkbook workbook = new XSSFWorkbook(fis);

        int sheets = workbook.getNumberOfSheets();

        for(int i=0; i<sheets; i++) //iterate through each sheet
        {
            if(workbook.getSheetName(i).equalsIgnoreCase("TestData"))
            {
                XSSFSheet sheet = workbook.getSheetAt(i); //grab "TestData" sheet
                Iterator<Row> rows = sheet.iterator(); //Iterate through each row
```

Figure 5

Step 6: Next, we will move to first row of the sheet by using rows.next()



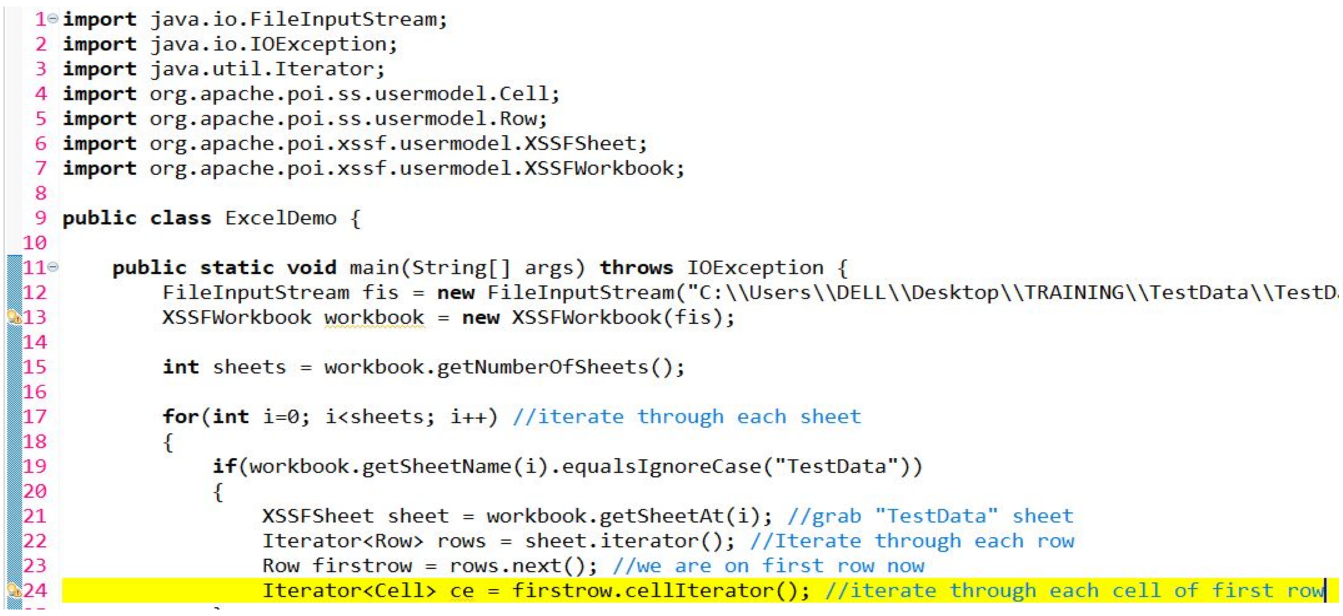
1	Testcases	SearchString	PageTitle



```
    public static void main(String[] args) throws IOException {
        FileInputStream fis = new FileInputStream("C:\\Users\\DELL\\Desktop\\TRAININ
        XSSFWorkbook workbook = new XSSFWorkbook(fis);

        int sheets = workbook.getNumberOfSheets();

        for(int i=0; i<sheets; i++) //iterate through each sheet
        {
            if(workbook.getSheetName(i).equalsIgnoreCase("TestData"))
            {
                XSSFSheet sheet = workbook.getSheetAt(i); //grab "TestData" sheet
                Iterator<Row> rows = sheet.iterator(); //Iterate through each row
                Row firstrow = rows.next(); //we are on first row now
```

Figure 6

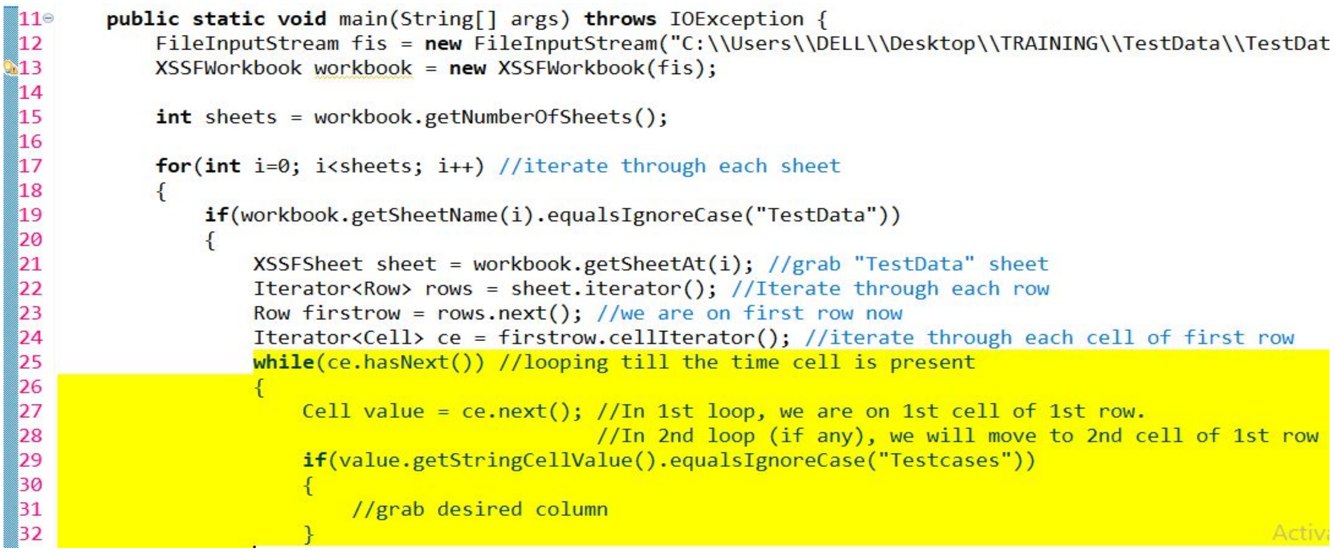
Step 7: Next we will iterate through each cell of first row (import org.apache.poi.ss.usermodel.Cell;)



```
 1 import java.io.FileInputStream;
 2 import java.io.IOException;
 3 import java.util.Iterator;
 4 import org.apache.poi.ss.usermodel.Cell;
 5 import org.apache.poi.ss.usermodel.Row;
 6 import org.apache.poi.xssf.usermodel.XSSFSheet;
 7 import org.apache.poi.xssf.usermodel.XSSFWorkbook;
 8
 9 public class ExcelDemo {
10
11     public static void main(String[] args) throws IOException {
12         FileInputStream fis = new FileInputStream("C:\\Users\\DELL\\Desktop\\TRAINING\\TestData\\TestD
13         XSSFWorkbook workbook = new XSSFWorkbook(fis);
14
15         int sheets = workbook.getNumberOfSheets();
16
17         for(int i=0; i<sheets; i++) //iterate through each sheet
18         {
19             if(workbook.getSheetName(i).equalsIgnoreCase("TestData"))
20             {
21                 XSSFSheet sheet = workbook.getSheetAt(i); //grab "TestData" sheet
22                 Iterator<Row> rows = sheet.iterator(); //Iterate through each row
23                 Row firstrow = rows.next(); //we are on first row now
24                 Iterator<Cell> ce = firstrow.cellIterator(); //iterate through each cell of first row
```

Figure 7

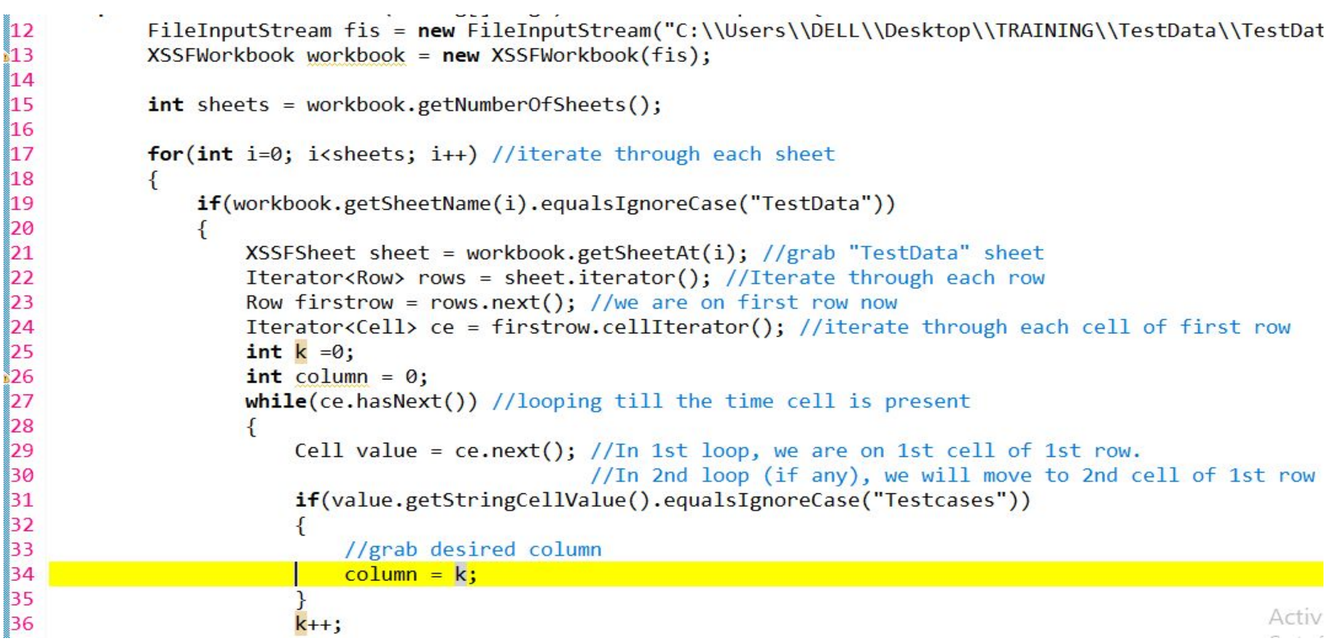
Step 8: Next, we will create a ‘while’ loop that will loop till the time a cell is present. If the value of first cell is equal to “Testcases”, we will grab the column



```
11     public static void main(String[] args) throws IOException {
12         FileInputStream fis = new FileInputStream("C:\\Users\\DELL\\Desktop\\TRAINING\\TestData\\TestDat
13         XSSFWorkbook workbook = new XSSFWorkbook(fis);
14
15         int sheets = workbook.getNumberOfSheets();
16
17         for(int i=0; i<sheets; i++) //iterate through each sheet
18         {
19             if(workbook.getSheetName(i).equalsIgnoreCase("TestData"))
20             {
21                 XSSFSheet sheet = workbook.getSheetAt(i); //grab "TestData" sheet
22                 Iterator<Row> rows = sheet.iterator(); //Iterate through each row
23                 Row firstrow = rows.next(); //we are on first row now
24                 Iterator<Cell> ce = firstrow.cellIterator(); //iterate through each cell of first row
25                 while(ce.hasNext()) //looping till the time cell is present
26                 {
27                     Cell value = ce.next(); //In 1st loop, we are on 1st cell of 1st row.
28                                             //In 2nd loop (if any), we will move to 2nd cell of 1st row
29                     if(value.getStringCellValue().equalsIgnoreCase("Testcases"))
30                     {
31                         //grab desired column
32                     }
```

Figure 8

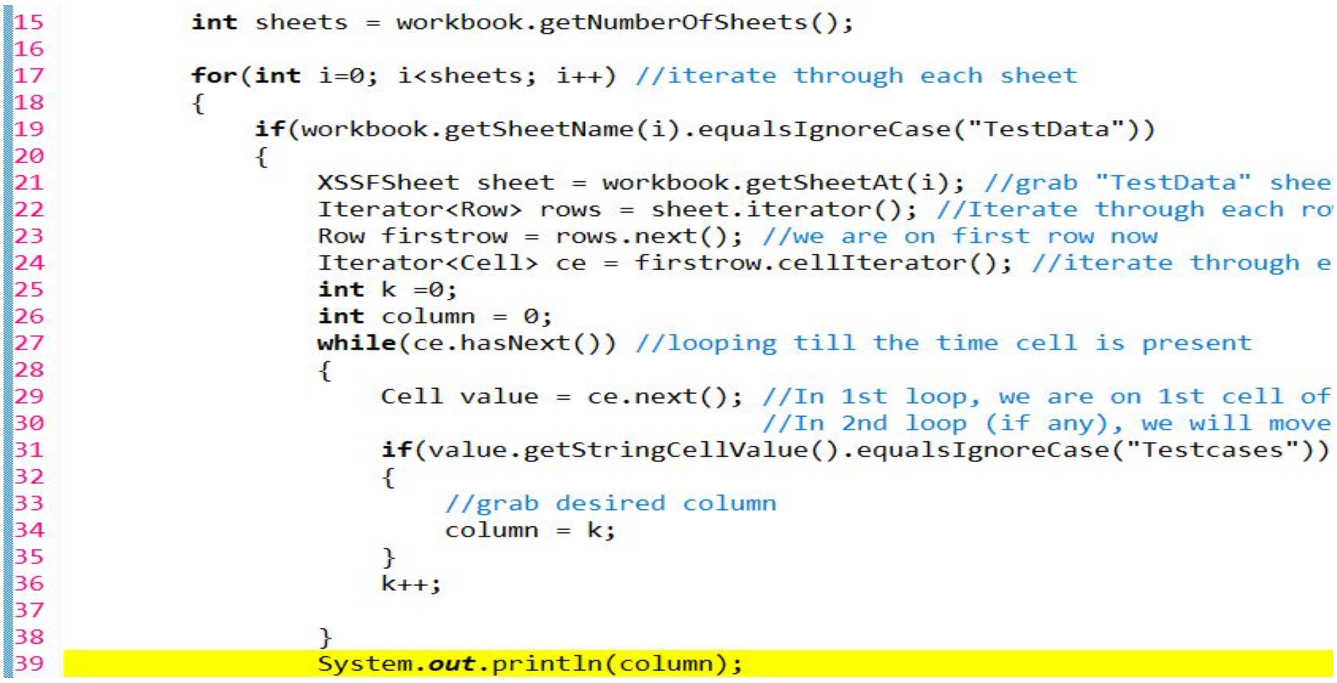
Step 9: Next, we will define 2 variables: ‘k’ and ‘column’. The ‘column’ variable will store the column index (starts from 0). The variable ‘k’ will keep getting incremented with each ‘while’ loop. Initialize both these variables to 0 (notice lines 25, 26 in the figure below). You will understand why we are doing this after a while (notice lines 34, 36)



```
12        FileInputStream fis = new FileInputStream("C:\\Users\\DELL\\Desktop\\TRAINING\\TestData\\TestDat
13        XSSFWorkbook workbook = new XSSFWorkbook(fis);
14
15        int sheets = workbook.getNumberOfSheets();
16
17        for(int i=0; i<sheets; i++) //iterate through each sheet
18        {
19            if(workbook.getSheetName(i).equalsIgnoreCase("TestData"))
20            {
21                XSSFSheet sheet = workbook.getSheetAt(i); //grab "TestData" sheet
22                Iterator<Row> rows = sheet.iterator(); //Iterate through each row
23                Row firstrow = rows.next(); //we are on first row now
24                Iterator<Cell> ce = firstrow.cellIterator(); //iterate through each cell of first row
25                int k =0;
26                int column = 0;
27                while(ce.hasNext()) //looping till the time cell is present
28                {
29                    Cell value = ce.next(); //In 1st loop, we are on 1st cell of 1st row.
30                                            //In 2nd loop (if any), we will move to 2nd cell of 1st row
31                    if(value.getStringCellValue().equalsIgnoreCase("Testcases"))
32                    {
33                        //grab desired column
34                        column = k;
35                    }
36                    k++;
```

Figure 9

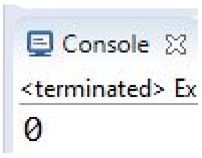
Step 10: Let us add sop at line#39 to print the value of 'column' variable, see below



```
15        int sheets = workbook.getNumberOfSheets();
16
17        for(int i=0; i<sheets; i++) //iterate through each sheet
18        {
19            if(workbook.getSheetName(i).equalsIgnoreCase("TestData"))
20            {
21                XSSFSheet sheet = workbook.getSheetAt(i); //grab "TestData" shee
22                Iterator<Row> rows = sheet.iterator(); //Iterate through each ro
23                Row firstrow = rows.next(); //we are on first row now
24                Iterator<Cell> ce = firstrow.cellIterator(); //iterate through e
25                int k =0;
26                int column = 0;
27                while(ce.hasNext()) //looping till the time cell is present
28                {
29                    Cell value = ce.next(); //In 1st loop, we are on 1st cell of
30                                            //In 2nd loop (if any), we will move
31                    if(value.getStringCellValue().equalsIgnoreCase("Testcases"))
32                    {
33                        //grab desired column
34                        column = k;
35                    }
36                    k++;
37
38                }
39                System.out.println(column);
```

Figure 10

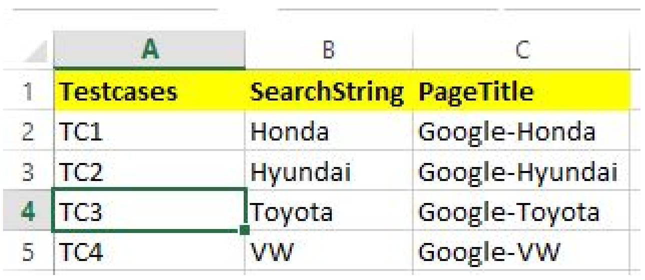
Step 11: Close the 'TestDataExcel' & run the above code. It prints the value of 'column' variable as 0. So we have found the 'Testcases' column at column 0 (remember that the index starts at 0). If you see figure 'B', we do see 'Testcases' column at column 0 (remember that the index starts at 0)



```
<terminated> Ex
0
```

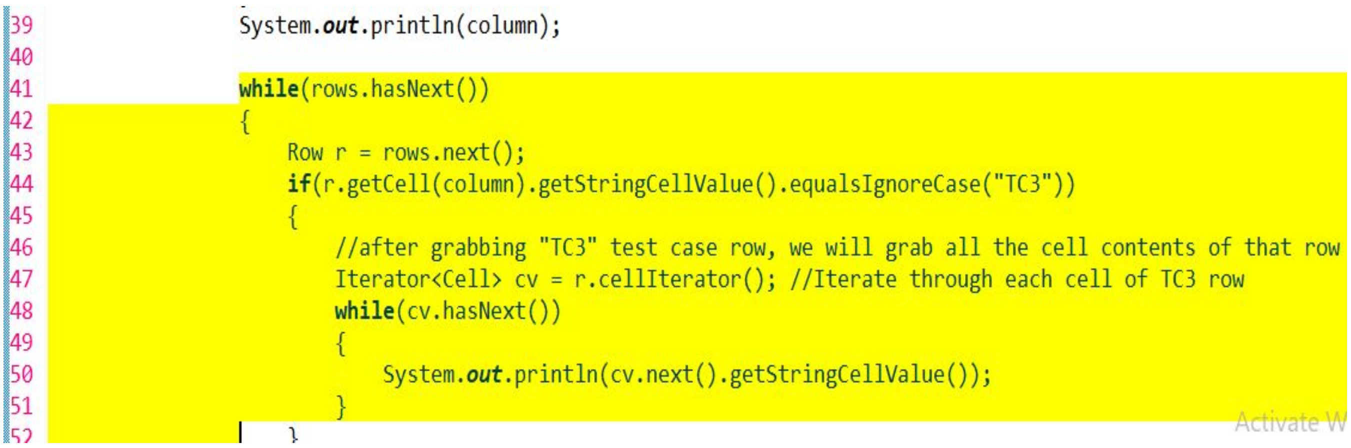
Figure 11

Step 12: Next we will scan all the rows of 'Testcases' column and find where 'TC3' test case row is. Once we have access to desired row, we will get access to all cells of the 'TC3' test case row



	A	B	C
1	Testcases	SearchString	PageTitle
2	TC1	Honda	Google-Honda
3	TC2	Hyundai	Google-Hyundai
4	TC3	Toyota	Google-Toyota
5	TC4	VW	Google-VW

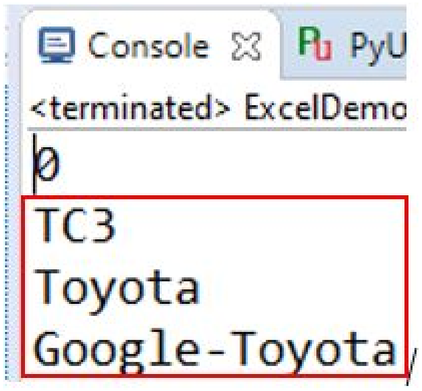
Figure 12A



```
            System.out.println(column);

            while(rows.hasNext())
            {
                Row r = rows.next();
                if(r.getCell(column).getStringCellValue().equalsIgnoreCase("TC3"))
                {
                    //after grabbing "TC3" test case row, we will grab all the cell contents of that row
                    Iterator<Cell> cv = r.cellIterator(); //Iterate through each cell of TC3 row
                    while(cv.hasNext())
                    {
                        System.out.println(cv.next().getStringCellValue());
                    }
                }
```

Figure 12B

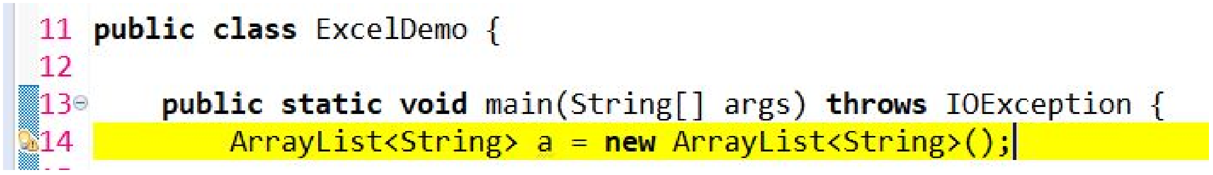
Run the script, see below, all the cell contents of TC3 row get printed



```
<terminated> ExcelDemo
0
TC3
Toyota
Google-Toyota
```

Figure 12C

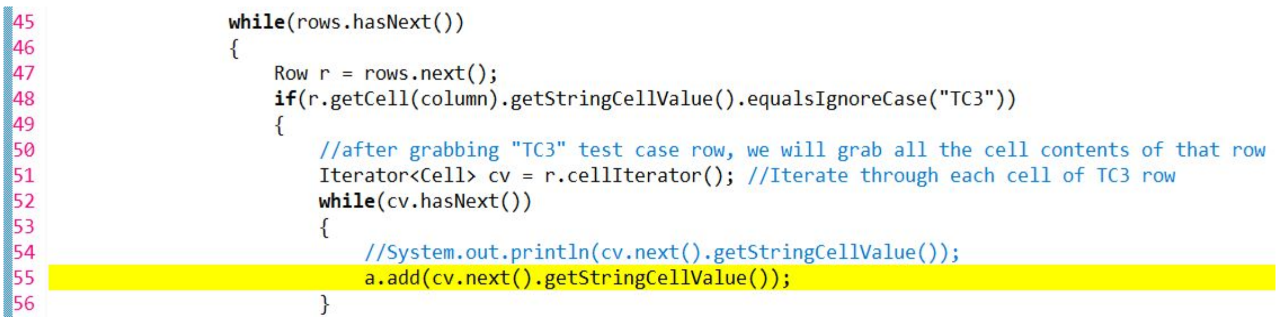
Step 13: The above printed data is slightly raw, so let's store it in array. To do that first create an ArrayList of type String



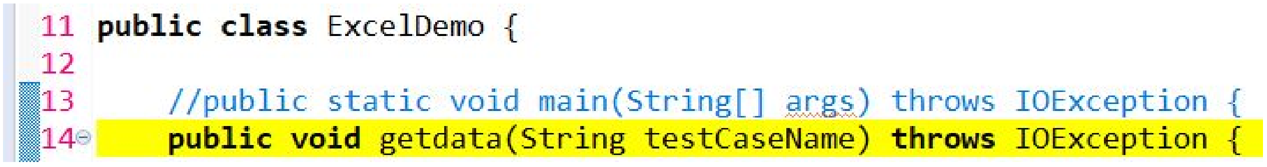
```
public class ExcelDemo {

    public static void main(String[] args) throws IOException {
        ArrayList<String> a = new ArrayList<String>();
```

Figure 13

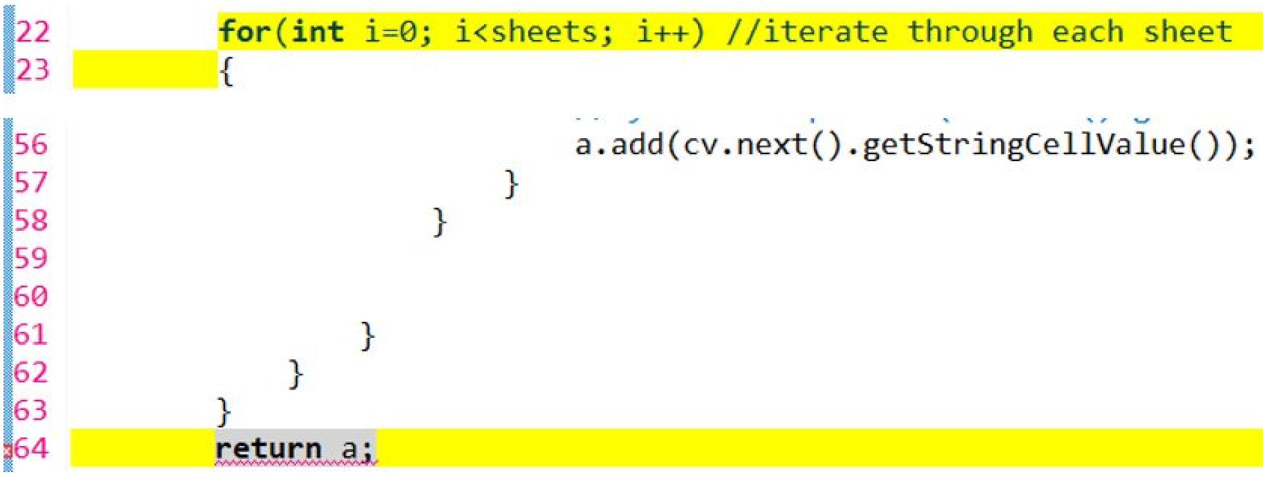
Step 14: Comment line#54, add line#55 to collect data in array list


```
45                  while(rows.hasNext())
46                  {
47                      Row r = rows.next();
48                      if(r.getCell(column).getStringCellValue().equalsIgnoreCase("TC3"))
49                      {
50                          //after grabbing "TC3" test case row, we will grab all the cell contents of that row
51                          Iterator<Cell> cv = r.cellIterator(); //Iterate through each cell of TC3 row
52                          while(cv.hasNext())
53                          {
54                              //System.out.println(cv.next().getStringCellValue());
55                              a.add(cv.next().getStringCellValue());
56                          }
```
Figure 14

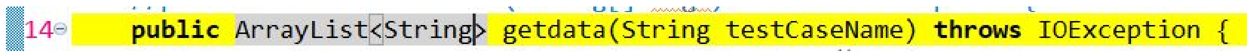
Step 15: Comment line#13. Add line#14 to create getdata() utility


```
11 public class ExcelDemo {
12 
13     //public static void main(String[] args) throws IOException {
14     public void getdata(String testCaseName) throws IOException {
```
Figure 15

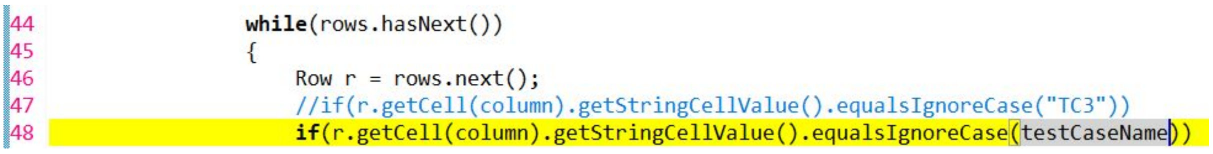
Step 16: Return the array after the first 'for' loop is over, line#64


```
22          for(int i=0; i<sheets; i++) //iterate through each sheet
23          {

56                              a.add(cv.next().getStringCellValue());
57                          }
58                      }
59 
60 
61                  }
62              }
63          }
64          return a;
```
Figure 16

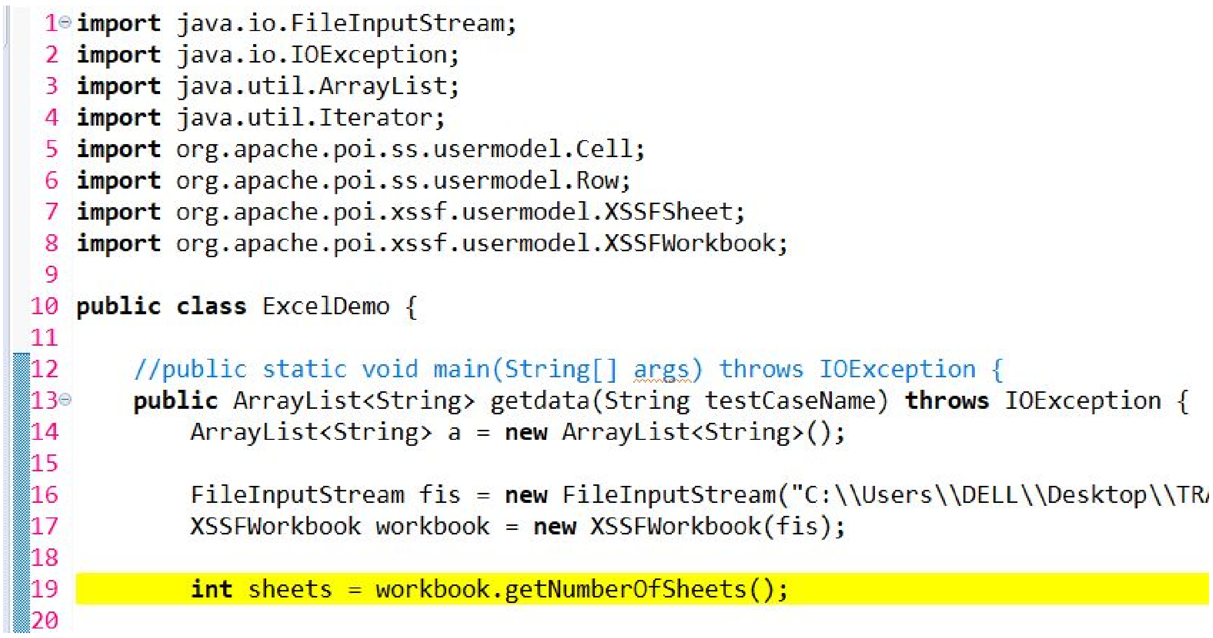
Step 17: Change return type of getdata() utility to ArrayList<String>


```
14     public ArrayList<String> getdata(String testCaseName) throws IOException {
```
Figure 17

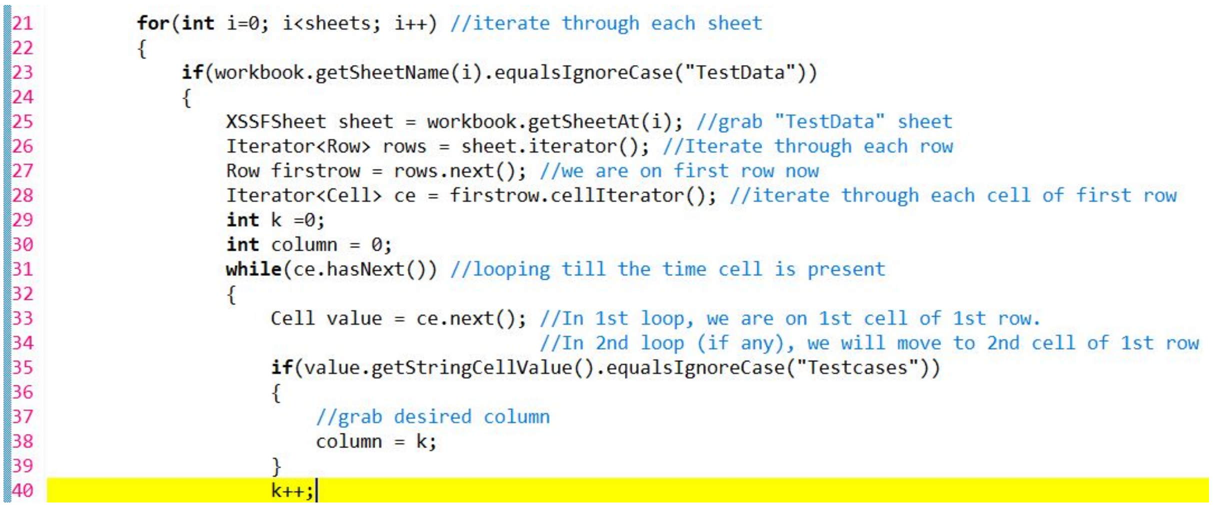
Step 18: Save, the error should disappear. We will now remove the hardcoded test case "TC3" from our code and replaced it with parameter 'testCaseName' that is being passed from the getdata() utility. To do that, comment line#47, add line#48.


```
44                  while(rows.hasNext())
45                  {
46                      Row r = rows.next();
47                      //if(r.getCell(column).getStringCellValue().equalsIgnoreCase("TC3"))
48                      if(r.getCell(column).getStringCellValue().equalsIgnoreCase(testCaseName))
```
Figure 18

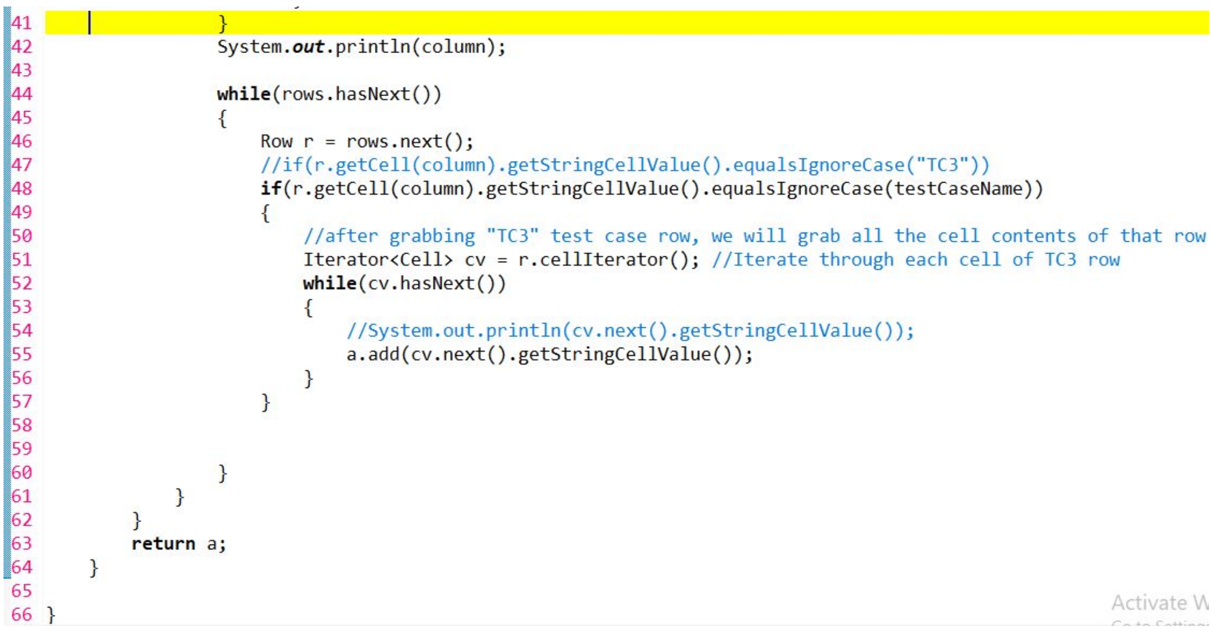
Step 19: That's it. We have created a utility. Below is entire code



```java
import java.io.FileInputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;

public class ExcelDemo {

    //public static void main(String[] args) throws IOException {
    public ArrayList<String> getdata(String testCaseName) throws IOException {
        ArrayList<String> a = new ArrayList<String>();

        FileInputStream fis = new FileInputStream("C:\\Users\\DELL\\Desktop\\TR
        XSSFWorkbook workbook = new XSSFWorkbook(fis);

        int sheets = workbook.getNumberOfSheets();

```



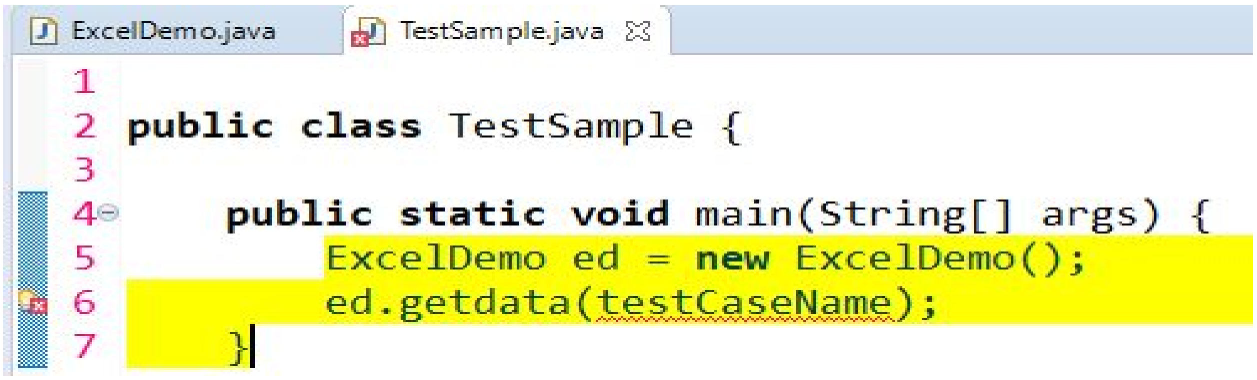
```java
        for(int i=0; i<sheets; i++) //iterate through each sheet
        {
            if(workbook.getSheetName(i).equalsIgnoreCase("TestData"))
            {
                XSSFSheet sheet = workbook.getSheetAt(i); //grab "TestData" sheet
                Iterator<Row> rows = sheet.iterator(); //Iterate through each row
                Row firstrow = rows.next(); //we are on first row now
                Iterator<Cell> ce = firstrow.cellIterator(); //iterate through each cell of first row
                int k =0;
                int column = 0;
                while(ce.hasNext()) //looping till the time cell is present
                {
                    Cell value = ce.next(); //In 1st loop, we are on 1st cell of 1st row.
                                            //In 2nd loop (if any), we will move to 2nd cell of 1st row
                    if(value.getStringCellValue().equalsIgnoreCase("Testcases"))
                    {
                        //grab desired column
                        column = k;
                    }
                    k++;
```



```
                }
                System.out.println(column);

                while(rows.hasNext())
                {
                    Row r = rows.next();
                    //if(r.getCell(column).getStringCellValue().equalsIgnoreCase("TC3"))
                    if(r.getCell(column).getStringCellValue().equalsIgnoreCase(testCaseName))
                    {
                        //after grabbing "TC3" test case row, we will grab all the cell contents of that row
                        Iterator<Cell> cv = r.cellIterator(); //Iterate through each cell of TC3 row
                        while(cv.hasNext())
                        {
                            //System.out.println(cv.next().getStringCellValue());
                            a.add(cv.next().getStringCellValue());
                        }
                    }


                }
            }
        }
        return a;
    }

}
```

Figure 19

Step 20: To test our utility, let's create a class 'TestSample' having main function. Create object of ExcelDemo (name of the above class) and call the above utility

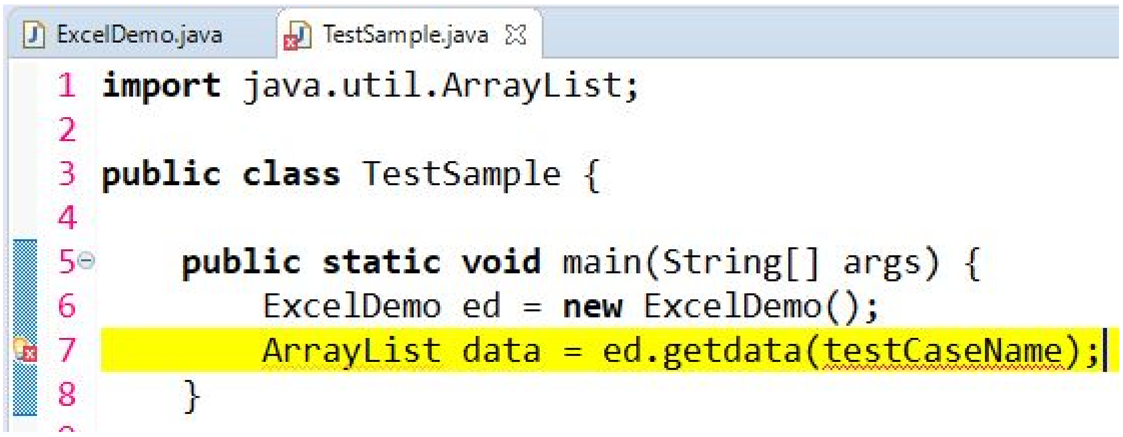


ExcelDemo.java TestSample.java

```

public class TestSample {

    public static void main(String[] args) {
        ExcelDemo ed = new ExcelDemo();
        ed.getdata(testCaseName);
    }
```

Figure 20

Step 21: Since getData method returns an array list, so

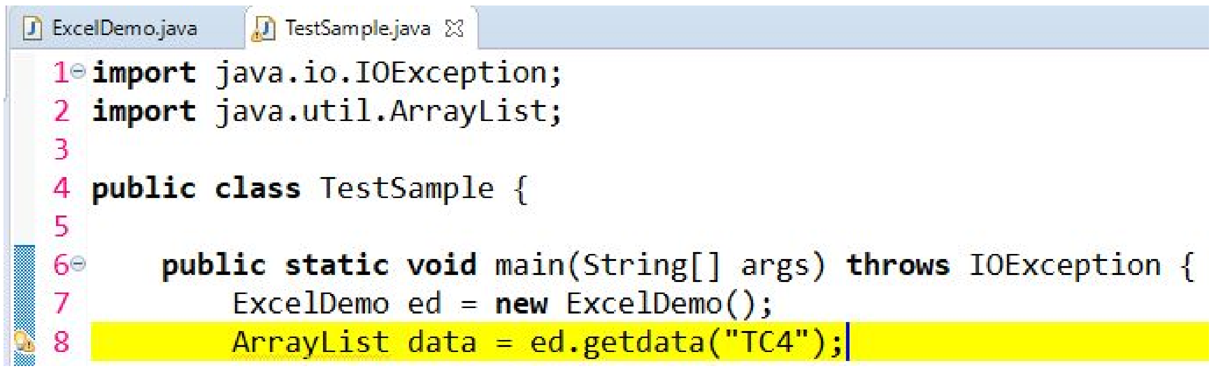


ExcelDemo.java TestSample.java

```
import java.util.ArrayList;

public class TestSample {

    public static void main(String[] args) {
        ExcelDemo ed = new ExcelDemo();
        ArrayList data = ed.getdata(testCaseName);
    }
```

Figure 21

Step 22: Let us mention testCaseName as "TC4"

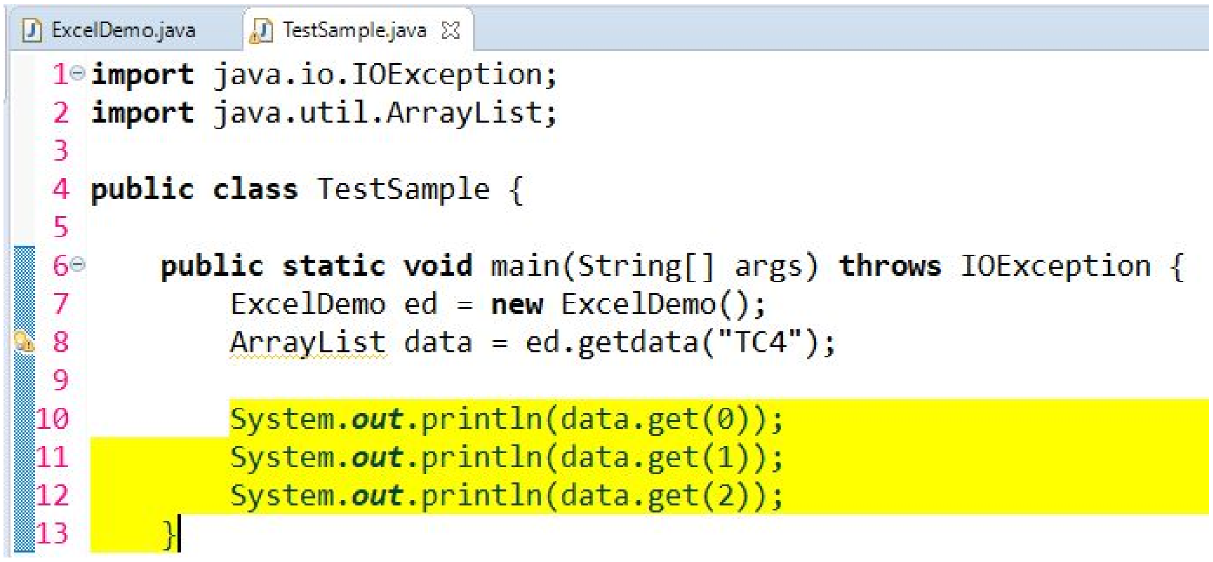


ExcelDemo.java TestSample.java

```java
import java.io.IOException;
import java.util.ArrayList;

public class TestSample {

    public static void main(String[] args) throws IOException {
        ExcelDemo ed = new ExcelDemo();
        ArrayList data = ed.getdata("TC4");
```

Figure 22

Step 23: Now we know that an array stores data in indices, so

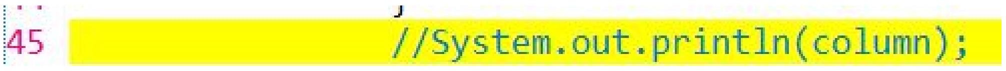


ExcelDemo.java TestSample.java

```java
import java.io.IOException;
import java.util.ArrayList;

public class TestSample {

    public static void main(String[] args) throws IOException {
        ExcelDemo ed = new ExcelDemo();
        ArrayList data = ed.getdata("TC4");

        System.out.println(data.get(0));
        System.out.println(data.get(1));
        System.out.println(data.get(2));
    }
```

Figure 23

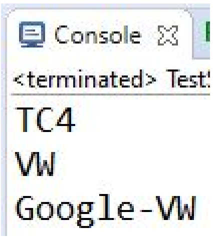
Step 24: Close the excel. Comment line#45 in 'ExcelDemo' class



```java
45                  //System.out.println(column);
```

Figure 24

Step 25: Run 'TestSample' class, we get the TC4 row details

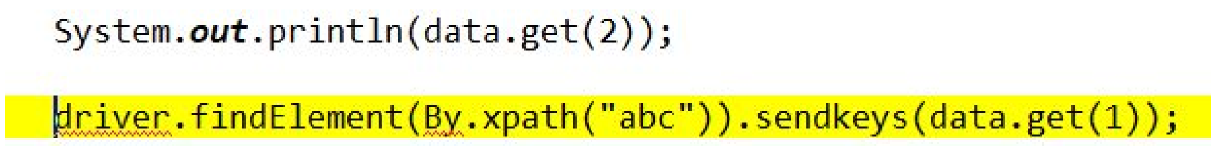


Console

```
<terminated> Test!
TC4
VW
Google-VW
```

Figure 25

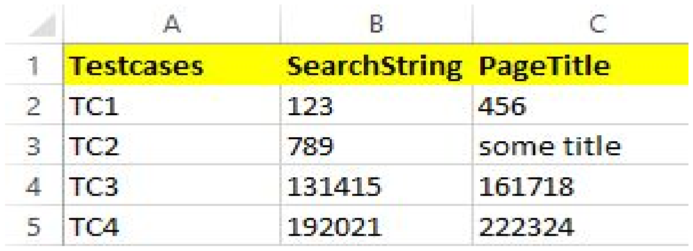
Step 26: Similarly you can getData of other test cases as well. We can now call this test data that is present in excel into our selenium code. So assuming data.get(1) contain username, we can pass it like below



```
System.out.println(data.get(2));

driver.findElement(By.xpath("abc")).sendkeys(data.get(1));
```

Figure 26

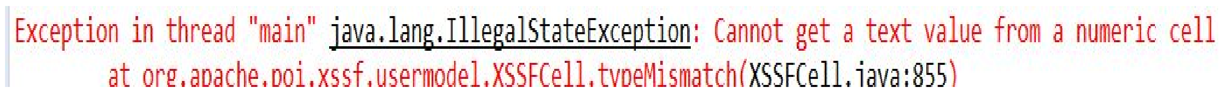
Step 27: In “TestDataExcel” excel, let us change few of the cell values to **integer**



	A	B	C
1	Testcases	SearchString	PageTitle
2	TC1	123	456
3	TC2	789	some title
4	TC3	131415	161718
5	TC4	192021	222324

Figure 27

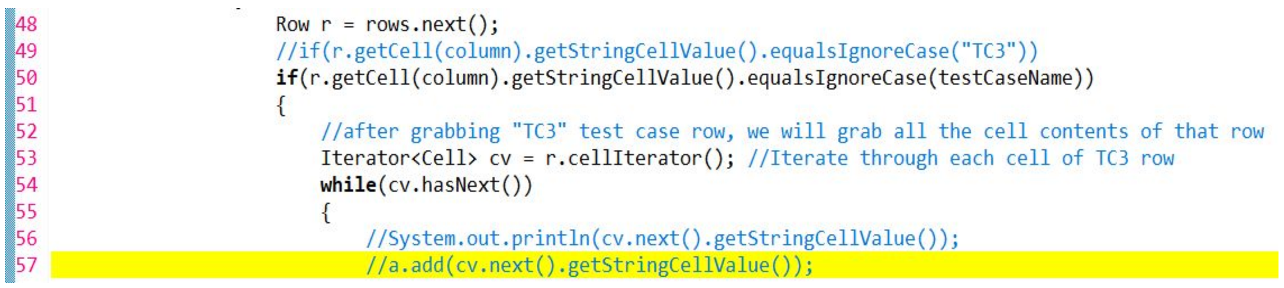
Step 28: Save excel & close it. Run ‘TestSample’ class, we will get below error



```
Exception in thread "main" java.lang.IllegalStateException: Cannot get a text value from a numeric cell
	at org.apache.poi.xssf.usermodel.XSSFCell.typeMismatch(XSSFCell.java:855)
```

Figure 28

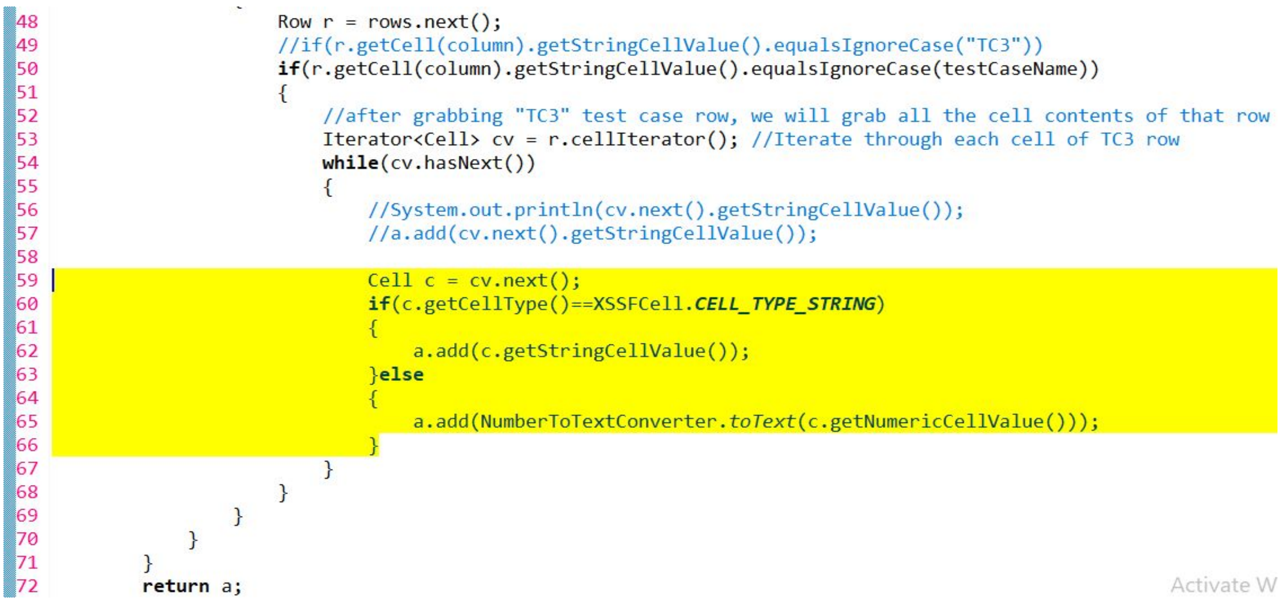
Step 29: Let us resolve this error. In ExcelDemo.java, comment below line where we are storing the string values to an array



```
48                      Row r = rows.next();
49                      //if(r.getCell(column).getStringCellValue().equalsIgnoreCase("TC3"))
50                      if(r.getCell(column).getStringCellValue().equalsIgnoreCase(testCaseName))
51                      {
52                          //after grabbing "TC3" test case row, we will grab all the cell contents of that row
53                          Iterator<Cell> cv = r.cellIterator(); //Iterate through each cell of TC3 row
54                          while(cv.hasNext())
55                          {
56                              //System.out.println(cv.next().getStringCellValue());
57                              //a.add(cv.next().getStringCellValue());
```

Figure 29

Step 30: Extract the value of cv.next shown in line#59. In line#60 we are checking if cell type is string. Under else loop, in line#65, we are converting the numeric value to String (since we have defined the ArrayList of type string, the ArrayList cannot accept numeric values)

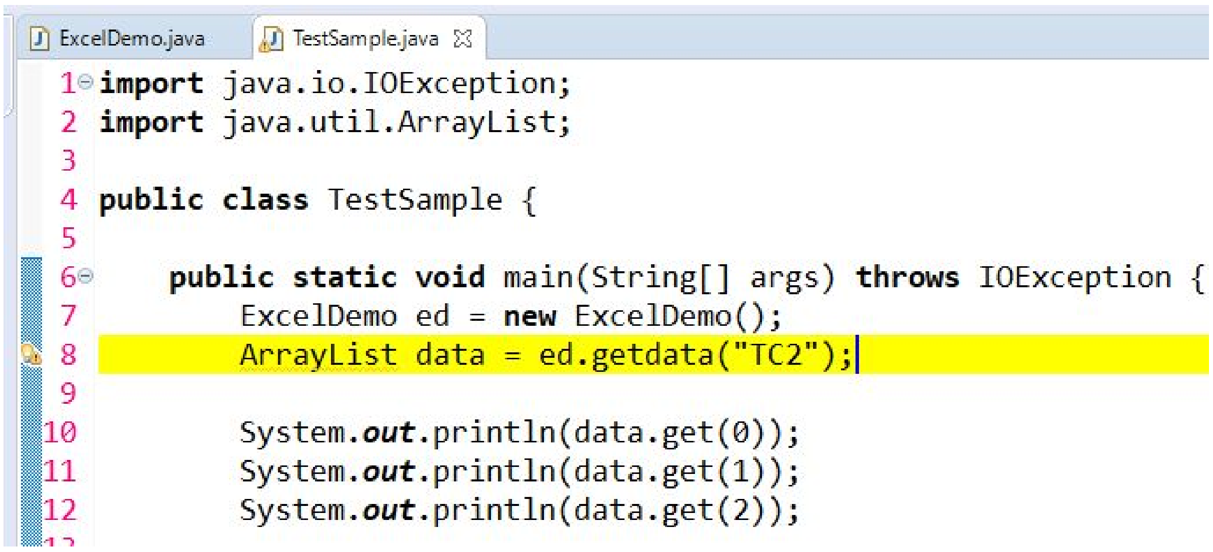


```java
                        Row r = rows.next();
                        //if(r.getCell(column).getStringCellValue().equalsIgnoreCase("TC3"))
                        if(r.getCell(column).getStringCellValue().equalsIgnoreCase(testCaseName))
                        {
                            //after grabbing "TC3" test case row, we will grab all the cell contents of that row
                            Iterator<Cell> cv = r.cellIterator(); //Iterate through each cell of TC3 row
                            while(cv.hasNext())
                            {
                                //System.out.println(cv.next().getStringCellValue());
                                //a.add(cv.next().getStringCellValue());

                                Cell c = cv.next();
                                if(c.getCellType()==XSSFCell.CELL_TYPE_STRING)
                                {
                                    a.add(c.getStringCellValue());
                                }else
                                {
                                    a.add(NumberToTextConverter.toText(c.getNumericCellValue()));
                                }
                            }
                        }
                    }
                }
            }
            return a;
```

Figure 30

So basically we are converting the numeric integers to string and then adding it to array

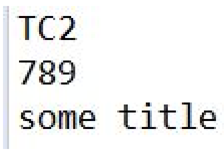
Step 31: Save the code. Change to “TC2”



```java
import java.io.IOException;
import java.util.ArrayList;

public class TestSample {

    public static void main(String[] args) throws IOException {
        ExcelDemo ed = new ExcelDemo();
        ArrayList data = ed.getdata("TC2");

        System.out.println(data.get(0));
        System.out.println(data.get(1));
        System.out.println(data.get(2));
```

Figure 31

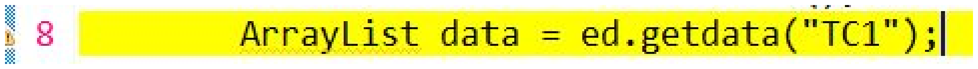
Step 32: Run TestSample.java, notice that we now are getting numeric as well as string values



```
TC2
789
some title
```

Figure 32

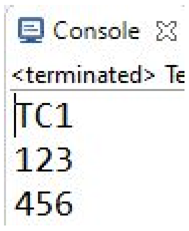
Step 33: Change to “TC1”



```java
        ArrayList data = ed.getdata("TC1");
```

Figure 33

Step 34: Run TestSample.java


Console

<terminated> Te

```
TC1
123
456
```

Figure 34

<u>Source code</u>

```
import java.io.FileInputStream;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Iterator;

import org.apache.poi.ss.usermodel.Cell;

import org.apache.poi.ss.usermodel.Row;

import org.apache.poi.ss.util.NumberToTextConverter;

import org.apache.poi.xssf.usermodel.XSSFCell;

import org.apache.poi.xssf.usermodel.XSSFSheet;

import org.apache.poi.xssf.usermodel.XSSFWorkbook;

public class ExcelDemo {

	//public static void main(String[] args) throws IOException {

	public ArrayList<String> getdata(String testCaseName) throws IOException {

		ArrayList<String> a = new ArrayList<String>();

		FileInputStream fis = new
FileInputStream("C:\\Users\\DELL\\Desktop\\TRAINING\\TestData\\TestDataExcel.xlsx");

		XSSFWorkbook workbook = new XSSFWorkbook(fis);

		int sheets = workbook.getNumberOfSheets();

		for(int i=0; i<sheets; i++) //iterate through each sheet

		{

			if(workbook.getSheetName(i).equalsIgnoreCase("TestData"))

			{

				XSSFSheet sheet = workbook.getSheetAt(i); //grab "TestData" sheet

				Iterator<Row> rows = sheet.iterator(); //Iterate through each row
```

```
                Row firstrow = rows.next(); //we are on first row now
                Iterator<Cell> ce = firstrow.cellIterator(); //iterate through each cell
of first row
                int k =0;
                int column = 0;
                while(ce.hasNext()) //looping till the time cell is present
                {
                    Cell value = ce.next(); //In 1st loop, we are on 1st cell of 1st
row.
                                                    //In 2nd
loop (if any), we will move to 2nd cell of 1st row
                    if(value.getStringCellValue().equalsIgnoreCase("Testcases"))
                    {
                        //grab desired column
                        column = k;
                    }
                    k++;
                }
                //System.out.println(column);
                while(rows.hasNext())
                {
                    Row r = rows.next();
    //if(r.getCell(column).getStringCellValue().equalsIgnoreCase("TC3"))
    if(r.getCell(column).getStringCellValue().equalsIgnoreCase(testCaseName))
                    {
                        //after grabbing "TC3" test case row, we will grab all
the cell contents of that row
                        Iterator<Cell> cv = r.cellIterator(); //Iterate through
each cell of TC3 row
                        while(cv.hasNext())
                        {

//System.out.println(cv.next().getStringCellValue());
```

```
                                                        //a.add(cv.next().getStringCellValue());
                                                        Cell c = cv.next();

if(c.getCellType()==XSSFCell.CELL_TYPE_STRING)
                                                        {
                                                              a.add(c.getStringCellValue());
                                                        }else
                                                        {

a.add(NumberToTextConverter.toText(c.getNumericCellValue()));
                                                        }
                                                  }
                                            }
                                      }
                                }
                          }
                          return a;
                    }
}
```