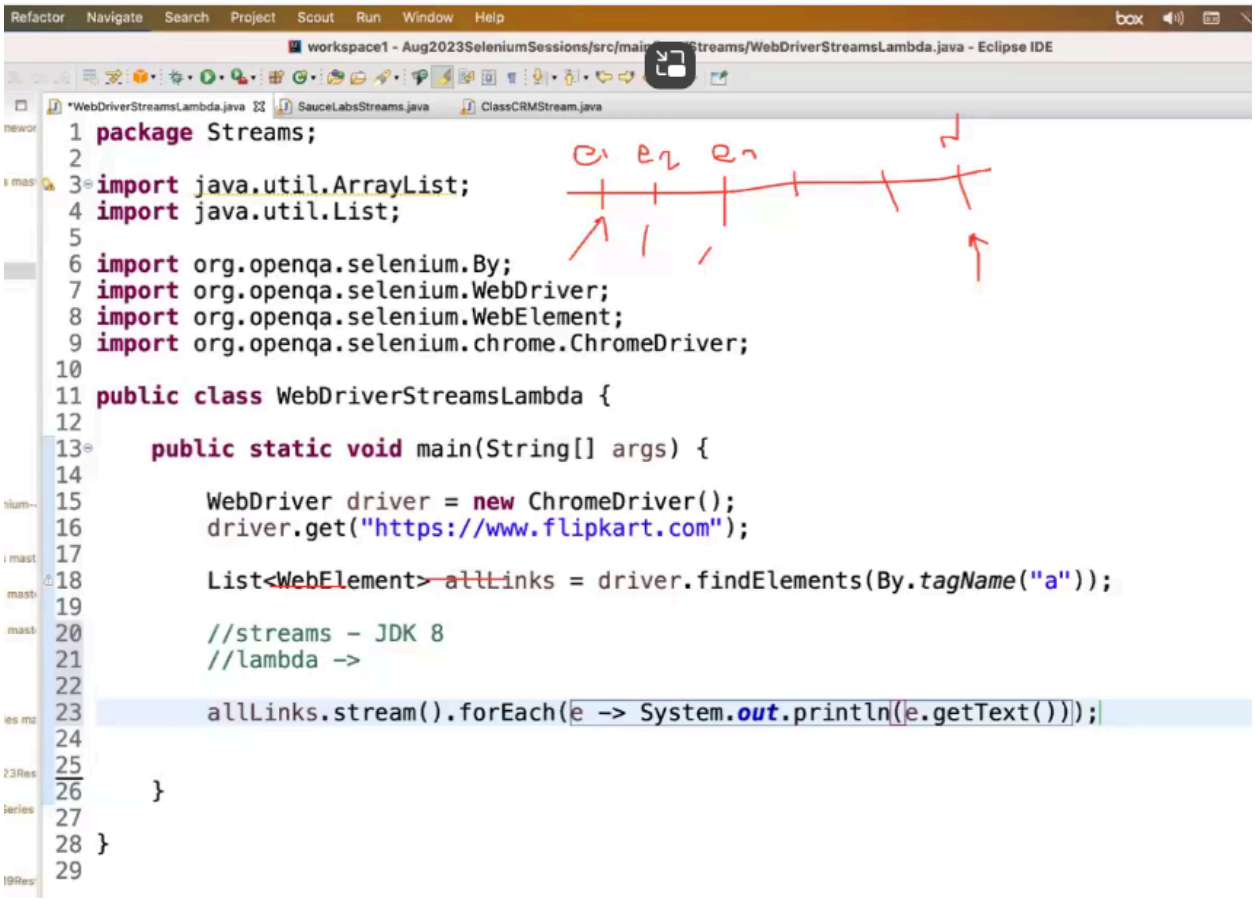
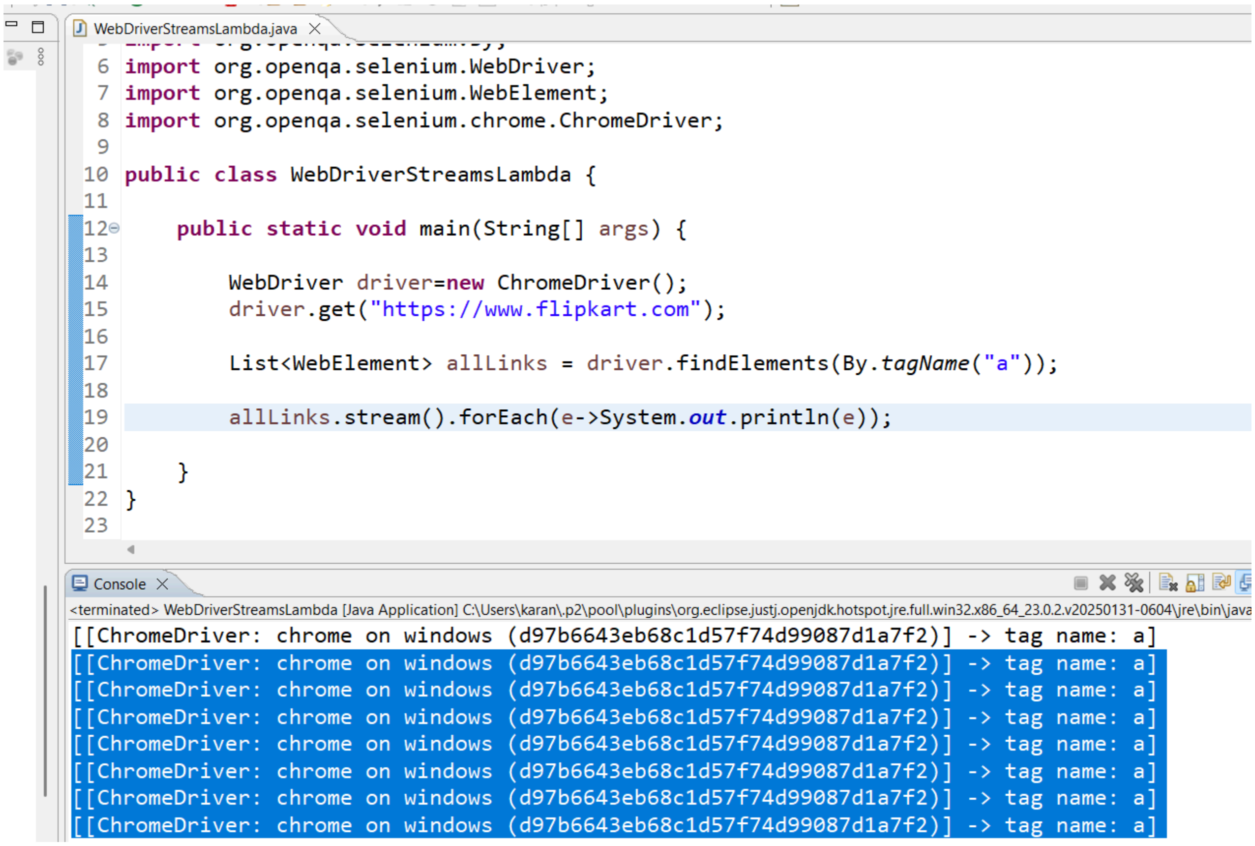
Replacement for **for** and **for each** loop.
Used with collections only.
Applied mostly with lists.
Stream comes from **java.util** package.
Return type is stream<generics>, if we are getting webelement then it is stream<webelement>.
To iterate over stream use foreach. Foreach requires consumer as parameter. Write any variable name you like. Use lambda. Right side give the expression. The variable will go to each and every element. We just print the text. Simply printing the variable will give webelement, we want text.

Get all links from flipkart-


Refactor Navigate Search Project Scout Run Window Help
workspace1 - Aug2023SeleniumSessions/src/main/java/Streams/WebDriverStreamsLambda.java - Eclipse IDE
*WebDriverStreamsLambda.java
SauceLabsStreams.java
ClassCRMStream.java
1 package Streams;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 import org.openqa.selenium.By;
7 import org.openqa.selenium.WebDriver;
8 import org.openqa.selenium.WebElement;
9 import org.openqa.selenium.chrome.ChromeDriver;
10
11 public class WebDriverStreamsLambda {
12
13 public static void main(String[] args) {
14
15 WebDriver driver = new ChromeDriver();
16 driver.get("https://www.flipkart.com");
17
18 List<WebElement> allLinks = driver.findElements(By.tagName("a"));
19
20 //streams - JDK 8
21 //lambda ->
22
23 allLinks.stream().forEach(e -> System.out.println(e.getText()));
24
25
26 }
27
28 }
29

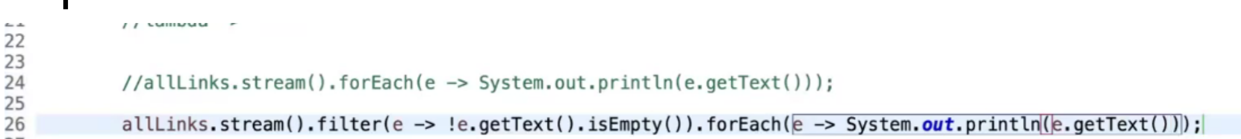
If we try to print the webelement, we get like this–

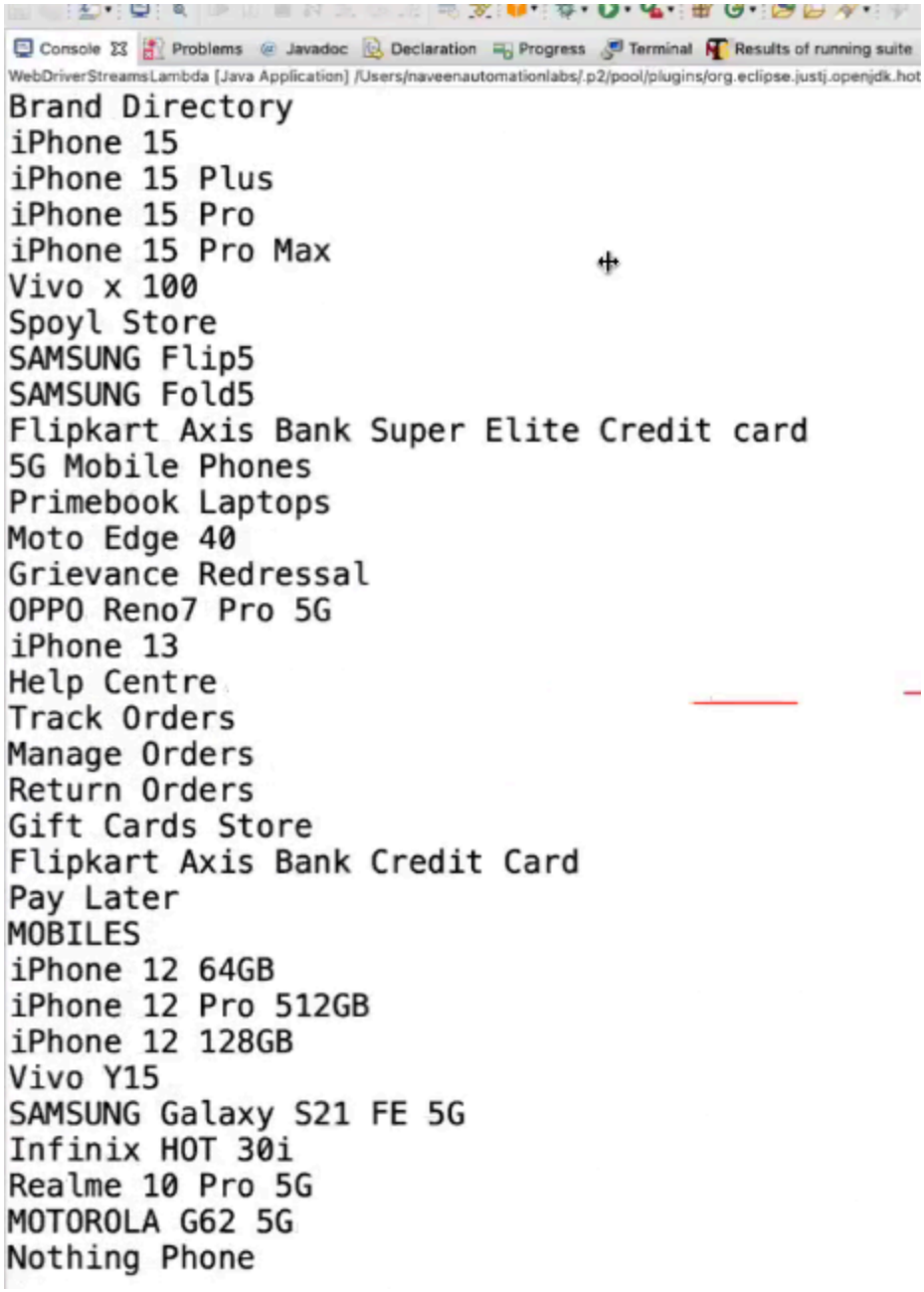
//prints all internal ids of webelements.



Filter method-

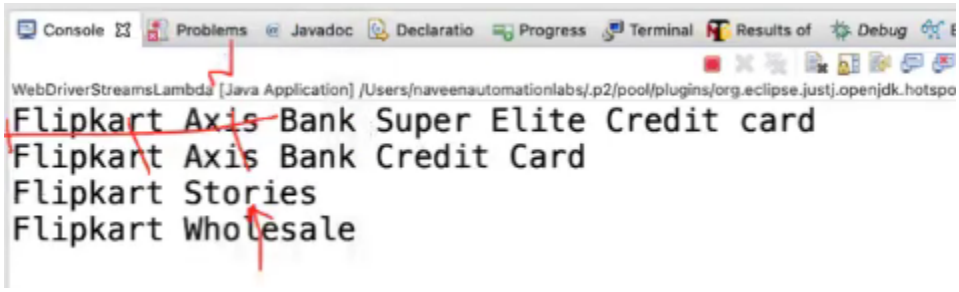
How to exclude empty links from flipkart page. If any text is empty ignore it. Once filtered, send to for each to print.





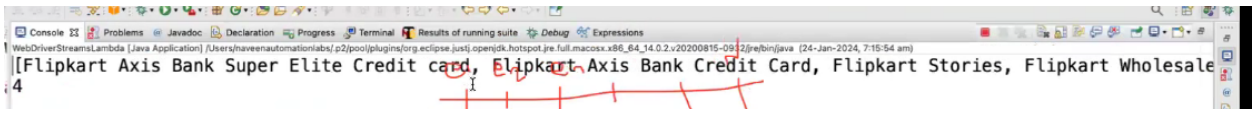
Apply as many filters as you like-
Filter to get links which are not empty and starting with word flipkart.

```
23
24          //allLinks.stream().forEach(e -> System.out.println(e.getText()));
25
26          //allLinks.stream().filter(e -> !e.getText().isEmpty()).forEach(e -> System.out.println(e.getText()));
27
28          allLinks
29              .stream()
30                  .filter(e -> !e.getText().isEmpty())
31                      .filter(e -> e.getText().startsWith("Flipkart"))
32                          .forEach(e -> System.out.println(e.getText()));
33
```

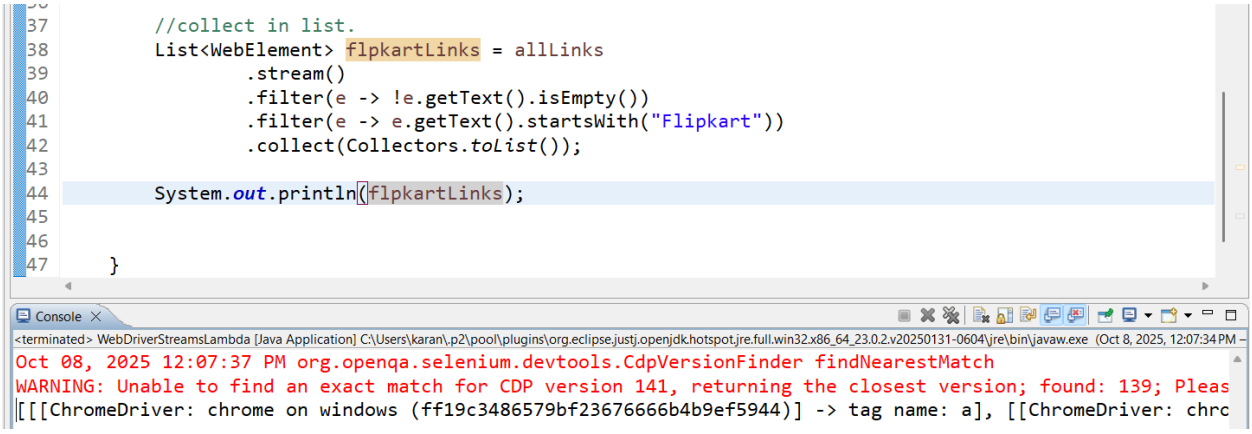


Collect in a list format-
Use collect.
Pass collectors.tolist.
Collect returns list of webelement. Again depends on what is our generics.
We can apply stream on the list.
Whenever you want to do some conversion from integer to string or webelement to string use map.
Collect all the text. Returns list of string this time.

```
35
36          List<WebElement> flpkartLinks= allLinks
37                                          .stream()
38                                                  .filter(e -> !e.getText().isEmpty())
39                                                          .filter(e -> e.getText().startsWith("Flipkart"))
40                                                                  .collect(Collectors.toList());
41
42          List<String> flpkartLinksText = flpkartLinks.stream().map(e -> e.getText()).collect(Collectors.toList());
43          System.out.println(flpkartLinksText);
44          System.out.println(flpkartLinksText.size());
```



If we try to print the list of webelement- we get the internal webelement id and the locator-

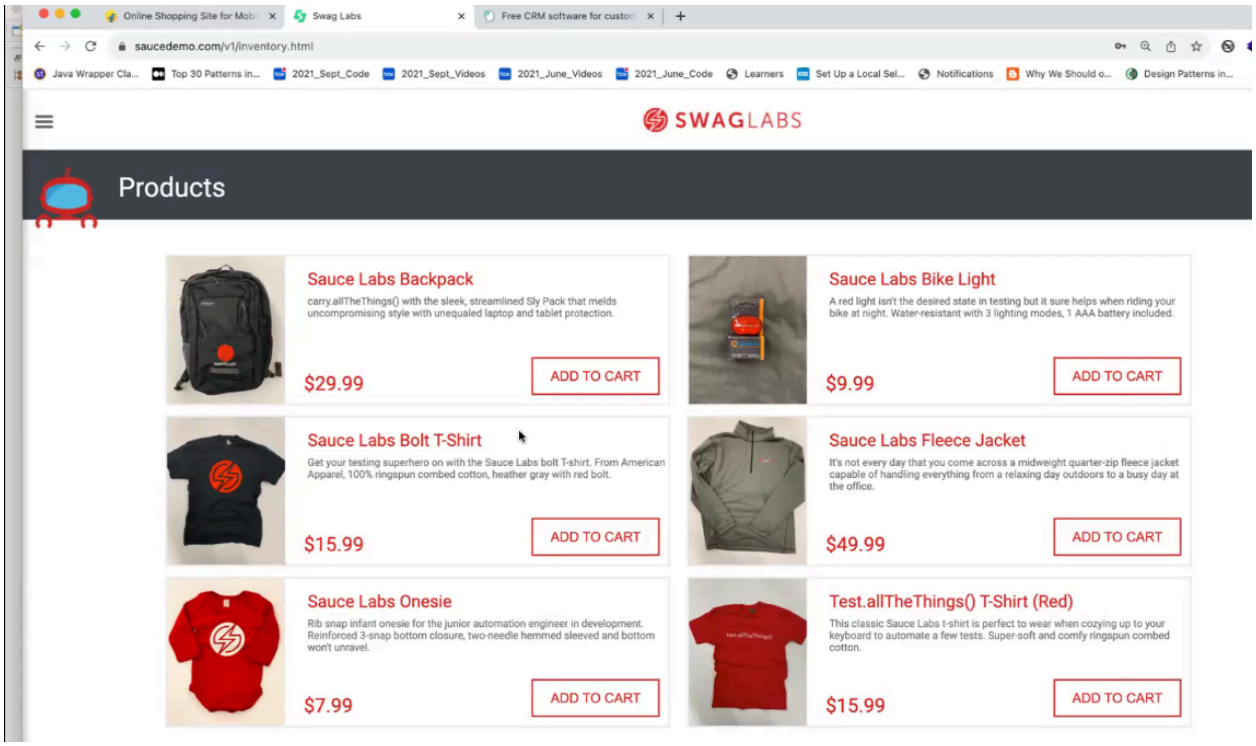


```
        //collect in list.
        List<WebElement> flpkartLinks = allLinks
                .stream()
                .filter(e -> !e.getText().isEmpty())
                .filter(e -> e.getText().startsWith("Flipkart"))
                .collect(Collectors.toList());

        System.out.println(flpkartLinks);


    }
```

```
Console
<terminated> WebDriverStreamsLambda [Java Application] C:\Users\karan\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_23.0.2.v20250131-0604\jre\bin\javaw.exe (Oct 8, 2025, 12:07:34 PM –
Oct 08, 2025 12:07:37 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
WARNING: Unable to find an exact match for CDP version 141, returning the closest version; found: 139; Pleas
[[[ChromeDriver: chrome on windows (ff19c3486579bf23676666b4b9ef5944)] -> tag name: a], [[ChromeDriver: chrc
```

Streams cons-
Time consuming.
For and foreach are the fastest.
Java says use stream for smaller data sets not millions of records.

Capture only price of all products-

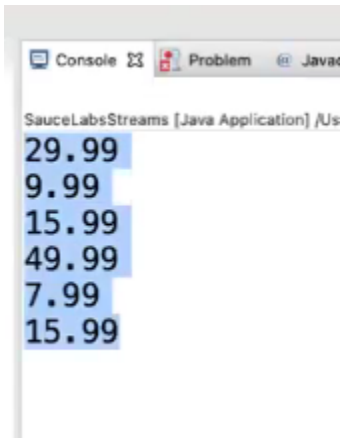


Capture price and print on console.

```java
package Streams;

import java.util.Comparator;
import java.util.List;
import java.util.stream.Collectors;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class SauceLabsStreams {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        WebDriver driver = new ChromeDriver();
        driver.get("https://www.saucedemo.com/v1/index.html");

        driver.findElement(By.id("user-name")).sendKeys("standard_user");
        driver.findElement(By.id("password")).sendKeys("secret_sauce");
        driver.findElement(By.id("login-button")).click();

        List<WebElement> pricesList = driver.findElements(By.cssSelector("div.inventory_item_price"));

        pricesList.stream().forEach(e -> System.out.println(e.getText()));


    }

}
```

Capture the value and ignore dollar-
Use substring and ignore dollar.
Later convert string to double.

```
27
28          //$29.99 --> "29.99" --> 29.99
29
30          pricesList.stream()
31                      .map(e -> Double.parseDouble(e.getText().substring(1)))
32                              .forEach(e -> System.out.println(e));
33
```



Now lets collect -
Before collecting sort in ascending order. Use sorted.
Collect returns list of double now.

```
29
30      List<Double> sorted_prices =    pricesList.stream()
31                                              .map(e -> Double.parseDouble(e.getText().substring(1)))
32                                                      .sorted()
33                                                              .collect(Collectors.toList());
34
35
36          System.out.println(sorted_prices);
37
```

```
SauceLabsStreams [Java Application] /Users/naveenautomationlabs/.p2/pool/plugins/org.eclipse.justj.openjdk.
[7.99, 9.99, 15.99, 15.99, 29.99, 49.99]
```

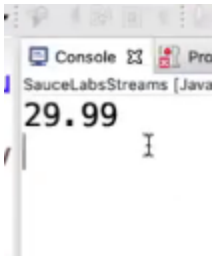
Sort in descending order-
Sorted has one parameter, comparator.reverseorder.

```
38
39          List<Double> sorted_prices_desc =   pricesList.stream()
40                  .map(e -> Double.parseDouble(e.getText().substring(1)))
41                          .sorted(Comparator.reverseOrder())
42                                  .collect(Collectors.toList());
43
44
45                  System.out.println(sorted_prices_desc);
46
```

```
Console  Problems  Javadoc  Declaration  Progress  Terminal  Results of r
SauceLabsStreams [Java Application]
[49.99, 29.99, 15.99, 15.99, 9.99, 7.99]
```

Capture all elements and give me the first element- Find first method to be used. Find first method returns optional<double>. We dont want optional double, we want actual values, so use get(). Get returns double here.

```
47
48          double firstPrice = pricesList.stream()
49                                  .map(e -> Double.parseDouble(e.getText().substring(1)))
50                                          .findFirst().get();
51
52          System.out.println(firstPrice);
```

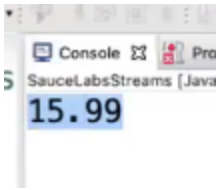


If we use collect, then store in list. Then use get(index) and get the value.

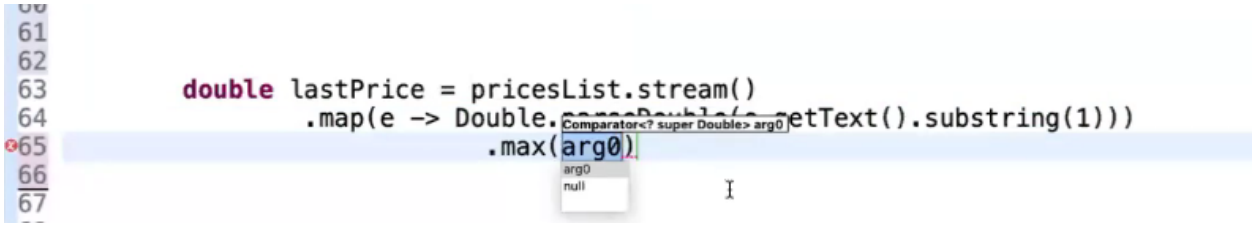
Find any method will give any random price the code likes.

Capture last value-
Use reduce.
Useful for manipulations.
Reduce takes two variable, what ever name you want to give is ok. Then we need the last value so pass the second variable after arrow, and use get to get the value. How reduce works internally, keeps swapping the first variable and second variable till end of list.
OR
Collect in list and use get(list.size-1).

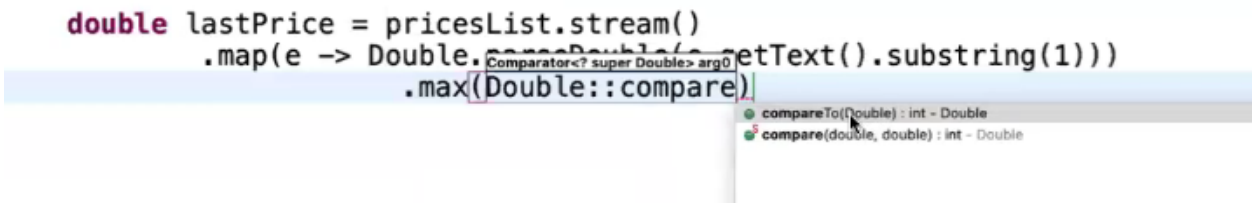
```
54
55          double lastPrice = pricesList.stream()
56                  .map(e -> Double.parseDouble(e.getText().substring(1)))
57                          .reduce((first, second) -> second).get();
58
59                                      System.out.println(lastPrice);
60
```



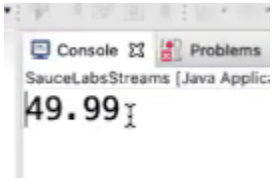
Return product with max price-
We can use sorted and then capture the lowest or highest value as per our requirement.
OR
What does the comparator<double> mean.



Use wrapper class double. After jdk1.8 if we want to use method of any class use **double::** this is called method expression. You dont have to pass double value and all here as parameter.compare to will compare internally and produce max or min as we want.

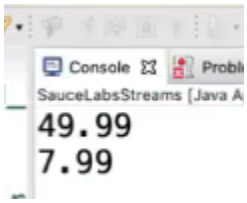


```
62
63          double maxPrice = pricesList.stream()
64                  .map(e -> Double.parseDouble(e.getText().substring(1)))
65                          .max(Double::compareTo).get();
66
67
68          System.out.println(maxPrice);
```



Get the min price-

```
70
71          double minPrice = pricesList.stream()
72                  .map(e -> Double.parseDouble(e.getText().substring(1)))
73                          .min(Double::compareTo).get();
74
75
76          System.out.println(minPrice);
77
```



Without using min and max-
Use Sorted.
Then capture first value for min and last value for max.

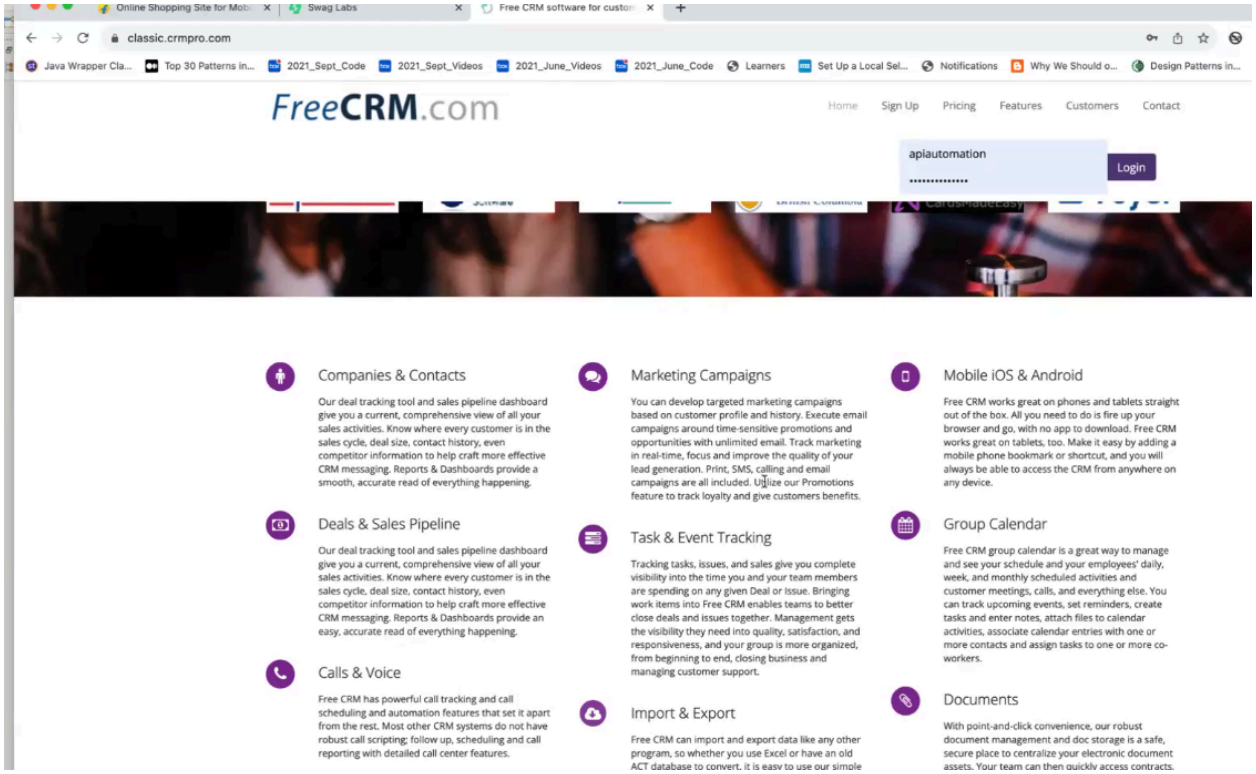
Home work-
Collect all add to cart buttons. Click on first, third and fifth.

Home work-
Capture all prices. If price is 15.99 then click add to cart. Stream will be complex. Go for simple scenarios with streams.

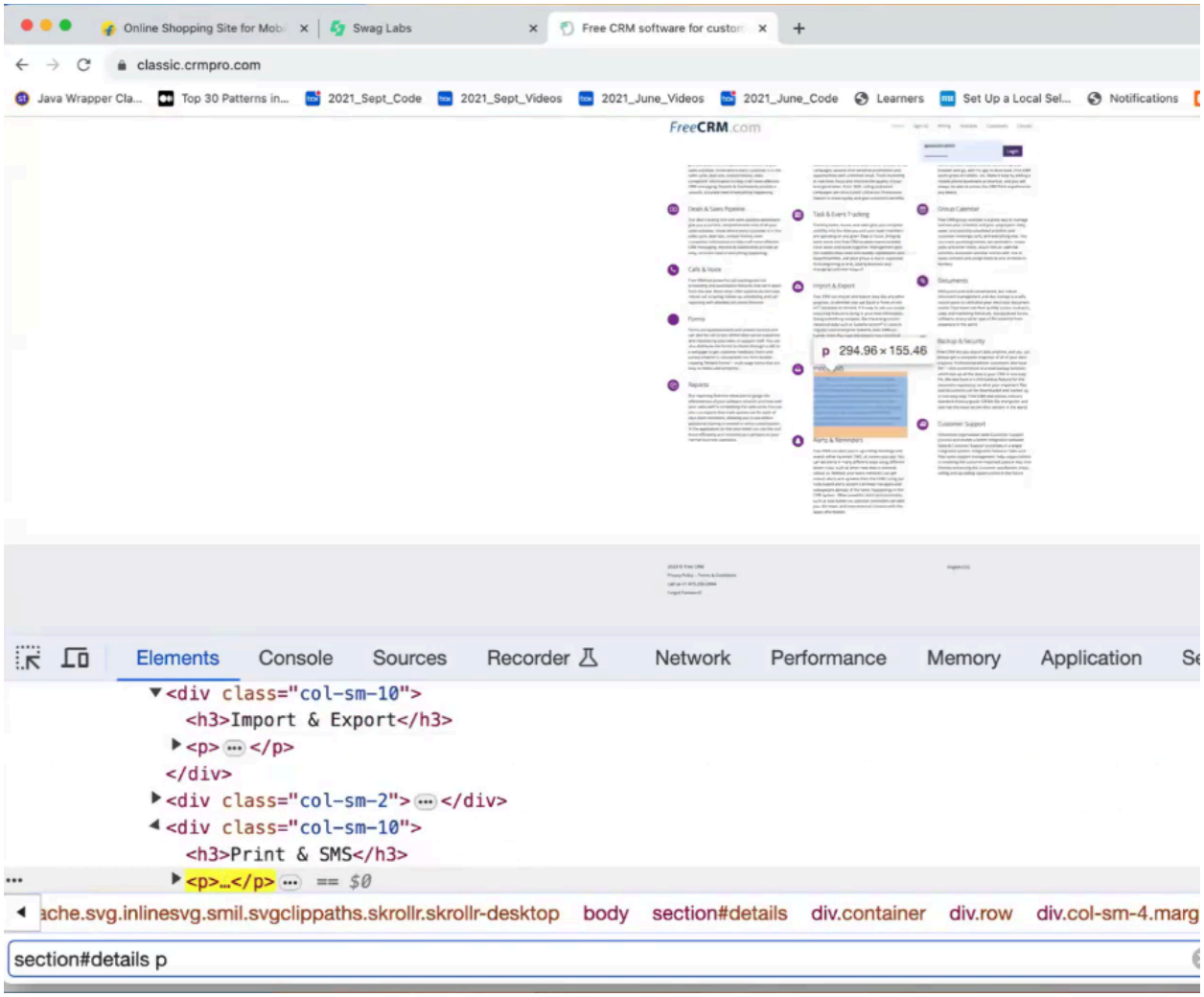
Home work-
Capture all prices. If price is 15.99 then click add to cart. Use Without streams. Use selenium 4 features, relative locators, and older concepts like sibling concept

Home work-
Give me all the items list where price less than 10 dollar. Apply filter and then Collect in list.
Streams only for list interface.49.00.

Join all paragraphs by line-
Make them as single string. Or join all headers into single string not list.



Collect all paragraphs, start from section parent and then move to para -



We dont want to use reduce function because we want all the elements not reduce the element count. So reduce is used when we want sum or reduce the original count and get the aggregate.
Joining method to join all paragraphs. if we use toList(), then again we get para in list format. Pass the separator. Collect returns string here.
See how naveen reads the return type of collect - he told it returns string.

```
paragraphs.stream()
                .map(e -> e.getText())
                        .collect(Collectors.joining("\n"));
```

<String, ?> String java.util.stream.Stream.collect(Collector<? super String, ?, String> arg0)

Note: The Javadoc for this element could neither be found in the attached source nor the attached Javadoc.

```
}
```

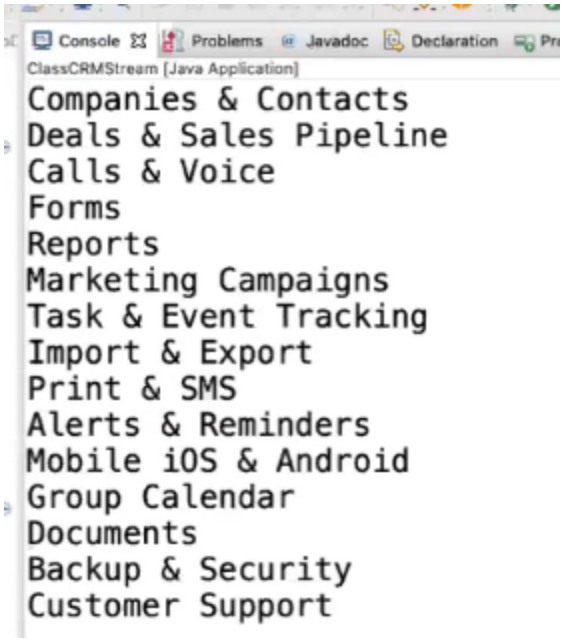
WebDriverStreamsLambda.java | SauceLabsStreams.java | ClassCRMStream.java

```java
package Streams;

import java.util.List;
import java.util.stream.Collectors;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class ClassCRMStream {

    public static void main(String[] args) {
        WebDriver driver = new ChromeDriver();
        driver.get("https://classic.crmpro.com/");

        List<WebElement> paragraphs = driver.findElements(By.cssSelector("section#details p"));

        String allPara = paragraphs.stream()
                        .map(e -> e.getText())
                                .collect(Collectors.joining("\n"));

        System.out.println(allPara);


    }

}
```

```
Our deal tracking tool and sales pipeline dashboard give you a current, comprehensive view of all your sales activities.
Our deal tracking tool and sales pipeline dashboard give you a current, comprehensive view of all your sales activities.
Free CRM has powerful call tracking and call scheduling and automation features that set it apart from the rest. Most of
Forms are questionnaires and answer surveys and can also be call scripts where data can be explained and inputted by you
Our reporting features allow you to gauge the effectiveness of your software solution and how well your sales staff is
You can develop targeted marketing campaigns based on customer profile and history. Execute email campaigns around time-
Tracking tasks, issues, and sales give you complete visibility into the time you and your team members are spending on a
Free CRM can import and export data like any other program, so whether you use Excel or have an old ACT database to conv
Free CRM supports direct mail and mail merge campaigns using our advanced Microsoft Word integration, which lets you cre
Free CRM can alert you to upcoming meetings and events either by email, SMS, or screen pop-ups. You can set alerts in ma
Free CRM works great on phones and tablets straight out of the box. All you need to do is fire up your browser and go, v
Free CRM group calendar is a great way to manage and see your schedule and your employees' daily, week, and monthly sche
With point-and-click convenience, our robust document management and doc storage is a safe, secure place to centralize y
Free CRM lets you export data anytime, and you can always get a complete snapshot of all of your data anytime. Professio
Streamline organization-wide Customer Support process and enable a better integration between Sales & Customer Support p
```

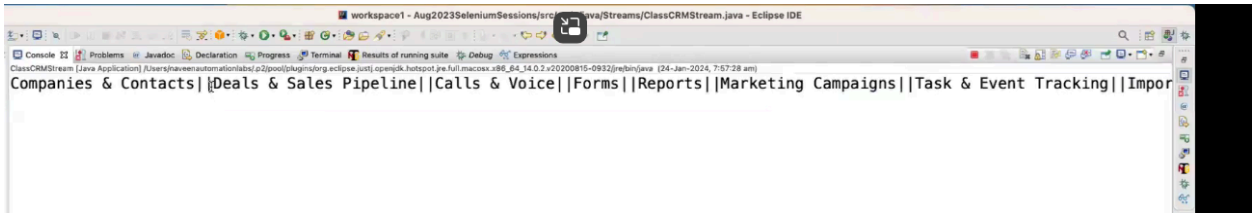
Capture all headers of the paragraph only-

```
        List<WebElement> headers = driver.findElements(By.cssSelector("section#details h3"));

        String allHeaders = headers.stream()
                        .map(e -> e.getText())
                                .collect(Collectors.joining("\n"));

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



Changed joining delimiter to pipe-

```
16
17        List<WebElement> headers = driver.findElements(By.cssSelector("section#details h3"));
18
19        String allHeaders = headers.stream()
20                        .map(e -> e.getText())
21                                .collect(Collectors.joining("||"));
22
```



Get the header name and the tag name-

This we are doing in one liner code.

//get tag name returns string.

//get text returns string.

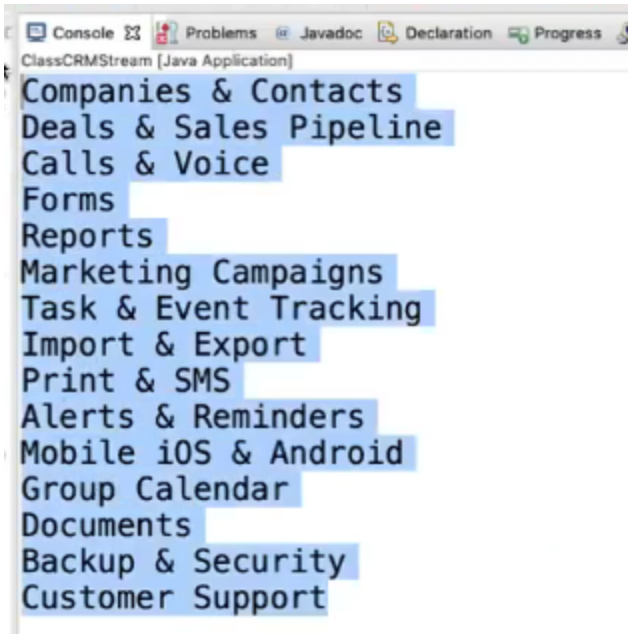
```
16
17        List<WebElement> headers = driver.findElements(By.cssSelector("section#details h3"));
18
19 //      String allHeaders = headers.stream()
20 //                      .map(e -> e.getText())
21 //                              .collect(Collectors.joining("||"));
22 //
23 //      System.out.println(allHeaders);
24
25
26
27        headers.stream().forEach(e -> System.out.println(e.getText() + " -- " + e.getTagName()));
28
```

```
ClassCRMStream [Java Application] /Users/naveenautomationlabs/.p2/pool/plugins/org.eclipse
Companies & Contacts -- h3
Deals & Sales Pipeline -- h3
Calls & Voice -- h3
Forms -- h3
Reports -- h3
Marketing Campaigns -- h3
Task & Event Tracking -- h3
Import & Export -- h3
Print & SMS -- h3
Alerts & Reminders -- h3
Mobile iOS & Android -- h3
Group Calendar -- h3
Documents -- h3
Backup & Security -- h3
Customer Support -- h3
```

Because **e** has webelement, we have to use **get text.** 58.00

Same code without any variables-
Get all the header names and print.

```
16
17        //List<WebElement> headers = driver.findElements(By.cssSelector("section#details h3"));
18
19 //      String allHeaders = headers.stream()
20 //                      .map(e -> e.getText())
21 //                              .collect(Collectors.joining("||"));
22 //
23 //      System.out.println(allHeaders);
24
25
26
27        //headers.stream().forEach(e -> System.out.println(e.getText() + " -- " + e.getTagName()));
28
29        driver.findElements(By.cssSelector("section#details h3"))
30                              .stream()
31                                      .forEach(e -> System.out.println(e.getText()));
32    }
```



Generic function-

For headers list.

Added 12.

Updated 14.

```
10
11 public class ClassCRMStream {
12     static WebDriver driver;
13     public static void main(String[] args) {
14         driver = new ChromeDriver();
15         driver.get("https://classic.crmpro.com/");
```

Print all headers static generic method-

```
36     public static void printHeadersList(By locator) {
37         driver.findElements(locator)
38                 .stream()
39                         .forEach(e -> System.out.println(e.getText()));
40     }
```

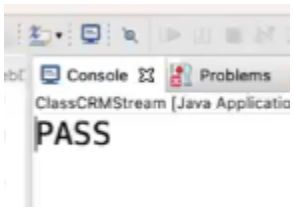
Get all headers static generic method-

We first have to convert webelement to text and then collect, else we will have webelements inside list.
1.05

```
42    public static List<String> getHeadersList(By locator) {
43        return driver.findElements(locator).stream().map(e -> e.getText()).collect(Collectors.toList());
44    }
```

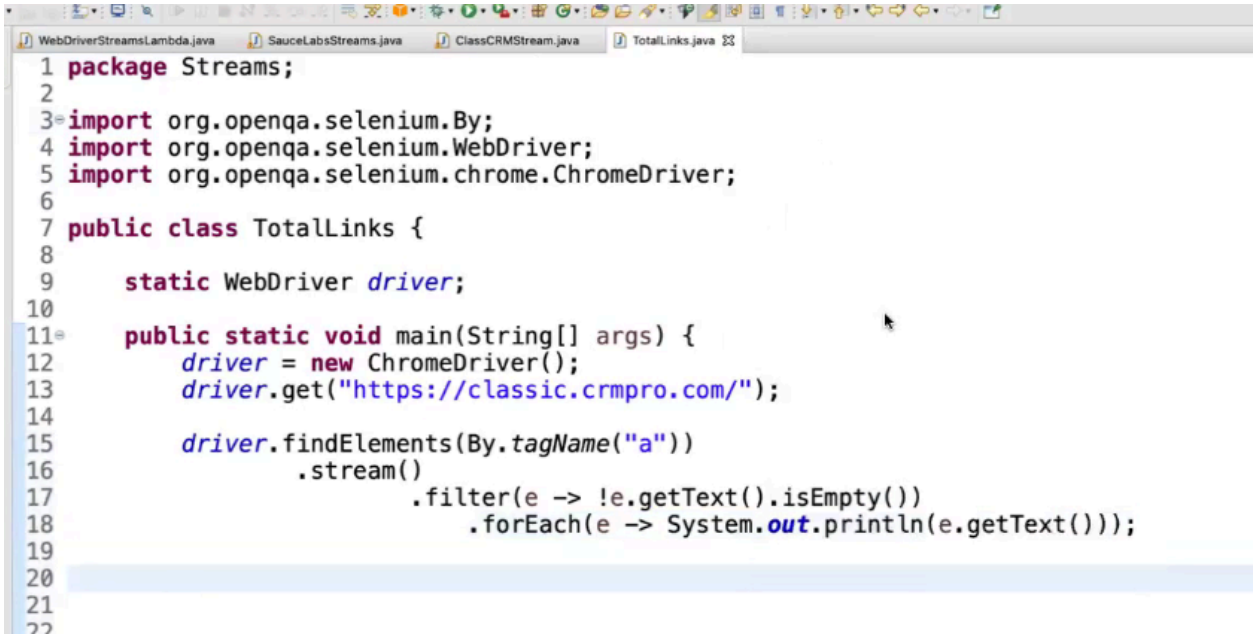
Call generic methods-
Just showing how to use the collected headers further. We are checking if header contains import and export.

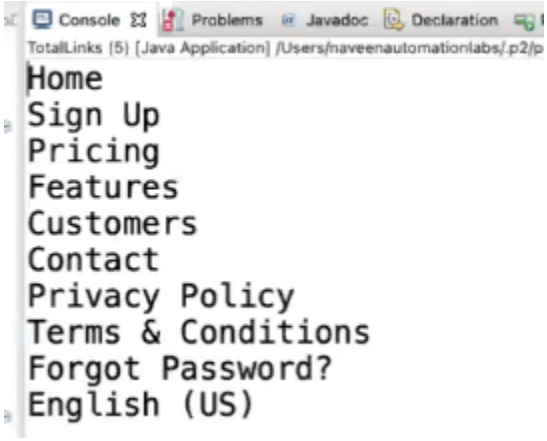
```
29
30          By headers = By.cssSelector("section#details h3");
31
32          if(getHeadersList(headers).contains("Import & Export")) {
33              System.out.println("PASS");
34          }
35
36      }
```



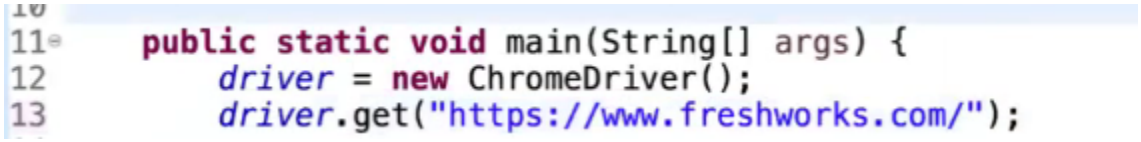
Cons–Difficult to debug also in streams.

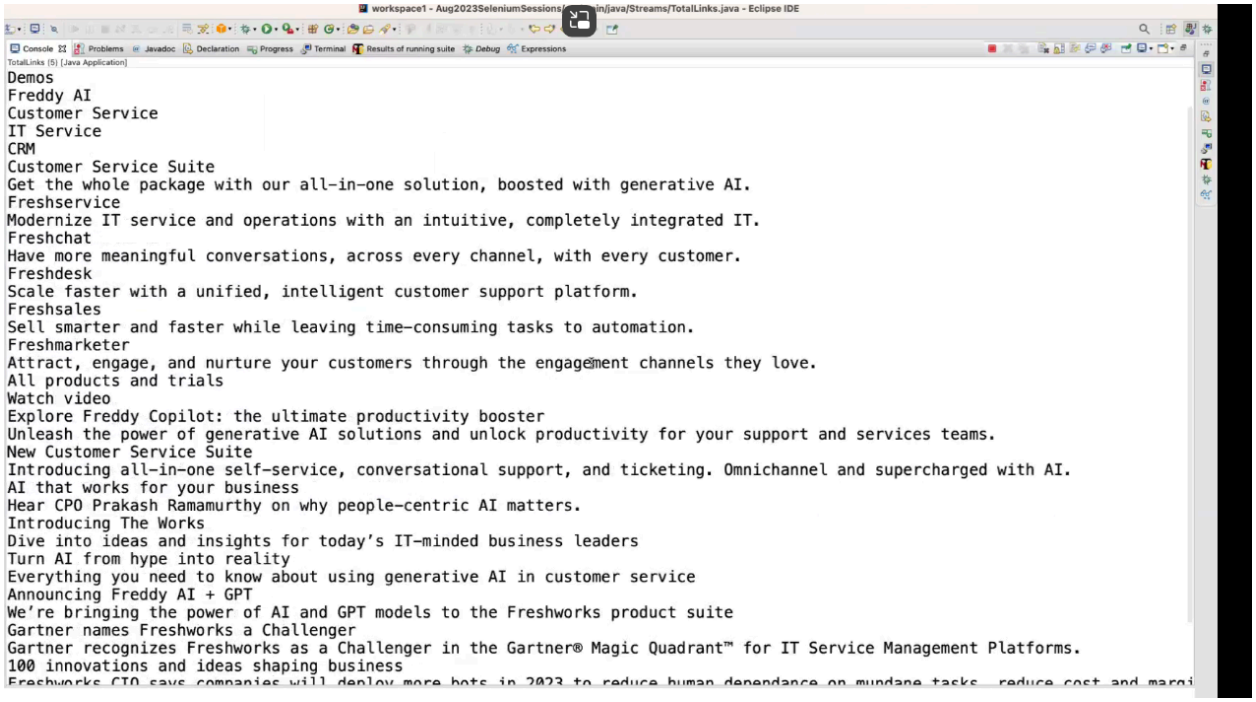
Count number of links-
Capture all non empty links.
Print them.





Changed website where we have more links-





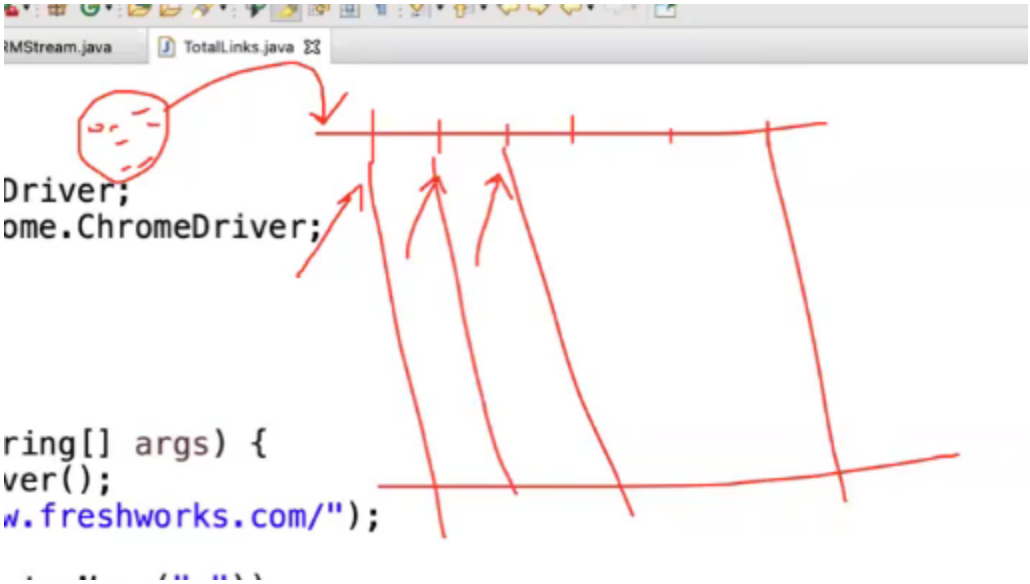
Parallel stream-

Asynchronous process.

Anybody from the stream runs whenever they want.

Sequential stream-

One element at a time. After everything completed on that element move to next element and so on.



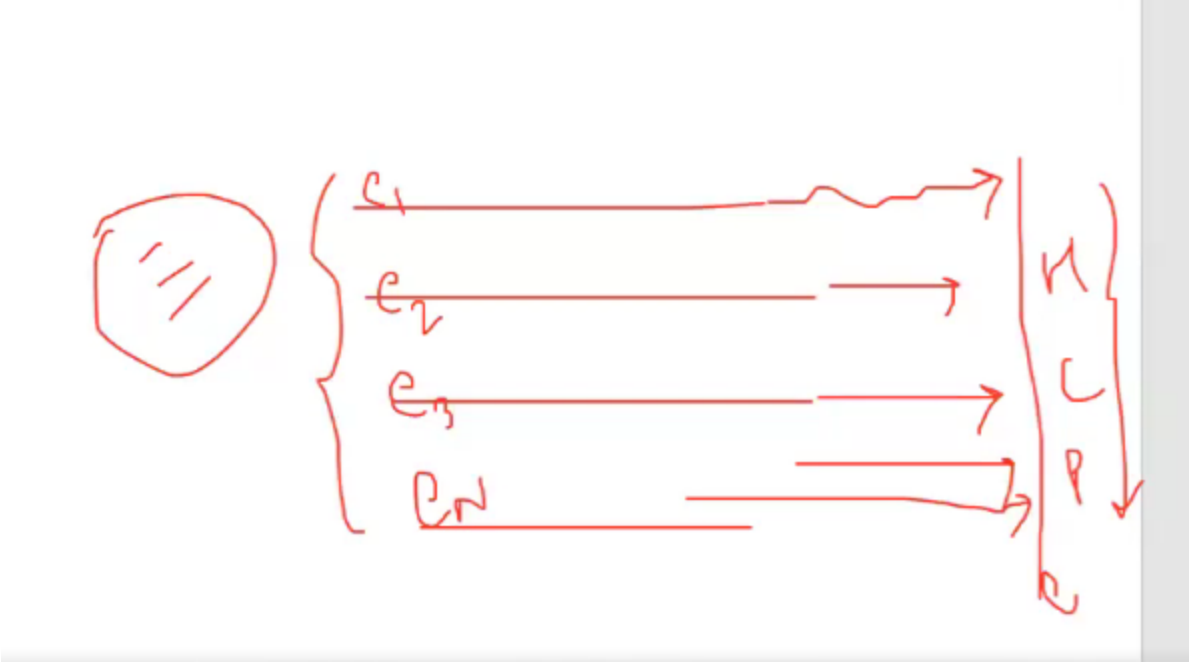
Every element completes operation and then next element comes.

Parallel stream-

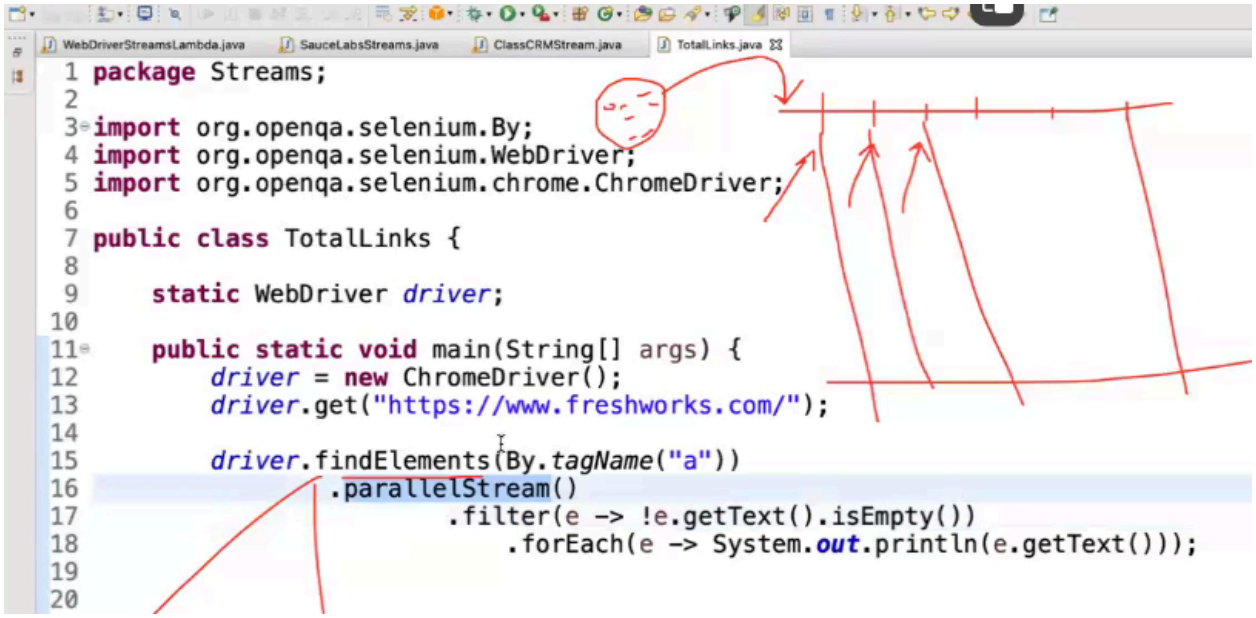
Race.

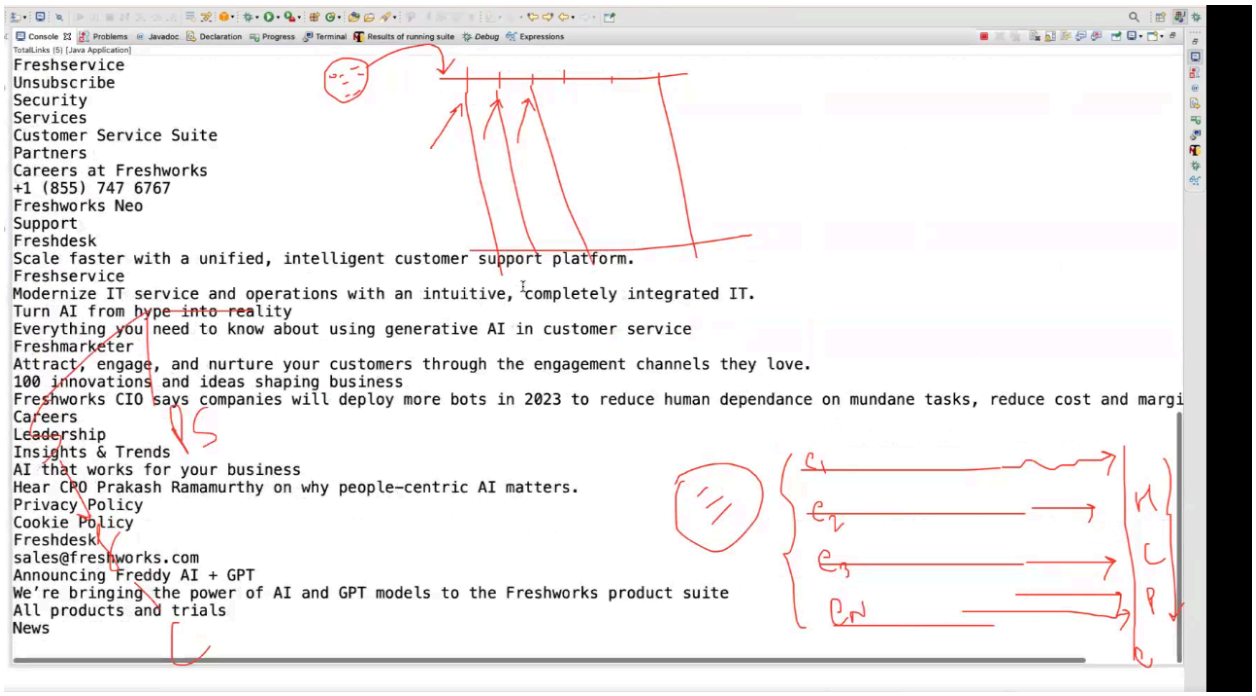
Anybody can come and finish and go.

Example, homepage finished first, then contact us page finished, then privacy page finished, then login page finished.



When sequence doesnt matter then we can use parallel. You are not bothered about order of what comes first or last.





Homework-

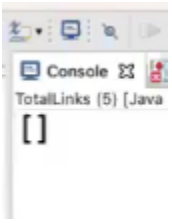
Use parallel streams and print all country names from orange crm.

Cons-
Lots of ram and cpu usage when data is more. If 1000 web elements, then for every web element a stream is created(1.16). Program may not work properly.

Create list and store data using streams-
Introducing list collection in between streams.

```
24          List<String> linksList = new ArrayList<String>();//pc=0
25
26          driver.findElements(By.tagName("a"))
27          .stream()
28                  .filter(e -> !e.getText().isEmpty())
29                      .map(e -> linksList.add(e.getText()));
30
31          System.out.println(linksList);
32      }
```

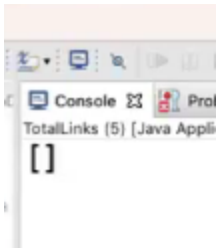
Getting empty list.



Updated code-

```
24          List<String> linksList = new ArrayList<String>();//pc=0
25
26          driver.findElements(By.tagName("a"))
27          .stream()
28                      .map(e -> linksList.add(e.getText()));
29
30          System.out.println(linksList);
31      }
```

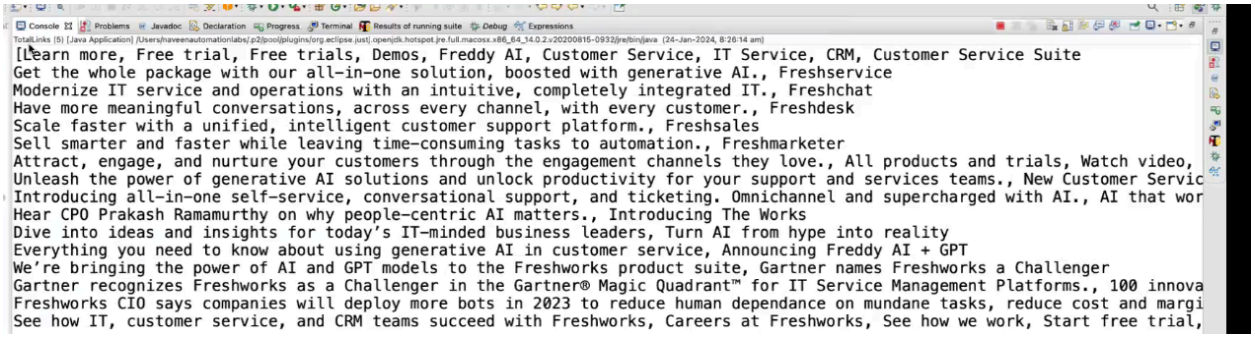
Still empty list.



Did something on code and made it work-
Used collect.
Then used for each on the list.
Grabbed text.
Added to our own list.
Printed our own list.

```
12 public class TotalLinks {
13
14     static WebDriver driver;
15
16     public static void main(String[] args) {
17         driver = new ChromeDriver();
18         driver.get("https://www.freshworks.com/");
19
20 //      driver.findElements(By.tagName("a"))
21 //              .stream()
22 //                      .filter(e -> !e.getText().isEmpty())
23 //                          .forEach(e -> System.out.println(e.getText()));
24
25
26         List<String> linksList = new ArrayList<String>();//pc=0
27
28
29
30
31                     List<WebElement> linksAllList= driver.findElements(By.tagName("a"))
32                                      .stream()
33                                              .filter(e -> !e.getText().isEmpty())
34                                                      .collect(Collectors.toList());
35
36                     linksAllList.forEach(e -> linksList.add(e.getText()));
37                     System.out.println(linksList);
38
39     }
```

First collect all things as a list. Then add to the list we created. And then print the list we created.



Sometimes this code gives stale element exception-

```
//lets collect it in list.
                List<WebElement> linksAllList =
driver.findElements(By.tagName("a"))
                .stream().filter(e->!e.getText().isEmpty())
                .collect(Collectors.toList());
                //then add to our list.
                linksAllList.forEach(e->linksList.add(e.getText()));
                //print our list
                System.out.println(linksList); //gives all link names
```

🧩 **Root cause**

`StaleElementReferenceException` means:

> The WebElement you're trying to access is no longer attached to the current DOM.

That usually happens when:

- The page **reloads or updates dynamically** after you fetched the elements.

- You stored a list of `WebElement`s (`linksAllList`) and then accessed their text later — but by then the page had re-rendered (for example, animations or lazy loading on `freshworks.com`).

So when this line runs:

```
linksAllList.forEach(e -> linksList.add(e.getText()));
```

the `e` objects refer to elements that may no longer exist in the current DOM snapshot.

✅ **How to fix it**

Option 1: Extract text immediately during streaming

This avoids holding references to WebElements.

```
List<String> linksList =
driver.findElements(By.tagName("a"))
    .stream()
    .filter(e -> !e.getText().isEmpty())
    .map(WebElement::getText)
    .collect(Collectors.toList());

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

→ Here we never store the WebElement itself, so it can’t go stale.
This is the **best and simplest** fix.

Option 2: Re-find elements before accessing them

If you really must process `WebElement`s later, re-fetch them right before you call `getText()`:

```
List<WebElement> linksAllList =
driver.findElements(By.tagName("a"))
    .stream()
    .filter(e -> !e.getText().isEmpty())
    .collect(Collectors.toList());

for (int i = 0; i < linksAllList.size(); i++) {
    WebElement refreshedElement =
driver.findElements(By.tagName("a")).get(i);
    linksList.add(refreshedElement.getText());
}
```

But that’s less efficient and generally unnecessary.

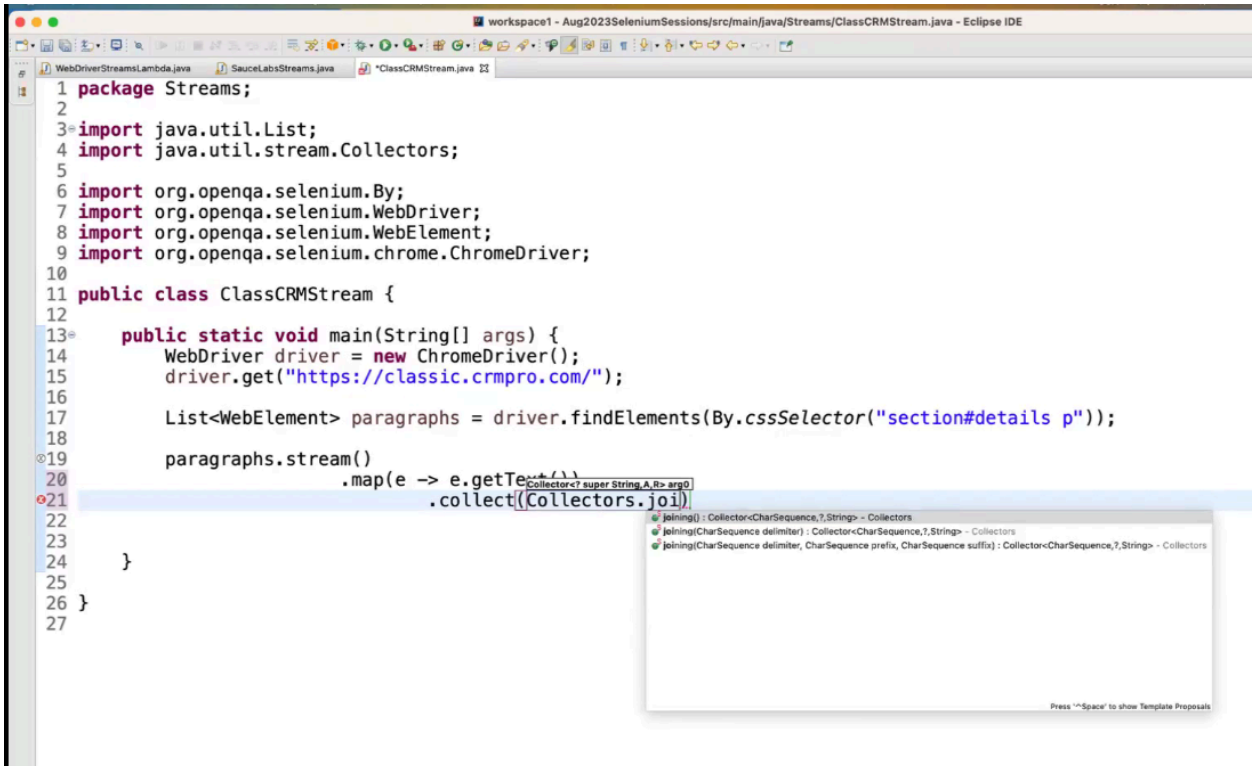
Option 3: Small wait for stability

If the page is still loading or rendering, insert a short wait:

Thread.sleep(2000); // 2 seconds

before collecting links.
Or use Selenium's `WebDriverWait` to wait for a stable condition (e.g. `ExpectedConditions.presenceOfAllElementsLocatedBy`).

Joining method parameters-
The parameters it can accept.
53.00



Clearer picture-