

Replacement for **for** and **for each** loop.

Used with collections only.

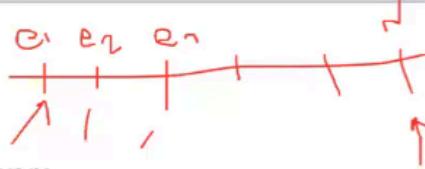
Applied mostly with lists.

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
newor
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
```



The screenshot shows an IDE interface with a central code editor window. The window displays a series of product names listed vertically. At the top of the window, there are several tabs: 'Console' (highlighted in blue), 'Problems', 'Javadoc', 'Declaration', and 'Progress'. Below the tabs, the title bar reads 'WebDriverStreamsLambda [Java Application] /Users/naveenautomationlabs/'. The main content area contains the following text:

```
Best Air Cooler
Bags
Hitachi Refrigerator 3 Door
Books
Toys
Candles
Helmets
Wall Clocks
Baby Food
Chocolates
Cycles
Calculators
Lipsticks
Mask
Vertiv UPS
Fastrack Watches
Wallets
Earrings
Gold Coins
Realme Pad Mini
Handbags
conekt SW2 Smartwatch
Mivi DuoPods a350
Speaker Cleaner
FURNITURE
Furniture
Sofas
Beds
Dining sets
Wardrobes
Mattresses
TV Units
Tables
Chairs
Shelves
Bean Bags
```

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

//prints all internal ids of webelements.

The screenshot shows the Eclipse IDE interface. The top part is the code editor with the file 'WebDriverStreamsLambda.java' open. The code uses Java 8 streams to print all links from a Flipkart page. The bottom part is the 'Console' tab, which shows the output of the program: a list of URLs found on the page.

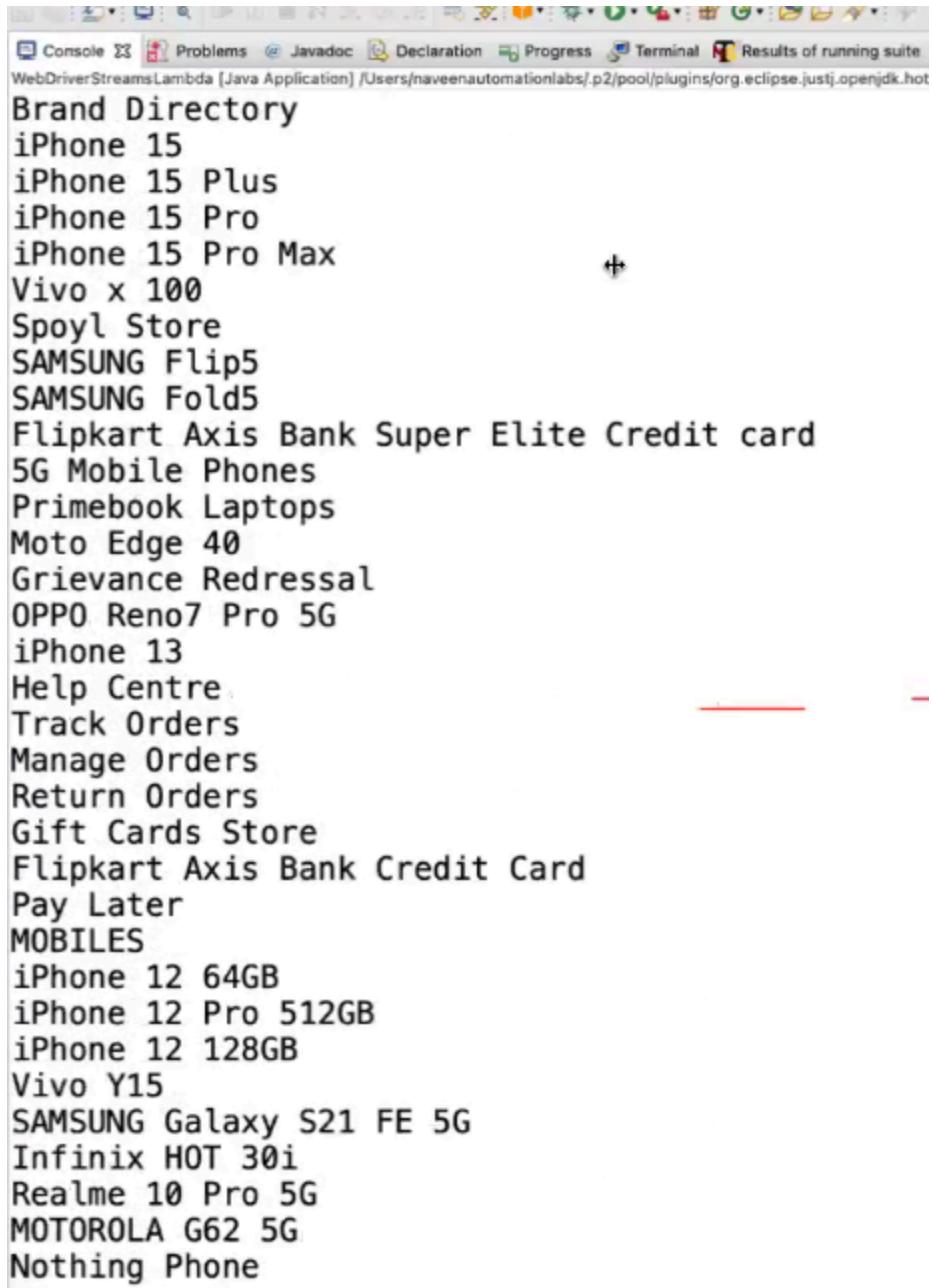
```
(WebDriverStreamsLambda.java)
6 import org.openqa.selenium.WebDriver;
7 import org.openqa.selenium.WebElement;
8 import org.openqa.selenium.chrome.ChromeDriver;
9
10 public class WebDriverStreamsLambda {
11
12    public static void main(String[] args) {
13
14        WebDriver driver=new ChromeDriver();
15        driver.get("https://www.flipkart.com");
16
17        List<WebElement> allLinks = driver.findElements(By.tagName("a"));
18
19        allLinks.stream().forEach(e->System.out.println(e));
20
21    }
22
23 }
```

```
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\java
[[ChromeDriver: chrome on windows (d97b6643eb68c1d57f74d99087d1a7f2)] -> tag name: a]
[[ChromeDriver: chrome on windows (d97b6643eb68c1d57f74d99087d1a7f2)] -> tag name: a]
[[ChromeDriver: chrome on windows (d97b6643eb68c1d57f74d99087d1a7f2)] -> tag name: a]
[[ChromeDriver: chrome on windows (d97b6643eb68c1d57f74d99087d1a7f2)] -> tag name: a]
[[ChromeDriver: chrome on windows (d97b6643eb68c1d57f74d99087d1a7f2)] -> tag name: a]
[[ChromeDriver: chrome on windows (d97b6643eb68c1d57f74d99087d1a7f2)] -> tag name: a]
[[ChromeDriver: chrome on windows (d97b6643eb68c1d57f74d99087d1a7f2)] -> tag name: a]
```

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.

```
22
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()));
```



The screenshot shows the Eclipse IDE interface with the 'Console' tab selected. The console window displays a list of URLs, likely generated by a script or application. The URLs include various product names and categories from e-commerce websites like Flipkart, Amazon, and other stores. Some URLs are preceded by a '+' sign, indicating they are expandable or part of a larger list.

```
Console Problems Javadoc Declaration Progress Terminal Results of running suite
WebDriverStreamsLambda [Java Application] /Users/naveenautomationlabs/.p2/pool/plugins/org.eclipse.justj.openjdk.hot
Brand Directory
iPhone 15
iPhone 15 Plus
iPhone 15 Pro
iPhone 15 Pro Max +
Vivo x 100
Spoyl Store
SAMSUNG Flip5
SAMSUNG Fold5
Flipkart Axis Bank Super Elite Credit card
5G Mobile Phones
Primebook Laptops
Moto Edge 40
Grievance Redressal
OPPO Reno7 Pro 5G
iPhone 13
Help Centre
Track Orders
Manage Orders
Return Orders
Gift Cards Store
Flipkart Axis Bank Credit Card
Pay Later
MOBILES
iPhone 12 64GB
iPhone 12 Pro 512GB
iPhone 12 128GB
Vivo Y15
SAMSUNG Galaxy S21 FE 5G
Infinix HOT 30i
Realme 10 Pro 5G
MOTOROLA G62 5G
Nothing Phone
```

Apply as many filters as you like-

Filter to get links which are not empty and starting with word flipkart.

```

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

```

```

Console Problems Javadoc Declaration Progress Terminal Results of Debug E
WebDriverStreamsLambda [Java Application] /Users/naveenautomationlabs/p2/pool/plugins/org.eclipse.justj.openjdk.hotspo
Flipkart Axis Bank Super Elite Credit card
Flipkart Axis Bank Credit Card
Flipkart Stories
Flipkart Wholesale

```

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 List<WebElement> flpkartLinks= allLinks
36     .stream()
37         .filter(e -> !e.getText().isEmpty())
38             .filter(e -> e.getText().startsWith("Flipkart"))
39                 .collect(Collectors.toList());
40
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());

```

```

Console Problems Javadoc Declaration Progress Terminal Results of running suite Debug Expressions
WebDriverStreamsLambda [Java Application] /Users/naveenautomationlabs/p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.x86_64_14.0.2.v20200815-0912/rebin/java (24-Jan-2024, 7:15:54 am)
[Flipkart Axis Bank Super Elite Credit card, Flipkart Axis Bank Credit Card, Flipkart Stories, Flipkart Wholesale
4

```

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

The screenshot shows an IDE interface with a code editor and a console window. The code editor contains Java code demonstrating the use of streams to filter and print web element links. The console window shows the output of the execution, which includes Selenium logs and the printed list of filtered links.

```
37     //collect in list.
38     List<WebElement> flpkartLinks = allLinks
39         .stream()
40         .filter(e -> !e.getText().isEmpty())
41         .filter(e -> e.getText().startsWith("Flipkart"))
42         .collect(Collectors.toList());
43
44     System.out.println(flpkartLinks);
45
46
47 }
```

Console Output:

```
<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
```

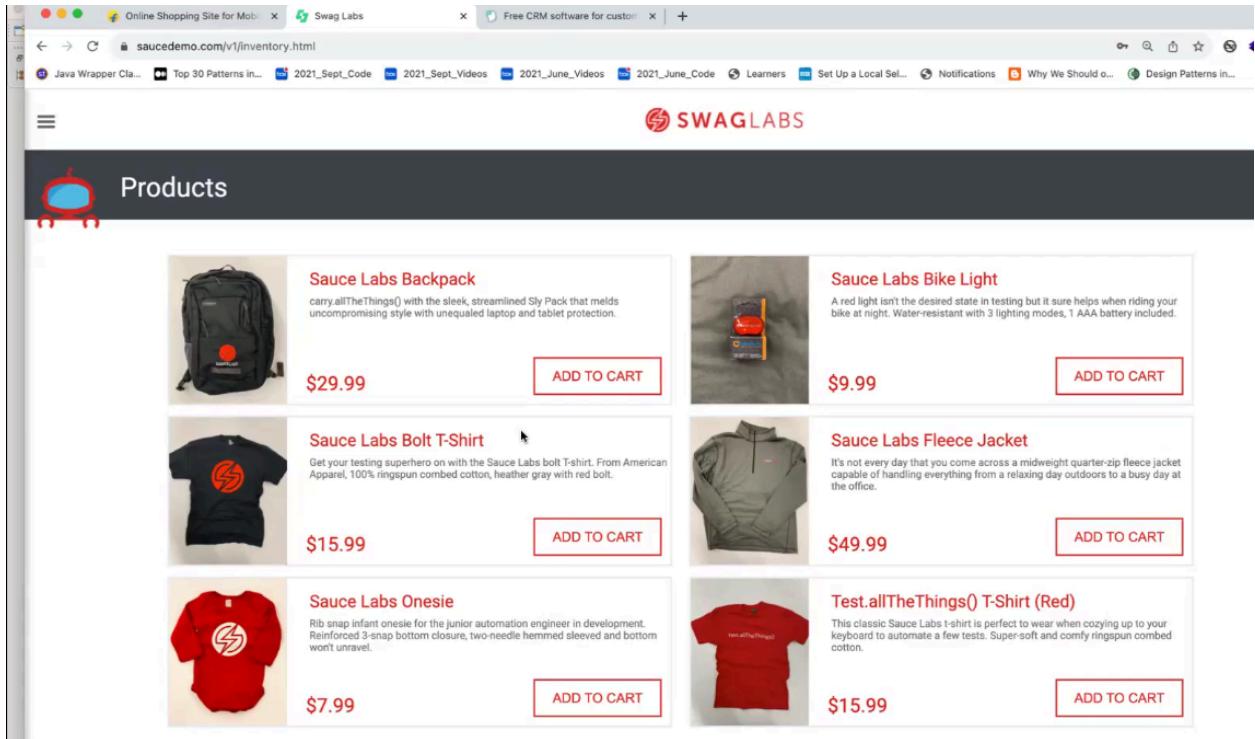
Streams cons-

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.

```

1 package Streams;
2
3 import java.util.Comparator;
4 import java.util.List;
5 import java.util.stream.Collectors;
6
7 import org.openqa.selenium.By;
8 import org.openqa.selenium.WebDriver;
9 import org.openqa.selenium.WebElement;
10 import org.openqa.selenium.chrome.ChromeDriver;
11
12 public class SauceLabsStreams {
13
14     public static void main(String[] args) {
15         // TODO Auto-generated method stub
16         WebDriver driver = new ChromeDriver();
17         driver.get("https://www.saucedemo.com/v1/index.html");
18
19         driver.findElement(By.id("user-name")).sendKeys("standard_user");
20         driver.findElement(By.id("password")).sendKeys("secret_sauce");
21         driver.findElement(By.id("login-button")).click();
22
23         List<WebElement> pricesList = driver.findElements(By.cssSelector("div.inventory_item_price"));
24
25         pricesList.stream().forEach(e -> System.out.println(e.getText()));
26
27
28     }
29
30 }

```



```
SauceLabsStreams [Java Application]
$29.99
$9.99
$15.99
$49.99
$7.99
$15.99
```

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
```



```
SauceLabsStreams [Java Application] /Users/...
29.99
9.99
15.99
49.99
7.99
15.99
```

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

```

SauceLabsStreams [Java Application]
[49.99, 29.99, 15.99, 15.99, 9.99, 7.99]

Capture all elements and give me the first element-

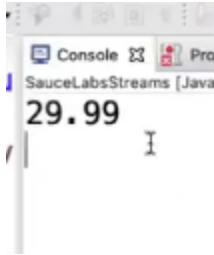
Find first method to be used. Find first method

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
```

A screenshot of a Java IDE's console window. The title bar says "SauceLabsStreams [Java]". The console area displays the number "15.99" in a blue-highlighted box.

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.

A screenshot of a Java IDE showing a piece of code. Line 65 contains ".max(arg0)". A tooltip appears over "arg0", showing the type "Comparator<? super Double> arg0" and two methods: "arg0" and "null".

```
60  
61  
62  
63     double lastPrice = pricesList.stream()  
64         .map(e -> Double.parseDouble(setText().substring(1)))  
65     .max(arg0)  
66     .arg0  
67     null
```

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.

```
double lastPrice = pricesList.stream()  
    .map(e -> Double.parseDouble(setText().substring(1)))  
    .max(Double::compare)
```

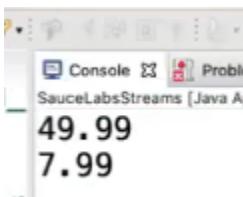
A screenshot of a Java IDE showing a tooltip for the "Double::compare" method. It shows two signatures: "compareTo(Double) : int - Double" and "compare(double, double) : int - Double".

```
61  
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-

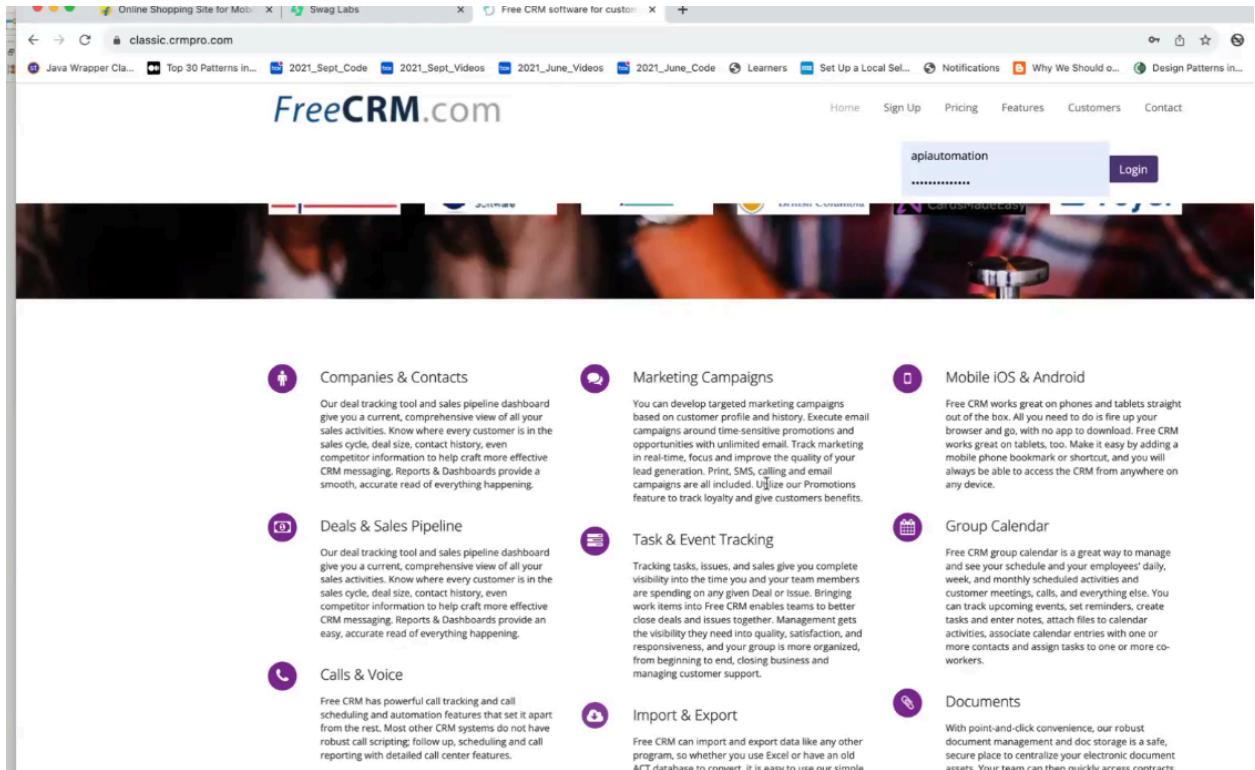
Capture all prices. If price is 15.99 then click add to cart. Use Without streams. Use selenium 4 features, relative locators, and older concepts like sibling concept

Home work-

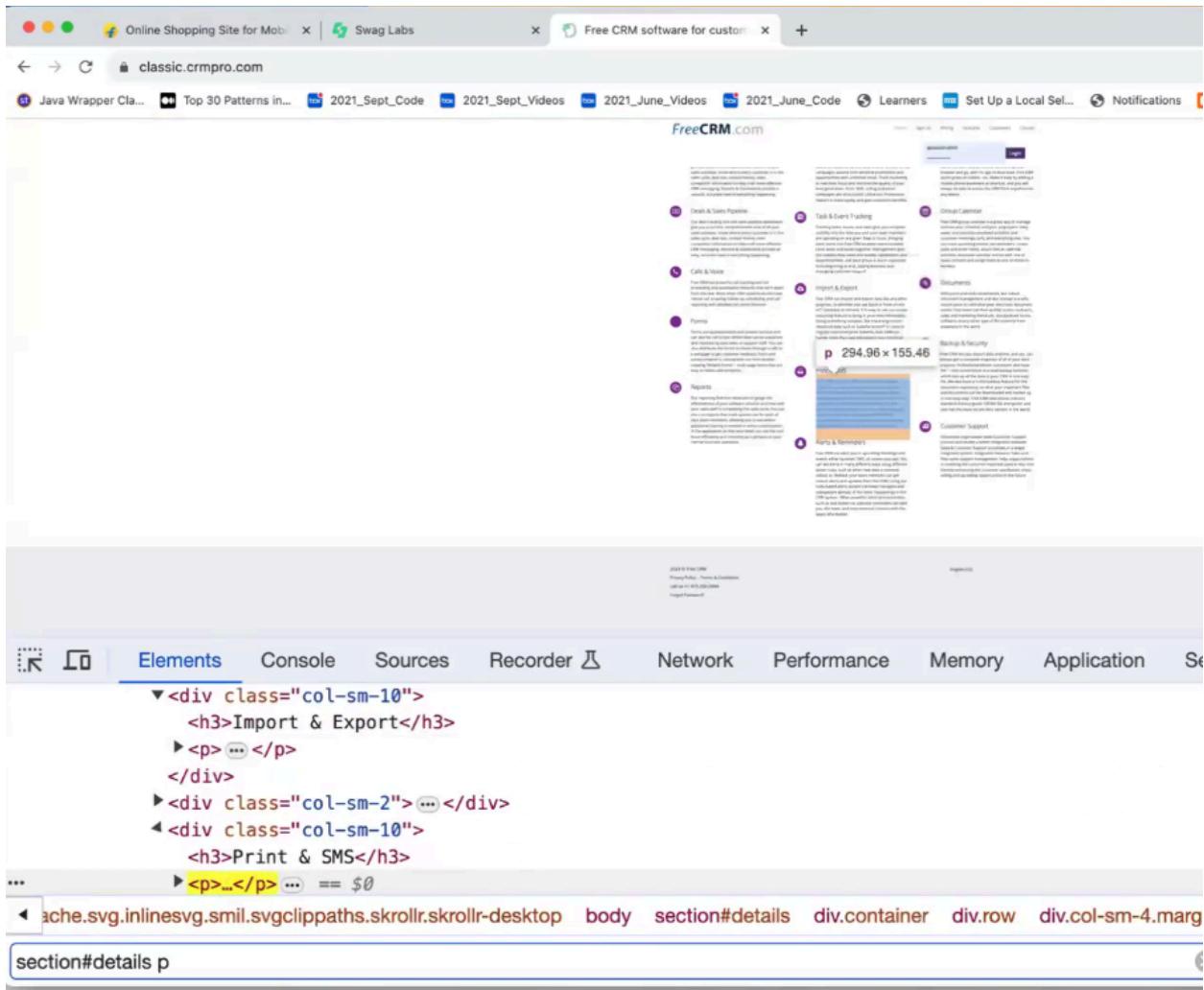
Give me all the items list where price less than 10 dollar. Apply filter and then Collect in list.
Streams only for list interface.49.00.

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 don't 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"));
}

Note: The Javadoc for this element could neither be found in the attached source nor the attached Javadoc.
Press F2 for focus

```

```

1 package Streams;
2
3 import java.util.List;
4 import java.util.stream.Collectors;
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 ClassCRMStream {
12
13     public static void main(String[] args) {
14         WebDriver driver = new ChromeDriver();
15         driver.get("https://classic.crmpro.com/");
16
17         List<WebElement> paragraphs = driver.findElements(By.cssSelector("section#details p"));
18
19         String allPara = paragraphs.stream()
20             .map(e -> e.getText())
21             .collect(Collectors.joining("\n"));
22
23         System.out.println(allPara);
24
25     }
26
27
28 }
29

```

```

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("\n"));
22
23     System.out.println(allHeaders);
24

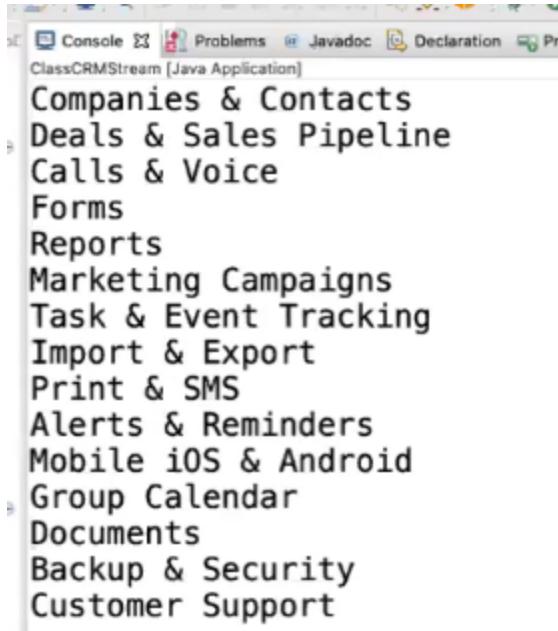
```

Capture all headers of the paragraph only-

```

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("\n"));
22
23     System.out.println(allHeaders);
24

```



Changed joining delimiter to pipe-

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

```

```

1 Companies & Contacts -- h3
2 Deals & Sales Pipeline -- h3
3 Calls & Voice -- h3
4 Forms -- h3
5 Reports -- h3
6 Marketing Campaigns -- h3
7 Task & Event Tracking -- h3
8 Import & Export -- h3
9 Print & SMS -- h3
0 Alerts & Reminders -- h3
1 Mobile iOS & Android -- h3
2 Group Calendar -- h3
3 Documents -- h3
4 Backup & Security -- h3
5 Customer Support -- h3
6
7

```

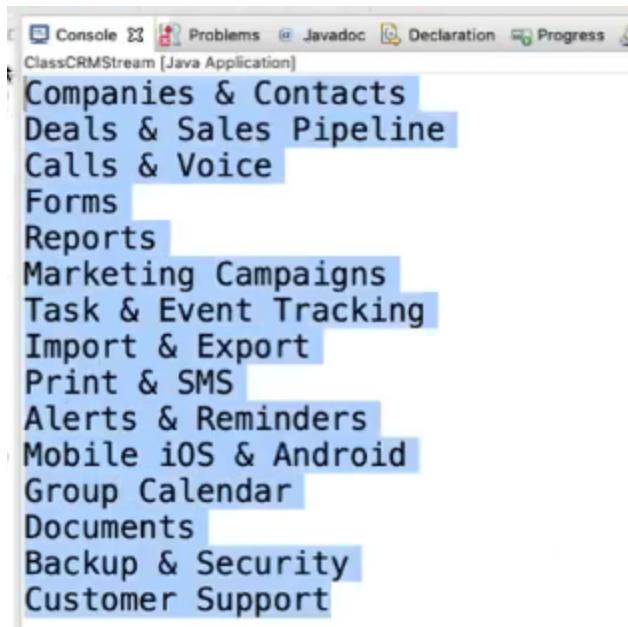
Because **e** has webelement, we have to use **get text**. 58.00

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     }
41
```

Get all headers static generic method-

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

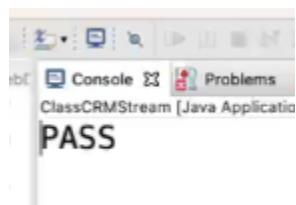
1.05

```
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.

The screenshot shows an IDE interface with several tabs at the top: WebDriverStreamsLambda.java, SauceLabsStreams.java, ClassCRMStream.java, and TotalLinks.java. The TotalLinks.java tab is active, displaying the following Java code:

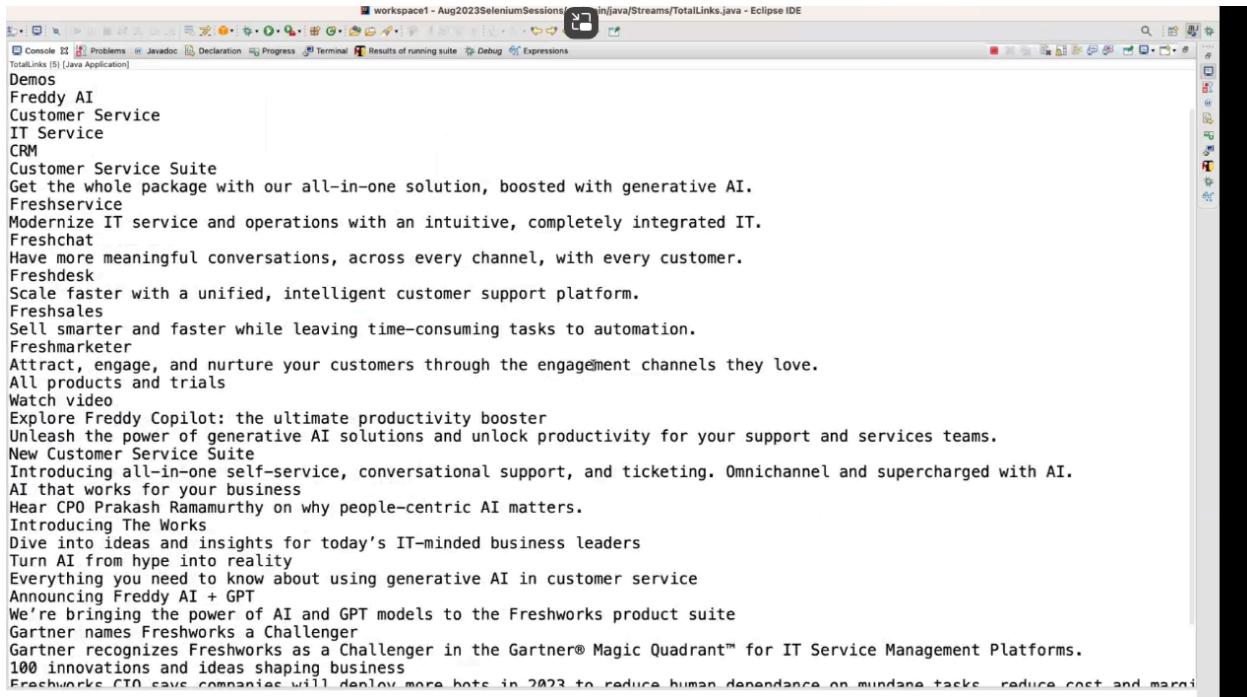
```
1 package Streams;
2
3 import org.openqa.selenium.By;
4 import org.openqa.selenium.WebDriver;
5 import org.openqa.selenium.chrome.ChromeDriver;
6
7 public class TotalLinks {
8
9     static WebDriver driver;
10
11    public static void main(String[] args) {
12        driver = new ChromeDriver();
13        driver.get("https://classic.crmpro.com/");
14
15        driver.findElements(By.tagName("a"))
16            .stream()
17            .filter(e -> !e.getText().isEmpty())
18            .forEach(e -> System.out.println(e.getText()));
19
20    }
21
22}
```

Below the code editor, there is a preview of the webpage loaded in the browser. The URL is "https://classic.crmpro.com/". The page content includes:

- Home
- Sign Up
- Pricing
- Features
- Customers
- Contact
- Privacy Policy
- Terms & Conditions
- Forgot Password?
- English (US)

Changed website where we have more links-

```
11    public static void main(String[] args) {
12        driver = new ChromeDriver();
13        driver.get("https://www.freshworks.com/");
```



The screenshot shows the Eclipse IDE interface with the title bar "workspace1 - Aug2023SeleniumSessions in java/Streams/TotalLinks.java - Eclipse IDE". The code editor displays the following Java code:

```
workspace1 - Aug2023SeleniumSessions in java/Streams/TotalLinks.java - Eclipse IDE
TotalLinks (5) [Java Application]
Console Problems Javadoc Declaration Progress Terminal Results of running suite Debug Expressions
Demos
Freddy AI
Customer Service
IT Service
CRM
Customer Service Suite
Get the whole package with our all-in-one solution, boosted with generative AI.
Freshservice
Modernize IT service and operations with an intuitive, completely integrated IT.
Freshchat
Have more meaningful conversations, across every channel, with every customer.
Freshdesk
Scale faster with a unified, intelligent customer support platform.
Freshsales
Sell smarter and faster while leaving time-consuming tasks to automation.
Freshmarketeer
Attract, engage, and nurture your customers through the engagement channels they love.
All products and trials
Watch video
Explore Freddy Copilot: the ultimate productivity booster
Unleash the power of generative AI solutions and unlock productivity for your support and services teams.
New Customer Service Suite
Introducing all-in-one self-service, conversational support, and ticketing. Omnichannel and supercharged with AI.
AI that works for your business
Hear CPO Prakash Ramamurthy on why people-centric AI matters.
Introducing The Works
Dive into ideas and insights for today's IT-minded business leaders
Turn AI from hype into reality
Everything you need to know about using generative AI in customer service
Announcing Freddy AI + GPT
We're bringing the power of AI and GPT models to the Freshworks product suite
Gartner names Freshworks a Challenger
Gartner recognizes Freshworks as a Challenger in the Gartner® Magic Quadrant™ for IT Service Management Platforms.
100 innovations and ideas shaping business
Freshworks CTO says companies will deploy more bots in 2023 to reduce human dependence on mundane tasks, reduce cost and marci
```

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.

```
Driver;  
ome.ChromeDriver;  
  
ring[] args) {  
ver();  
w.freshworks.com/");  
...  
}
```

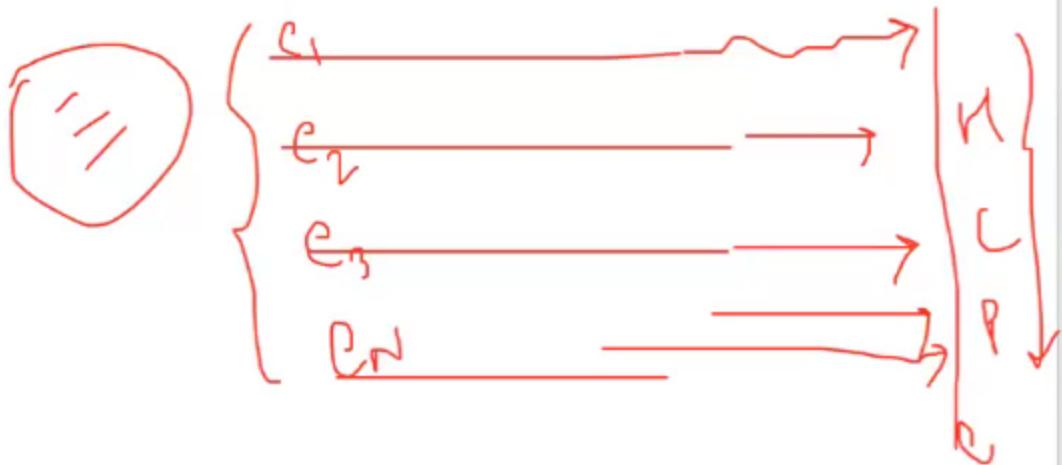
Every element completes operation and then next element comes.

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 doesn't matter then we can use parallel. You are not bothered about order of what comes first or last.

```

1 package Streams;
2
3 import org.openqa.selenium.By;
4 import org.openqa.selenium.WebDriver;
5 import org.openqa.selenium.chrome.ChromeDriver;
6
7 public class TotalLinks {
8
9     static WebDriver driver;
10
11    public static void main(String[] args) {
12        driver = new ChromeDriver();
13        driver.get("https://www.freshworks.com/");
14
15        driver.findElements(By.tagName("a"))
16            .parallelStream()
17            .filter(e -> !e.getText().isEmpty())
18            .forEach(e -> System.out.println(e.getText()));
19
20    }

```

```

Freshservice
Unsubscribe
Security
Services
Customer Service Suite
Partners
Careers at Freshworks
+1 (855) 747 6767
Freshworks Neo
Support
Freshdesk
Scale faster with a unified, intelligent customer support platform.
Freshservice
Modernize IT service and operations with an intuitive, completely integrated IT.
Turn AI from hype into reality
Everything you need to know about using generative AI in customer service
Freshmarketer
Attract, engage, and nurture your customers through the engagement channels they love.
100+ innovations and ideas shaping business
Freshworks CIO says companies will deploy more bots in 2023 to reduce human dependence on mundane tasks, reduce cost and margin
Careers
Leadership
Insights & Trends
AI that works for your business
Hear CMO Prakash Ramamurthy on why people-centric AI matters.
Privacy Policy
Cookie Policy
Freshdesk
sales@freshworks.com
Announcing Freddy AI + GPT
We're bringing the power of AI and GPT models to the Freshworks product suite
All products and trials
News

```

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

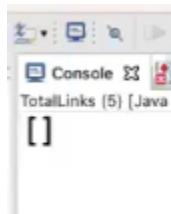
Cons-

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 }
```

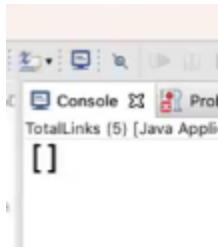
Getting empty list.



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.

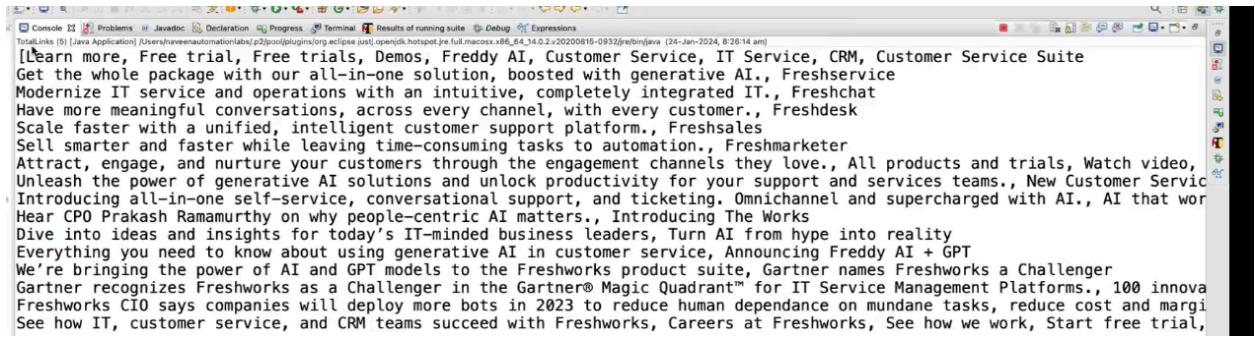
Grabbed text.

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     List<WebElement> linksAllList= driver.findElements(By.tagName("a"))
31             .stream()
32                 .filter(e -> !e.getText().isEmpty())
33                     .collect(Collectors.toList());
34
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.



```
[Learn more, Free trial, Trials, Demos, Freddy AI, Customer Service, IT Service, CRM, Customer Service Suite
Get the whole package with our all-in-one solution, boosted with generative AI., Freshservice
Modernize IT service and operations with an intuitive, completely integrated IT., Freshchat
Have more meaningful conversations, across every channel, with every customer., Freshdesk
Scale faster with unified, intelligent customer support platform., Freshsales
Sell smarter and faster while leaving time-consuming tasks to automation., Freshmarketeer
Attract, engage, and nurture your customers through the engagement channels they love., All products and trials, Watch video,
Unleash the power of generative AI solutions and unlock productivity for your support and services teams., New Customer Service
Introducing all-in-one self-service, conversational support, and ticketing. Omnichannel and supercharged with AI., AI that works
Hear CPO Prakash Ramamurthy on why people-centric AI matters., Introducing The Works
Dive into ideas and insights for today's IT-minded business leaders, Turn AI from hype into reality
Everything you need to know about using generative AI in customer service, Announcing Freddy AI + GPT
We're bringing the power of AI and GPT models to the Freshworks product suite, Gartner names Freshworks a Challenger
Gartner recognizes Freshworks as a Challenger in the Gartner® Magic Quadrant™ for IT Service Management Platforms., 100 innovators
Freshworks CIO says companies will deploy more bots in 2023 to reduce human dependence on mundane tasks, reduce cost and margin
See how IT, customer service, and CRM teams succeed with Freshworks, Careers at Freshworks, See how we work, Start free trial,
```

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 `WebElements` (`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 WebElements 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

The screenshot shows the Eclipse IDE interface with the title bar "workspace1 - Aug2023SeleniumSessions/src/main/java/Streams/ClassCRMStream.java - Eclipse IDE". The code editor displays the following Java code:

```
1 package Streams;
2
3 import java.util.List;
4 import java.util.stream.Collectors;
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 ClassCRMStream {
12
13     public static void main(String[] args) {
14         WebDriver driver = new ChromeDriver();
15         driver.get("https://classic.crmpro.com/");
16
17         List<WebElement> paragraphs = driver.findElements(By.cssSelector("section#details p"));
18
19         paragraphs.stream()
20             .map(e -> e.getText())
21             .collect(Collectors.joining());
22
23     }
24
25 }
26
27 }
```

A tooltip is displayed over the line ".collect(Collectors.joining());", listing three methods from the `Collectors` class:

- joining() : Collector<CharSequence,?,String> - Collectors
- joining(CharSequence delimiter) : Collector<CharSequence,?,String> - Collectors
- joining(CharSequence delimiter, CharSequence prefix, CharSequence suffix) : Collector<CharSequence,?,String> - Collectors

At the bottom right of the tooltip, it says "Press 'Space' to show Template Proposals".

Clearer picture-

The screenshot shows the Eclipse IDE interface with the title bar "*ClassicCRMStream.java". The code editor displays the following Java code:

```
15     static WebDriver driver;
16
17     public static void main(String[] args) {
18
19         driver=new ChromeDriver();
20         driver.get("https://classic.crmpro.com/");
21
22         List<WebElement> paragraphs = driver.findElements(By.cssSelector("section#details p"));
23
24         String allPara = paragraphs.stream().
25             map(e->e.getText())
26             .collect(Collectors.join("\n"));
27
28     }
29
30
31
32 }
```

A tooltip is displayed over the line ".collect(Collectors.join("\n"));", providing documentation and alternative implementations:

Returns a Collector that concatenates the input elements into a String, in encounter order.

Returns:

a Collector that concatenates the input elements into a String, in encounter order

Details

Press 'Tab' from proposal table or click/hover for focus

Press 'Ctrl+Space' to show Template Proposals

The tooltip lists three methods from the `Collectors` class:

- joining() : Collector<CharSequence,?,String> - Collectors
- joining(CharSequence delimiter) : Collector<CharSequence,?,String> - Collectors
- joining(CharSequence delimiter, CharSequence prefix, CharSequence suffix) : Collector<CharSequence,?,String> - Collectors