

# Scraping Task:

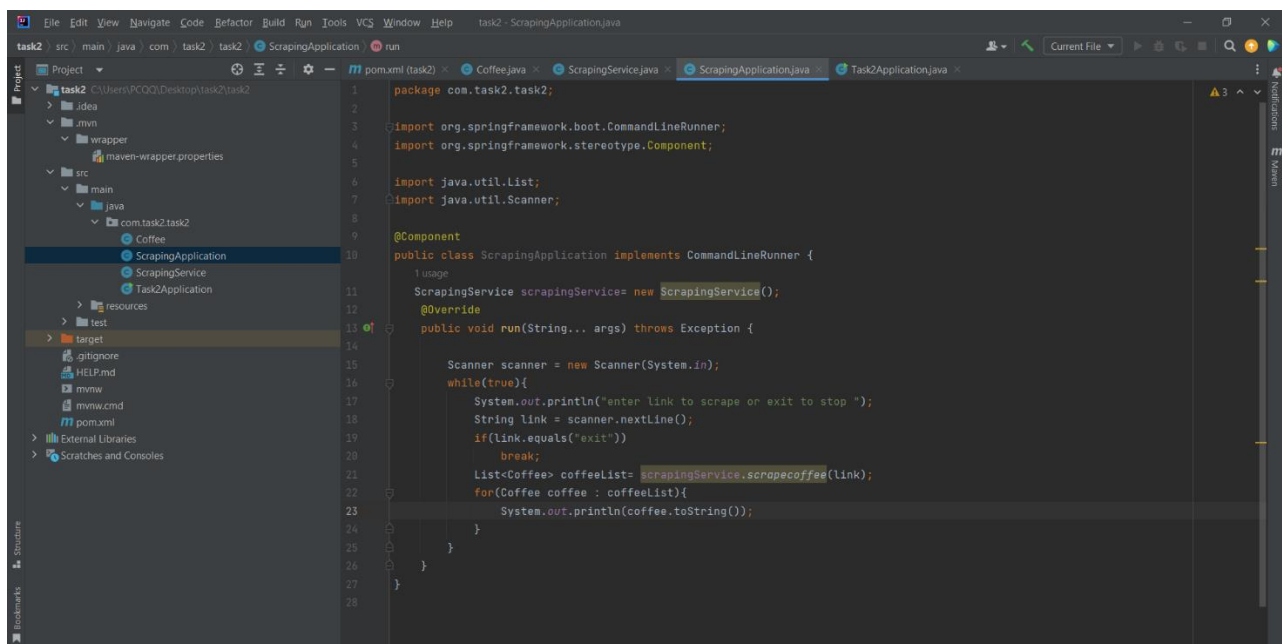
## • Description:

In this task, we enhanced a Java program that scrapes coffee product information from a website using Jsoup. Specifically, we focused on refining the process of extracting and storing data. The key changes involved:

1. **Data Extraction:** We extracted the title, link, image URL, summary, and price of each coffee product as individual strings directly from the HTML elements on the web page.
2. **Object Creation:** Instead of assigning values to object properties individually, we utilized a parameterized constructor to create `Coffee` objects. This constructor takes the extracted strings as arguments, streamlining the object creation process.
3. **Data Storage:** Each `Coffee` object, initialized with the extracted data, was then added to a list of coffee products.

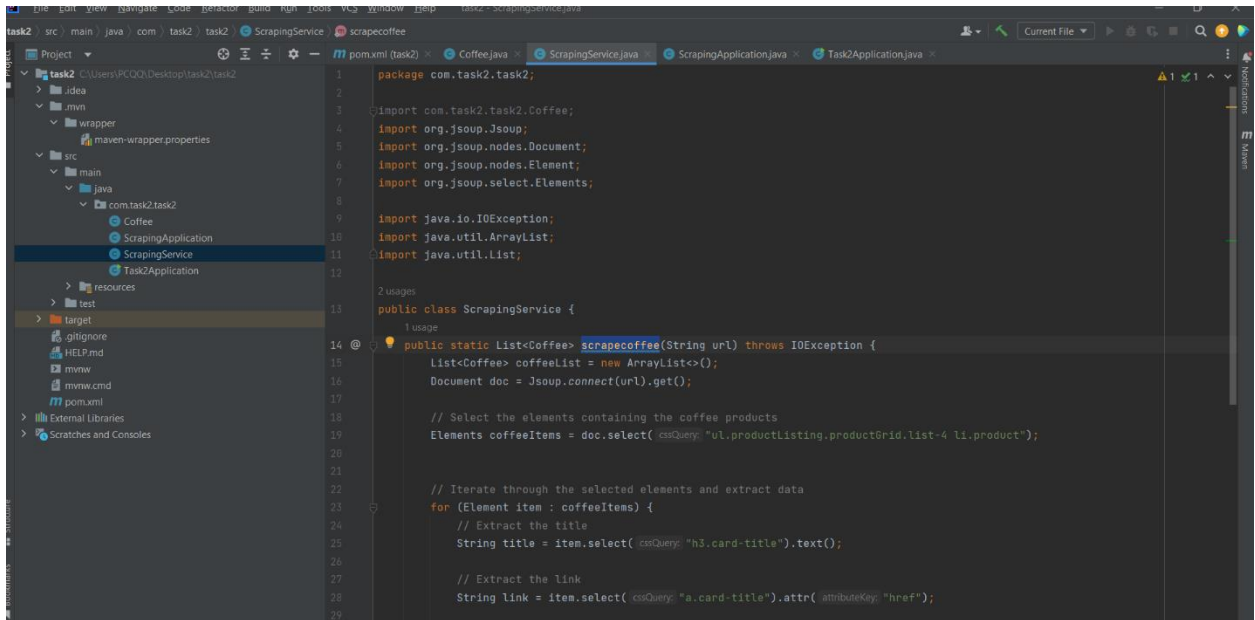
This approach improves code clarity and ensures that the `Coffee` objects are instantiated with all required data in a concise manner.

## • ScrapingApplication code :

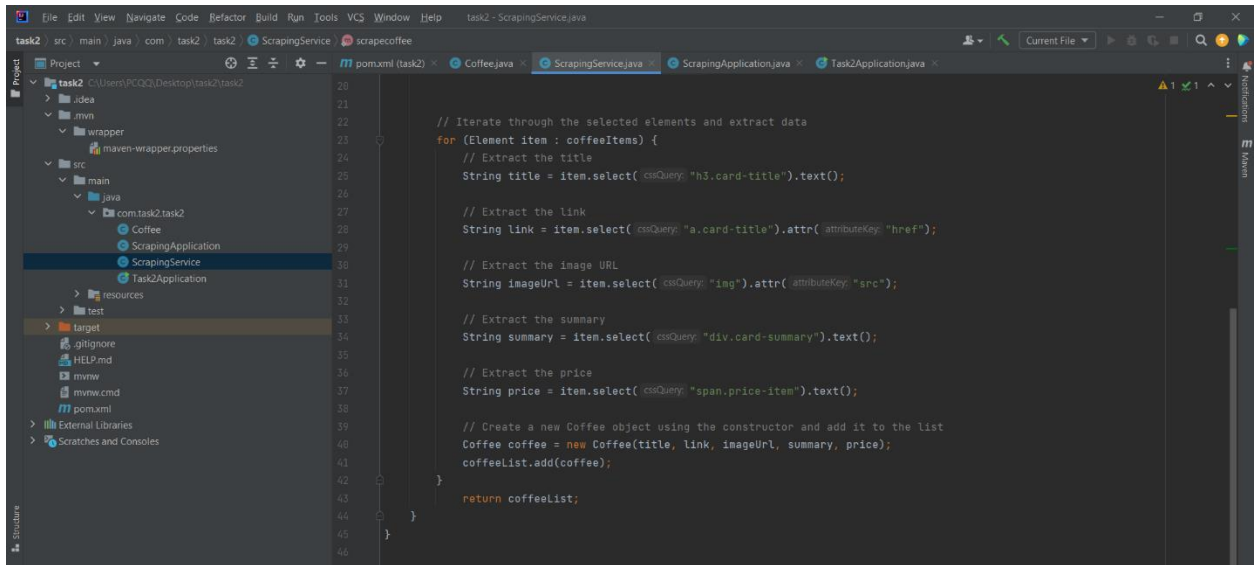


```
1 package com.task2.task2;
2
3 import org.springframework.boot.CommandLineRunner;
4 import org.springframework.stereotype.Component;
5
6 import java.util.List;
7 import java.util.Scanner;
8
9 @Component
10 public class ScrapingApplication implements CommandLineRunner {
11     @Autowired
12     ScrapingService scrapingService = new ScrapingService();
13     @Override
14     public void run(String... args) throws Exception {
15
16         Scanner scanner = new Scanner(System.in);
17         while(true){
18             System.out.println("enter link to scrape or exit to stop ");
19             String link = scanner.nextLine();
20             if(link.equals("exit"))
21                 break;
22             List<Coffee> coffeeList = scrapingService.scrapecoffee(link);
23             for(Coffee coffee : coffeeList){
24                 System.out.println(coffee.toString());
25             }
26         }
27     }
28 }
```

- ScrapingService code:



```
1 package com.task2.task2;
2
3 import com.task2.task2.Coffee;
4 import org.jsoup.Jsoup;
5 import org.jsoup.nodes.Document;
6 import org.jsoup.nodes.Element;
7 import org.jsoup.select.Elements;
8
9 import java.io.IOException;
10 import java.util.ArrayList;
11 import java.util.List;
12
13 2 usages
14 public class ScrapingService {
15     1 usage
16     public static List<Coffee> scrapeCoffee(String url) throws IOException {
17         List<Coffee> coffeeList = new ArrayList<>();
18         Document doc = Jsoup.connect(url).get();
19
20         // Select the elements containing the coffee products
21         Elements coffeeItems = doc.select(cssQuery: "ul.productListing.productGrid.list-4 li.product");
22
23         // Iterate through the selected elements and extract data
24         for (Element item : coffeeItems) {
25             // Extract the title
26             String title = item.select(cssQuery: "h3.card-title").text();
27
28             // Extract the link
29             String link = item.select(cssQuery: "a.card-title").attr(attributeKey: "href");
```



```
30
31
32         // Iterate through the selected elements and extract data
33         for (Element item : coffeeItems) {
34             // Extract the title
35             String title = item.select(cssQuery: "h3.card-title").text();
36
37             // Extract the link
38             String link = item.select(cssQuery: "a.card-title").attr(attributeKey: "href");
39
40             // Extract the image URL
41             String imageUrl = item.select(cssQuery: "img").attr(attributeKey: "src");
42
43             // Extract the summary
44             String summary = item.select(cssQuery: "div.card-summary").text();
45
46             // Extract the price
47             String price = item.select(cssQuery: "span.price-item").text();
48
49             // Create a new Coffee object using the constructor and add it to the list
50             Coffee coffee = new Coffee(title, link, imageUrl, summary, price);
51             coffeeList.add(coffee);
52         }
53         return coffeeList;
54     }
55 }
```

- Output :

