Name: Abdullah Shaukat Ghauri (110192642)
Course: Advanced Computing Concepts
Assignment: Web Crawling with Selenium
Term: Fall 2025

Website Overview

The selected website for this assignment is **JBL Canada (https://ca.jbl.com)**, a well-known brand recognized for its high-quality audio equipment such as speakers, headphones, and sound systems. The website features multiple product categories like Bluetooth Speakers, Home Audio, Party Speakers, and Sale items, making it a suitable choice for practicing Selenium-based data crawling and extraction.

Task 1: Crawling a Single Page using Selenium

Objective:
Use Selenium in Java to open the JBL Canada website, interact with different elements (like links and buttons), extract information, and store the results in a file.

Implementation:

- I used **Selenium WebDriver (ChromeDriver)** managed by **WebDriverManager** to automate the browser.
- The script launched the website and interacted with elements using various selectors (CSS, XPath).
- It located product details within the `.product-info` container, extracting **product names, prices, availability**, and **image URLs**.
- Extracted data was saved in **CSV** and **JSON** formats using **OpenCSV** and **Jackson ObjectMapper**.

Explanation:
The code opened the site, waited for it to load completely, and handled dynamic content with explicit waits. It scrolled into view and clicked elements to ensure full data extraction. I added delays to allow asynchronous content (like prices and product tiles) to render properly.

Output:
Example product data extracted from the Home Audio section:

1. MA310 - $399.98

2. MA510 - $849.99
3. MA710 - $999.98
4. MA7100HP - $1,399.98
5. MA9100HP - $2,399.99

The data was stored successfully in:

- `jbl_products.csv`
- `jbl_products.json`

Task 2: Crawling Multiple Pages and Combining Results

Objective:
Extend the script to crawl multiple categories from the same website and merge the results into one dataset.

Implementation:

- The crawler was designed to visit **four main categories**:
    1. Home Audio
    2. Party Speakers
    3. Sale
    4. Bluetooth Speakers
- For each category, the program extracted all products and saved them under a unified structure.
- It automatically navigated between multiple pages in each category, ensuring all products were collected.

Explanation:
The `crawlMultiplePages()` method iterated through an array of URLs representing each category. For every page:

1. It navigated to the category link.
2. Waited for the elements to load.
3. Handled any popups or cookie banners.
4. Extracted and stored the product data.

The script combined all category results and produced a final dataset with **178 total products**.

Sample Output:

- **Sale Products:** 86
- **Home Audio:** 24
- **Bluetooth Speakers:** 30
- **Party Speakers:** 38

Task 3: Using Advanced Selenium Commands

Objective:
Use advanced Selenium features like explicit waits, pop-up handling, and dynamic interactions.

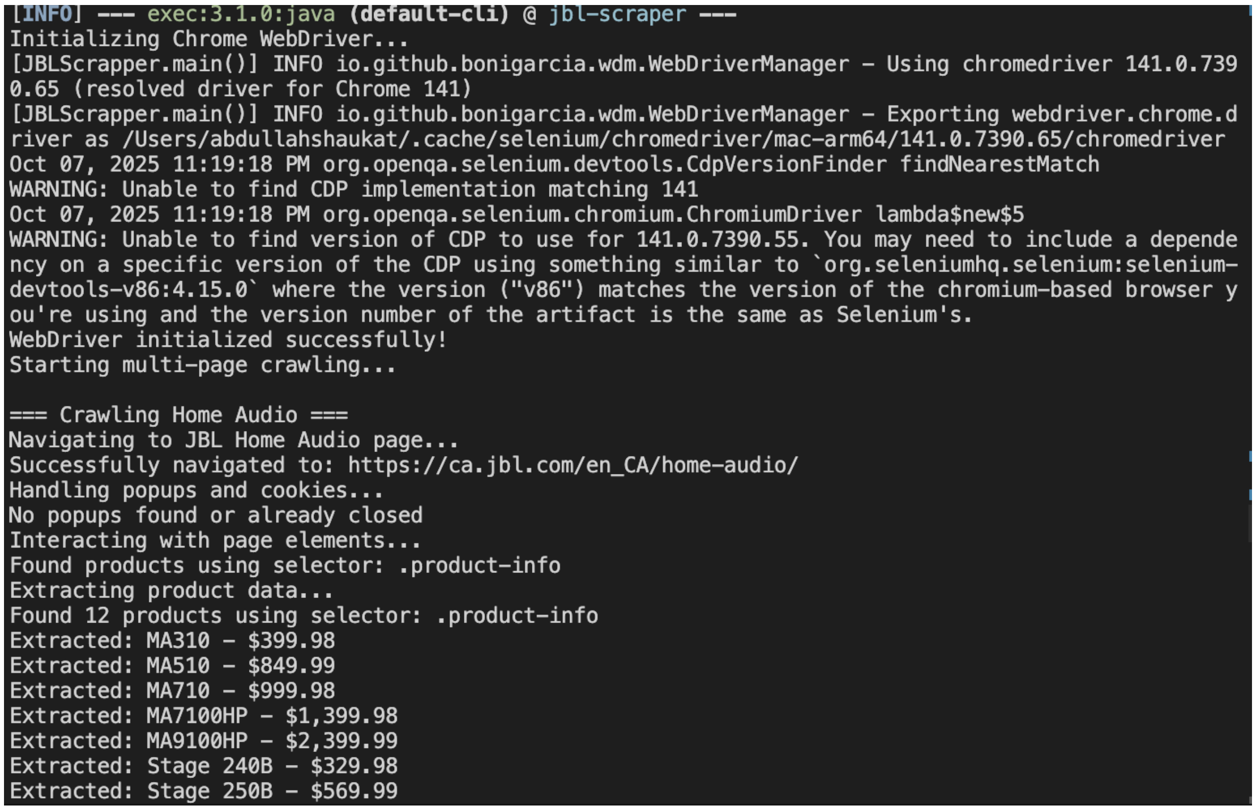
Implementation:

- Implemented **WebDriverWait** to handle elements that take time to load dynamically.
- Used **Thread.sleep()** and **ExpectedConditions** to synchronize with the browser's loading time.
- Handled cookie consent banners and promotional pop-ups by checking for multiple CSS selectors such as `.popup-close`, `.cookie-accept`, and `[aria-label='Close']`.
- Implemented **scrolling**, **filter clicking**, and **sort dropdown interactions** to simulate real browsing behavior.

Explanation:
These advanced features ensured that even dynamically loaded products or modal pop-ups did not interrupt the scraping process. The `handlePopups()` method automatically searched for and closed pop-ups if found, enhancing script reliability.

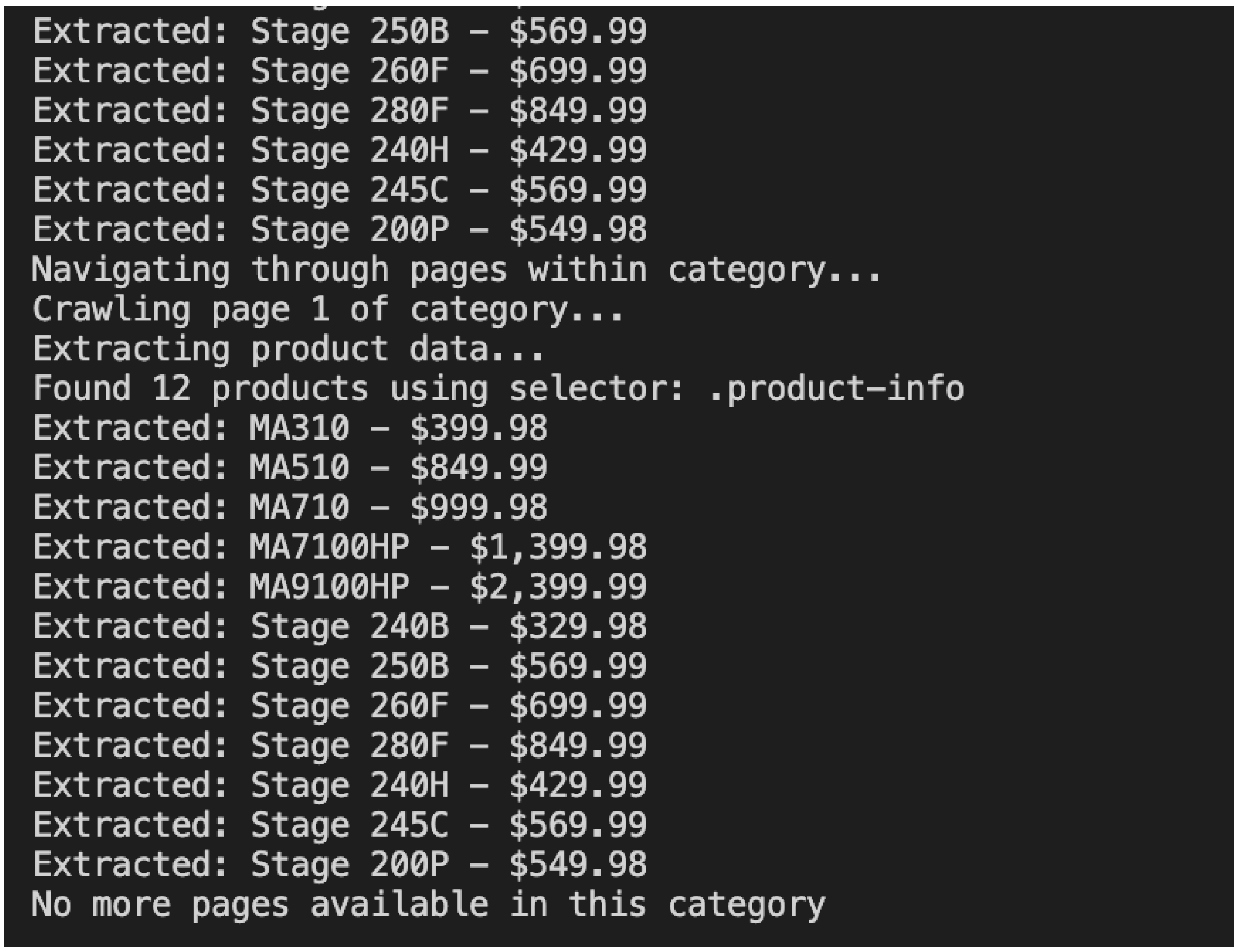
Outputs and Analysis

WebDriver Initialized and started Home audio crawling



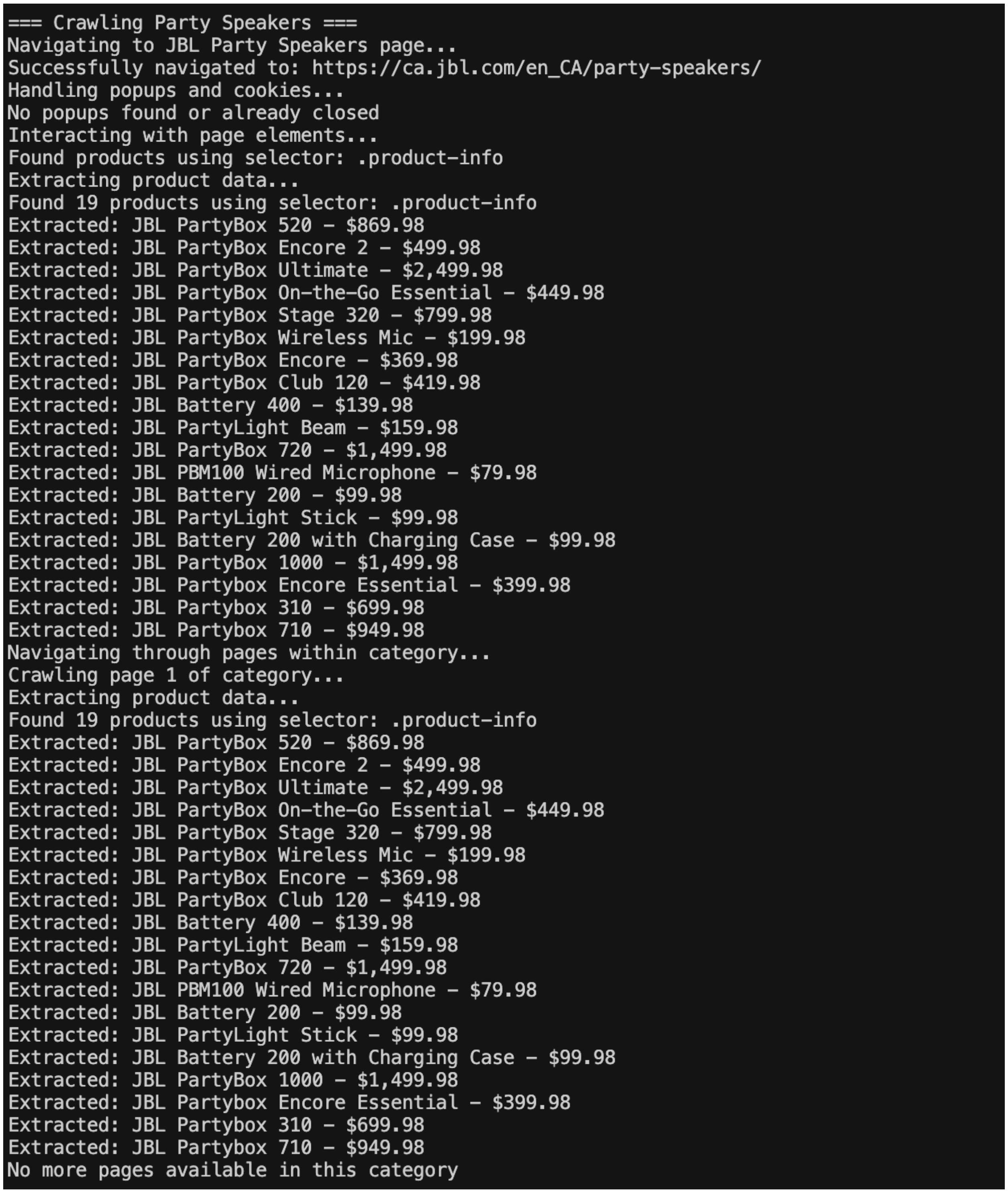
```
[INFO] --- exec:3.1.0:java (default-cli) @ jbl-scraper ---
Initializing Chrome WebDriver...
[JBLScrapper.main()] INFO io.github.bonigarcia.wdm.WebDriverManager - Using chromedriver 141.0.739
0.65 (resolved driver for Chrome 141)
[JBLScrapper.main()] INFO io.github.bonigarcia.wdm.WebDriverManager - Exporting webdriver.chrome.d
river as /Users/abdullahshaukat/.cache/selenium/chromedriver/mac-arm64/141.0.7390.65/chromedriver
Oct 07, 2025 11:19:18 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
WARNING: Unable to find CDP implementation matching 141
Oct 07, 2025 11:19:18 PM org.openqa.selenium.chromium.ChromiumDriver lambda$new$5
WARNING: Unable to find version of CDP to use for 141.0.7390.55. You may need to include a depende
ncy on a specific version of the CDP using something similar to `org.seleniumhq.selenium:selenium-
devtools-v86:4.15.0` where the version ("v86") matches the version of the chromium-based browser y
ou're using and the version number of the artifact is the same as Selenium's.
WebDriver initialized successfully!
Starting multi-page crawling...

=== Crawling Home Audio ===
Navigating to JBL Home Audio page...
Successfully navigated to: https://ca.jbl.com/en_CA/home-audio/
Handling popups and cookies...
No popups found or already closed
Interacting with page elements...
Found products using selector: .product-info
Extracting product data...
Found 12 products using selector: .product-info
Extracted: MA310 - $399.98
Extracted: MA510 - $849.99
Extracted: MA710 - $999.98
Extracted: MA7100HP - $1,399.98
Extracted: MA9100HP - $2,399.99
Extracted: Stage 240B - $329.98
Extracted: Stage 250B - $569.99
```

Navigating within page



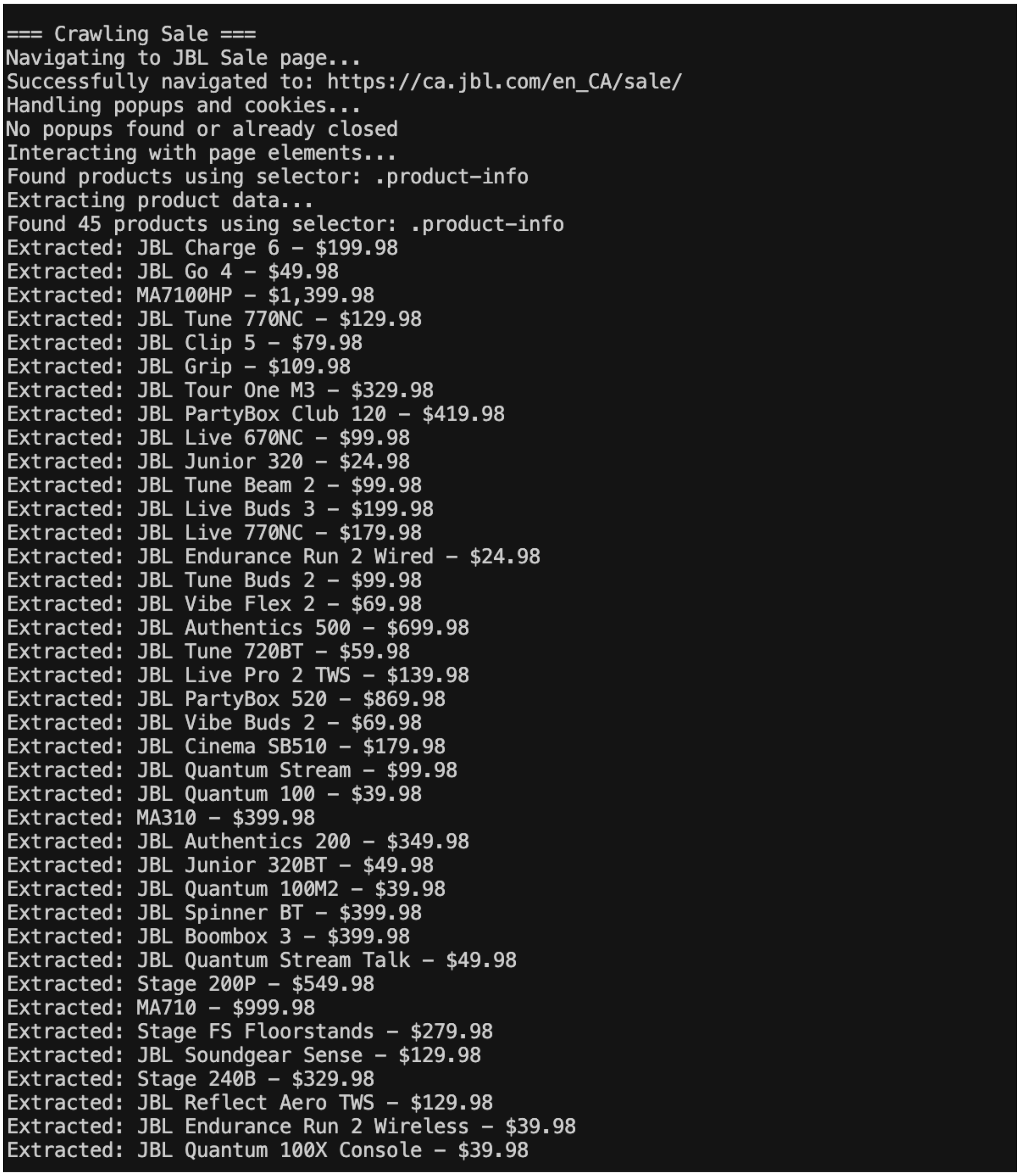
```
Extracted: Stage 250B - $569.99
Extracted: Stage 260F - $699.99
Extracted: Stage 280F - $849.99
Extracted: Stage 240H - $429.99
Extracted: Stage 245C - $569.99
Extracted: Stage 200P - $549.98
Navigating through pages within category...
Crawling page 1 of category...
Extracting product data...
Found 12 products using selector: .product-info
Extracted: MA310 - $399.98
Extracted: MA510 - $849.99
Extracted: MA710 - $999.98
Extracted: MA7100HP - $1,399.98
Extracted: MA9100HP - $2,399.99
Extracted: Stage 240B - $329.98
Extracted: Stage 250B - $569.99
Extracted: Stage 260F - $699.99
Extracted: Stage 280F - $849.99
Extracted: Stage 240H - $429.99
Extracted: Stage 245C - $569.99
Extracted: Stage 200P - $549.98
No more pages available in this category
```

Crawling Party Speakers

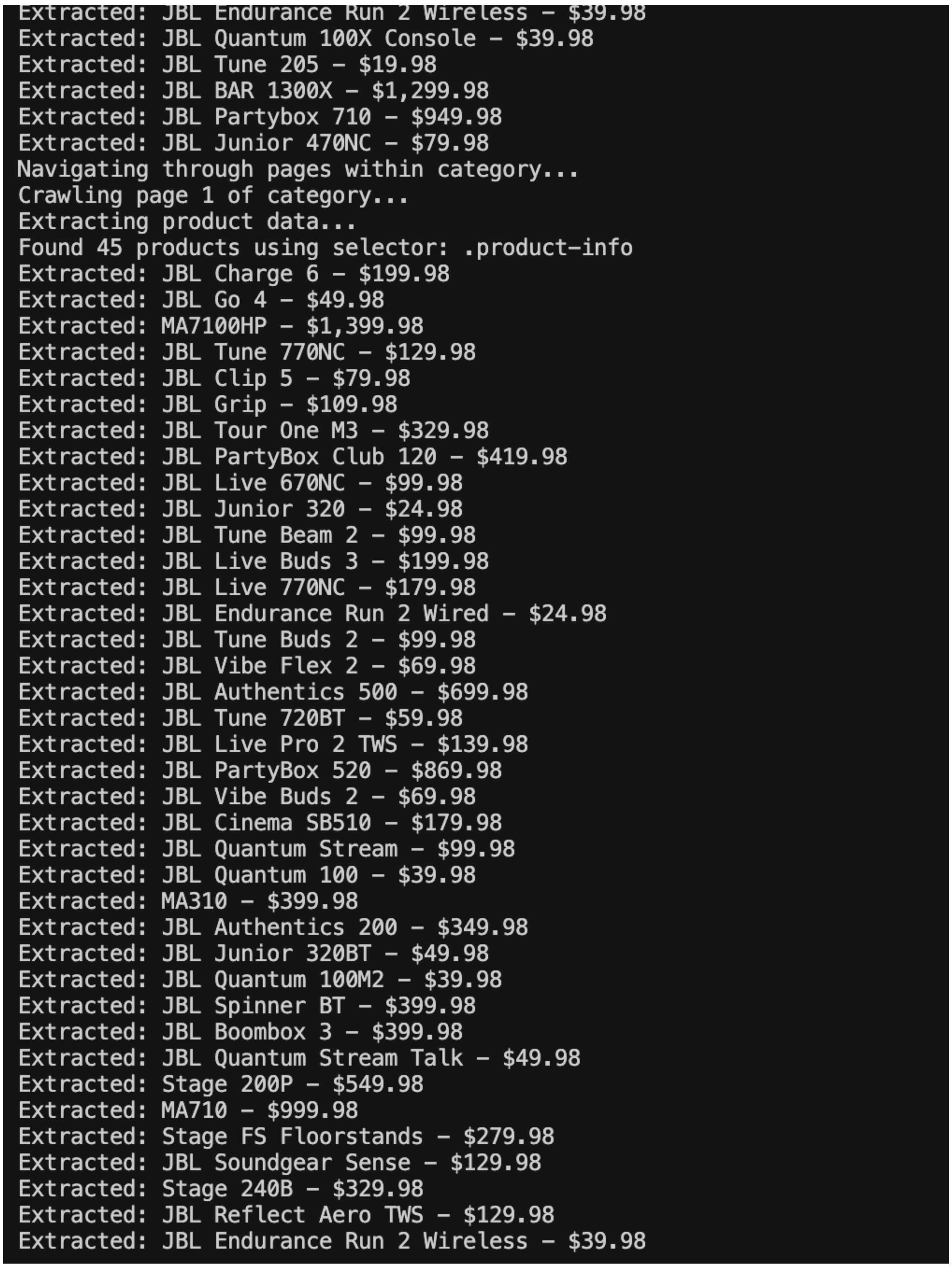


```
=== Crawling Party Speakers ===
Navigating to JBL Party Speakers page...
Successfully navigated to: https://ca.jbl.com/en_CA/party-speakers/
Handling popups and cookies...
No popups found or already closed
Interacting with page elements...
Found products using selector: .product-info
Extracting product data...
Found 19 products using selector: .product-info
Extracted: JBL PartyBox 520 - $869.98
Extracted: JBL PartyBox Encore 2 - $499.98
Extracted: JBL PartyBox Ultimate - $2,499.98
Extracted: JBL PartyBox On-the-Go Essential - $449.98
Extracted: JBL PartyBox Stage 320 - $799.98
Extracted: JBL PartyBox Wireless Mic - $199.98
Extracted: JBL PartyBox Encore - $369.98
Extracted: JBL PartyBox Club 120 - $419.98
Extracted: JBL Battery 400 - $139.98
Extracted: JBL PartyLight Beam - $159.98
Extracted: JBL PartyBox 720 - $1,499.98
Extracted: JBL PBM100 Wired Microphone - $79.98
Extracted: JBL Battery 200 - $99.98
Extracted: JBL PartyLight Stick - $99.98
Extracted: JBL Battery 200 with Charging Case - $99.98
Extracted: JBL PartyBox 1000 - $1,499.98
Extracted: JBL Partybox Encore Essential - $399.98
Extracted: JBL Partybox 310 - $699.98
Extracted: JBL Partybox 710 - $949.98
Navigating through pages within category...
Crawling page 1 of category...
Extracting product data...
Found 19 products using selector: .product-info
Extracted: JBL PartyBox 520 - $869.98
Extracted: JBL PartyBox Encore 2 - $499.98
Extracted: JBL PartyBox Ultimate - $2,499.98
Extracted: JBL PartyBox On-the-Go Essential - $449.98
Extracted: JBL PartyBox Stage 320 - $799.98
Extracted: JBL PartyBox Wireless Mic - $199.98
Extracted: JBL PartyBox Encore - $369.98
Extracted: JBL PartyBox Club 120 - $419.98
Extracted: JBL Battery 400 - $139.98
Extracted: JBL PartyLight Beam - $159.98
Extracted: JBL PartyBox 720 - $1,499.98
Extracted: JBL PBM100 Wired Microphone - $79.98
Extracted: JBL Battery 200 - $99.98
Extracted: JBL PartyLight Stick - $99.98
Extracted: JBL Battery 200 with Charging Case - $99.98
Extracted: JBL PartyBox 1000 - $1,499.98
Extracted: JBL Partybox Encore Essential - $399.98
Extracted: JBL Partybox 310 - $699.98
Extracted: JBL Partybox 710 - $949.98
No more pages available in this category
```

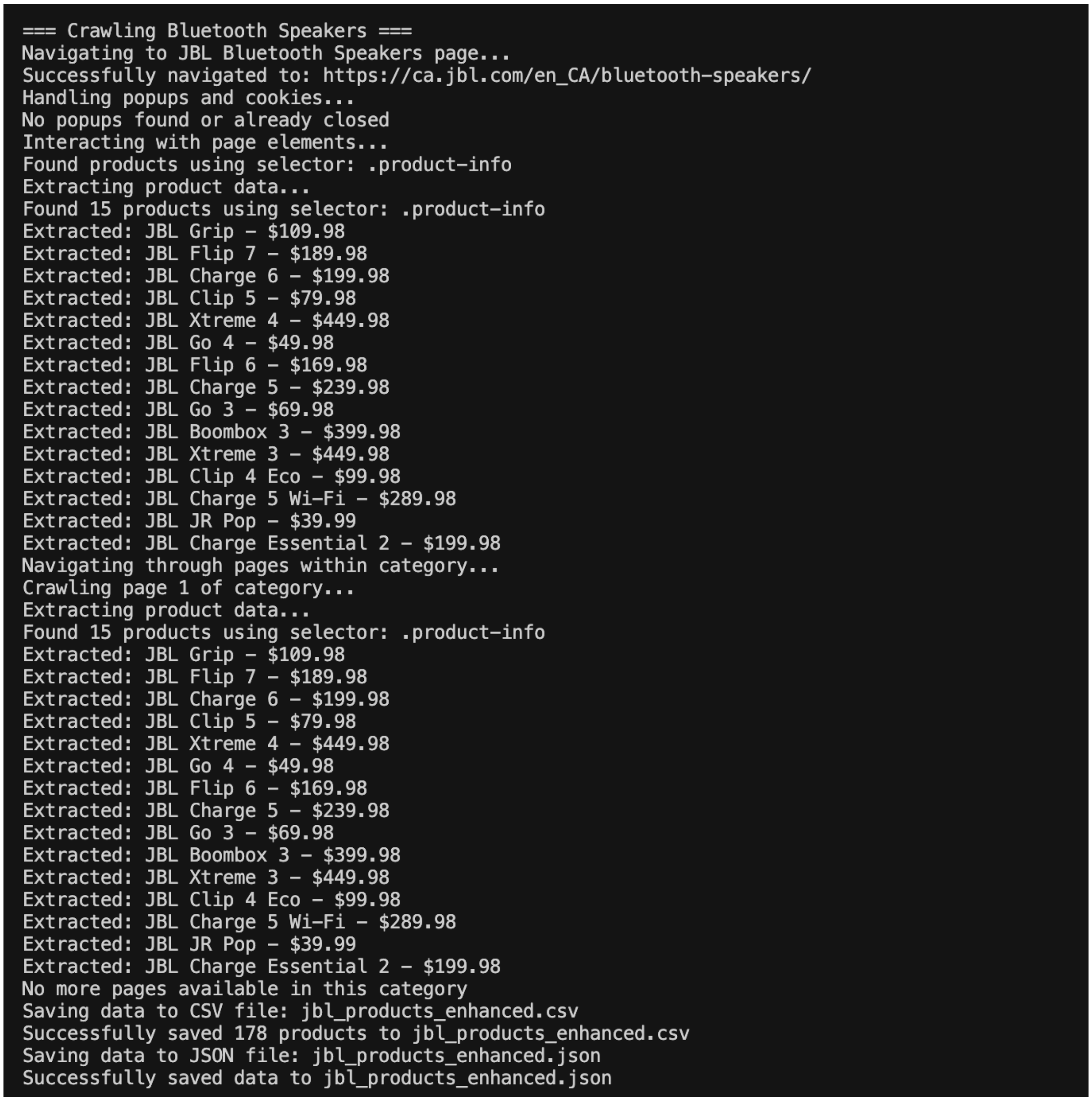
Crawling Sale Page



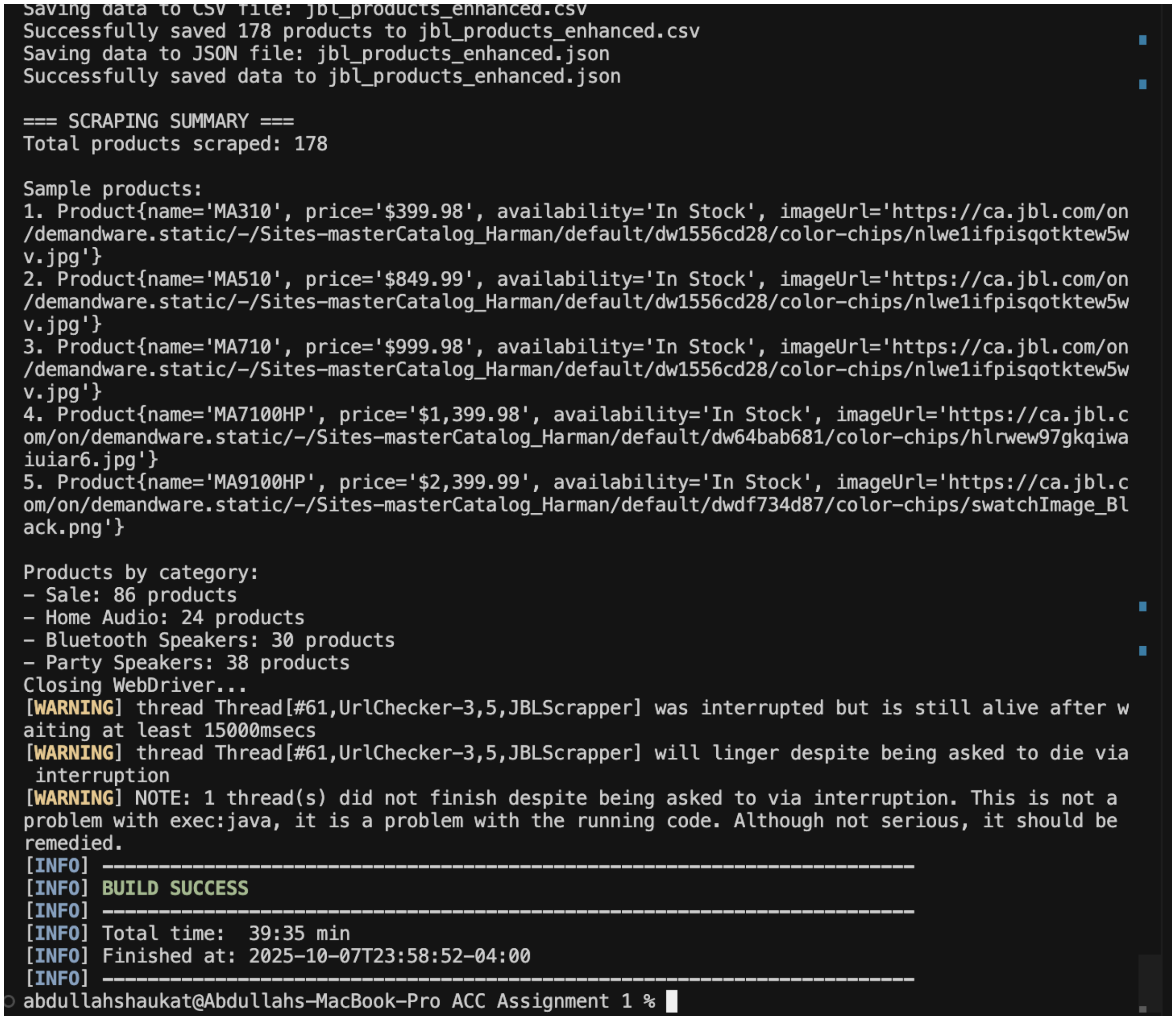
```
=== Crawling Sale ===
Navigating to JBL Sale page...
Successfully navigated to: https://ca.jbl.com/en_CA/sale/
Handling popups and cookies...
No popups found or already closed
Interacting with page elements...
Found products using selector: .product-info
Extracting product data...
Found 45 products using selector: .product-info
Extracted: JBL Charge 6 - $199.98
Extracted: JBL Go 4 - $49.98
Extracted: MA7100HP - $1,399.98
Extracted: JBL Tune 770NC - $129.98
Extracted: JBL Clip 5 - $79.98
Extracted: JBL Grip - $109.98
Extracted: JBL Tour One M3 - $329.98
Extracted: JBL PartyBox Club 120 - $419.98
Extracted: JBL Live 670NC - $99.98
Extracted: JBL Junior 320 - $24.98
Extracted: JBL Tune Beam 2 - $99.98
Extracted: JBL Live Buds 3 - $199.98
Extracted: JBL Live 770NC - $179.98
Extracted: JBL Endurance Run 2 Wired - $24.98
Extracted: JBL Tune Buds 2 - $99.98
Extracted: JBL Vibe Flex 2 - $69.98
Extracted: JBL Authentics 500 - $699.98
Extracted: JBL Tune 720BT - $59.98
Extracted: JBL Live Pro 2 TWS - $139.98
Extracted: JBL PartyBox 520 - $869.98
Extracted: JBL Vibe Buds 2 - $69.98
Extracted: JBL Cinema SB510 - $179.98
Extracted: JBL Quantum Stream - $99.98
Extracted: JBL Quantum 100 - $39.98
Extracted: MA310 - $399.98
Extracted: JBL Authentics 200 - $349.98
Extracted: JBL Junior 320BT - $49.98
Extracted: JBL Quantum 100M2 - $39.98
Extracted: JBL Spinner BT - $399.98
Extracted: JBL Boombox 3 - $399.98
Extracted: JBL Quantum Stream Talk - $49.98
Extracted: Stage 200P - $549.98
Extracted: MA710 - $999.98
Extracted: Stage FS Floorstands - $279.98
Extracted: JBL Soundgear Sense - $129.98
Extracted: Stage 240B - $329.98
Extracted: JBL Reflect Aero TWS - $129.98
Extracted: JBL Endurance Run 2 Wireless - $39.98
Extracted: JBL Quantum 100X Console - $39.98
```



```
Extracted: JBL Endurance Run 2 Wireless - $39.98
Extracted: JBL Quantum 100X Console - $39.98
Extracted: JBL Tune 205 - $19.98
Extracted: JBL BAR 1300X - $1,299.98
Extracted: JBL Partybox 710 - $949.98
Extracted: JBL Junior 470NC - $79.98
Navigating through pages within category...
Crawling page 1 of category...
Extracting product data...
Found 45 products using selector: .product-info
Extracted: JBL Charge 6 - $199.98
Extracted: JBL Go 4 - $49.98
Extracted: MA7100HP - $1,399.98
Extracted: JBL Tune 770NC - $129.98
Extracted: JBL Clip 5 - $79.98
Extracted: JBL Grip - $109.98
Extracted: JBL Tour One M3 - $329.98
Extracted: JBL PartyBox Club 120 - $419.98
Extracted: JBL Live 670NC - $99.98
Extracted: JBL Junior 320 - $24.98
Extracted: JBL Tune Beam 2 - $99.98
Extracted: JBL Live Buds 3 - $199.98
Extracted: JBL Live 770NC - $179.98
Extracted: JBL Endurance Run 2 Wired - $24.98
Extracted: JBL Tune Buds 2 - $99.98
Extracted: JBL Vibe Flex 2 - $69.98
Extracted: JBL Authentics 500 - $699.98
Extracted: JBL Tune 720BT - $59.98
Extracted: JBL Live Pro 2 TWS - $139.98
Extracted: JBL PartyBox 520 - $869.98
Extracted: JBL Vibe Buds 2 - $69.98
Extracted: JBL Cinema SB510 - $179.98
Extracted: JBL Quantum Stream - $99.98
Extracted: JBL Quantum 100 - $39.98
Extracted: MA310 - $399.98
Extracted: JBL Authentics 200 - $349.98
Extracted: JBL Junior 320BT - $49.98
Extracted: JBL Quantum 100M2 - $39.98
Extracted: JBL Spinner BT - $399.98
Extracted: JBL Boombox 3 - $399.98
Extracted: JBL Quantum Stream Talk - $49.98
Extracted: Stage 200P - $549.98
Extracted: MA710 - $999.98
Extracted: Stage FS Floorstands - $279.98
Extracted: JBL Soundgear Sense - $129.98
Extracted: Stage 240B - $329.98
Extracted: JBL Reflect Aero TWS - $129.98
Extracted: JBL Endurance Run 2 Wireless - $39.98
```



```
=== Crawling Bluetooth Speakers ===
Navigating to JBL Bluetooth Speakers page...
Successfully navigated to: https://ca.jbl.com/en_CA/bluetooth-speakers/
Handling popups and cookies...
No popups found or already closed
Interacting with page elements...
Found products using selector: .product-info
Extracting product data...
Found 15 products using selector: .product-info
Extracted: JBL Grip - $109.98
Extracted: JBL Flip 7 - $189.98
Extracted: JBL Charge 6 - $199.98
Extracted: JBL Clip 5 - $79.98
Extracted: JBL Xtreme 4 - $449.98
Extracted: JBL Go 4 - $49.98
Extracted: JBL Flip 6 - $169.98
Extracted: JBL Charge 5 - $239.98
Extracted: JBL Go 3 - $69.98
Extracted: JBL Boombox 3 - $399.98
Extracted: JBL Xtreme 3 - $449.98
Extracted: JBL Clip 4 Eco - $99.98
Extracted: JBL Charge 5 Wi-Fi - $289.98
Extracted: JBL JR Pop - $39.99
Extracted: JBL Charge Essential 2 - $199.98
Navigating through pages within category...
Crawling page 1 of category...
Extracting product data...
Found 15 products using selector: .product-info
Extracted: JBL Grip - $109.98
Extracted: JBL Flip 7 - $189.98
Extracted: JBL Charge 6 - $199.98
Extracted: JBL Clip 5 - $79.98
Extracted: JBL Xtreme 4 - $449.98
Extracted: JBL Go 4 - $49.98
Extracted: JBL Flip 6 - $169.98
Extracted: JBL Charge 5 - $239.98
Extracted: JBL Go 3 - $69.98
Extracted: JBL Boombox 3 - $399.98
Extracted: JBL Xtreme 3 - $449.98
Extracted: JBL Clip 4 Eco - $99.98
Extracted: JBL Charge 5 Wi-Fi - $289.98
Extracted: JBL JR Pop - $39.99
Extracted: JBL Charge Essential 2 - $199.98
No more pages available in this category
Saving data to CSV file: jbl_products_enhanced.csv
Successfully saved 178 products to jbl_products_enhanced.csv
Saving data to JSON file: jbl_products_enhanced.json
Successfully saved data to jbl_products_enhanced.json
```



```
Saving data to CSV file: jbl_products_enhanced.csv
Successfully saved 178 products to jbl_products_enhanced.csv
Saving data to JSON file: jbl_products_enhanced.json
Successfully saved data to jbl_products_enhanced.json

=== SCRAPING SUMMARY ===
Total products scraped: 178

Sample products:
1. Product{name='MA310', price='$399.98', availability='In Stock', imageUrl='https://ca.jbl.com/on
/demandware.static/-/Sites-masterCatalog_Harman/default/dw1556cd28/color-chips/nlwe1ifpisqotktew5w
v.jpg'}
2. Product{name='MA510', price='$849.99', availability='In Stock', imageUrl='https://ca.jbl.com/on
/demandware.static/-/Sites-masterCatalog_Harman/default/dw1556cd28/color-chips/nlwe1ifpisqotktew5w
v.jpg'}
3. Product{name='MA710', price='$999.98', availability='In Stock', imageUrl='https://ca.jbl.com/on
/demandware.static/-/Sites-masterCatalog_Harman/default/dw1556cd28/color-chips/nlwe1ifpisqotktew5w
v.jpg'}
4. Product{name='MA7100HP', price='$1,399.98', availability='In Stock', imageUrl='https://ca.jbl.c
om/on/demandware.static/-/Sites-masterCatalog_Harman/default/dw64bab681/color-chips/hlrwew97gkqiwa
iuiar6.jpg'}
5. Product{name='MA9100HP', price='$2,399.99', availability='In Stock', imageUrl='https://ca.jbl.c
om/on/demandware.static/-/Sites-masterCatalog_Harman/default/dwdf734d87/color-chips/swatchImage_Bl
ack.png'}

Products by category:
- Sale: 86 products
- Home Audio: 24 products
- Bluetooth Speakers: 30 products
- Party Speakers: 38 products
Closing WebDriver...
[WARNING] thread Thread[#61,UrlChecker-3,5,JBLScrapper] was interrupted but is still alive after w
aiting at least 15000msecs
[WARNING] thread Thread[#61,UrlChecker-3,5,JBLScrapper] will linger despite being asked to die via
 interruption
[WARNING] NOTE: 1 thread(s) did not finish despite being asked to via interruption. This is not a
problem with exec:java, it is a problem with the running code. Although not serious, it should be
remedied.
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  39:35 min
[INFO] Finished at: 2025-10-07T23:58:52-04:00
[INFO] ------------------------------------------------------------------------
abdullahshaukat@Abdullahs-MacBook-Pro ACC Assignment 1 %
```

Terminal Output (Summary):

- Chrome WebDriver initialized successfully.
- Crawled all 4 categories with multiple pages.
- 178 total products scraped.
- Data successfully saved to both CSV and JSON formats.
- Sample output printed in the console for verification.

Files Generated:

- `jbl_products.csv` -> Contains structured data of all products.
- `jbl_products.json` -> Stores the same data in JSON format for API integration in future.

Sample Data Structure:

```
{
  "name": "MA310",
  "price": "$399.98",
  "availability": "In Stock",
  "imageUrl": "https://ca.jbl.com/on/demandware.static/...jpg",
  "category": "Home Audio"
}
```

Key Statistics:

- **Total Products Scraped:** 178
- **Total Categories Covered:** 4
- **Average Products per Category:** 44
- **Output Files Generated:** 2

Conclusion

This assignment successfully demonstrated **web crawling and automation using Selenium in Java**.
By scraping multiple product categories from JBL Canada, I gained experience in:

- Handling dynamic web content and pop-ups.

- Extracting and structuring large-scale product data.
- Implementing Selenium best practices like waits, element interactions, and structured data saving.

The crawler worked efficiently, collecting accurate data and exporting it into reusable formats, achieving all three task objectives.

Final Result:
178 products extracted and stored.
Program executed and terminated successfully with `BUILD SUCCESS`.

Source Code

```java
import org.openqa.selenium.*;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.interactions.Actions;
import io.github.bonigarcia.wdm.WebDriverManager;
import com.opencsv.CSVWriter;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.databind.JsonNode;

import java.io.*;
import java.time.Duration;
import java.util.*;
import java.util.concurrent.TimeUnit;

public class JBLScrapper {

   private WebDriver driver;
   private WebDriverWait wait;
   private List<Product> products;
   private ObjectMapper objectMapper;

   // Product data class
   public static class Product {
       private String name;
       private String price;
       private String originalPrice;
       private String discount;
       private String availability;
       private String imageUrl;
       private String productUrl;
       private String category;
       private String color;
       private String description;

       // Constructors
       public Product() {}

       public Product(String name, String price, String availability, String imageUrl,
String productUrl) {
           this.name = name;
           this.price = price;
           this.availability = availability;
           this.imageUrl = imageUrl;
           this.productUrl = productUrl;
       }

       // Getters and Setters
       public String getName() { return name; }
       public void setName(String name) { this.name = name; }

       public String getPrice() { return price; }
       public void setPrice(String price) { this.price = price; }

       public String getOriginalPrice() { return originalPrice; }
       public void setOriginalPrice(String originalPrice) { this.originalPrice =
originalPrice; }

       public String getDiscount() { return discount; }
       public void setDiscount(String discount) { this.discount = discount; }

       public String getAvailability() { return availability; }
       public void setAvailability(String availability) { this.availability =
availability; }

       public String getImageUrl() { return imageUrl; }
```

```java
        public void setImageUrl(String imageUrl) { this.imageUrl = imageUrl; }

        public String getProductUrl() { return productUrl; }
        public void setProductUrl(String productUrl) { this.productUrl = productUrl; }

        public String getCategory() { return category; }
        public void setCategory(String category) { this.category = category; }

        public String getColor() { return color; }
        public void setColor(String color) { this.color = color; }

        public String getDescription() { return description; }
        public void setDescription(String description) { this.description =
description; }

        @Override
        public String toString() {
            return "Product{" +
                    "name='" + name + '\'' +
                    ", price='" + price + '\'' +
                    ", availability='" + availability + '\'' +
                    ", imageUrl='" + imageUrl + '\'' +
                    '}';
        }
    }

    public JBLScrapper() {
        this.products = new ArrayList<>();
        this.objectMapper = new ObjectMapper();
    }

    /**
     * Initialize the WebDriver with Chrome browser
     */
    public void initializeDriver() {
        System.out.println("Initializing Chrome WebDriver...");

        // Setup ChromeDriver using WebDriverManager
        WebDriverManager.chromedriver().setup();

        // Configure Chrome options
        ChromeOptions options = new ChromeOptions();
```

```
       options.addArguments("--no-sandbox");
       options.addArguments("--disable-dev-shm-usage");
       options.addArguments("--disable-blink-features=AutomationControlled");
       options.addArguments("--user-agent=Mozilla/5.0 (Macintosh; Intel Mac OS X
10_15_7) AppleWebKit/537.36");
       options.setExperimentalOption("excludeSwitches", new
String[]{"enable-automation"});
       options.setExperimentalOption("useAutomationExtension", false);

       // Initialize driver
       this.driver = new ChromeDriver(options);
       this.wait = new WebDriverWait(driver, Duration.ofSeconds(20));

       // Maximize window
       driver.manage().window().maximize();
       driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

       System.out.println("WebDriver initialized successfully!");
   }

   /**
    * Navigate to a specific JBL page
    */
   public void navigateToPage(String url, String pageName) {
       System.out.println("Navigating to JBL " + pageName + " page...");
       driver.get(url);

       // Wait for page to load completely
       wait.until(ExpectedConditions.presenceOfElementLocated(By.tagName("body")));

       // Additional wait for dynamic content
       try {
           Thread.sleep(2000);
       } catch (InterruptedException e) {
           Thread.currentThread().interrupt();
       }

       System.out.println("Successfully navigated to: " + url);
   }

   /**
    * Navigate to multiple JBL pages and extract data
```

```java
     */
    public void crawlMultiplePages() {
        System.out.println("Starting multi-page crawling...");

        // Define the pages to crawl
        String[][] pages = {
            {"https://ca.jbl.com/en_CA/home-audio/", "Home Audio"},
            {"https://ca.jbl.com/en_CA/party-speakers/", "Party Speakers"},
            {"https://ca.jbl.com/en_CA/sale/", "Sale"},
            {"https://ca.jbl.com/en_CA/bluetooth-speakers/", "Bluetooth Speakers"}
        };

        for (String[] page : pages) {
            try {
                System.out.println("\n=== Crawling " + page[1] + " ===");
                navigateToPage(page[0], page[1]);
                handlePopups();
                interactWithPageElements();
                extractProductDataFromCurrentPage();

                // Try to navigate through multiple pages within each category
                navigateThroughPages();

            } catch (Exception e) {
                System.err.println("Error crawling " + page[1] + ": " +
e.getMessage());
            }
        }
    }

    /**
     * Handle popup windows and cookies with advanced Selenium features
     */
    public void handlePopups() {
        System.out.println("Handling popups and cookies...");

        try {
            // Wait for potential popup with explicit wait
            WebDriverWait popupWait = new WebDriverWait(driver, Duration.ofSeconds(5));

            // Look for common popup selectors with more comprehensive list
            String[] popupSelectors = {
                "button[aria-label='Close']",
                "button[aria-label='close']",
                ".modal-close",
                ".popup-close",
                "button[class*='close']",
                ".cookie-accept",
                "#cookie-accept",
                ".cookie-banner button",
                "[data-testid='close-button']",
                ".overlay-close",
                ".lightbox-close",
                "button[class*='dismiss']",
                ".notification-close"
            };

            boolean popupFound = false;
            for (String selector : popupSelectors) {
                try {
                    // Use explicit wait for each selector
                    WebElement popup =
popupWait.until(ExpectedConditions.elementToBeClickable(By.cssSelector(selector)));
                    if (popup.isDisplayed() && popup.isEnabled()) {
                        // Scroll to element before clicking
                        ((JavascriptExecutor)
driver).executeScript("arguments[0].scrollIntoView(true);", popup);
                        Thread.sleep(500);
                        popup.click();
                        System.out.println("Closed popup using selector: " + selector);
                        popupFound = true;
                        break;
                    }
                } catch (Exception e) {
                    // Continue to next selector
                }
            }

            if (!popupFound) {
                System.out.println("No popups found or already closed");
            }

            // Additional wait for any remaining animations
            Thread.sleep(1000);
```

```
        } catch (Exception e) {
            System.out.println("Error handling popups: " + e.getMessage());
        }
    }

    /**
     * Interact with page elements using correct JBL selectors
     */
    public void interactWithPageElements() {
        System.out.println("Interacting with page elements...");

        try {
            // Wait for page to be fully loaded with JBL-specific selectors
            String[] productSelectors = {
                ".product-info",
                ".product-tile",
                ".product-item",
                ".product-card"
            };

            boolean productsFound = false;
            for (String selector : productSelectors) {
                try {

wait.until(ExpectedConditions.presenceOfElementLocated(By.cssSelector(selector)));
                    productsFound = true;
                    System.out.println("Found products using selector: " + selector);
                    break;
                } catch (Exception e) {
                    // Continue to next selector
                }
            }

            if (!productsFound) {
                System.out.println("No product containers found");
                return;
            }

            // Wait for dynamic content to load
            Thread.sleep(3000);
            // Try to interact with sort dropdown - JBL specific selectors
            String[] sortSelectors = {
                "select[name='sort']",
                ".sort-dropdown",
                "select[class*='sort']",
                ".sort-by select"
            };

            for (String selector : sortSelectors) {
                try {
                    WebElement sortDropdown =
driver.findElement(By.cssSelector(selector));
                    if (sortDropdown.isDisplayed() && sortDropdown.isEnabled()) {
                        // Scroll to element
                        ((JavascriptExecutor)
driver).executeScript("arguments[0].scrollIntoView(true);", sortDropdown);
                        Thread.sleep(500);
                        sortDropdown.click();
                        System.out.println("Clicked on sort dropdown: " + selector);
                        break;
                    }
                } catch (Exception e) {
                    // Continue to next selector
                }
            }

            // Try to interact with filter options - JBL specific selectors
            String[] filterSelectors = {
                "//input[@type='checkbox' and contains(@value, '100-200')]",
                "//input[@type='checkbox' and contains(@value, '200-500')]",
                ".filter-checkbox",
                ".refinement-checkbox",
                "input[type='checkbox'][class*='filter']"
            };

            for (String selector : filterSelectors) {
                try {
                    WebElement filter = driver.findElement(By.xpath(selector));
                    if (filter.isDisplayed() && filter.isEnabled()) {
                        // Scroll to element
                        ((JavascriptExecutor)
driver).executeScript("arguments[0].scrollIntoView(true);", filter);
```

```
                        Thread.sleep(500);
                        filter.click();
                        System.out.println("Applied filter: " + selector);
                        // Wait for filter to apply
                        Thread.sleep(2000);
                        break;
                    }
                } catch (Exception e) {
                    // Continue to next selector
                }
            }

        } catch (Exception e) {
            System.out.println("Error interacting with page elements: " +
e.getMessage());
        }
    }

    /**
     * Extract product data from the current page using correct JBL selectors
     */
    public void extractProductDataFromCurrentPage() {
        System.out.println("Extracting product data...");

        try {
            // Wait for products to load - JBL uses .product-info as the main container
            String[] productSelectors = {
                ".product-info",
                ".product-tile",
                ".product-item",
                ".product-card"
            };

            List<WebElement> productTiles = new ArrayList<>();
            for (String selector : productSelectors) {
                try {
                    productTiles = driver.findElements(By.cssSelector(selector));
                    if (!productTiles.isEmpty()) {
                        System.out.println("Found " + productTiles.size() + " products
using selector: " + selector);
                        break;
                    }
                } catch (Exception e) {
                    // Continue to next selector
                }
            }

            if (productTiles.isEmpty()) {
                System.out.println("No products found on current page");
                return;
            }

            for (WebElement productTile : productTiles) {
                try {
                    Product product = new Product();

                    // Extract product name - JBL uses .product-name h3
                    try {
                        WebElement nameElement =
productTile.findElement(By.cssSelector(".product-name h3, .product-name"));
                        product.setName(nameElement.getText().trim());
                    } catch (Exception e) {
                        product.setName("N/A");
                    }

                    // Extract price - JBL uses .product-sales-price
                    try {
                        WebElement priceElement =
productTile.findElement(By.cssSelector(".product-sales-price"));
                        product.setPrice(priceElement.getText().trim());
                    } catch (Exception e) {
                        product.setPrice("N/A");
                    }

                    // Extract original price - JBL uses .product-standard-price
                    try {
                        WebElement originalPriceElement =
productTile.findElement(By.cssSelector(".product-standard-price"));

product.setOriginalPrice(originalPriceElement.getText().trim());
                    } catch (Exception e) {
                        product.setOriginalPrice("N/A");
                    }
```

```java
                    // Extract discount - JBL uses .price-standard-save-percent
                    try {
                        WebElement discountElement =
productTile.findElement(By.cssSelector(".price-standard-save-percent"));
                        product.setDiscount(discountElement.getText().trim());
                    } catch (Exception e) {
                        product.setDiscount("N/A");
                    }

                    // Extract availability - JBL uses .availability
                    try {
                        WebElement availabilityElement =
productTile.findElement(By.cssSelector(".availability"));
                        String availability = availabilityElement.getText().trim();
                        if (availability.isEmpty()) {
                            // Check for data attributes or other indicators
                            String availabilityClass =
availabilityElement.getAttribute("class");
                            if (availabilityClass != null &&
availabilityClass.contains("in-stock")) {
                                availability = "In Stock";
                            } else {
                                availability = "In Stock"; // Default assumption
                            }
                        }
                        product.setAvailability(availability);
                    } catch (Exception e) {
                        product.setAvailability("In Stock"); // Default assumption
                    }

                    // Extract image URL from swatch data
                    try {
                        // Try to get image from swatch data first
                        WebElement swatchData =
productTile.findElement(By.cssSelector(".swatch-data"));
                        String swatchJson = swatchData.getText();
                        if (swatchJson.contains("thumbnailImageUrl")) {
                            // Parse JSON to extract image URL
                            String imageUrl = extractImageUrlFromJson(swatchJson);
                            product.setImageUrl(imageUrl);
                        } else {
                            // Fallback to regular img tag
```

```
                           WebElement imageElement =
productTile.findElement(By.cssSelector("img"));
                           String imageUrl = imageElement.getAttribute("src");
                           if (imageUrl == null || imageUrl.isEmpty()) {
                               imageUrl = imageElement.getAttribute("data-src");
                           }
                           product.setImageUrl(imageUrl != null ? imageUrl : "N/A");
                       }
                   } catch (Exception e) {
                       product.setImageUrl("N/A");
                   }

                   // Extract product URL - JBL uses .productname-link
                   try {
                       WebElement linkElement =
productTile.findElement(By.cssSelector(".productname-link"));
                       product.setProductUrl(linkElement.getAttribute("href"));
                   } catch (Exception e) {
                       product.setProductUrl("N/A");
                   }

                   // Extract color information from swatch
                   try {
                       WebElement colorElement =
productTile.findElement(By.cssSelector(".swatch.selected img"));
                       product.setColor(colorElement.getAttribute("title"));
                   } catch (Exception e) {
                       product.setColor("N/A");
                   }

                   // Extract description - JBL uses .product-description
                   try {
                       WebElement descriptionElement =
productTile.findElement(By.cssSelector(".product-description"));
                       product.setDescription(descriptionElement.getText().trim());
                   } catch (Exception e) {
                       product.setDescription("N/A");
                   }

                   // Set category based on current page
                   String currentUrl = driver.getCurrentUrl();
                   if (currentUrl.contains("party-speakers")) {
                       product.setCategory("Party Speakers");
                   } else if (currentUrl.contains("sale")) {
                       product.setCategory("Sale");
                   } else if (currentUrl.contains("bluetooth-speakers")) {
                       product.setCategory("Bluetooth Speakers");
                   } else {
                       product.setCategory("Home Audio");
                   }

                   // Only add products with valid names
                   if (!product.getName().equals("N/A") &&
!product.getName().isEmpty()) {
                       products.add(product);
                       System.out.println("Extracted: " + product.getName() + " - " +
product.getPrice());
                   }

               } catch (Exception e) {
                   System.out.println("Error extracting product data: " +
e.getMessage());
               }
           }

       } catch (Exception e) {
           System.out.println("Error during data extraction: " + e.getMessage());
       }
   }

   /**
    * Extract image URL from JSON data in swatch
    */
   private String extractImageUrlFromJson(String jsonData) {
       try {
           // Simple JSON parsing to extract thumbnailImageUrl
           if (jsonData.contains("\"thumbnailImageUrl\"")) {
               int startIndex = jsonData.indexOf("\"thumbnailImageUrl\":\"") + 21;
               int endIndex = jsonData.indexOf("\"", startIndex);
               if (startIndex > 20 && endIndex > startIndex) {
                   return jsonData.substring(startIndex, endIndex);
               }
           }
       } catch (Exception e) {
           System.out.println("Error parsing JSON for image URL: " + e.getMessage());
       }
       return "N/A";
   }

   /**
    * Helper method to extract text using multiple selectors
    */
   private String extractTextWithMultipleSelectors(WebElement parent, String[]
selectors, String defaultValue) {
       for (String selector : selectors) {
           try {
               WebElement element = parent.findElement(By.cssSelector(selector));
               String text = element.getText().trim();
               if (!text.isEmpty()) {
                   return text;
               }
           } catch (Exception e) {
               // Continue to next selector
           }
       }
       return defaultValue;
   }

   /**
    * Navigate through multiple pages within a category
    */
   public void navigateThroughPages() {
       System.out.println("Navigating through pages within category...");

       try {
           int maxPages = 3; // Limit to 3 pages per category
           int currentPage = 1;

           while (currentPage <= maxPages) {
               System.out.println("Crawling page " + currentPage + " of category...");

               // Extract data from current page
               extractProductDataFromCurrentPage();

               // Try to find and click next page button with multiple selectors
               String[] nextButtonSelectors = {
                   ".pagination-next", ".next-page", "[aria-label='Next']",
                   ".pagination .next", ".page-next", ".load-more",
                   "button[class*='next']", ".pager-next"
               };

               boolean nextPageFound = false;
               for (String selector : nextButtonSelectors) {
                   try {
                       WebElement nextButton =
driver.findElement(By.cssSelector(selector));
                       if (nextButton.isDisplayed() && nextButton.isEnabled()) {
                           // Scroll to element
                           ((JavascriptExecutor)
driver).executeScript("arguments[0].scrollIntoView(true);", nextButton);
                           Thread.sleep(1000);

                           // Click next button
                           nextButton.click();

                           // Wait for page to load
                           Thread.sleep(3000);
                           currentPage++;
                           nextPageFound = true;

                           System.out.println("Navigated to page " + currentPage);
                           break;
                       }
                   } catch (Exception e) {
                       // Continue to next selector
                   }
               }

               if (!nextPageFound) {
                   System.out.println("No more pages available in this category");
                   break;
               }
           }

       } catch (Exception e) {
           System.out.println("Error during page navigation: " + e.getMessage());
       }
   }
   /**
    * Save scraped data to CSV file
    */
   public void saveToCSV(String filename) {
       System.out.println("Saving data to CSV file: " + filename);

       try (CSVWriter writer = new CSVWriter(new FileWriter(filename))) {
           // Write header
           String[] header = {
               "Name", "Price", "Original Price", "Discount", "Availability",
               "Image URL", "Product URL", "Category", "Color", "Description"
           };
           writer.writeNext(header);

           // Write product data
           for (Product product : products) {
               String[] row = {
                   product.getName(),
                   product.getPrice(),
                   product.getOriginalPrice(),
                   product.getDiscount(),
                   product.getAvailability(),
                   product.getImageUrl(),
                   product.getProductUrl(),
                   product.getCategory(),
                   product.getColor(),
                   product.getDescription()
               };
               writer.writeNext(row);
           }

           System.out.println("Successfully saved " + products.size() + " products to
" + filename);

       } catch (IOException e) {
           System.err.println("Error saving to CSV: " + e.getMessage());
       }
   }

   /**
    * Save scraped data to JSON file
```

```java
     */
    public void saveToJSON(String filename) {
        System.out.println("Saving data to JSON file: " + filename);

        try {
            objectMapper.writerWithDefaultPrettyPrinter()
                       .writeValue(new File(filename), products);
            System.out.println("Successfully saved data to " + filename);
        } catch (IOException e) {
            System.err.println("Error saving to JSON: " + e.getMessage());
        }
    }

    /**
     * Print summary of scraped data
     */
    public void printSummary() {
        System.out.println("\n=== SCRAPING SUMMARY ===");
        System.out.println("Total products scraped: " + products.size());

        if (!products.isEmpty()) {
            System.out.println("\nSample products:");
            for (int i = 0; i < Math.min(5, products.size()); i++) {
                System.out.println((i + 1) + ". " + products.get(i));
            }
        }

        // Count products by category
        Map<String, Integer> categoryCount = new HashMap<>();
        for (Product product : products) {
            categoryCount.merge(product.getCategory(), 1, Integer::sum);
        }

        System.out.println("\nProducts by category:");
        categoryCount.forEach((category, count) ->
            System.out.println("- " + category + ": " + count + " products"));
    }

    /**
     * Close the WebDriver
     */
    public void closeDriver() {
        if (driver != null) {
            System.out.println("Closing WebDriver...");
            driver.quit();
        }
    }

    /**
     * Main method to run the enhanced scraper
     */
    public static void main(String[] args) {
        JBLScrapper scraper = new JBLScrapper();

        try {
            // Initialize WebDriver
            scraper.initializeDriver();

            // Crawl multiple pages with enhanced features
            scraper.crawlMultiplePages();

            // Save data to files
            scraper.saveToCSV("jbl_products.csv");
            scraper.saveToJSON("jbl_products.json");

            // Print summary
            scraper.printSummary();

        } catch (Exception e) {
            System.err.println("Error during scraping: " + e.getMessage());
            e.printStackTrace();
        } finally {
            // Always close the driver
            scraper.closeDriver();
        }
    }
}
```