ALGORITHM AND DATA STRUCTURES

ASSIGNMENT 2 – LINKEDLIST

By:

I Kadek Mahesa Permana Putra

F1D02410052

FACULTY OF ENGINEERING

DEPARTEMENT OF INFORMATICS ENGINEERING

UNIVERSITY OF MATARAM

2025

TASK:

- Implements the Singly LinkedList with ADT
- Implements the Doubly LinkedList with ADT
- Implements the Circular LinkedList with ADT

A. DIRECTORY

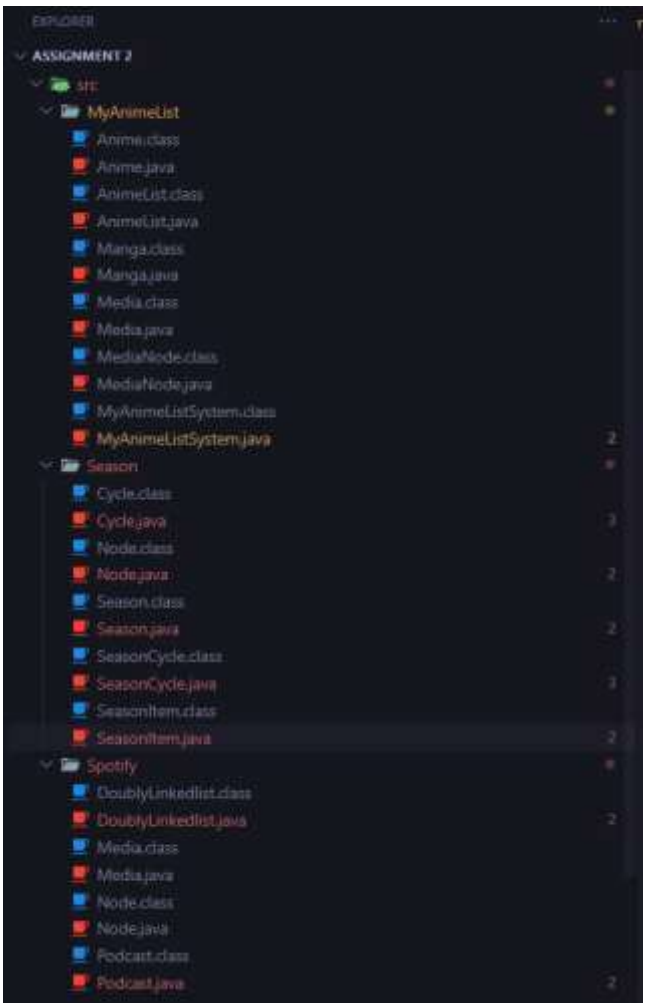


Image 1. Project Directory

The following is the directory hierarchy used for this assignment. There are three main programs for implementing the three types of linked lists, divided into the **MyAnimeList**, **Spotify**, and **SeasonCycle** programs. In each directory,

several files with the *.class* extension can be seen, which are the compiled results of Java source files, produced using *javac* (Java compiler).

B. SOURCE CODE

MyAnimeList Program

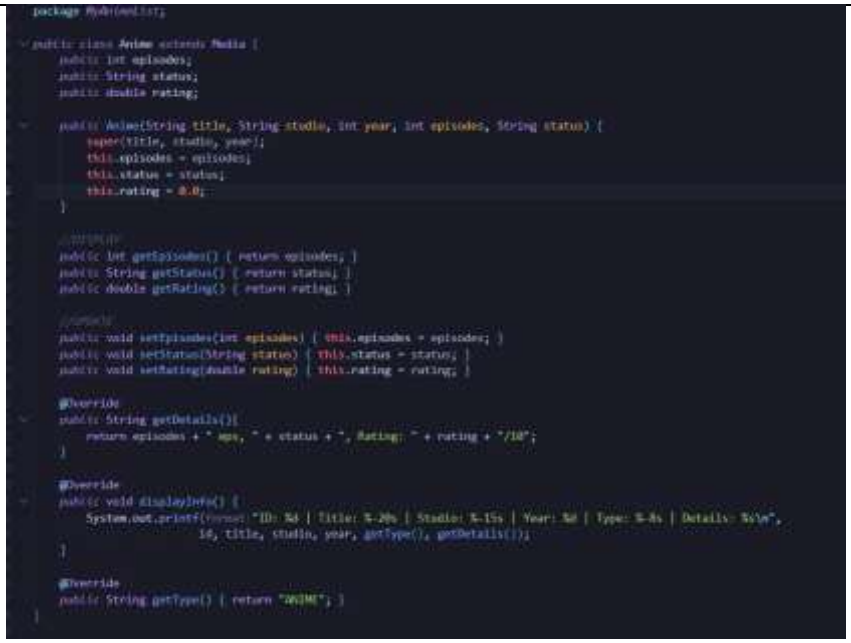
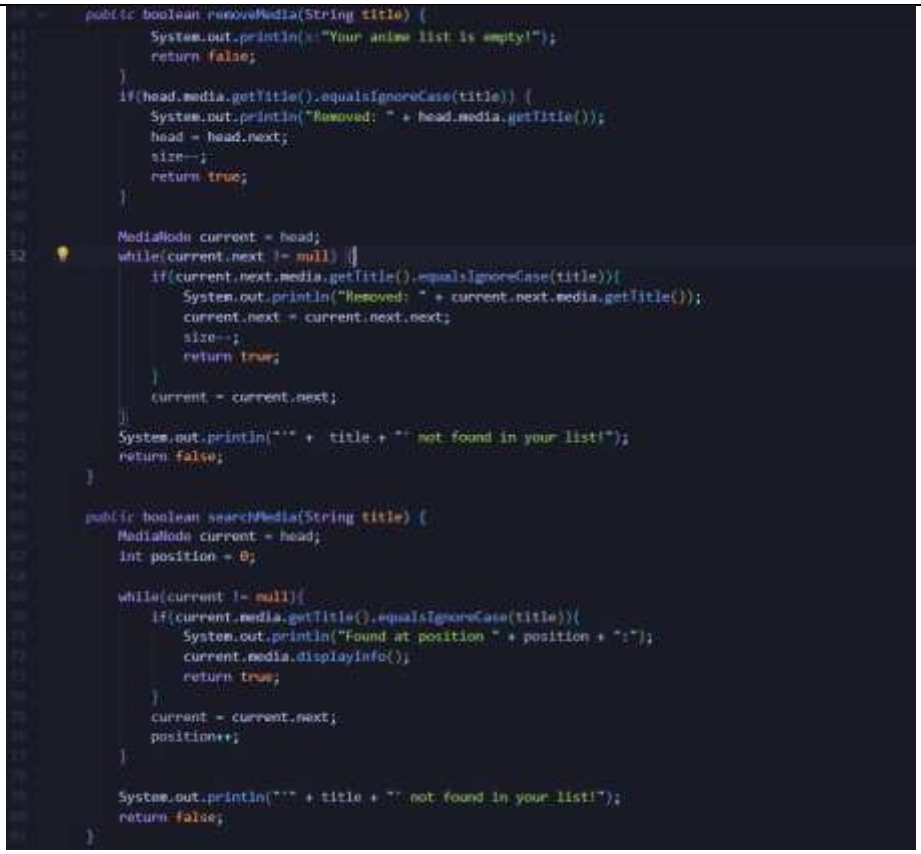
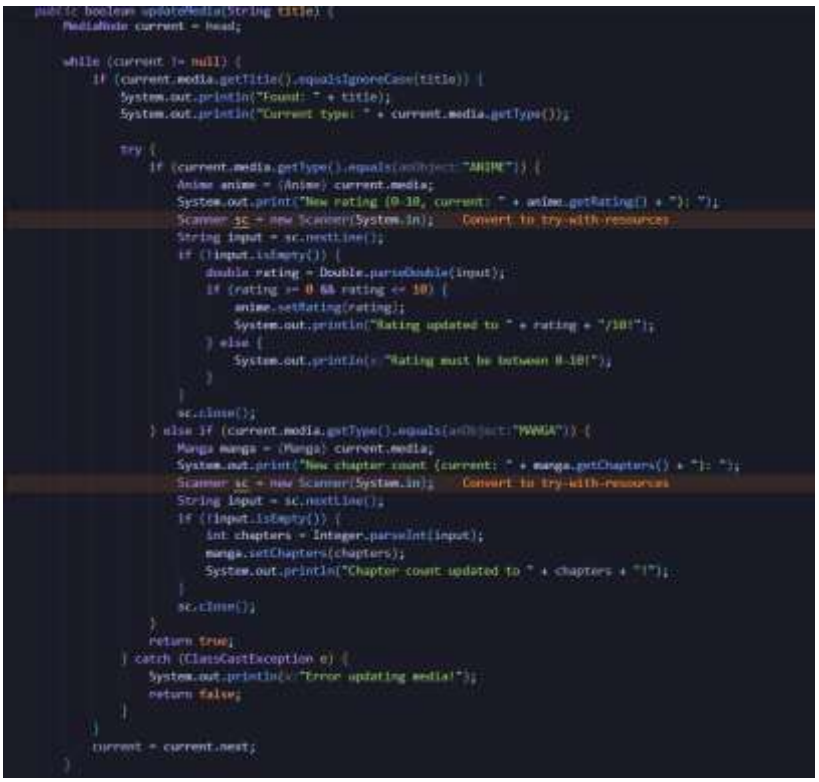


Image 2. Anime Class



```
public boolean removeMedia(String title) {
        System.out.println(x: "Your anime list is empty!");
        return false;
    }
    if(head.media.getTitle().equalsIgnoreCase(title)) {
        System.out.println("Removed: " + head.media.getTitle());
        head = head.next;
        size--;
        return true;
    }

    MediaNode current = head;
    while(current.next != null) {
        if(current.next.media.getTitle().equalsIgnoreCase(title)){
            System.out.println("Removed: " + current.next.media.getTitle());
            current.next = current.next.next;
            size--;
            return true;
        }
        current = current.next;
    }
    System.out.println("'" +  title + "' not found in your list!");
    return false;
}

public boolean searchMedia(String title) {
    MediaNode current = head;
    int position = 0;

    while(current != null){
        if(current.media.getTitle().equalsIgnoreCase(title)){
            System.out.println("Found at position " + position + ":");
            current.media.displayInfo();
            return true;
        }
        current = current.next;
        position++;
    }

    System.out.println("'" + title + "' not found in your list!");
    return false;
}
```



```
public boolean updateMedia(String title) {
    MediaNode current = head;

    while (current != null) {
        if (current.media.getTitle().equalsIgnoreCase(title)) {
            System.out.println("Found: " + title);
            System.out.println("Current type: " + current.media.getType());

            try {
                if (current.media.getType().equals(anObject:"ANIME")) {
                    Anime anime = (Anime) current.media;
                    System.out.print("New rating (0-10, current: " + anime.getRating() + "): ");
                    Scanner sc = new Scanner(System.in);    Convert to try-with-resources
                    String input = sc.nextLine();
                    if (!input.isEmpty()) {
                        double rating = Double.parseDouble(input);
                        if (rating >= 0 && rating <= 10) {
                            anime.setRating(rating);
                            System.out.println("Rating updated to " + rating + "/10!");
                        } else {
                            System.out.println(x: "Rating must be between 0-10!");
                        }
                    }
                    sc.close();
                } else if (current.media.getType().equals(anObject:"MANGA")) {
                    Manga manga = (Manga) current.media;
                    System.out.print("New chapter count (current: " + manga.getChapters() + "): ");
                    Scanner sc = new Scanner(System.in);    Convert to try-with-resources
                    String input = sc.nextLine();
                    if (!input.isEmpty()) {
                        int chapters = Integer.parseInt(input);
                        manga.setChapters(chapters);
                        System.out.println("Chapter count updated to " + chapters + "!");
                    }
                    sc.close();
                }
                return true;
            } catch (ClassCastException e) {
                System.out.println(x: "Error updating media!");
                return false;
            }
        }
        current = current.next;
    }
```

```java
    //SHOW BY CATEGORY
    public void showByCategory(String category) {
        if (head == null) {
            System.out.println(x:"Your list is empty!");
            return;
        }

        System.out.println("\nFILTERING BY: " + category.toUpperCase());
        System.out.println("=".repeat(count:95));

        MediaNode current = head;
        int found = 0;

        while (current != null) {
            if (current.media.getType().equals(category.toUpperCase())) {
                current.media.displayInfo();
                found++;
            }
            current = current.next;
        }

        if (found == 0) {
            System.out.println("No " + category.toLowerCase() + " found in your list!");
        } else {
            System.out.println("Found " + found + " " + category.toLowerCase() + "(s)");
        }
    }

    public boolean isEmpty() {
        return head == null;
    }

}
```

Image 3. AnimeList class

```java
package MyAnimeList;

public class Manga extends Media {
    public String mangaStatus;
    public int chapters;
    public String author;

    public Manga(String title, String studio, int year, int chapters, String mangaStatus, String author){
        super(title, studio, year);
        this.mangaStatus = mangaStatus;
        this.chapters = chapters;
        this.author = author;
    }

    //DISPLAY
    public int getChapters() { return chapters; }
    public String getAuthor() { return author; }
    public String getMangaStatus() { return mangaStatus; }

    //UPDATE
    public void setChapters(int chapters) { this.chapters = chapters; }
    public void setAuthor(String author) { this.author = author; }
    public void setMangaStatus(String mangaStatus) { this.mangaStatus = mangaStatus; }

    @Override
    public String getType() { return "MANGA"; }

    @Override
    public String getDetails() {
        return chapters + " chapters, " + mangaStatus + ", Author: " + author;
    }

    @Override
    public void displayInfo(){
        System.out.printf(format:"ID: %d | Title: %-20s | Studio: %-15s | Year: %d | Type: %-8s | Details: %s\n",
                          id, title, studio, year, getType(), getDetails());
    }

}
```

Image 4. Manga Class

```java
package MyAnimeList;

public class Media {
    public String title;
    public String studio;
    public int year;
    public int id;
    public int idCounter = 1001;

    public Media(String title, String studio, int year) {
        this.title =  title;
        this.studio = studio;
        this.year = year;
        this.id = ++idCounter;
    }

    public String getTitle() { return title; }
    public String getStudio() { return studio; }
    public int getYear() { return year; }
    public int getId() { return id; }
    public void displayInfo() {
        System.out.printf(format:"ID: %d | Title: %-20s | Studio: %-15s | Year: %d",
                          id, title, studio, year);
    }
    public String getType() { return "MEDIA"; }
    public String getDetails() { return "Basic Media"; }
}
```

Image 5. Media Class

```java
package MyAnimeList;

public class MediaNode {
    Media media;
    MediaNode next;

    public MediaNode(Media media){
        this.media = media;
        this.next = null;
    }
}
```

Image 6. MediaNode class

```java
package MyAnimeList;
import java.util.Scanner;

public class MyAnimeListSystem {
    public static Scanner scanner = new Scanner(System.in);
    public static AnimeList myList = new AnimeList();

    public static void main(String[] args) {
        System.out.println("MY ANIME LIST MANAGEMENT SYSTEM");
        System.out.println("=".repeat(40));

        myList.addAnime("Attack on Titan", "Mappa", 2013, 87, "Completed");
        myList.addAnime("One Piece", "Toei Animation", 1999, 1000, "Ongoing");
        myList.addManga("Demon Slayer", "Shuelsha", 2016, 205, "Completed", "Koyoharu Gotouge");
        myList.addAnime("Naruto", "Pierrot", 2002, 720, "Completed");
        myList.addAnime("Re:zero", "White Fox", 2016, 90, "Ongoing");
        myList.addManga("Jujutsu Kaisen", "Shueisha", 2020, 170, "Completed", "Gege Akutami");

        System.out.println("Sample dataset added to your list");

        boolean running = true;
        while(running){
            showMenu();
            int choice = getInput("Choose your option: ");

            switch(choice){    Convert switch to rule switch
                case 1: addAnime(); break;
                case 2: addManga(); break;
                case 3: removeMedia(); break;
                case 4: searchMedia(); break;
                case 5: myList.displayAll(); break;
                case 6: showByCategory(); break;
                case 7: myList.showStatistics(); break;
                case 8:
                    running = false;
                    System.out.println("Arigatou Gozaimashita, See You!");
                    break;
                default:
                    System.out.println("Invalid option, please try again!");
            }
            if (running) {
                System.out.println("\nPress Enter to continue...");
                scanner.nextLine();
            }
```

```java
    public static void showMenu() {
        System.out.println("\n" + "=".repeat(35));
        System.out.println("MY ANIME LIST MENU:");
        System.out.println("1.  Add Anime Series");
        System.out.println("2.  Add Manga");
        System.out.println("3.  Remove from List");
        System.out.println("4.  Search Title");
        System.out.println("5.  Show My Complete List");
        System.out.println("6.  Filter by Category");
        System.out.println("7.  Show Statistics");
        System.out.println("8.  Exit");
        System.out.println("=".repeat(35));
    }

    public static void addAnime() {
        System.out.println("\n ADD NEW ANIME:");
        System.out.print("Title: ");
        String title = scanner.nextLine();
        System.out.print("Studio: ");
        String studio = scanner.nextLine();
        System.out.print("Year: ");
        int year = Integer.parseInt(scanner.nextLine());
        System.out.print("Episodes: ");
        int episodes = Integer.parseInt(scanner.nextLine());
        System.out.print("Status (Ongoing/Completed/On Hold): ");
        String status = scanner.nextLine();

        myList.addAnime(title, studio, year, episodes, status);
    }

    public static void addManga() {
        System.out.println("\n ADD NEW MANGA:");
        System.out.print("Title: ");
        String title = scanner.nextLine();
        System.out.print("Publisher: ");
        String publisher = scanner.nextLine();
        System.out.print("Year: ");
        int year = Integer.parseInt(scanner.nextLine());
        System.out.print("Chapters: ");
        int chapters = Integer.parseInt(scanner.nextLine());
        System.out.print("Status (Ongoing/Completed/Hiatus): ");
        String status = scanner.nextLine();
        System.out.print("Author: ");
```

```
    public static void removeMedia() {
        if (myList.isEmpty()) {
            System.out.println(x:"Your list is empty!");
            return;
        }
        System.out.print(s:"Title to remove: ");
        String title = scanner.nextLine();
        myList.removeMedia(title);
    }

    public static void searchMedia() {
        if (myList.isEmpty()) {
            System.out.println(x:"Your list is empty!");
            return;
        }
        System.out.print(s:"Search title: ");
        String title = scanner.nextLine();
        myList.searchMedia(title);
    }

    public static void showByCategory() {
        if (myList.isEmpty()) {
            System.out.println(x:"Your list is empty!");
            return;
        }
        System.out.println(x:"Categories:");
        System.out.println(x:"1. Anime");
        System.out.println(x:"2. Manga");
        int choice = getInput(prompt:"Choose category: ");

        switch (choice) {    Convert switch to rule switch
            case 1: myList.showByCategory(category:"ANIME"); break;
            case 2: myList.showByCategory(category:"MANGA"); break;
            default: System.out.println(x:"Invalid choice!");
        }
    }

    public static int getInput(String prompt) {
        System.out.print(prompt);
        return Integer.parseInt(scanner.nextLine());
    }
```

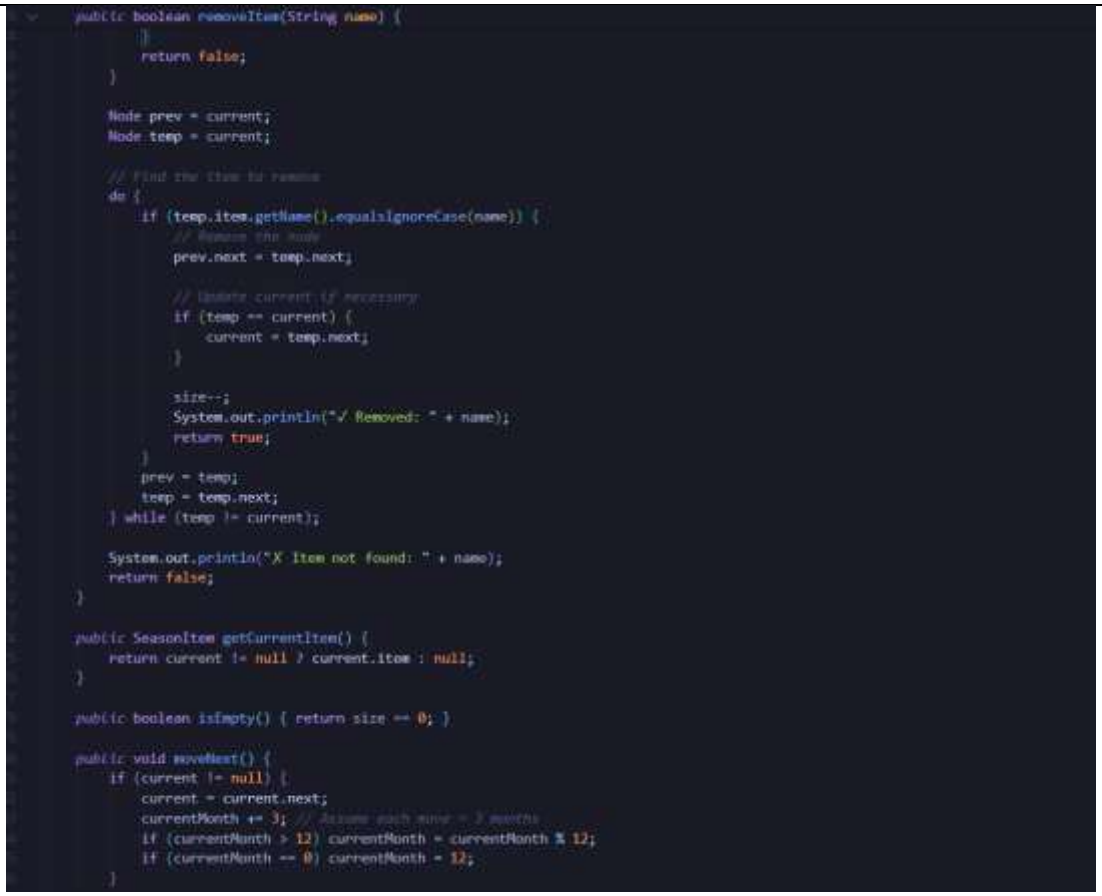
Image 7. MyAnimeListSystem class (Main)

The image above shows the implementation of an ADT, namely the **Anime class**, which inherits properties from another class called **Media**. This class also declares many methods that can later be invoked from other classes. As shown in **Image 3**, there are many methods implemented, such as methods to add data into the list, remove, search, and update. **Image 4** is quite similar to the **Anime** class, which is a subclass of the **Media** superclass in **Image 5**. **Image 6** shows the main **Node** used to form this linked list, with the initial next Node set to null or not pointing to any Node (which will later point to another Node when new data is added). **Image 7** is the main method (main function) of this program, where most

of the code is the initialization of static methods to display a text-based User Interface (terminal UI), using the **Scanner** library to read input from the terminal and applied in the logic of the main menu switch-case.

SeasonCycle Program

```
public class Cycle {
    public void autoCycle(int steps) {

        for (int i = 1; i <= steps; i++) {
            System.out.println("Step " + i + ": " + current.item.getName());
            moveNext();

            try {
                Thread.sleep(millis:800); // Pause    Thread.sleep called in loop
            } catch (InterruptedException e) {
                break;
            }
        }

        System.out.println("-".repeat(count:30));
        System.out.println("Current: " + current.item.getName());
    }
}
```



```
    public boolean removeItem(String name) {
        }
        return false;
    }

    Node prev = current;
    Node temp = current;

    // Find the item to remove
    do {
        if (temp.item.getName().equalsIgnoreCase(name)) {
            // Remove the node
            prev.next = temp.next;

            // Update current if necessary
            if (temp == current) {
                current = temp.next;
            }

            size--;
            System.out.println("✓ Removed: " + name);
            return true;
        }
        prev = temp;
        temp = temp.next;
    } while (temp != current);

    System.out.println("X Item not found: " + name);
    return false;
}

public SeasonItem getCurrentItem() {
    return current != null ? current.item : null;
}

public boolean isEmpty() { return size == 0; }

public void moveNext() {
    if (current != null) {
        current = current.next;
        currentMonth += 3;  // Assume each move = 3 months
        if (currentMonth > 12) currentMonth = currentMonth % 12;
        if (currentMonth == 0) currentMonth = 12;
    }
```

```
80      public void moveNext() {
81          if (current != null) {
82              current = current.next;
83              currentMonth += 3; // Assume each move = 3 months
84              if (currentMonth > 12) currentMonth = currentMonth % 12;
85              if (currentMonth == 0) currentMonth = 12;
86          }
87      }
88
89      public void movePrevious() {
90          if (current == null || size <= 1) return;
91
92          // Find previous node
93          Node temp = current;
94          while (temp.next != current) {
95              temp = temp.next;
96          }
97          current = temp;
98          currentMonth -= 3;
99          if (currentMonth <= 0) currentMonth = currentMonth + 12;
00      }
01
```

Image 8. Cycle Class

```java
package Season;     Incorrect Package

public class Node {
    SeasonItem item;
    Node next;

    public Node(SeasonItem item) {
        this.item = item;
        this.next = null;
    }
}

```

Image 9. Node Class

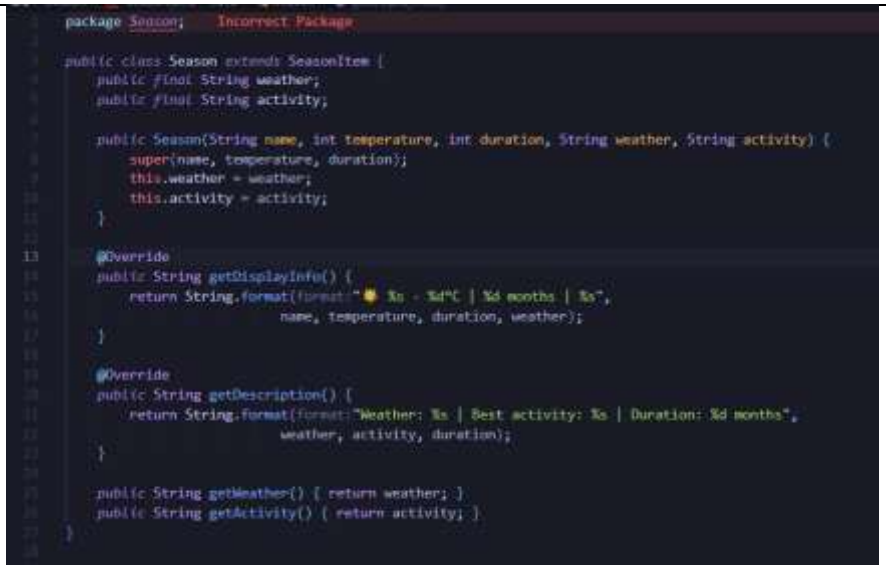


Image 10. Season Class

```java
package Season;     Incorrect Package

public abstract class SeasonItem {
    public String name;
    public int temperature;
    public int duration;

    public SeasonItem(String name, int temperature, int duration) {
        this.name = name;
        this.temperature = temperature;
        this.duration = duration;
    }

    public abstract String getDisplayInfo();
    public abstract String getDescription();

    public String getName() { return name; }
    public int getTemperature() { return temperature; }
    public int getDuration() { return duration; }
}

```

Image 11. SeasonItem Class

```java
package season;    Incorrect Package

import java.util.Scanner;

public class SeasonCycle {
    static Cycle cycle = new Cycle();
    static Scanner scanner = new Scanner(System.in);

    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        initializeSeasons();
        System.out.println("🌸 Welcome to Season Cycle Manager 🌸");

        boolean running = true;
        while(running) {
            displayMenu();
            int choice = getIntInput("Choose your option: ");

            switch(choice) {    Convert switch to rule switch
                case 1: addSeason(); break;
                case 2: removeSeason(); break;
                case 3: nextSeason(); break;
                case 4: prevSeason(); break;
                case 5: autoCycleDemo(); break;
                case 0:
                    System.out.println("👋 Goodbye! Enjoy all seasons!");
                    running = false;
                    break;
                default:
                    System.out.println("❌ Invalid choice!");
            }
            if (running) {
                System.out.println("\nPress Enter to continue...");
                scanner.nextLine();
            }
        }
        scanner.close();
    }
```

```java
public static void displayMenu() {
    System.out.println("\n" + "=".repeat(40));
    System.out.println("🌸 SEASON CYCLE MENU:");
    System.out.println("1. Add New Season");
    System.out.println("2. Remove Season");
    System.out.println("3. Next Season");
    System.out.println("4. Previous Season");
    System.out.println("5. Auto Cycle Demo");
    System.out.println("0. Exit");
    System.out.println("=".repeat(40));
}

private static int getIntInput(String prompt) {
    while (true) {
        try {
            System.out.print(prompt);
            return Integer.parseInt(scanner.nextLine());
        } catch (NumberFormatException e) {
            System.out.println("❌ Enter valid number!");
        }
    }
}

private static void initializeSeasons() {
    cycle.addItem(new Season("Spring", 15, 3, "Mild and rainy", "Planting flowers"));
    cycle.addItem(new Season("Summer", 25, 3, "Hot and sunny", "Beach vacation"));
    cycle.addItem(new Season("Autumn", 10, 3, "Cool and windy", "Leaf watching"));
    cycle.addItem(new Season("Winter", -5, 3, "Cold and snowy", "Skiing"));

    System.out.println("🌸 Basic seasons initialized!");
}

private static void autoCycleDemo() {
    if (cycle.isEmpty()) {
        System.out.println("❌ Cycle is empty!");
        return;
    }

    int steps = getIntInput("Number of steps: ");
    cycle.autoCycle(steps);
}
```

Image 12. SeasonCycle class (Main)

In this program, the type of linked list used is a **circular linked list**, where the difference lies in the **Cycle** class shown in **Image 8**. In the add method, the pointer points back to itself, as an illustration of the head pointing to the head (circular). Overall, this program has many similarities with the previous one, with the only difference being in how the data is inserted into this linked list. This program also uses **Thread** to provide a delay (sleep) in each cycle session to enhance the user experience.

Spotify Program

```
package Spotify;    Incorrect Package

public class DoublyLinkedlist {
    Node head;
    Node tail;
    Node current;
    int size;

    public DoublyLinkedlist() {
        this.head = null;
        this.tail = null;
        this.current = null;
        this.size = 0;
    }

    public void addToEnd(Media item) {
        Node newNode = new Node(item);

        if (head == null) {
            head = tail = current = newNode;
        } else {
            tail.next = newNode;
            newNode.prev = tail;
            tail = newNode;
        }
        size++;
        System.out.println("✓ Added: " + item.getTitle() + " to playlist");
    }

    public void addToBeginning(Media item) {
        Node newNode = new Node(item);

        if (head == null) {
            head = tail = current = newNode;
        } else {
            newNode.next = head;
            head.prev = newNode;
            head = newNode;
        }
        size++;
        System.out.println("✓ Added: " + item.getTitle() + " to playlist");
    }

    public boolean remove(String title) {
        Node temp = head;
```

```
    public boolean remove(String title) {
        Node temp = head;

        while(temp != null) {
            if(temp.data.getTitle().equalsIgnoreCase(title)) {
                if(temp == current) {
                    if(temp.next != null) {
                        current = temp.next;
                    } else if(temp.prev != null) {
                        current = temp.prev;
                    } else {
                        current = null;
                    }
                }
                if(temp.prev != null) {
                    temp.prev.next = temp.next;
                } else {
                    head = temp.next;
                }

                if(temp.next != null) {
                    temp.next.prev = temp.prev;
                } else {
                    tail = temp.prev;
                }

                size--;
                System.out.println("✓ Removed: " + title + " from playlist");
                return true;
            }
            temp = temp.next;
        }
        System.out.println("✗ Song: " + title + " not found in playlist");
        return false;
    }

    public boolean next(){
        if(current != null && current.next != null) {
            current = current.next;
            return true;
        }
        return false;
    }
```

```
public boolean previous(){
    if(current != null && current.prev != null) {
        current = current.prev;
        return true;
    }
    return false;
}

public Media getCurrentItem() {
    return current != null ? current.data : null;
}

public void shuffle() {
    if(size <= 1) return;

    Media[] items = new Media[size];
    Node temp = head;
    int index = 0;

    while(temp != null) {
        items[index++] = temp.data;
        temp = temp.next;
    }
    for (int i = items.length - 1; i > 0; i--) {
        int j = (int) (Math.random() * (i + 1));
        Media tempItem = items[i];
        items[i] = items[j];
        items[j] = tempItem;
    }

    clear();
    for (Media item : items) {
        addToEnd(item);
    }
    System.out.println("✓ Playlist shuffled");
}

public void clear() {
    head = tail = current = null;
    size = 0;
}
```

```
public void displayPlaylist() {
        return;
    }

    System.out.println("\n=== PLAYLIST ===");
    Node temp = head;
    int index = 1;

    while (temp != null) {
        String marker = (temp == current) ? "▶ " : "  ";
        System.out.println(marker + index + ". " + temp.data.getDisplayInfo());
        temp = temp.next;
        index++;
    }
    System.out.println("================");
    System.out.println("Total items: " + size);
}

public int getSize() { return size; }
public boolean isEmpty() { return size == 0; }
public void search(String keyword) {
    System.out.println("\n=== SEARCH RESULTS ===");
    Node temp = head;
    int found = 0;

    while (temp != null) {
        if (temp.data.getTitle().toLowerCase().contains(keyword.toLowerCase()) ||
            temp.data.getArtist().toLowerCase().contains(keyword.toLowerCase())) {
            System.out.println((found + 1) + ". " + temp.data.getDisplayInfo());
            found++;
        }
        temp = temp.next;
    }

    if (found == 0) {
        System.out.println("No results found for: " + keyword);
    } else {
        System.out.println("Found " + found + " item(s)");
    }
    System.out.println("====================");
}
}
```

Image 13. DoublyLinkedlist Class

```java
package Spotify;    Incorrect Package

public abstract class Media {
    String title;
    String artist;
    int duration;

    public Media(String title, String artist, int duration) {
        this.title = title;
        this.artist = artist;
        this.duration = duration;
    }

    public abstract String getDisplayInfo();
    public abstract String getType();

    public String getTitle(){ return title; }
    public String getArtist(){ return artist; }
    public int getDuration(){ return duration; }

    public String formatDuration() {
        int minutes = duration / 60;
        int seconds = duration % 60;
        return String.format(format:"%d:%02d", minutes, seconds);
    }
}
```

Image 14. Media Class

src > Spotify > Node.java > ...

```java
package Spotify;

public class Node {
    Media data;    Variable data is never read
    Node next;    Variable next is never read
    Node prev;    Variable prev is never read

    public Node(Media data) {
        this.data = data;
        this.next = null;
        this.prev = null;
    }
}
```

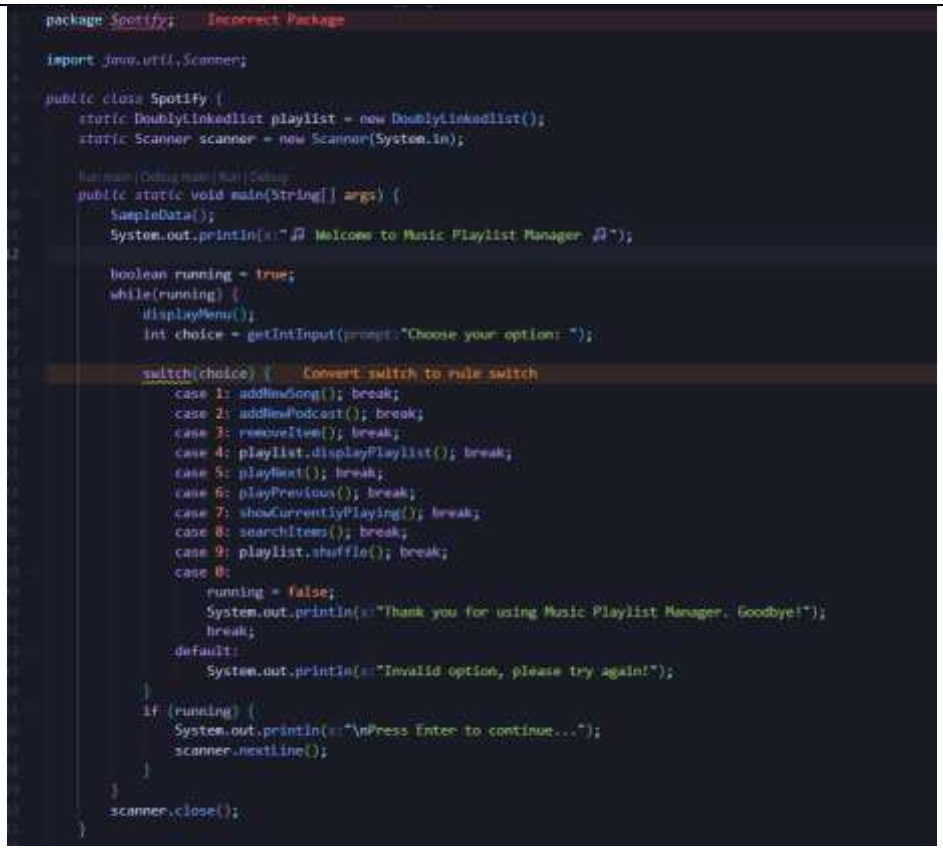
Image 15. Node Class

```
package Spotify;    Incorrect Package

public class Podcast extends Media {
    String episode;
    String description;

    public Podcast(String title, String artist, int duration, String episode, String description) {
        super(title, artist, duration);
        this.episode = episode;
        this.description = description;
    }

    @Override
    public String getDisplayInfo() {
        return String.format(format:"🎙 %s - %s [%s] (%s) - %s",
                             title, artist, episode, description, formatDuration());
    }

    @Override
    public String getType() {
        return "Podcast";
    }

    public String getEpisode() { return episode; }
    public String getDescription() { return description; }

}
```

Image 16. Podcast Class

```
package Spotify;    Incorrect Package

public class Song extends Media{
    String album;
    String genre;

    public Song(String title, String artist, int duration, String album, String genre) {
        super(title, artist, duration);
        this.album = album;
        this.genre = genre;
    }

    @Override
    public String getDisplayInfo() {
        return String.format(format:"♪ %s - %s [%s] (%s) - %s",
                             title, artist, album, genre, formatDuration());
    }

    @Override
    public String getType() {
        return "Song";
    }

    public String getAlbum() { return album; }
    public String getGenre() { return genre; }
}
```

Image 17. Song Class



```
package Spotify;    Incorrect Package

import java.util.Scanner;

public class Spotify {
    static DoublyLinkedlist playlist = new DoublyLinkedlist();
    static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        SampleData();
        System.out.println("🎵 Welcome to Music Playlist Manager 🎵");

        boolean running = true;
        while(running) {
            displayMenu();
            int choice = getIntInput("Choose your option: ");

            switch(choice) {    Convert switch to rule switch
                case 1: addNewSong(); break;
                case 2: addNewPodcast(); break;
                case 3: removeItem(); break;
                case 4: playlist.displayPlaylist(); break;
                case 5: playNext(); break;
                case 6: playPrevious(); break;
                case 7: showCurrentlyPlaying(); break;
                case 8: searchItems(); break;
                case 9: playlist.shuffle(); break;
                case 0:
                    running = false;
                    System.out.println("Thank you for using Music Playlist Manager. Goodbye!");
                    break;
                default:
                    System.out.println("Invalid option, please try again!");
            }
            if (running) {
                System.out.println("\nPress Enter to continue...");
                scanner.nextLine();
            }
        }
        scanner.close();
    }
```

Image 18. Spotify Class (Main)

The following class above (especially DoublyLinkedlist on **image 13**) stores four main attributes: **head**, **tail**, **current**, and **size**. head points to the first node, tail points to the last node, and current points to the active/selected item. Since each Node has two pointers (next and prev) allowing to move forward and backward easily. The **addToEnd** and **addToBeginning** methods handle insertion in O(1). If the list is empty, all three pointers (head, tail, current) are set to the new node. Otherwise, the next/prev pointers are linked accordingly, and tail or head is updated. The size is always incremented after insertion. Because the pointer updates are direct, adding elements is always fast without traversal.

The **remove(String title)** method performs a linear search starting from head to find the first node whose title matches (using **equalsIgnoreCase**). Once found, it handles several cases, if the node is current, current is moved to next if available, otherwise to prev, or finally set to null if the list is empty. Then, neighboring pointers (**prev.next** and **next.prev**) are updated, if the removed node

was head or tail, those fields are reassigned. The size decreases, and the method returns a boolean.

Navigation with **next()** and **previous()** is straightforward, they simply check if current and its next/prev exist, then move current and return true if successful. There is no wrap-around mechanism when at the tail, next() returns false and does not jump to head. This fits a non-looping playlist, but could be modified to circular for a repeat mode.

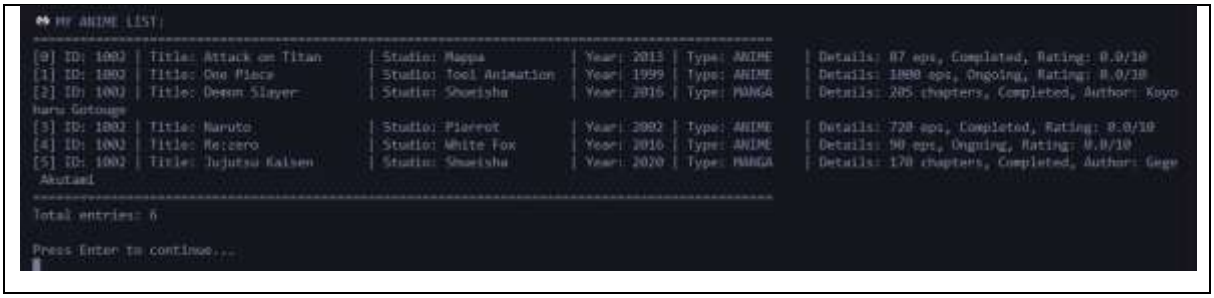
getCurrentItem() returns the Media at current or null if none; **displayPlaylist()** traverses from head to tail and marks the current item with the "▶" symbol. **search(keyword)** performs a case-insensitive search on title and artist using contains, printing all matching results.

C. RESULT

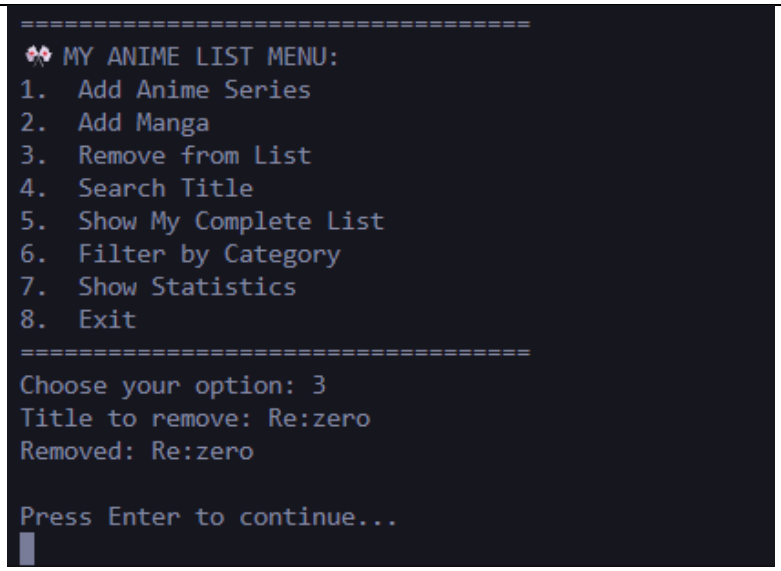
I will show several screenshots of the display from each program, although not in too much detail only the important parts will be highlighted.

MyAnimeList Program

This is the initial display of the **MyAnimeList** program. At the start, the program shows several menu options that the user can choose from. The program is initialized with some data to make the demonstration easier.



When the user selects option 5, it will open the **Complete List** menu (to display all the data in the linked list). The data shown here comes from the initial initialization, and it is displayed in the order it was inserted. If new data is added later, it will appear at the bottom of the list.

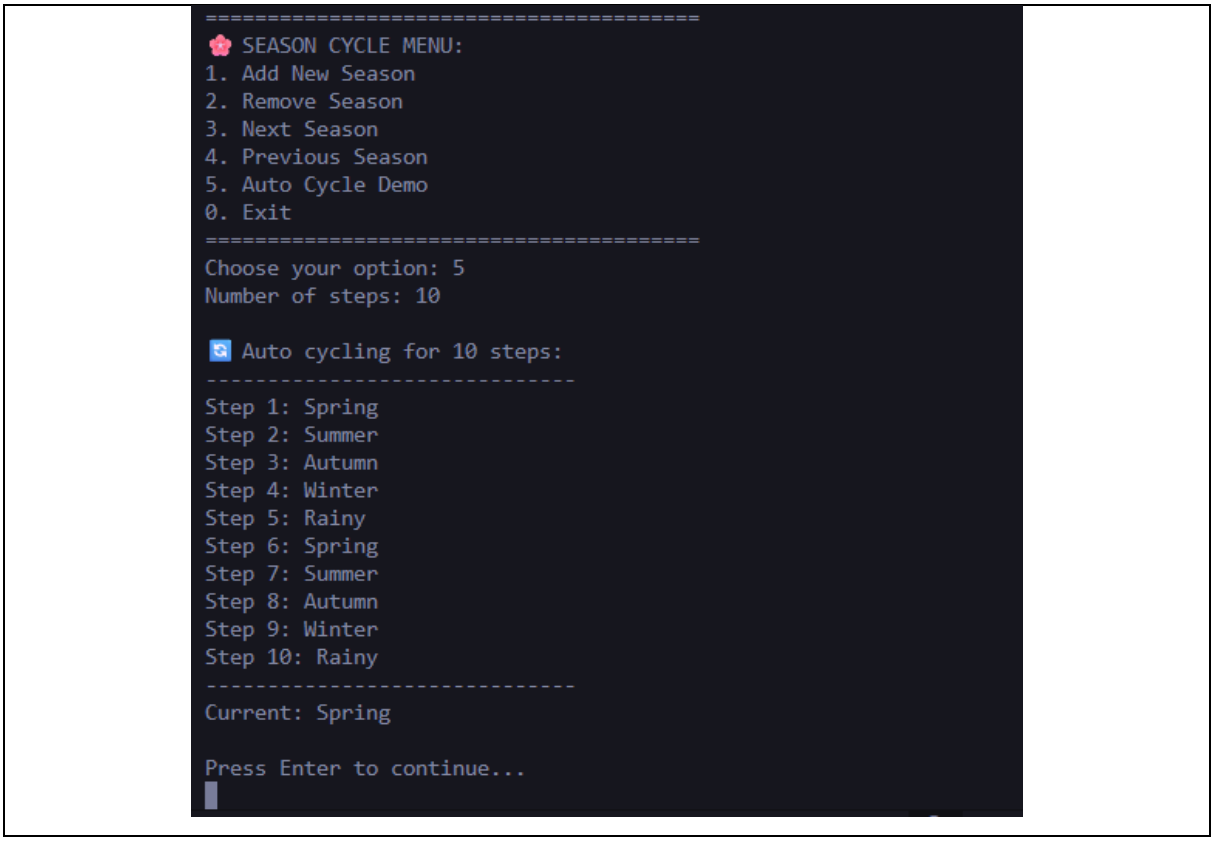


In this image, when the user wants to remove data from the list, they only need to search by entering the title (the program handles both uppercase and lowercase cases, so the input format does not matter). The remove feature is almost the same as the **Search Title** menu, which is why I did not include it separately in the program output. The way it works is by traversing all the data based on the user's input, if it is not found in the first Node, it continues to the next Node, and so on, until a matching title is found. The remove algorithm itself works by shifting the next pointer to the Node after the one being removed, making the data inaccessible or in other words, deleted from the list.

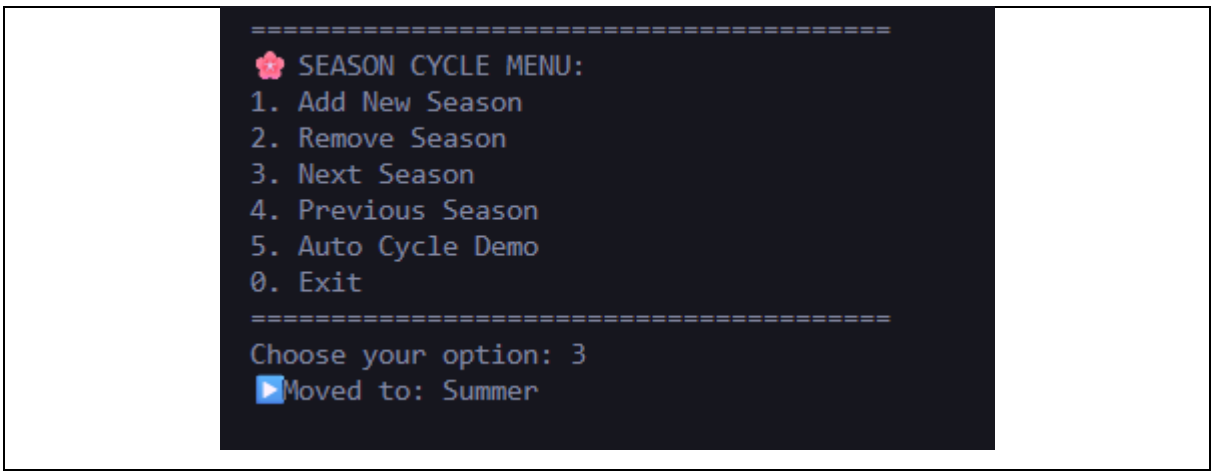
SeasonCycle Program

This is the initial display of the program, which shows several options for the user. The main menu of this program is **Auto Cycle Demo**, which is used to visualize the Circular Linked List.

The image above is a demonstration of adding new data in the program. The user is asked to provide several inputs, which are then processed by the program to modify the structure of the linked list.

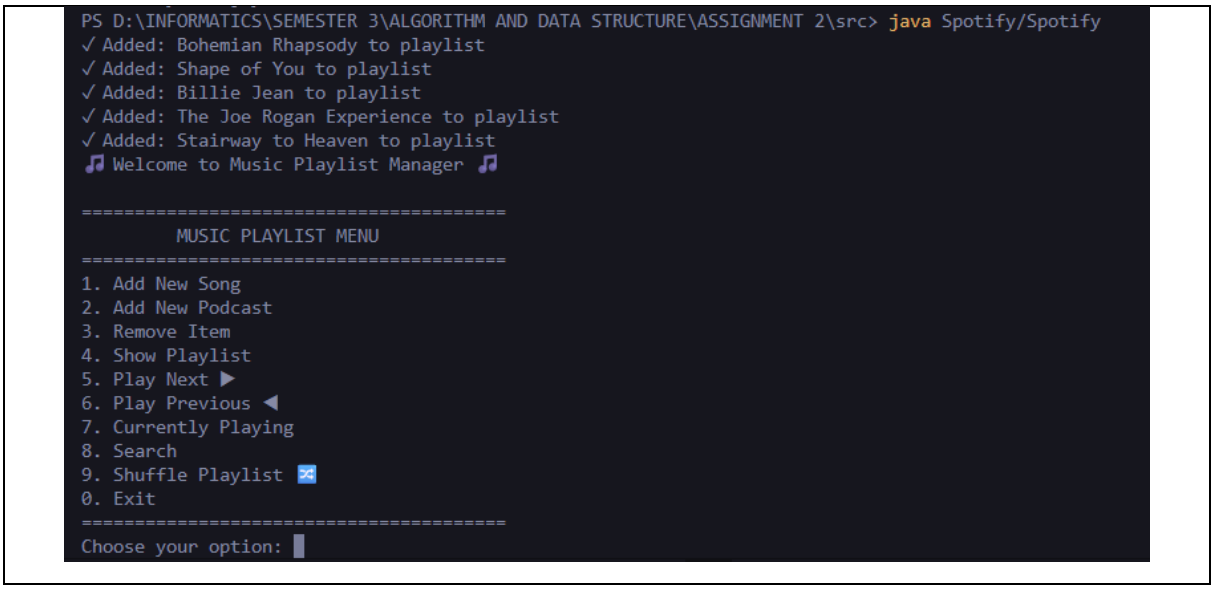


This is the visualization of the circular linked list, where the data loops back to the head. In this program, the user is asked to enter a step value, which represents how many loops will be used to display the data. Since there are 5 data items, on the 6th step the linked list will display the first data again (the head).

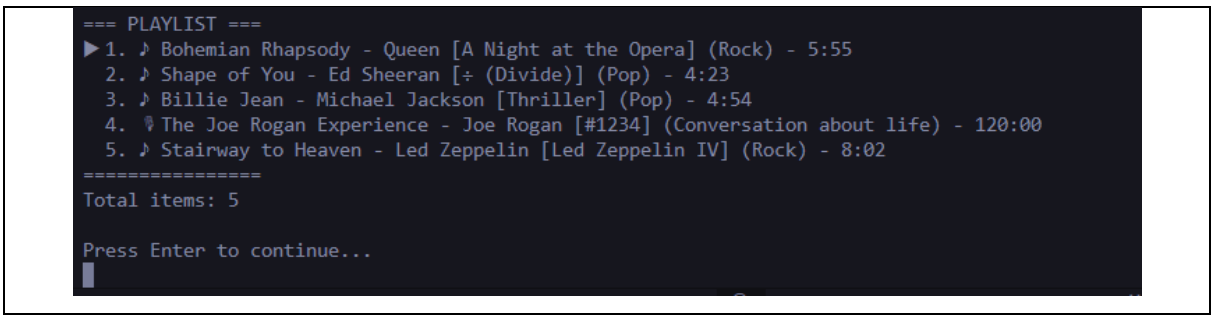


This is the usage of the next function. The next season shifts the data to the following Node, where the default data is Spring, then moves to Summer, and will shift to Autumn if pressed again.

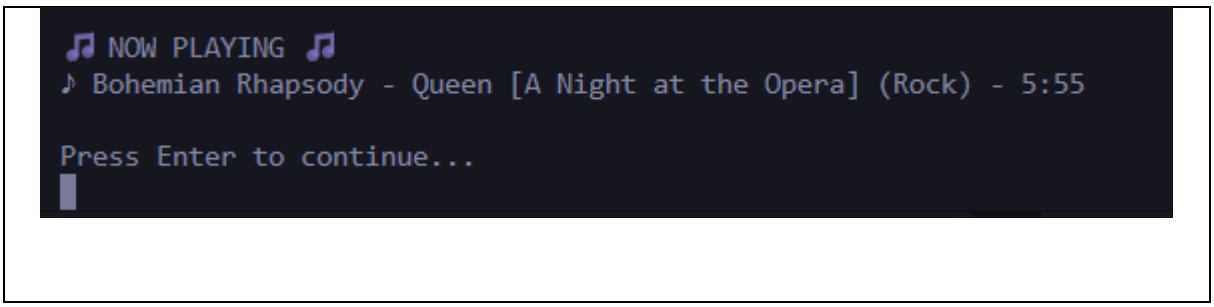
Spotify Program

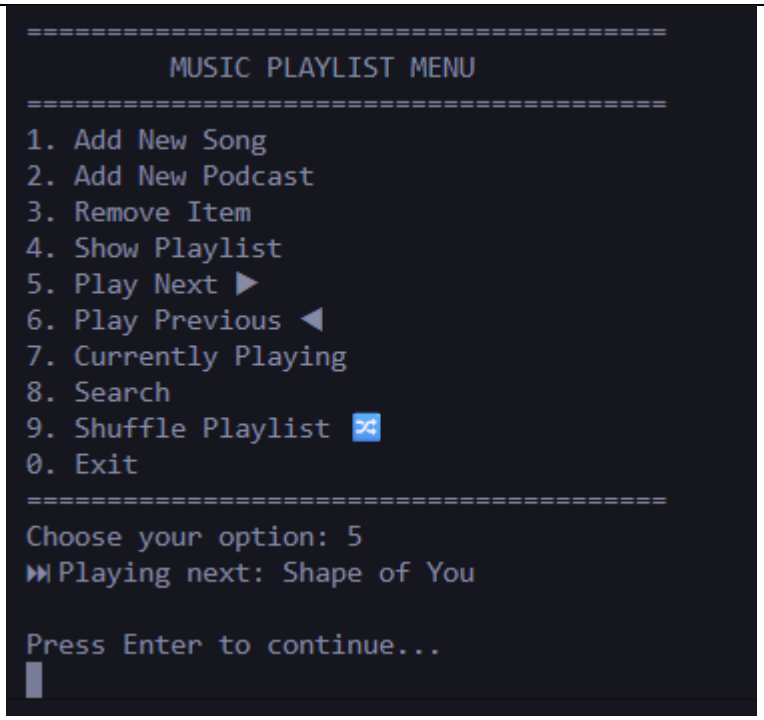


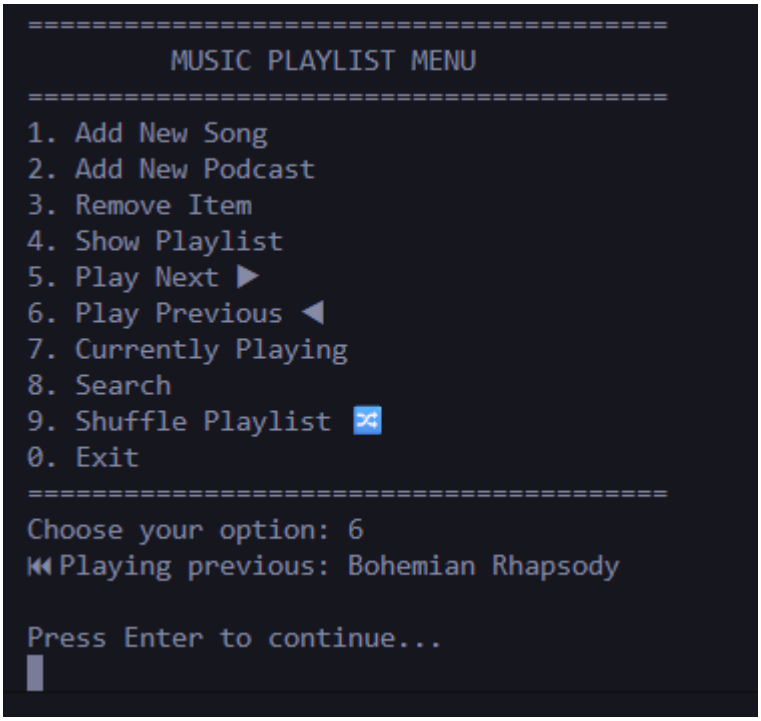
Just like the previous programs, this is the initial display of this program. There are several options such as **add**, **remove**, **show playlist**, **play next** and **previous** (these are the main features of this doubly linked list), **currently playing**, **search song**, and **shuffle**.



This is the display when the user chooses to show all the songs in the list (the song data in the list). The data was initialized beforehand in the program to make the demonstration easier.







The following is the display of three menus at once: **currently playing**, **next song**, and **previous song**. As the names suggest, this feature shows which song is currently playing, while the next and previous options are used to move the pointer to the Node containing the respective data, making it appear as if the song is moving from one Node to another.