MUSIC PLAYLIST

A report submitted in partial fulfillment of the requirements for the Award of Degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

PATHAPATI LAKSHMI MANASA

Regd.No:21B91A05N2

Henotic Technology Pvt Ltd, Hyderabad

(Duration: 5th July 2023 to 5th September 2023)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

S.R.K.R. ENGINEERING COLLEGE

(Autonomous)

SRKR MARG, CHINNA AMIRAM, BHIMAVARAM-534204, A.P

(Recognized by A.I.C.T.E New Delhi) (Accredited by NBA & NAAC)

(Affiliated to JNTU, KAKINADA)

SAGI RAMA KRISHNAM RAJU ENGINEERING COLLEGE
(Autonomous)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Summer Internship Report titled “Music Playlist” is the bonafide work done by MS. PATHAPATI LAKSHMI MANASA 21B91A05N2 at the end of second year second semester at **Henotic Technology Pvt Ltd, Hyderabad** from 5th July 2023 to 5th September 2023 in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering.

Department Internship coordinator **Dean -T & P Cell** **Head of the Department**

INTRODUCTION:

Data Structures and Algorithms in Java:

Data Structures and Algorithms (DSA) are fundamental concepts in computer science and software development. They form the building blocks of efficient and optimized software solutions. In the Java programming language, DSA plays a vital role in solving complex computational problems, managing data effectively, and improving the performance of applications.

Data Structures in Java

Java offers a rich set of data structures, which are essential for storing, organizing, and manipulating data efficiently. Some common data structures in Java include:

1)Arrays:

A basic, fixed-size data structure that stores elements of the same data type in contiguous memory locations.

2)ArrayList:

A dynamic array implementation that can grow or shrink as needed, offering more flexibility than traditional arrays.

3)LinkedList:

A data structure where elements are connected through nodes, making it efficient for insertion and deletion operations.

4)Stack and Queue:

Data structures for implementing Last-In-First-Out (LIFO) and First-In-First-Out (FIFO) behavior, respectively.

5)Tree and Graph:

Hierarchical data structures used to represent various relationships between data elements.

6)HashMap and HashSet:

Implementations of hash tables for efficient key-value storage and unique element storage, respectively.

Algorithms in Java

Algorithms are step-by-step procedures for solving specific problems or performing tasks. Java provides a platform for implementing a wide range of algorithms. Common categories of algorithms in Java include:

1)Sorting Algorithms:

Java offers built-in sorting algorithms such as QuickSort and MergeSort for arranging data in specific orders.

2)Searching Algorithms:

Algorithms like Binary Search help find elements in sorted data structures efficiently.

3)Graph Algorithms:

Java supports graph algorithms like Breadth-First Search (BFS) and Depth-First Search (DFS) for traversing graph structures.

4)Dynamic Programming:

This approach breaks down problems into smaller subproblems and caches solutions to avoid redundant calculations.

5)Greedy Algorithms:

These algorithms make locally optimal choices at each step, aiming for a global optimum.

6)Backtracking:

Algorithms that explore all possible solutions and backtrack when a solution is not found.

In Java, DSA are integral to many application areas, including search engines, machine learning, database management, game development, financial modeling, and cybersecurity.

Understanding DSA in Java is crucial for building efficient, scalable, and reliable software solutions, and it is a valuable skill for software developers and computer scientists alike.

IMPORTANCE OF DSA

1)Optimization:

DSA enables the development of optimized solutions to various problems by efficiently managing data and operations.

2)Scalability:

DSA allows for the scaling of applications and systems, making them adaptable to handle larger datasets.

3)Performance:

Well-designed DSA implementations in Java can significantly enhance application performance.

TYPES OF DSA APPLICATIONS

1)Search Engines:

DSA are fundamental for search engines like Google to quickly retrieve and rank relevant search results.

2)Machine Learning:

DSA play a crucial role in machine learning algorithms, enabling efficient data manipulation and model training.

3)Database Management:

Data structures like B-trees and hashing are used in database systems for efficient data retrieval.

4)Game Development:

DSA are utilized in game engines for rendering, physics simulations, and pathfinding.

5)Financial Modeling:

Complex financial models depend on efficient algorithms for risk assessment and portfolio optimization.

6)Cybersecurity:

Cryptographic algorithms are used to secure data and communications.

MAIN PAGE:

1)Data Structure:

Data structure like an LinkedList to store the list of songs in the playlist.

2)Algorithms:

There is a use of implementing various algorithms for searching, sorting, and iterating through the song list.

Menu Page for Adding Songs:

1)Data Structure:

When adding a new song to the playlist, we create a Song object that contains details like the song's title, author, and runtime. We can add this Song object to your playlist data structure.

2)Algorithms:

Implement an algorithm to add songs to the playlist, which usually involves inserting the song object at a specific position in your data structure.

Deleting Songs:

1)Data Structure:

To delete a song, we'll need a way to identify and locate the song to be removed, such as its title or unique identifier.

2)Algorithms:

Implement an algorithm to search for the song in the playlist and remove it from your data structure.

Play Next Song and Play Previous Song:

1)Data Structure:

Keep track of the current playing song and its position in the playlist.

2)Algorithms:

Implement algorithms to find the next and previous songs in the playlist, given the current playing song's position.

Updating Details of Playlist, Author, Runtime, etc.:

1)Data Structure:

We'll need a way to access and modify the details of a song, so make sure your Song object allows for updates.

2)Algorithms:

Implement algorithms for updating the song details, such as author or runtime, and make sure the changes are reflected in your data structure.

CODE:

MUSIC PLAYER:

```
package Project;


import java.util.ArrayList;

import java.util.List; import

java.util.Scanner;



public class MusicPlayer { private

List<Song> playlist; private

List<Song> favoritePlaylist; private

int currentSongIndex;


 public MusicPlayer() { this.playlist =

new ArrayList<>(); this.favoritePlaylist =

new ArrayList<>();

this.currentSongIndex = -1;

 }


 public void addSong(String name, String singer, int length, String type) {

Song song = new Song(name, singer, length, type); playlist.add(song);

 System.out.println("Added " + song.toString() + " to the playlist.");   }
```

```
public void deleteSong(String songName) {
for (Song song : playlist) { if
(song.getName().equals(songName)) {
playlist.remove(song);
favoritePlaylist.remove(song);
System.out.println("Deleted " + song.toString() + " from the playlist."); return;
}
}
System.out.println("'" + songName + "' not found in the playlist."); }

public void updateSong(String oldSongName, String newName, String newSinger, int
newLength, String newType) {
for (Song song : playlist) { if
(song.getName().equals(oldSongName)) {
song.setName(newName);
song.setSinger(newSinger);
song.setLength(newLength);
song.setType(newType);
System.out.println("Updated '" + oldSongName +
"' to " + song.toString() + " in the playlist.");
return;
}
}
System.out.println("'" + oldSongName + "' not found in the playlist."); }

public void addToFavoritePlaylist(String songName) {
```

```
for (Song song : playlist) { if
(song.getName().equals(songName)) {
favoritePlaylist.add(song);
 System.out.println("'" + songName + "' added to the favorite playlist.");   return;
 }
 }
 System.out.println("'" + songName + "' not found in the playlist.");   }

 public void removeFromFavoritePlaylist(String songName) {
for (Song song : favoritePlaylist) { if
(song.getName().equals(songName)) {
favoritePlaylist.remove(song);
 System.out.println("'" + songName + "' removed from the favorite playlist.");   return;
 }
 }
 System.out.println("'" + songName + "' not found in the favorite playlist.");   }
 public String getNextSong() {
if (currentSongIndex == -1) {
currentSongIndex = 0;
 } else {
 currentSongIndex = (currentSongIndex + 1) % playlist.size();   }
 return playlist.get(currentSongIndex).getName();
 }

 public String getPreviousSong() {
if (currentSongIndex == -1) {
```

```
return null;
 }
 currentSongIndex = (currentSongIndex - 1 + playlist.size()) % playlist.size();
return playlist.get(currentSongIndex).getName();
 }

 public void displayPlaylist(List<Song> songs, String playlistName) {
System.out.println("Current " + playlistName + ":"); for (int i =  0; i
< songs.size(); i++) {
 System.out.println((i + 1) + ". " + songs.get(i).toString());   }
 }

 public static void main(String[] args) {
 MusicPlayer player = new MusicPlayer();
 Scanner scanner = new Scanner(System.in);
while (true) {
 System.out.println("\n--- Music Player Menu ---");
 System.out.println("1. Add Song");
 System.out.println("2. Delete Song");
 System.out.println("3. Update Song");
 System.out.println("4. Add to Favorite Playlist");
 System.out.println("5. Remove from Favorite Playlist");
 System.out.println("6. Play Next Song");
 System.out.println("7. Play Previous Song");
 System.out.println("8. Display Playlist");
 System.out.println("9. Display Favorite Playlist");
 System.out.println("10. Exit");
```

```
System.out.print("Enter your choice (1-10): "); int choice = scanner.nextInt();
scanner.nextLine(); // Consume the newline character after reading the integer input.

switch (choice) {
case 1:
System.out.print("Enter the song name: ");
String name = scanner.nextLine();
System.out.print("Enter the song singer: ");
String singer = scanner.nextLine();
System.out.print("Enter the song length (in seconds): "); int length = scanner.nextInt();
scanner.nextLine(); // Consume the newline character after reading the integer input.
System.out.print("Enter the song type: ");
String type = scanner.nextLine();
player.addSong(name, singer, length, type);
break;
case 2:
System.out.print("Enter the song name to delete: ");
String songToDelete = scanner.nextLine();
player.deleteSong(songToDelete);
break;
case 3:
System.out.print("Enter the song name to update: ");
String oldSongName = scanner.nextLine();
System.out.print("Enter the new song name: ");
String newSongName = scanner.nextLine();
```

```
System.out.print("Enter the new song author: ");
String newSinger = scanner.nextLine();
System.out.print("Enter the new song length (in seconds): "); int newLength = scanner.nextInt(); scanner.nextLine(); // Consume the newline character after reading the integer input.
System.out.print("Enter the new song type: ");
String newType = scanner.nextLine();
player.updateSong(oldSongName, newSongName, newSinger, newLength, newType);
break;
case 4:
System.out.print("Enter the song name to add to favorite playlist: ");
String songToAddToFavorites = scanner.nextLine();
player.addToFavoritePlaylist(songToAddToFavorites);
break;
case 5:
System.out.print("Enter the song name to remove from favorite playlist: "); String songToRemoveFromFavorites = scanner.nextLine();
player.removeFromFavoritePlaylist(songToRemoveFromFavorites);
break;
case 6:
String nextSong = player.getNextSong();
System.out.println("Playing Next Song: " + nextSong); break;
case 7:
String prevSong = player.getPreviousSong();
if (prevSong != null) {
```

```
 System.out.println("Playing Previous Song: " + prevSong);   } else {

 System.out.println("No previous song. Playing the first song.");   }

break;

case 8:

 player.displayPlaylist(player.playlist, "Playlist");

break; case 9:

 player.displayPlaylist(player.favoritePlaylist, "Favorite Playlist");   break;

case 10:
 System.out.println("Goodbye!"); scanner.close(); //

 Close the scanner before exiting the program.

 return;

default:

 System.out.println("Invalid choice. Please enter a valid option (1-10).");   }

 }

 }

}


}
```

SONG:

```
package Project;

class Song { private

String name; private

String singer; private

int length; private

String type;


 public Song(String name, String singer, int length, String type) {

this.name = name; this.singer= singer; this.length =

length; this.type = type;

 }


 public String getName() {

return name;

 }
 public void setName(String name) {

this.name = name;

 }


 public String getSinger() {

return singer;

 }


 public void setSinger(String singer) {

this.singer = singer;
```

```
}

public int getLength() {
return length;
}

public void setLength(int length) {
this.length = length;
}

public String getType() {
return type;

}

public void setType(String type) {
this.type = type;
}

@Override public String toString() { return "'" + name + "' by " + singer + " |
Length: " + length + " seconds | Type: " + type;
}
}
```

OUTPUT:

--- Music Player Menu ---

1. Add Song

2. Delete Song

3. Update Song

4. Add to Favorite Playlist

5. Remove from Favorite Playlist

6. Play Next Song

7. Play Previous Song

8. Display Playlist

9. Display Favorite Playlist

10. Exit

Enter your choice (1-10): 1

Enter the song name: Pournami

Enter the song singer: Chitra
Enter the song length (in seconds): 677

Enter the song type: melody

Added 'Pournami' by Chitra | Length: 677 seconds | Type: melody to the playlist.

--- Music Player Menu ---

1. Add Song

2. Delete Song

3. Update Song

4. Add to Favorite Playlist

5. Remove from Favorite Playlist

6. Play Next Song

7. Play Previous Song

8. Display Playlist

9. Display Favorite Playlist 10. Exit

Enter your choice (1-10): 1

Enter the song name: Bulleya

Enter the song singer: Arjit singh

Enter the song length (in seconds): 256

Enter the song type: melody

Added 'Bulleya' by Arjit singh | Length: 256 seconds | Type: melody to the playlist. --- Music Player Menu ---

1. Add Song

2. Delete Song

3. Update Song

4. Add to Favorite Playlist

5. Remove from Favorite Playlist

6. Play Next Song

7. Play Previous Song
8. Display Playlist

9. Display Favorite Playlist

10. Exit

Enter your choice (1-10): 2

Enter the song name to delete: Bulleya

Deleted 'Bulleya' by Arjit singh | Length: 256 seconds | Type: melody from the playlist.

--- Music Player Menu ---

1. Add Song

2. Delete Song

3. Update Song

4. Add to Favorite Playlist

5. Remove from Favorite Playlist

6. Play Next Song

7. Play Previous Song

8. Display Playlist

9. Display Favorite Playlist

10. Exit

Enter your choice (1-10): 4

Enter the song name to add to favorite playlist: Pournami

'Pournami' added to the favorite playlist.

--- Music Player Menu ---

1. Add Song

2. Delete Song

3. Update Song

4. Add to Favorite Playlist
5. Remove from Favorite Playlist

6. Play Next Song

7. Play Previous Song

8. Display Playlist

9. Display Favorite Playlist 10. Exit

Enter your choice (1-10): 5

Enter the song name to remove from favorite playlist: Pournami

'Pournami' removed from the favorite playlist.

--- Music Player Menu ---

1. Add Song

2. Delete Song

3. Update Song

4. Add to Favorite Playlist

5. Remove from Favorite Playlist

6. Play Next Song

7. Play Previous Song

8. Display Playlist

9. Display Favorite Playlist 10. Exit

Enter your choice (1-10): 6

Playing Next Song: Pournami

--- Music Player Menu ---

1. Add Song

2. Delete Song

3. Update Song

4. Add to Favorite Playlist
5. Remove from Favorite Playlist

6. Play Next Song

7. Play Previous Song

8. Display Playlist

9. Display Favorite Playlist 10. Exit

Enter your choice (1-10): 7

Playing Previous Song: Pournami

--- Music Player Menu ---

1. Add Song

2. Delete Song

3. Update Song

4. Add to Favorite Playlist

5. Remove from Favorite Playlist

6. Play Next Song

7. Play Previous Song

8. Display Playlist

9. Display Favorite Playlist 10. Exit

Enter your choice (1-10): 8

Current Playlist:

1. 'Pournami' by Chitra | Length: 677 seconds | Type: melody

--- Music Player Menu ---

1. Add Song

2. Delete Song

3. Update Song

4. Add to Favorite Playlist
5. Remove from Favorite Playlist

6. Play Next Song

7. Play Previous Song

8. Display Playlist

9. Display Favorite Playlist 10. Exit

Enter your choice (1-10): 9

Current Favorite Playlist:

--- Music Player Menu ---

1. Add Song

2. Delete Song

3. Update Song

4. Add to Favorite Playlist

5. Remove from Favorite Playlist

6. Play Next Song

7. Play Previous Song

8. Display Playlist

9. Display Favorite Playlist 10. Exit

Enter your choice (1-10): 10

Goodbye!

CONCLUSION:

Data Structures and Algorithms in Java are essential for developing efficient and scalable software solutions. Understanding the concepts, choosing the right data structures, and implementing algorithms effectively can significantly impact the performance and reliability of software systems. A strong foundation in DSA and proficiency in Java can open doors to a wide range of applications in the field of computer science and software development.