BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

Work Integrated Learning Programmes Division

Data Structures and Algorithms

Lab Sheet -1 Topic: ADT (Song Playlist ADT)

General Instructions for Programming :

1. All inputs to the program must be command line arguments.

2. Use standard Java coding conventions. Each class: Node, Linked List and Main code should be written in separate .java files.

Problem Statement:

Implement a **Song Playlist ADT** using doubly linked list.
Each node in the list should contain the song title and duration of the song

You will have to implement the following operations using the concepts of doubly linked lists

1. Add a song to the playlist
   - Adding from beginning or end of the DLL is okay
2. Delete a song from the playlist
   - Delete a song based on a position in the linked list which is input through a prompt
   - Ensure the input position is a valid one
3. Find a song by name
   - Prompt for the song name. Exact name matching is fine. Partial match is not required.
4. Next track / Previous track
   - First prompt for a starting position and then traverse forward or backward based on user input. This should be done through a sub menu.
   - Ensure relevant messages are displayed when the beginning and end of the playlist is reached.
5. Sort playlist by song title
   - Use one of the standard sorting algorithms
6. Display playlist
   - Display all songs in the playlist

Also ensure to include a proper exit sequence from the program.

Following are the operations to be performed on the **Song Playlist ADT**.

1) public void addSong(String t, String dur)
Precond: t and dur are the title and duration as input in the command prompt
Effect : a new node is created and added to the beginning of the doubly linked list. The title t and duration d is applied to the new node. If the list is empty, a new node is created as the starting node of the list.

2) public int findSong(String t)
Precond : Playlist is already populated with multiple nodes using addSong()
Effect : the program finds the exact match of the song t and returns the track number / position of the song in the playlist.

3) public void deleteAtPos(int pos)
Precond: Playlist is already populated with multiple nodes using addSong()
Effect: Deletes the song at position pos input in the command prompt. Ensure the position is a valid one.

4) public String getSongAtPos(int pos)
Precond : Playlist is already populated with multiple nodes using addSong()
Effect: Returns the song title at the given position. This is used to traverse the playlist (next track / previous track)

5) public void sortList()
Precond : Playlist is already populated with multiple nodes using addSong()
Effect : Sorts the songs in the playlist in alphabetical order based on the song title

6) public void display()
Precond: none
Effect : Displays the position, song title and duration of all the songs in the playlist.

Input:
The input to the program should be provided real time through the command prompt.
The following input considerations need to be made:

1. Each of the operations should be driven by a menu and sub menu option.
2. Two separate prompts can be used to take in the song title and duration.

Sample Input for Playlist:

```
1. Believer <3:23>
2. Shotgun <3:21>
3. Titanium <4:07>
4. Kiki do you love me <3:39>
5. Shape of you <3:52>
```

Output:
Depending on the operation selected display the relevant details like:

1. Entire playlist and total number of songs in the playlist
2. Current track in the case of next / previous track.
3. Track name that is being deleted
4. Track position in the case of find track
5. Sorted playlist

Sample Output:

1. Addition of Songs

```
          Playlist Test
```

```
Playlist Operations:

1. Add a song to the playlist
2. Delete a song from the playlist
3. Find a song by name
4. Next track / Previous track
5. Sort playlist by song title
6. Display playlist
7. Exit

Enter Menu Option:
1

Enter Song Title:
Shape of you

Enter Song Duration:
3:52

Added the song 'Shape of you' to the playlist
```

2. Display playlist

```
Playlist Operations:

1. Add a song to the playlist
2. Delete a song from the playlist
3. Find a song by name
4. Next track / Previous track
5. Sort playlist by song title
6. Display playlist
7. Exit

Enter Menu Option:
6

Playlist currently contains 5 song(s)
1. Believer <3:23>
2. Shotgun <3:21>
3. Titanium <4:07>
4. Kiki do you love me <3:39>
5. Shape of you <3:52>
```

3. Find a song by name

```
Playlist Operations:

1. Add a song to the playlist
2. Delete a song from the playlist
3. Find a song by name
4. Next track / Previous track
5. Sort playlist by song title
6. Display playlist
```

```
7. Exit

Enter Menu Option:
3
Enter the Name of the Song:
Titanium
The Song 'Titanium' is present at position 3
```

4. Next track / previous track

```
Playlist Operations:

1. Add a song to the playlist
2. Delete a song from the playlist
3. Find a song by name
4. Next track / Previous track
5. Sort playlist by song title
6. Display playlist
7. Exit

Enter Menu Option:
4
Enter a track number to start from:
2
Current Song is 2. Shotgun
Enter n for Next Track, p for Previous Track, e to exit:

p
Current Song is 1. Believer
```

Deliverables:

1. node.java – should contain the basic node definition and basic get and set functions pertaining to the node.
2. LinkedList.java – should contain the doubly linked list functions and all functions for the operations mentioned in the problem statement.
3. SongPlaylist.java – should contain the main function call and the input / output requirements and menu switching options.

Required Text Book Reading:

1) Section 2.2 of Algorithm Design by Michael Goodrich and Roberto Tamassia