LAB # 08

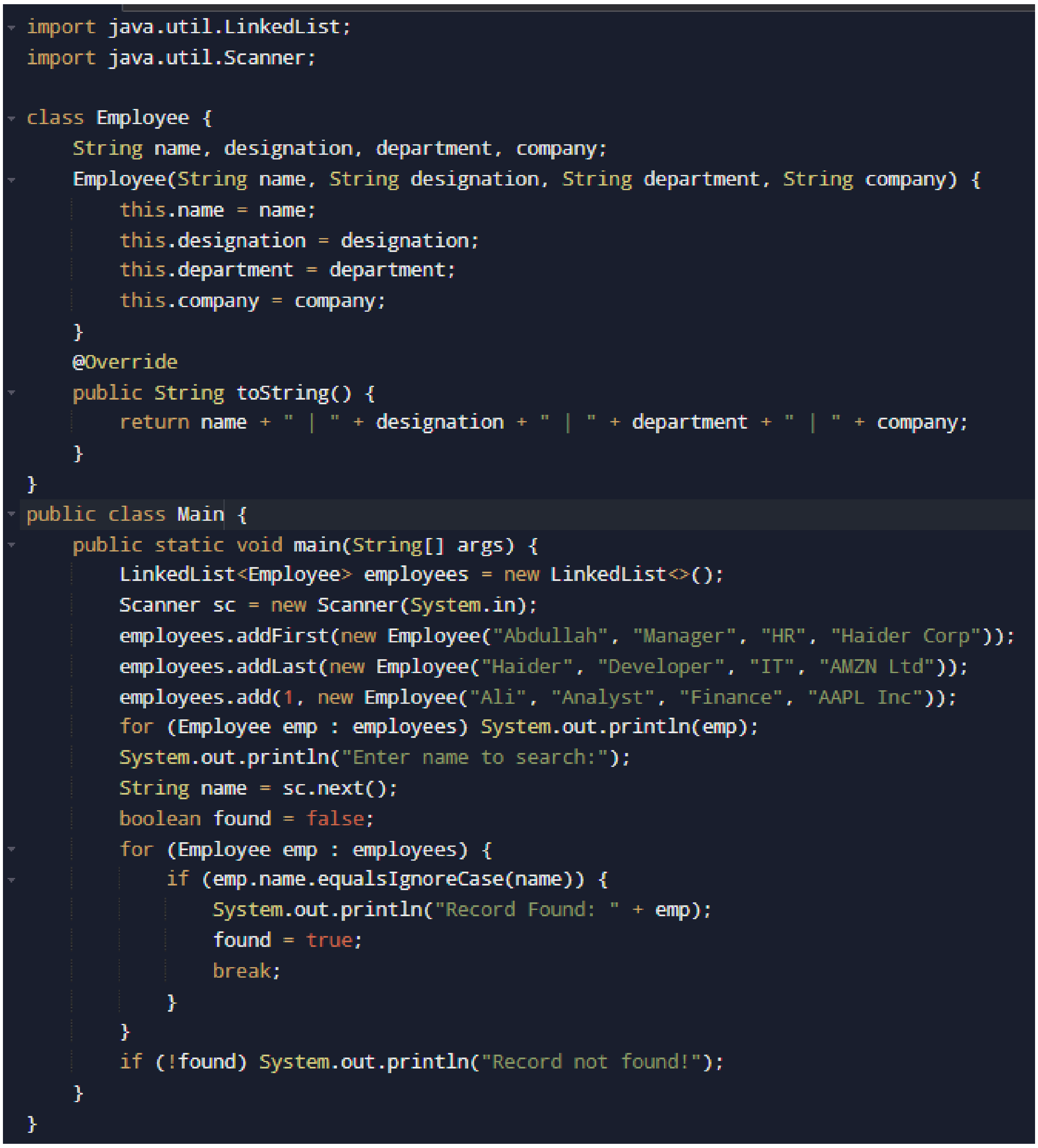
Singly Linked List Implementation And Doubly Linked List implementation of the list ADT

OBJECTIVE: Implementing singly linked list, associated operations and Runner technique and Implementing doubly linked list, associated operations and LRU technique.

LAB TASKS

1. Write a program that can insert the records of employees in a link list. The record includes employee's name, designation, department and company name. The program should be able to insert the record as first, last and as middle node in the list and search any record.

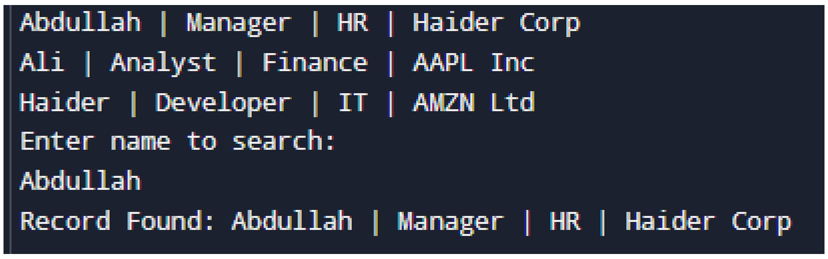
Code:



```java
import java.util.LinkedList;
import java.util.Scanner;

class Employee {
    String name, designation, department, company;
    Employee(String name, String designation, String department, String company) {
        this.name = name;
        this.designation = designation;
        this.department = department;
        this.company = company;
    }
    @Override
    public String toString() {
        return name + " | " + designation + " | " + department + " | " + company;
    }
}
public class Main {
    public static void main(String[] args) {
        LinkedList<Employee> employees = new LinkedList<>();
        Scanner sc = new Scanner(System.in);
        employees.addFirst(new Employee("Abdullah", "Manager", "HR", "Haider Corp"));
        employees.addLast(new Employee("Haider", "Developer", "IT", "AMZN Ltd"));
        employees.add(1, new Employee("Ali", "Analyst", "Finance", "AAPL Inc"));
        for (Employee emp : employees) System.out.println(emp);
        System.out.println("Enter name to search:");
        String name = sc.next();
        boolean found = false;
        for (Employee emp : employees) {
            if (emp.name.equalsIgnoreCase(name)) {
                System.out.println("Record Found: " + emp);
                found = true;
                break;
            }
        }
        if (!found) System.out.println("Record not found!");
    }
}
```

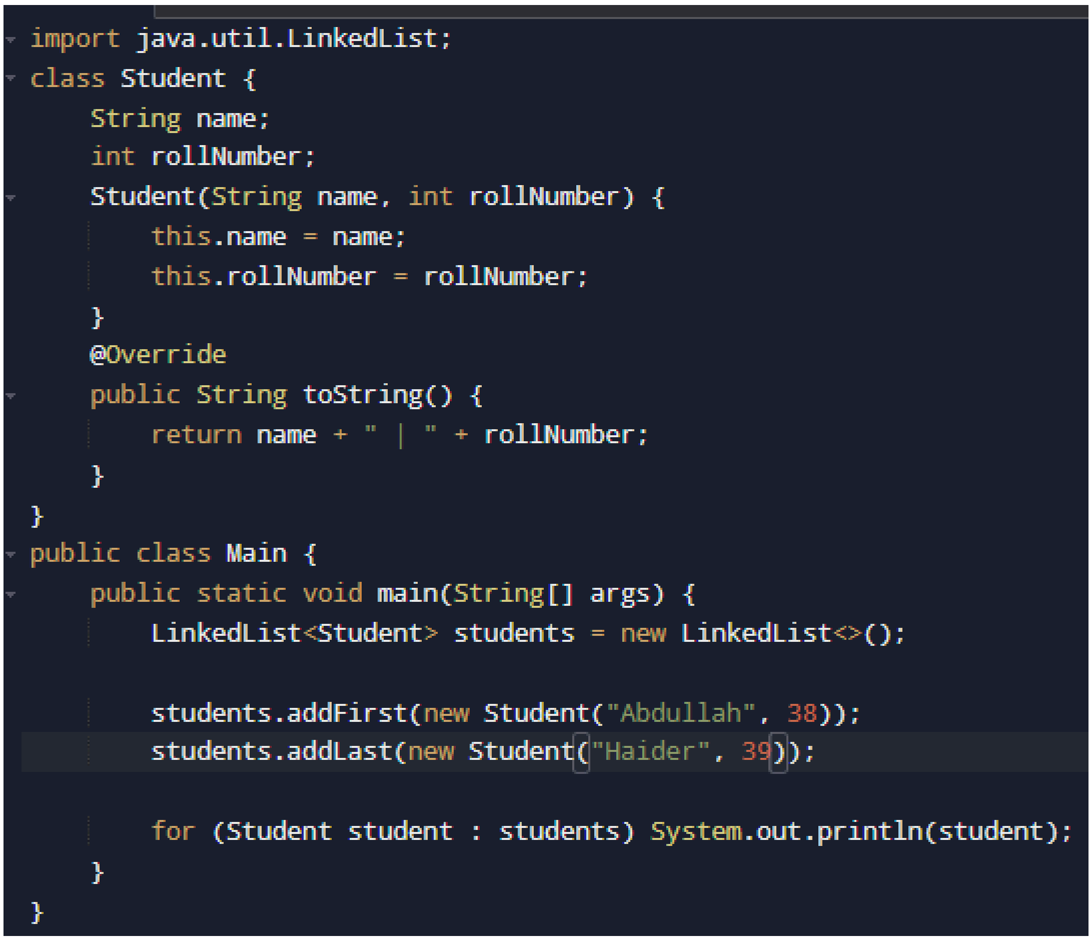
Output:



```
Abdullah | Manager | HR | Haider Corp
Ali | Analyst | Finance | AAPL Inc
Haider | Developer | IT | AMZN Ltd
Enter name to search:
Abdullah
Record Found: Abdullah | Manager | HR | Haider Corp
```

2. Write a program to insert the records of students in a Doubly linked list and insert elements at first and last node using Deque.

Code:

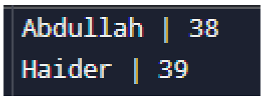


```java
import java.util.LinkedList;
class Student {
    String name;
    int rollNumber;
    Student(String name, int rollNumber) {
        this.name = name;
        this.rollNumber = rollNumber;
    }
    @Override
    public String toString() {
        return name + " | " + rollNumber;
    }
}
public class Main {
    public static void main(String[] args) {
        LinkedList<Student> students = new LinkedList<>();

        students.addFirst(new Student("Abdullah", 38));
        students.addLast(new Student("Haider", 39));

        for (Student student : students) System.out.println(student);
    }
}
```

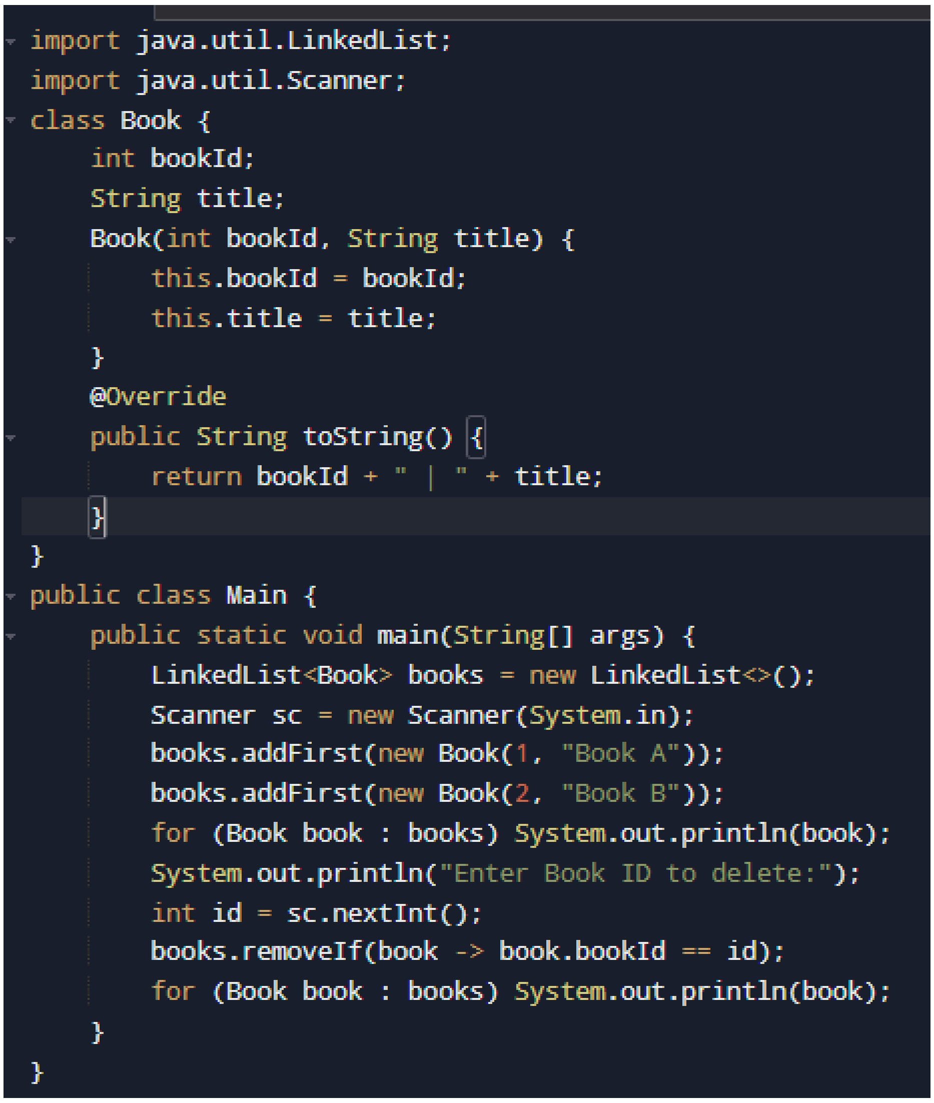
Output:



```
Abdullah | 38
Haider | 39
```

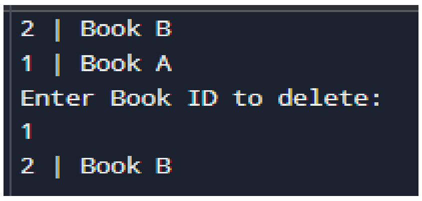
3. You are managing a library system where each book is represented by a node in a doubly linked list. Each node contains:

Code:



```java
import java.util.LinkedList;
import java.util.Scanner;
class Book {
    int bookId;
    String title;
    Book(int bookId, String title) {
        this.bookId = bookId;
        this.title = title;
    }
    @Override
    public String toString() {
        return bookId + " | " + title;
    }
}
public class Main {
    public static void main(String[] args) {
        LinkedList<Book> books = new LinkedList<>();
        Scanner sc = new Scanner(System.in);
        books.addFirst(new Book(1, "Book A"));
        books.addFirst(new Book(2, "Book B"));
        for (Book book : books) System.out.println(book);
        System.out.println("Enter Book ID to delete:");
        int id = sc.nextInt();
        books.removeIf(book -> book.bookId == id);
        for (Book book : books) System.out.println(book);
    }
}
```

Output:

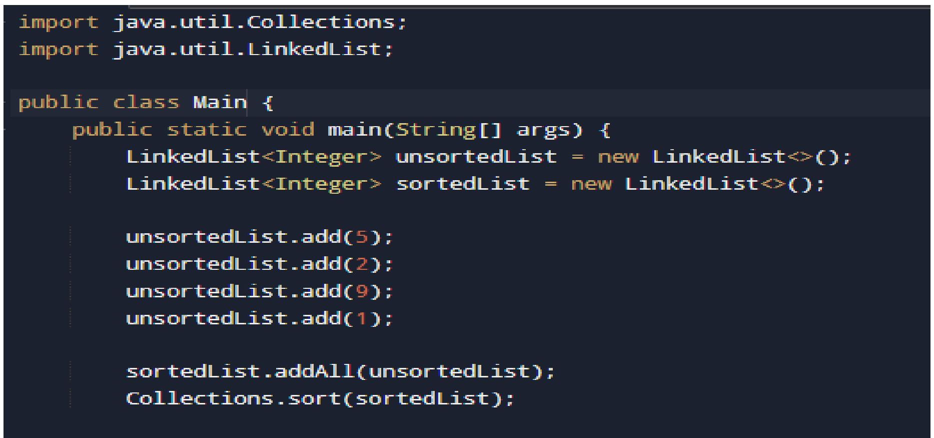


```
2 | Book B
1 | Book A
Enter Book ID to delete:
1
2 | Book B
```

Home Tasks

1. Write a program to create Unsorted LinkedList as well as Sorted LinkedList.

Code:



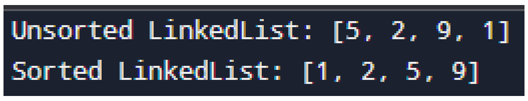
```java
import java.util.Collections;
import java.util.LinkedList;

public class Main {
    public static void main(String[] args) {
        LinkedList<Integer> unsortedList = new LinkedList<>();
        LinkedList<Integer> sortedList = new LinkedList<>();

        unsortedList.add(5);
        unsortedList.add(2);
        unsortedList.add(9);
        unsortedList.add(1);

        sortedList.addAll(unsortedList);
        Collections.sort(sortedList);
```

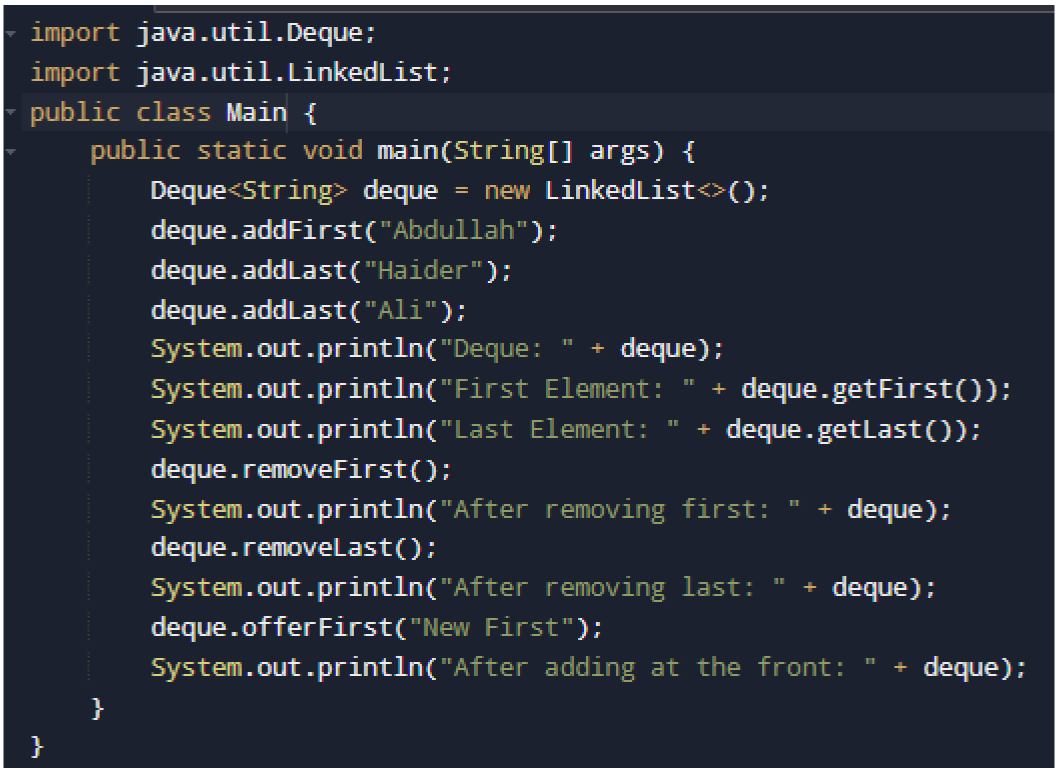
Output:



```
Unsorted LinkedList: [5, 2, 9, 1]
Sorted LinkedList: [1, 2, 5, 9]
```

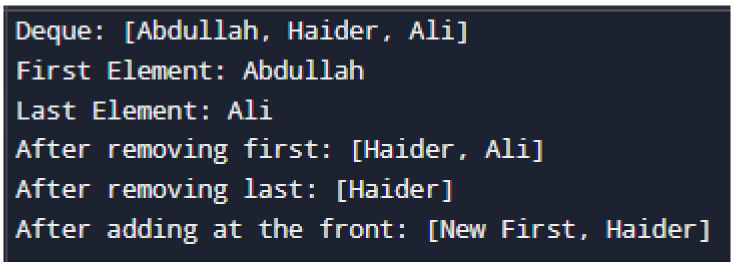
2. Write a program to create LinkedList using Deque and apply any five methods of Deque interface.

   Code:



```java
import java.util.Deque;
import java.util.LinkedList;
public class Main {
    public static void main(String[] args) {
        Deque<String> deque = new LinkedList<>();
        deque.addFirst("Abdullah");
        deque.addLast("Haider");
        deque.addLast("Ali");
        System.out.println("Deque: " + deque);
        System.out.println("First Element: " + deque.getFirst());
        System.out.println("Last Element: " + deque.getLast());
        deque.removeFirst();
        System.out.println("After removing first: " + deque);
        deque.removeLast();
        System.out.println("After removing last: " + deque);
        deque.offerFirst("New First");
        System.out.println("After adding at the front: " + deque);
    }
}
```

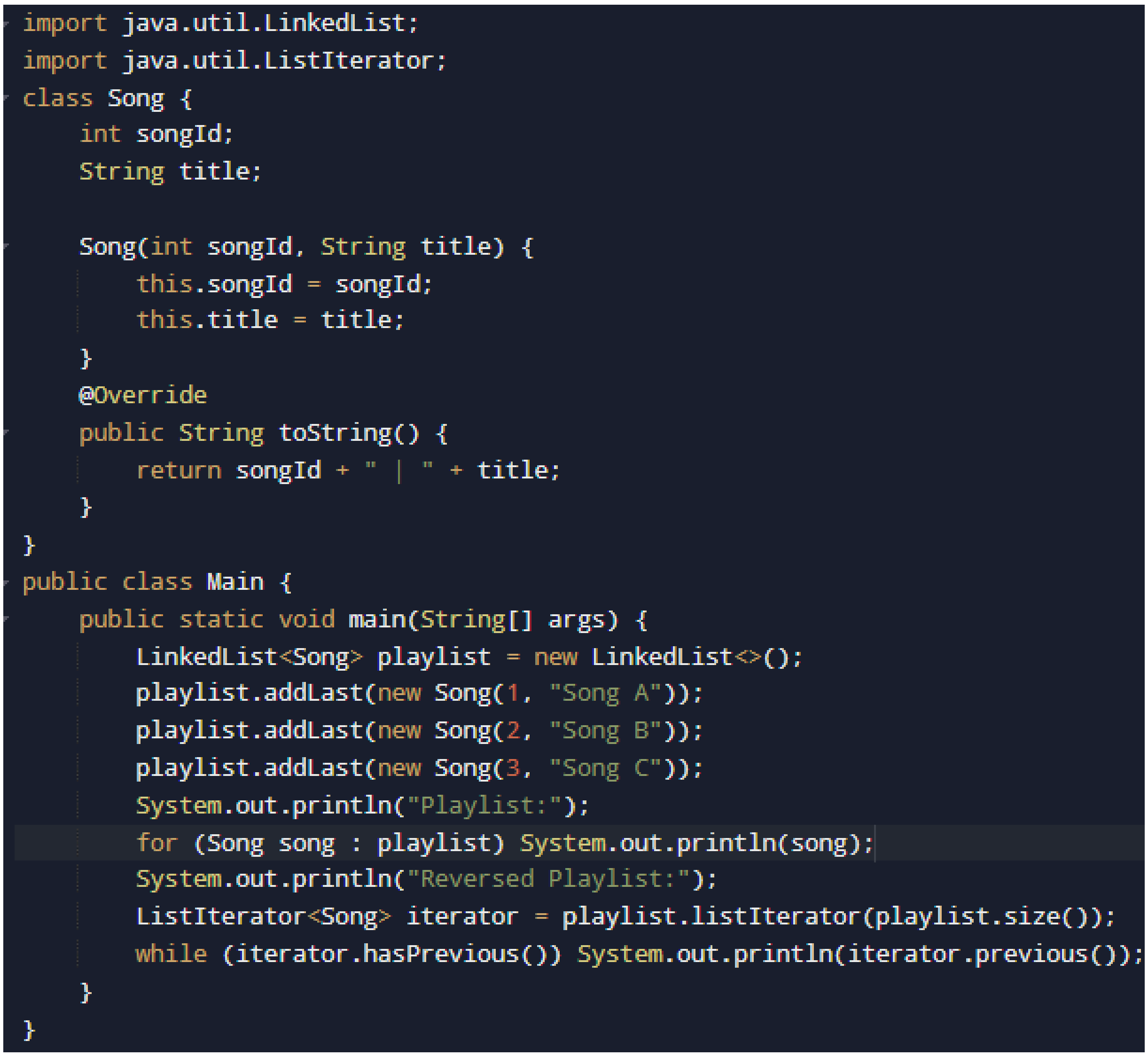
   Output:



```
Deque: [Abdullah, Haider, Ali]
First Element: Abdullah
Last Element: Ali
After removing first: [Haider, Ali]
After removing last: [Haider]
After adding at the front: [New First, Haider]
```

3. You are managing a playlist system where each song is represented by a node in a doubly linked list. Each node contains:

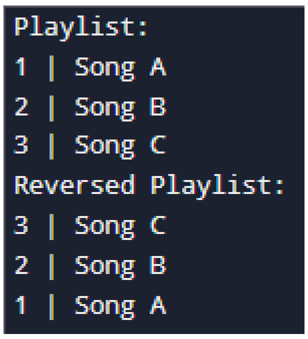
   Code:



```java
import java.util.LinkedList;
import java.util.ListIterator;
class Song {
    int songId;
    String title;

    Song(int songId, String title) {
        this.songId = songId;
        this.title = title;
    }
    @Override
    public String toString() {
        return songId + " | " + title;
    }
}
public class Main {
    public static void main(String[] args) {
        LinkedList<Song> playlist = new LinkedList<>();
        playlist.addLast(new Song(1, "Song A"));
        playlist.addLast(new Song(2, "Song B"));
        playlist.addLast(new Song(3, "Song C"));
        System.out.println("Playlist:");
        for (Song song : playlist) System.out.println(song);
        System.out.println("Reversed Playlist:");
        ListIterator<Song> iterator = playlist.listIterator(playlist.size());
        while (iterator.hasPrevious()) System.out.println(iterator.previous());
    }
}
```

Output:



```
Playlist:
1 | Song A
2 | Song B
3 | Song C
Reversed Playlist:
3 | Song C
2 | Song B
1 | Song A
```