In My console project, the Customer class manages the search for movies/Theaters and made bookings. It violates the Single Responsibility Principle.
I have to Create another class called,
1) SearchOperations which is mainly used for finding movies/heaters and
2) TicketBookings which focuses to book tickets , getTickets

```
package com.booking;

import java.sql.ResultSet;
import java.util.ArrayList;

import com.booking.Exceptions.NoShowAvailableException;

class Customer extends User implements CustomerInterface{
	ArrayList<Book> bookingHistory=new ArrayList<>();
	Customer(ResultSet rs) {
		super(rs);
		new Thread(new CollectHistory(this)).start();
	}
	Customer(){
		super();
	}
	@Override
	public void view() {
		boolean flag=true;
		while(flag) {
			System.out.print("**************************\n 1 - Search theater "
							+ "\n 2 - Search movie \n 3 - Booking History \n
4 - logout\n Enter your choice: ");
			switch(Choice.get()) {
			case 1:
				searchTheater();
				break;
			case 2:
				searchMovie();
```

```
                        break;
                case 3:
                        bookingDetails();
                        break;
                case 4:
                        System.out.println(name+" Log out Successfully...");
                        flag=false;
                        break;
                }
        }
}

private boolean searchMovie(){
        Movie movie=null;
        System.out.print("Enter Movie name to be searched/ 0- to back: ");
        String content=Input.in.nextLine();
        if(content.equals("0"))return false;
        SqlSearch t1=new SqlSearch("Movie","name",content);
        ThreadFunctions.join(t1);
        if(ResultSetOperations.next(t1.rs))movie = new Movie(t1.rs);
        else {
                System.out.println("Not available movie..");
                return false;
        }
        if(movie!=null) {
                System.out.println("\nMovie Details-(you searched)");
                movie.details(1);
                if(movie.theaterList.size()>0) {
                        boolean flag=true;
                        while(flag) {
                                Theater theater=movie.selectTheater();
                                if(theater!=null) {
                                        if(selectShow(theater,movie))flag=false;
                                }else flag=false;
                        }
                }
        }
        return true;
}
private boolean searchTheater() {
        Theater theater=null;
        System.out.print("Enter Theater name to be searched/ 0- to back:");
        String content=Input.in.nextLine();
        if(content.equals("0"))return false;
        SqlSearch t1=new SqlSearch("Theater","name", content);
        ThreadFunctions.join(t1);
        if(ResultSetOperations.next(t1.rs)) {
                theater = new Theater(t1.rs);
        }
        else {
                System.out.println(" Not available...");
                return false;
        }
        if(theater!=null) {
                System.out.println("Theater Details - (you searched)");
                theater.details(1);
                if(theater.movieList.size()>0) {
                        boolean flag=true;
                        while(flag) {
                                Movie movie=theater.selectMovie();
                                if(movie!=null) {
                                        if(selectShow(theater,movie))flag=false;
                                }else flag=false;
                        }

                }
        }
        return true;
}

private boolean selectShow(Theater theater,Movie movie ){
        ArrayList<ShowTime> shows=theater.getShowList(movie);
        ShowTime show=null;
        try {
```

```
				System.out.print("\n##### 0- back\n##### any 'Enter' to select
show: ");
				if(Input.in.nextLine().equals("0"))return false;
				show = new ShowTime(shows);
				System.out.println("\n$$$$$$$$$ Selected show $$$$$$$$$");
				show.details(0);
				bookTickets(movie,theater,show);
				return true;
			} catch (NoShowAvailableException e) {
				System.out.println(e+theater.name);
				return false;
			}
		}

		private void bookTickets(Movie movie,Theater theater,ShowTime show){
			System.out.print("Want to Book Tickets? \n1 - book\n2 - cancel\nEnter
your choice: ");
			switch(Choice.get()) {
			case 1:
				new Book(this,movie,theater,show);
				new Thread(new CollectHistory(this)).start();
				break;
			case 2:
				cancel();
				break;
			default:
				System.out.println("\n Invalid operations...");
				cancel();
				break;
			}
		}

		private void cancel() {
			System.out.println(".......Booking cancelled......");
		}

		@Override
```

```
	public String getUserName() {
		return this.userName;
	}

	private void bookingDetails() {
		System.out.println("\n***** Booking Details*****");
		int i=0;
		for(Book b:bookingHistory) {
			i++;
			b.details(i);
		}
		System.out.println("##################################");
	}
}
```