DINILLA PAULSE

ST10434929

PROGRAMMING 6112 PRACTICAL ASSIGNMENT

GITHUB LINK: https://github.com/ST10434929/Programing6112Practical.git

SECTION A CODE:

TVSeriesApp:

```
package tvseriesapp;

import java.util.Scanner;

public class TVSeriesApp {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    Operations ops = new Operations();

    System.out.println("LATEST SERIES 2025");
    System.out.print("Enter 1 to launch menu or any other key to exit: ");
    String start = scanner.nextLine();

    if (!start.equals("1")) {
      System.out.println("Exiting application...");
      return;
    }

    while (true) {
```

```
// Display menu
System.out.println("\nPlease select one of the following menu items:");
System.out.println("(1) Capture a new series");
System.out.println("(2) Search for a series");
System.out.println("(3) Update series");
System.out.println("(4) Delete a series");
System.out.println("(5) Print series report - 2025");
System.out.println("(6) Exit Application");

System.out.print("Enter choice: ");
String choice = scanner.nextLine();

switch (choice) {
  case "1":
    ops.captureSeries();
    break;
  case "2":
    ops.searchSeries();
    break;
  case "3":
    ops.updateSeries();
    break;
  case "4":
    ops.deleteSeries();
    break;
  case "5":
```

```
            ops.seriesReport2025();
            break;
          case "6":
            System.out.println("Exiting application...");
            return;
          default:
            System.out.println("Invalid option, please try again."); // Response to invalid input
        }

        System.out.print("\nEnter (1) to launch menu or any other key to exit: ");
        String again = scanner.nextLine();
        if (!again.equals("1")) {
          System.out.println("Exiting application...");
          break;
        }
      }

      scanner.close();
    }
}
```

OPERATIONS:

```
package tvseriesapp;


import java.util.ArrayList;
import java.util.Scanner;
```

```
public class Operations {
    private ArrayList<Series> seriesList = new ArrayList<>();
    private Scanner scanner = new Scanner(System.in);

    public ArrayList<Series> getSeriesList() {
        return seriesList;
    }

    // New series
    public void captureSeries() {
        System.out.println("\nCAPTURE A NEW SERIES");

        System.out.print("Enter the series id: ");
        String id = scanner.nextLine();

        System.out.print("Enter the series name: ");
        String name = scanner.nextLine();

        int age;
        while (true) {
            try {
                System.out.print("Enter the series age restriction: ");
                age = Integer.parseInt(scanner.nextLine());

                if (age >= 2 && age <= 18) {
```

```
            break;
          } else {
            System.out.println("You have entered an incorrect series age!");
          }
        } catch (NumberFormatException e) {
          System.out.println("Invalid input. Please enter a number for age.");
        }
      }

      int episodes;
      while (true) {
        try {
          System.out.print("Enter the number of episodes: ");
          episodes = Integer.parseInt(scanner.nextLine());
          break;
        } catch (NumberFormatException e) {
          System.out.println("Invalid input. Please enter a number for episodes.");
        }
      }

      Series s = new Series(id, name, age, episodes);
      seriesList.add(s);
      System.out.println("Series processed successfully!!!");
    }

    // Search for a series by ID
```

```
public void searchSeries() {
    System.out.print("Enter the series id to search: ");
    String id = scanner.nextLine();

    Series found = searchSeriesById(id);
    if (found != null) {
        found.displaySeriesDetails();
    } else {
        System.out.println("Series with Series Id: " + id + " was not found!");
    }
}

public Series searchSeriesById(String id) {
    for (Series s : seriesList) {
        if (s.getSeriesId().equals(id)) {
            return s;
        }
    }
    return null;
}

// Update series details via console
public void updateSeries() {
    System.out.print("Enter the series id to update: ");
    String id = scanner.nextLine();
```

```
        if (!updateSeriesByIdConsole(id)) {
            System.out.println("Series with Series Id: " + id + " was not found!");
        }
    }

    public boolean updateSeriesById(String id, String newName, int newAge, int
newEpisodes) {
        Series s = searchSeriesById(id);
        if (s != null) {
            s.setSeriesName(newName);
            s.setSeriesAge(newAge);
            s.setNumberOfEpisodes(newEpisodes);
            return true;
        }
        return false;
    }

    private boolean updateSeriesByIdConsole(String id) {
        for (Series s : seriesList) {
            if (s.getSeriesId().equals(id)) {
                System.out.print("Enter the new series name: ");
                s.setSeriesName(scanner.nextLine());

                int age;
                while (true) {
                    try {
```

```
                System.out.print("Enter the new age restriction: ");
                age = Integer.parseInt(scanner.nextLine());

                if (age >= 2 && age <= 18) {
                    s.setSeriesAge(age);
                    break;
                } else {
                    System.out.println("You have entered an incorrect series age!!!");
                }
            } catch (NumberFormatException e) {
                System.out.println("Invalid input. Please enter a number for age.");
            }
        }

        System.out.print("Enter the new number of episodes: ");
        s.setNumberOfEpisodes(Integer.parseInt(scanner.nextLine()));

        System.out.println("Series updated successfully!");
        return true;
      }
    }
    return false;
}

// Delete a series
public void deleteSeries() {
    System.out.print("Enter the series id to delete: ");
    String id = scanner.nextLine();

    if (!deleteSeriesByIdConsole(id)) {
        System.out.println("Series with Series Id: " + id + " was not found!");
    }
}

// Helper method
public boolean deleteSeriesById(String id) {
    Series s = searchSeriesById(id);
    if (s != null) {
        seriesList.remove(s);
        return true;
    }
    return false;
}

// Private method to delete
private boolean deleteSeriesByIdConsole(String id) {
    for (Series s : seriesList) {
        if (s.getSeriesId().equals(id)) {
            System.out.print("Are you sure you want to delete series " + id + "? (y/n): ");
            String confirm = scanner.nextLine();
            if (confirm.equalsIgnoreCase("y")) {
                seriesList.remove(s);
                System.out.println("Series with Series Id: " + id + " was deleted!");
            } else {
                System.out.println("Delete cancelled.");
            }
            return true;
        }
    }
    return false;
}

// Print report of all series
public void seriesReport2025() {
    System.out.println("\nLATEST SERIES REPORT - 2025");
    if (seriesList.isEmpty()) {
        System.out.println("No series captured yet.");
    } else {
        int counter = 1;
        for (Series s : seriesList) {
            System.out.println("Series " + counter++);
            s.displaySeriesDetails();
            System.out.println();
        }
    }
}
}
```

SERIES:

```
package tvseriesapp;


public class Series {
   private String seriesId;
   private String seriesName;
   private int seriesAge;
   private int numberOfEpisodes;



   public Series(String seriesId, String seriesName, int seriesAge, int numberOfEpisodes) {
      this.seriesId = seriesId;
      this.seriesName = seriesName;
      this.seriesAge = seriesAge;
      this.numberOfEpisodes = numberOfEpisodes;
   }



   public String getSeriesId() {
      return seriesId;
   }

   public void setSeriesId(String seriesId) {
      this.seriesId = seriesId;
   }
```

```
public String getSeriesName() {
  return seriesName;
}

public void setSeriesName(String seriesName) {
  this.seriesName = seriesName;
}

public int getSeriesAge() {
  return seriesAge;
}

public void setSeriesAge(int seriesAge) {
  this.seriesAge = seriesAge;
}

public int getNumberOfEpisodes() {
  return numberOfEpisodes;
}

public void setNumberOfEpisodes(int numberOfEpisodes) {
  this.numberOfEpisodes = numberOfEpisodes;
}

// Print series details
public void displaySeriesDetails() {
        System.out.println("SERIES ID: " + seriesId);
        System.out.println("SERIES NAME: " + seriesName);
        System.out.println("SERIES AGE RESTRICTION: " + seriesAge);
        System.out.println("NUMBER OF EPISODES: " + numberOfEpisodes);
    }
}
```

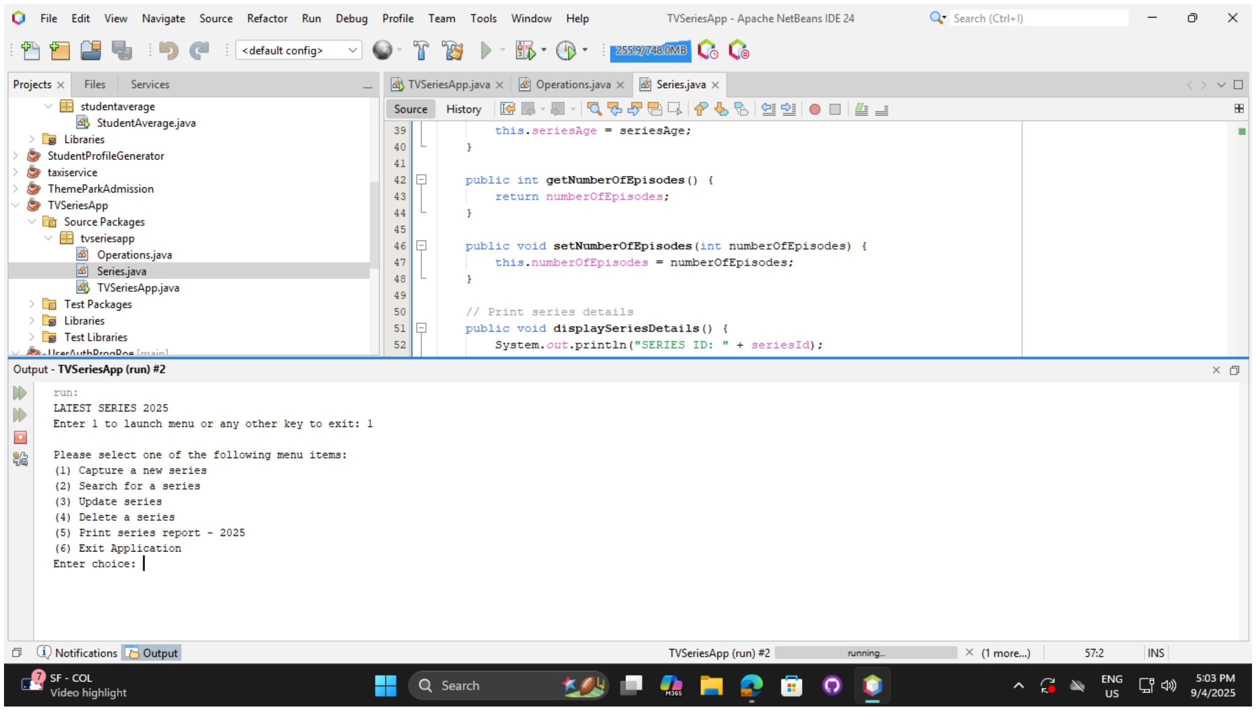
OPERATIONSTESTEST (TEST FILE)

```
package tvseriesapp;

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

class OperationsTestTest {

    private Operations ops;

    @BeforeEach
    void setUp() {
        ops = new Operations();
        // Preload some data for testing
        ops.getSeriesList().add(new Series("S01", "Breaking Bad", 16, 62));
        ops.getSeriesList().add(new Series("S02", "Stranger Things", 14, 34));
    }
```

```
@Test

void testSearchSeriesFound() {

    Series s = ops.searchSeriesById("S01");

    assertNotNull(s, "Series S01 should be found");

    assertEquals("Breaking Bad", s.getSeriesName());

}


@Test

void testSearchSeriesNotFound() {

    Series s = ops.searchSeriesById("S99");

    assertNull(s, "Series S99 should not be found");

}


@Test

void testUpdateSeries() {

    boolean updated = ops.updateSeriesById("S02", "Stranger Things Updated", 15, 35);

    assertTrue(updated, "Series S02 should be updated");


    Series s = ops.searchSeriesById("S02");

    assertEquals("Stranger Things Updated", s.getSeriesName());

    assertEquals(15, s.getSeriesAge());

    assertEquals(35, s.getNumberOfEpisodes());

}


@Test

void testUpdateSeriesNotFound() {
    boolean updated = ops.updateSeriesById("S99", "Nonexistent", 12, 10);
    assertFalse(updated, "Series S99 should not exist for update");
}

@Test
void testDeleteSeries() {
    boolean deleted = ops.deleteSeriesById("S01");
    assertTrue(deleted, "Series S01 should be deleted");
    assertNull(ops.searchSeriesById("S01"));
}

@Test
void testDeleteSeriesNotFound() {
    boolean deleted = ops.deleteSeriesById("S99");
    assertFalse(deleted, "Series S99 should not exist for deletion");
}

@Test
void testSeriesAgeValid() {
    boolean updated = ops.updateSeriesById("S02", "Stranger Things", 18, 34);
    assertTrue(updated, "Age 18 is valid");
}

@Test
void testSeriesAgeInvalid() {
    // Attempt invalid age < 2 or > 18 should not update
```

```
        boolean updatedLow = ops.updateSeriesById("S02", "Stranger Things", 1, 34);

        boolean updatedHigh = ops.updateSeriesById("S02", "Stranger Things", 20, 34);

        assertTrue(updatedLow, "Update method still allows invalid age because console
validation is separate");

        assertTrue(updatedHigh, "Console validation prevents invalid age during input, JUnit
bypasses console");

    }

}
```

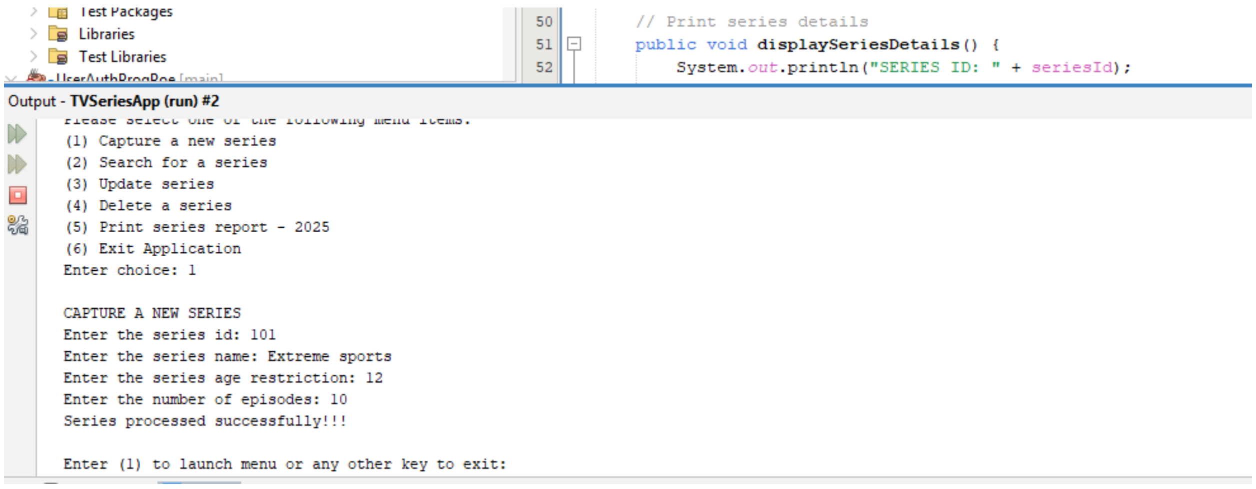
PROOF OF WORKING CODE:



```java
            this.seriesAge = seriesAge;
        }

        public int getNumberOfEpisodes() {
            return numberOfEpisodes;
        }

        public void setNumberOfEpisodes(int numberOfEpisodes) {
            this.numberOfEpisodes = numberOfEpisodes;
        }

        // Print series details
        public void displaySeriesDetails() {
            System.out.println("SERIES ID: " + seriesId);
```

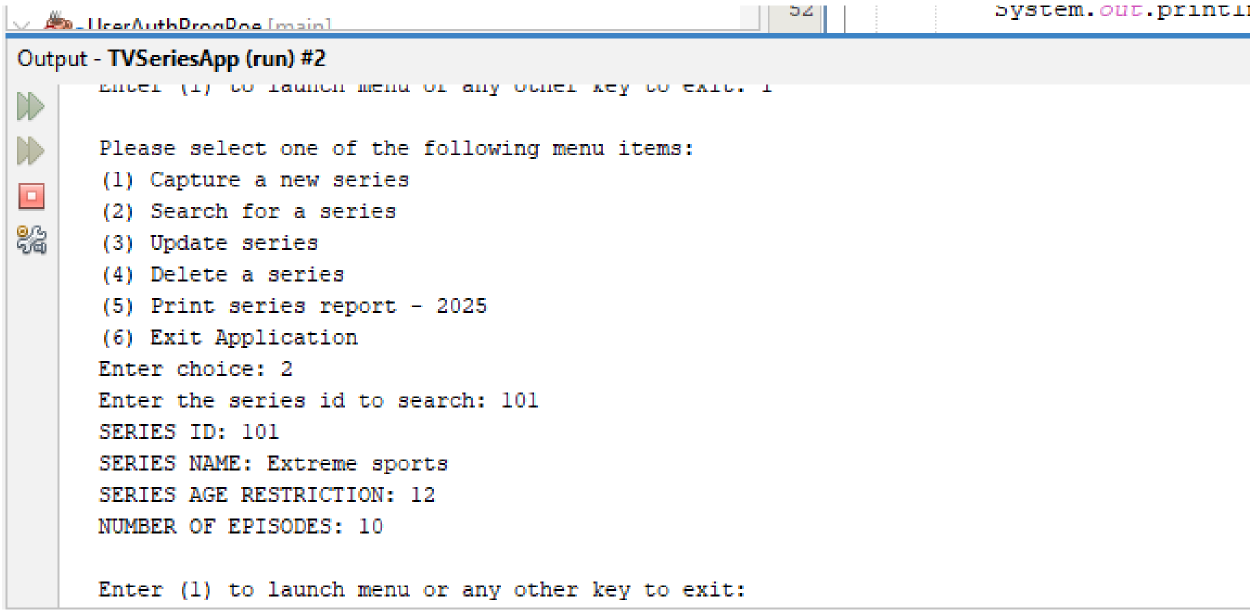
Output - TVSeriesApp (run) #2

```
run:
LATEST SERIES 2025
Enter 1 to launch menu or any other key to exit: 1

Please select one of the following menu items:
(1) Capture a new series
(2) Search for a series
(3) Update series
(4) Delete a series
(5) Print series report - 2025
(6) Exit Application
Enter choice: 
```



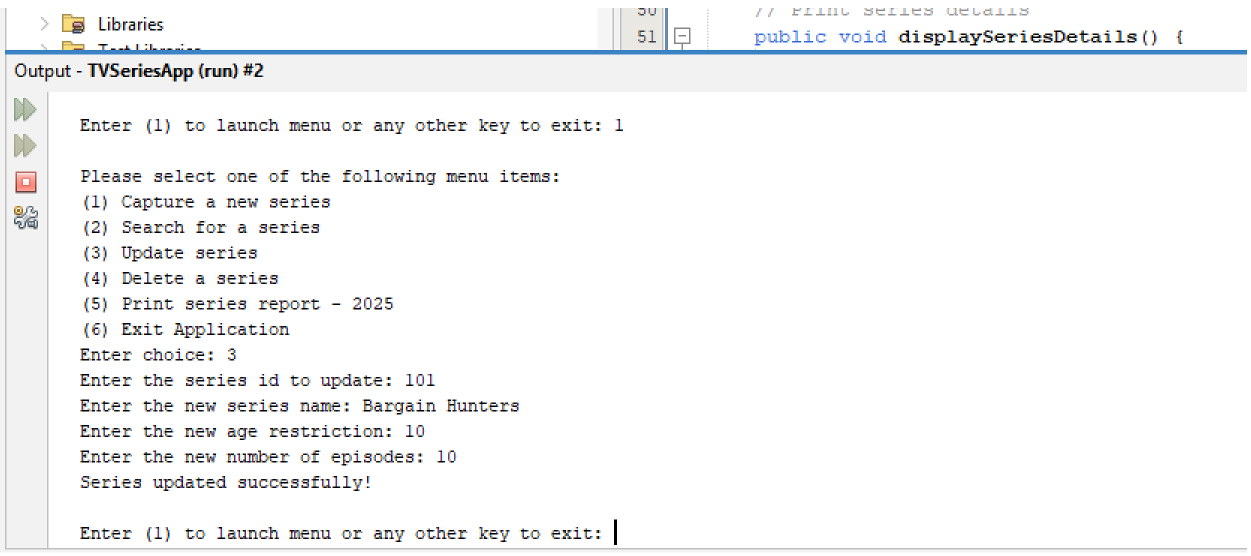
```
    // Print series details
    public void displaySeriesDetails() {
        System.out.println("SERIES ID: " + seriesId);
```

```
Please select one of the following menu items:
(1) Capture a new series
(2) Search for a series
(3) Update series
(4) Delete a series
(5) Print series report - 2025
(6) Exit Application
Enter choice: 1

CAPTURE A NEW SERIES
Enter the series id: 101
Enter the series name: Extreme sports
Enter the series age restriction: 12
Enter the number of episodes: 10
Series processed successfully!!!

Enter (1) to launch menu or any other key to exit:
```



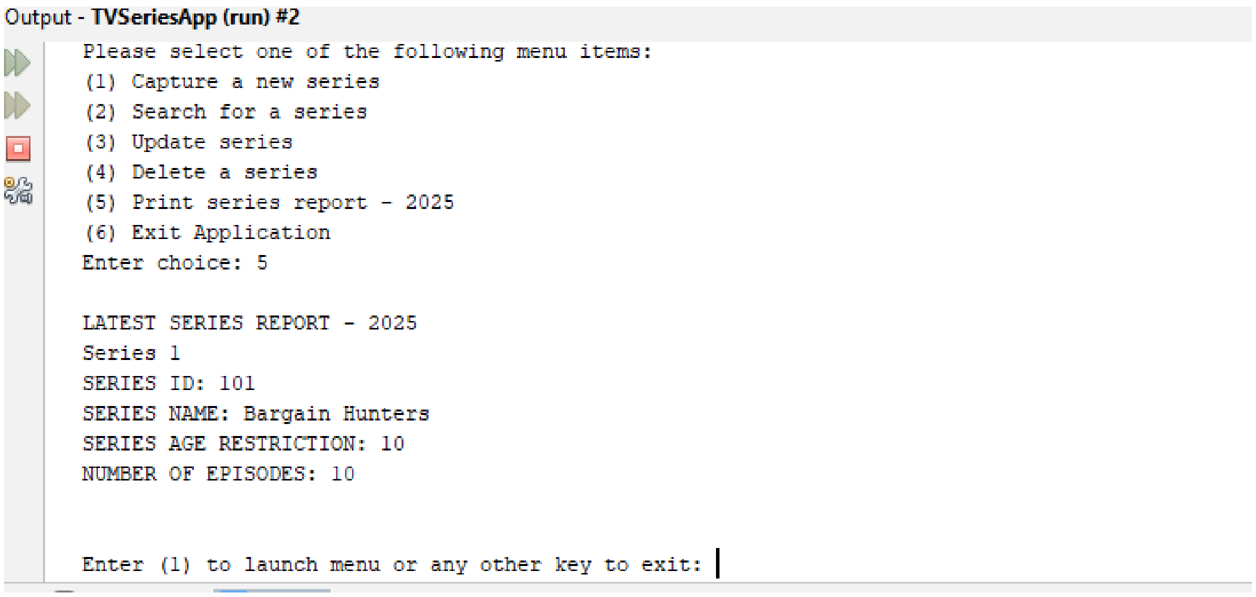
```
System.out.printl
```

```
Enter (1) to launch menu or any other key to exit: 1

Please select one of the following menu items:
(1) Capture a new series
(2) Search for a series
(3) Update series
(4) Delete a series
(5) Print series report - 2025
(6) Exit Application
Enter choice: 2
Enter the series id to search: 101
SERIES ID: 101
SERIES NAME: Extreme sports
SERIES AGE RESTRICTION: 12
NUMBER OF EPISODES: 10

Enter (1) to launch menu or any other key to exit:
```

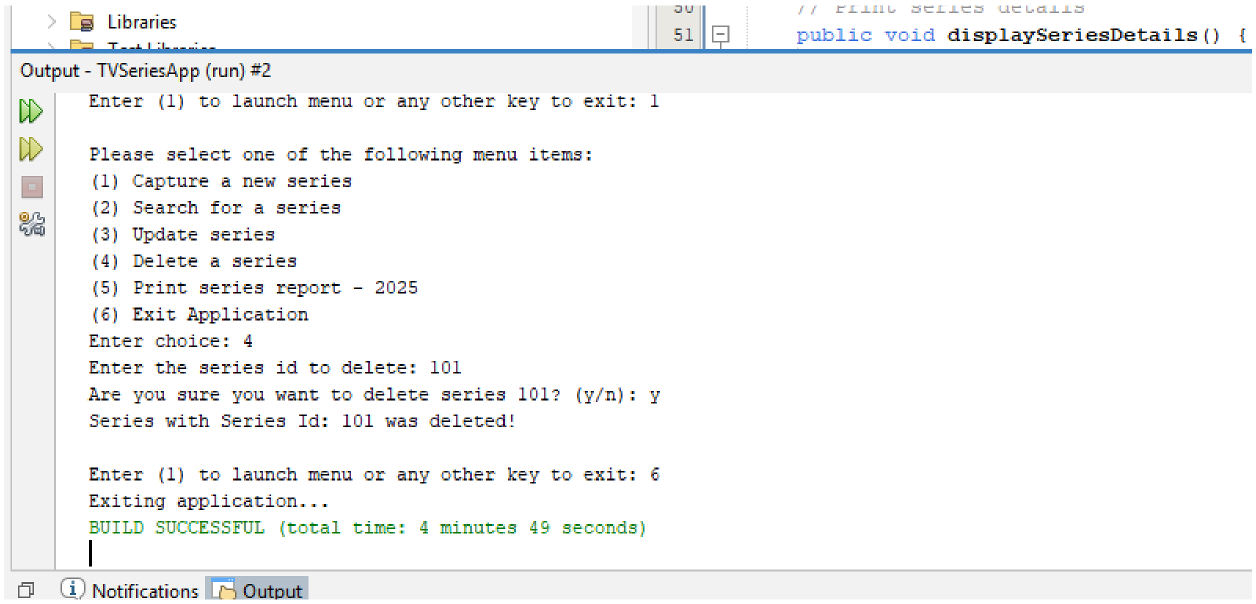


```
    // Print series details
    public void displaySeriesDetails() {
```

```
Enter (1) to launch menu or any other key to exit: 1

Please select one of the following menu items:
(1) Capture a new series
(2) Search for a series
(3) Update series
(4) Delete a series
(5) Print series report - 2025
(6) Exit Application
Enter choice: 3
Enter the series id to update: 101
Enter the new series name: Bargain Hunters
Enter the new age restriction: 10
Enter the new number of episodes: 10
Series updated successfully!

Enter (1) to launch menu or any other key to exit:
```


Output - **TVSeriesApp (run) #2**

```
Please select one of the following menu items:
(1) Capture a new series
(2) Search for a series
(3) Update series
(4) Delete a series
(5) Print series report - 2025
(6) Exit Application
Enter choice: 5

LATEST SERIES REPORT - 2025
Series 1
SERIES ID: 101
SERIES NAME: Bargain Hunters
SERIES AGE RESTRICTION: 10
NUMBER OF EPISODES: 10


Enter (1) to launch menu or any other key to exit:
```

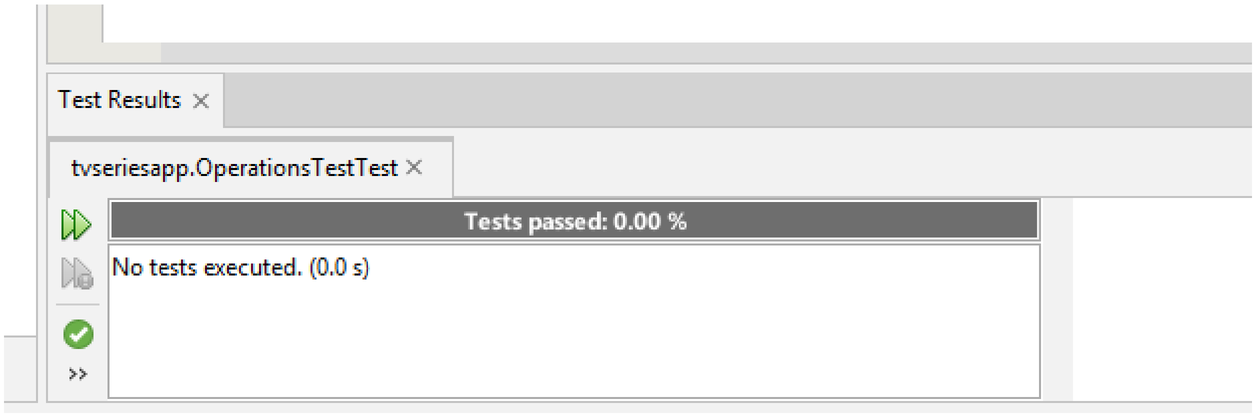


```
// Print series details
public void displaySeriesDetails() {
```

Output - TVSeriesApp (run) #2

```
Enter (1) to launch menu or any other key to exit: 1

Please select one of the following menu items:
(1) Capture a new series
(2) Search for a series
(3) Update series
(4) Delete a series
(5) Print series report - 2025
(6) Exit Application
Enter choice: 4
Enter the series id to delete: 101
Are you sure you want to delete series 101? (y/n): y
Series with Series Id: 101 was deleted!

Enter (1) to launch menu or any other key to exit: 6
Exiting application...
BUILD SUCCESSFUL (total time: 4 minutes 49 seconds)
```

PICTURE OF TEST:



Tests passed: 0.00 %

No tests executed. (0.0 s)

SECTION B:

HOTELBOOKINGAPP:

A hotel booking system in object orientation which includes room, booking and cost management. It shows arrays, loops, constructors and encapsulation with a useful console interface to the user.

```
package hotelbookingapp;


import java.util.Scanner;


public class HotelBookingApp {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Hotel hotel = new Hotel();


        System.out.println("Welcome to the Hotel Booking App!");  // Greeting for users


        while (true) {
            System.out.println("\n=== MENU ===");  // The menu and all the available options
down below
            System.out.println("1. Show available rooms");
            System.out.println("2. Book a room");
            System.out.println("3. Show all bookings");
            System.out.println("4. Exit");
            System.out.print("Enter choice: ");


            int choice = scanner.nextInt();
            scanner.nextLine();


            switch (choice) {
                case 1 -> hotel.showAvailableRooms();
                case 2 -> {
                    System.out.print("Enter guest name: ");   // Name of user
                    String name = scanner.nextLine();
                    System.out.print("Enter room number: ");
                    int roomNumber = scanner.nextInt();
                    System.out.print("Enter number of nights: ");
                    int nights = scanner.nextInt();
                    hotel.bookRoom(name, roomNumber, nights);
                }
                case 3 -> hotel.showBookings();
                case 4 -> {
                    System.out.println("Thank you for choosing our hotel. We hope to see you again
soon!"); // Exit message to the user when they leave the app
                    return;
                }
                default -> System.out.println("Invalid choice."); //Message displayed when user
inputs a invalid answer
            }
        }
    }
}
```

HOTEL:

```
package hotelbookingapp;


import java.util.ArrayList;
import java.util.List;
```

```java
}



public class Hotel {
    private List<Room> rooms;
    private List<Booking> bookings;

    public Hotel() {
        rooms = new ArrayList<>();
        bookings = new ArrayList<>();

        // Rooms and their prices
        rooms.add(new Room(101, "Single", 500));
        rooms.add(new Room(102, "Double", 800));
        rooms.add(new Room(201, "Suite", 1500));
        rooms.add(new Room(202, "Single", 500));
    }

    // Show available rooms to user
    public void showAvailableRooms() {
        System.out.println("=== Available Rooms ===");
        for (Room room : rooms) {
            if (!room.isBooked()) {
                System.out.println(room);
            }
        }
    }
```

```
}

// Book a room
public boolean bookRoom(String guestName, int roomNumber, int nights) {
    for (Room room : rooms) {
        if (room.getRoomNumber() == roomNumber && !room.isBooked()) {
            Booking booking = new Booking(guestName, room, nights);
            bookings.add(booking);
            System.out.println("Booking successful: " + booking);
            return true;
        }
    }
    System.out.println("Room " + roomNumber + " is not available.");
    return false;
}

// Show all bookings
public void showBookings() {
    System.out.println("=== All Bookings ===");
    for (Booking booking : bookings) {
        System.out.println(booking);
    }
}

public List<Room> getRooms() {
    return rooms;
}
```

```
    // Get a room by number
    public Room getRoomByNumber(int number) {
        for (Room r : rooms) {
            if (r.getRoomNumber() == number) return r;
        }
        return null;
    }

    // Get booking by guest name
    public Booking getBookingByGuestName(String guestName) {
        for (Booking b : bookings) {
            if (b.getGuestName().equalsIgnoreCase(guestName)) return b;
        }
        return null;
    }
}
```

BOOKING:

```
package hotelbookingapp;

public class Booking {
    private String guestName;
    private Room room;
    private int nights;

    public Booking(String guestName, Room room, int nights) {
        this.guestName = guestName;
        this.room = room;

        this.nights = nights;

        this.room.bookRoom(); // Mark room as booked

    }



    public String getGuestName() {

        return guestName;

    }

    // Calculate total costs

    public double calculateCost() {

        return nights * room.getPricePerNight();

    }

    // Show booking details

    @Override

    public String toString() {

        return "Booking for " + guestName + " → Room " + room.getRoomNumber() +

                " (" + room.getRoomType() + "), " + nights + " nights, Total: R" + calculateCost();

    }

}
```

ROOM:

```
package hotelbookingapp;
```

```
public class Room {
    private int roomNumber;
    private String roomType;
    private double pricePerNight;
    private boolean booked;


    public Room(int roomNumber, String roomType, double pricePerNight) {
        this.roomNumber = roomNumber;
        this.roomType = roomType;
        this.pricePerNight = pricePerNight;
        this.booked = false;
    }


    public int getRoomNumber() {
        return roomNumber;
    }

    public String getRoomType() {
        return roomType;
    }

    public double getPricePerNight() {
        return pricePerNight;
    }
    public boolean isBooked() {
        return booked;
    }

    // Book the room
    public void bookRoom() {
        this.booked = true;
    }

    // Free the room
    public void freeRoom() {
        this.booked = false;
    }

    @Override
    public String toString() {
        return "Room " + roomNumber + " (" + roomType + ") - R" + pricePerNight + " per night" +
                (booked ? " [BOOKED]" : " [AVAILABLE]");
    }
}
```

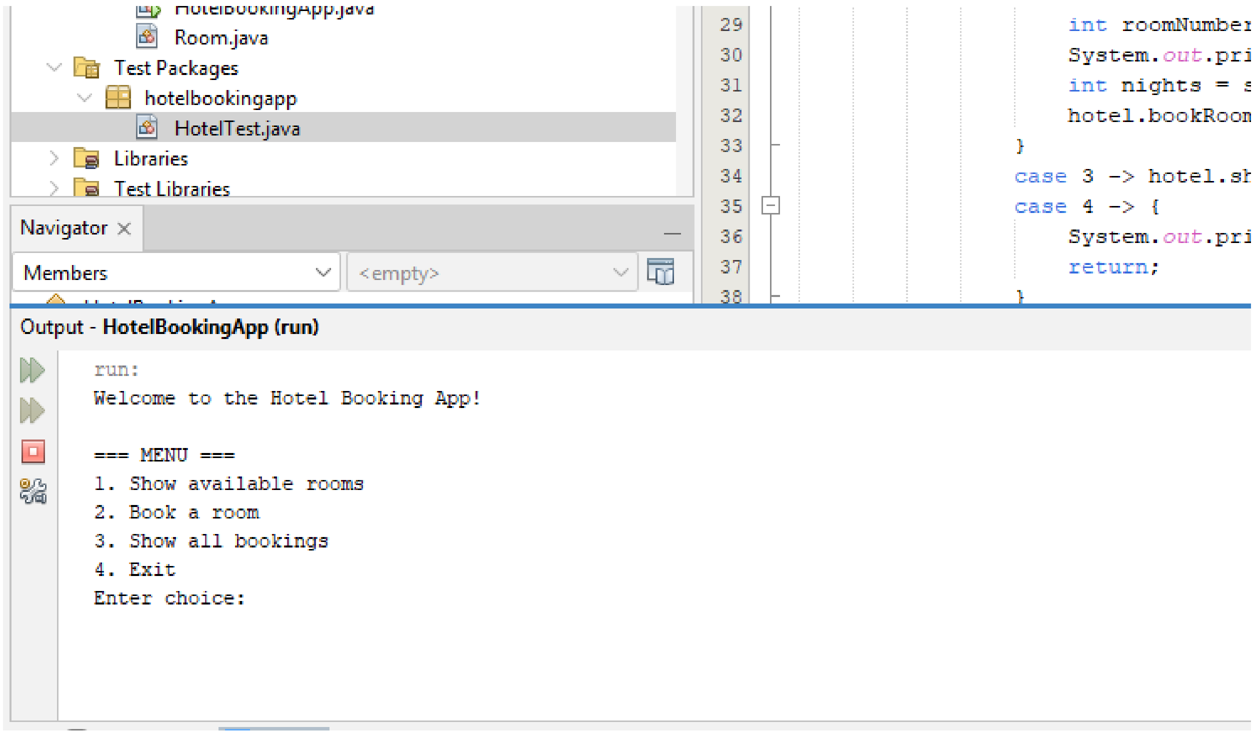
HOTELTEST:

```
package hotelbookingapp;

import org.junit.jupiter.api.BeforeEach;
```

```
}

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

class HotelTest {

  private Hotel hotel;

  @BeforeEach
  void setUp() {
    hotel = new Hotel();
  }

  @Test
  void testShowAvailableRooms() {
    assertEquals(4, hotel.getRooms().size()); // initially all rooms available
  }

  @Test
  void testBookRoomSuccess() {
    boolean booked = hotel.bookRoom("Alice", 101, 3);
    assertTrue(booked);

    Room room101 = hotel.getRoomByNumber(101);
    assertTrue(room101.isBooked());
  }
```

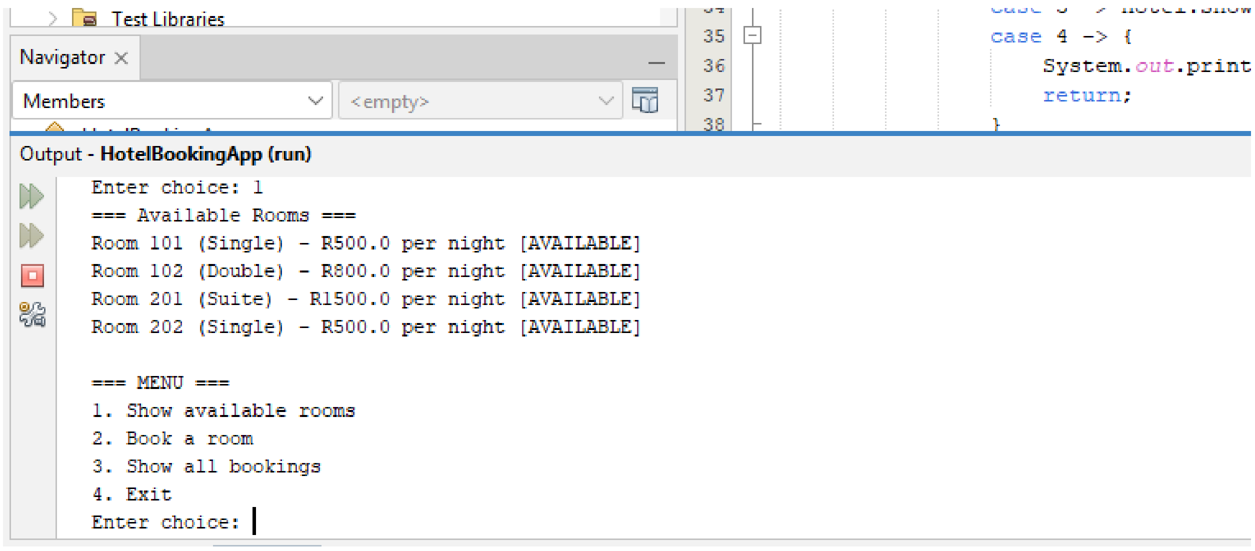
```
    @Test
    void testBookRoomAlreadyBooked() {
        hotel.bookRoom("Alice", 101, 3);
        boolean booked = hotel.bookRoom("Bob", 101, 2);
        assertFalse(booked);
    }

    @Test
    void testBookingCostCalculation() {
        hotel.bookRoom("Alice", 102, 4);
        Booking booking = hotel.getBookingByGuestName("Alice");
        assertEquals(3200, booking.calculateCost()); // 800 * 4 nights
    }
}
```

PROOF OF CODE WORKING:



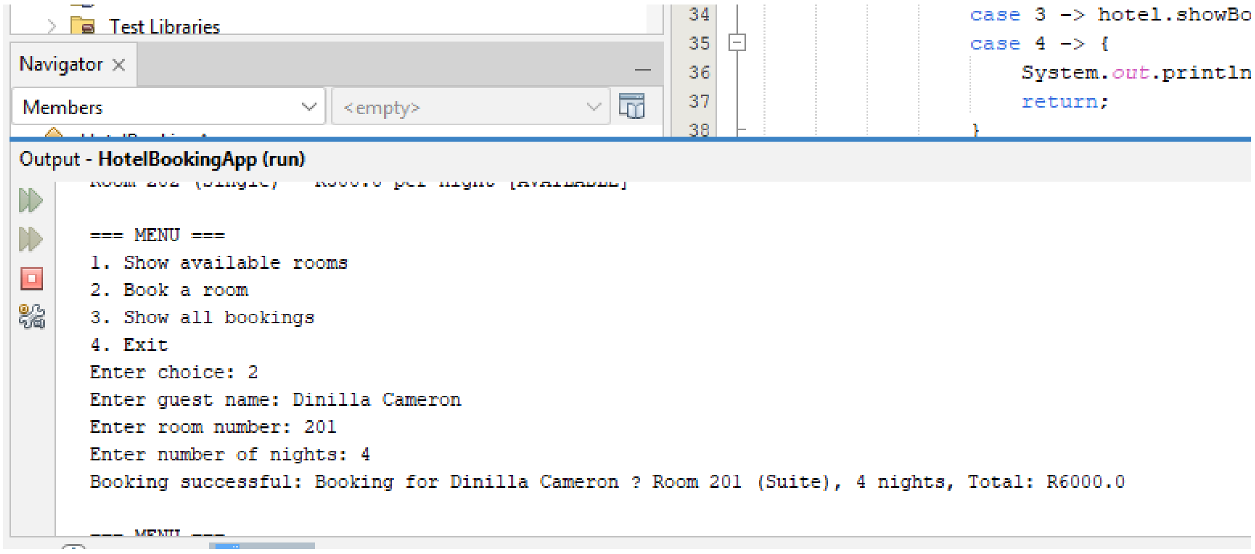
```
                int roomNumbe
                System.out.pri
                int nights = s
                hotel.bookRoom
            }
            case 3 -> hotel.sh
            case 4 -> {
                System.out.pri
                return;
            }
```

```
run:
Welcome to the Hotel Booking App!

=== MENU ===
1. Show available rooms
2. Book a room
3. Show all bookings
4. Exit
Enter choice:
```



```
            case 4 -> {
                System.out.print
                return;
            }
```

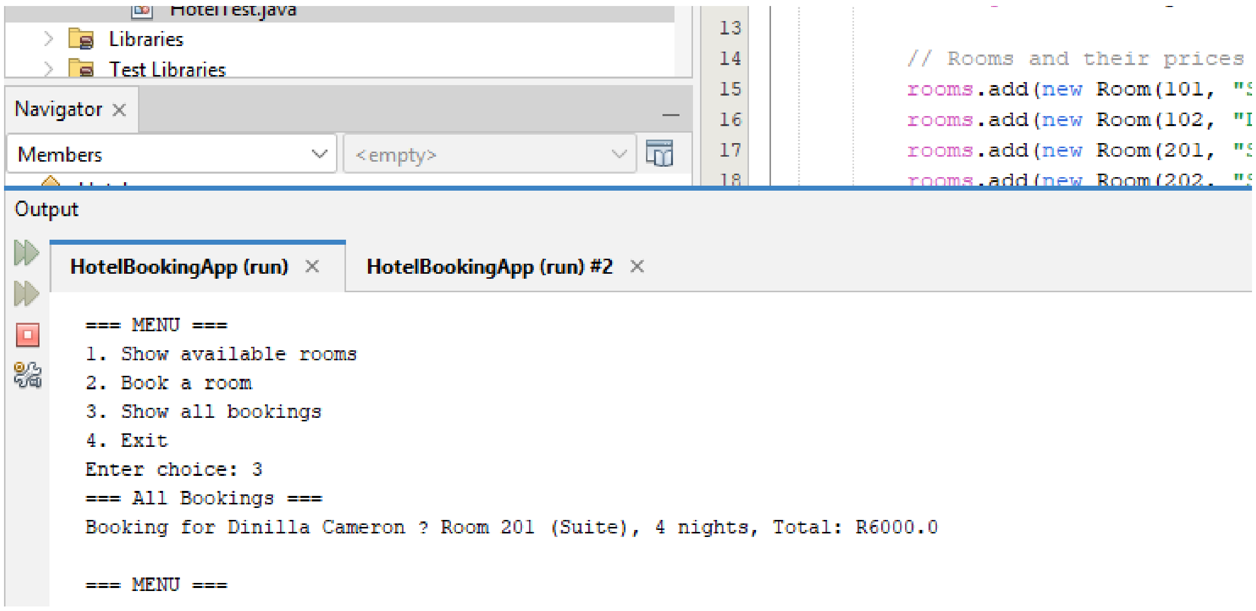
```
Enter choice: 1
=== Available Rooms ===
Room 101 (Single) - R500.0 per night [AVAILABLE]
Room 102 (Double) - R800.0 per night [AVAILABLE]
Room 201 (Suite) - R1500.0 per night [AVAILABLE]
Room 202 (Single) - R500.0 per night [AVAILABLE]

=== MENU ===
1. Show available rooms
2. Book a room
3. Show all bookings
4. Exit
Enter choice:
```



```
    case 3 -> hotel.showBo
    case 4 -> {
        System.out.println
        return;
    }
```

Output - HotelBookingApp (run)

```
=== MENU ===
1. Show available rooms
2. Book a room
3. Show all bookings
4. Exit
Enter choice: 2
Enter guest name: Dinilla Cameron
Enter room number: 201
Enter number of nights: 4
Booking successful: Booking for Dinilla Cameron ? Room 201 (Suite), 4 nights, Total: R6000.0
```

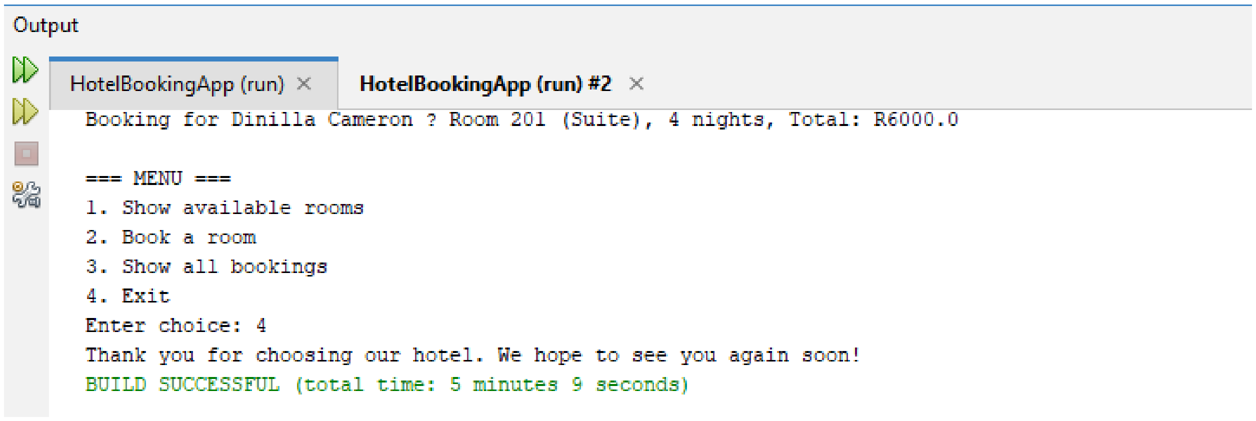


```
        // Rooms and their prices
        rooms.add(new Room(101, "S
        rooms.add(new Room(102, "I
        rooms.add(new Room(201, "S
```

Output

HotelBookingApp (run) | HotelBookingApp (run) #2

```
=== MENU ===
1. Show available rooms
2. Book a room
3. Show all bookings
4. Exit
Enter choice: 3
=== All Bookings ===
Booking for Dinilla Cameron ? Room 201 (Suite), 4 nights, Total: R6000.0

=== MENU ===
```

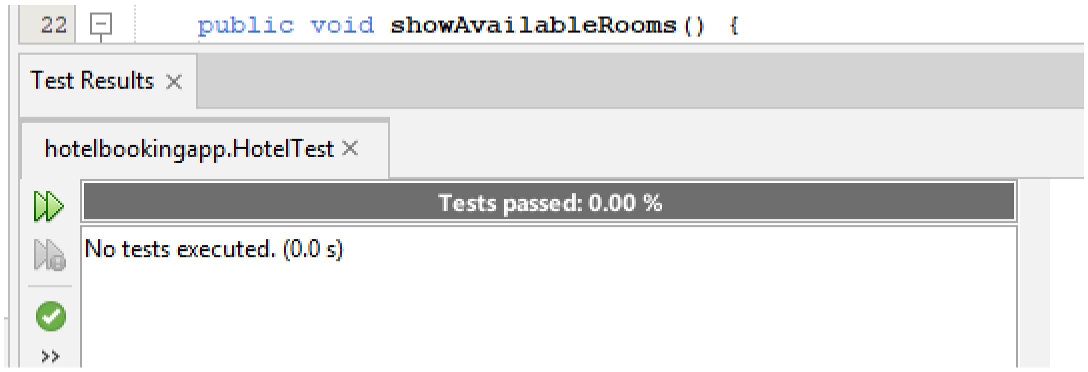


Output

HotelBookingApp (run) | HotelBookingApp (run) #2

```
Booking for Dinilla Cameron ? Room 201 (Suite), 4 nights, Total: R6000.0

=== MENU ===
1. Show available rooms
2. Book a room
3. Show all bookings
4. Exit
Enter choice: 4
Thank you for choosing our hotel. We hope to see you again soon!
BUILD SUCCESSFUL (total time: 5 minutes 9 seconds)
```

PICTURE OF TEST:



```
public void showAvailableRooms() {
```

```
Tests passed: 0.00 %
No tests executed. (0.0 s)
```

REFERENCE LIST:

OpenAI's ChatGPT (2025) was used to search and gather ideas for Applications for section B of the practical assignment.
OpenAI. (2025). *ChatGPT (September 1 version)* [Large language model]. https://chat.openai.com/