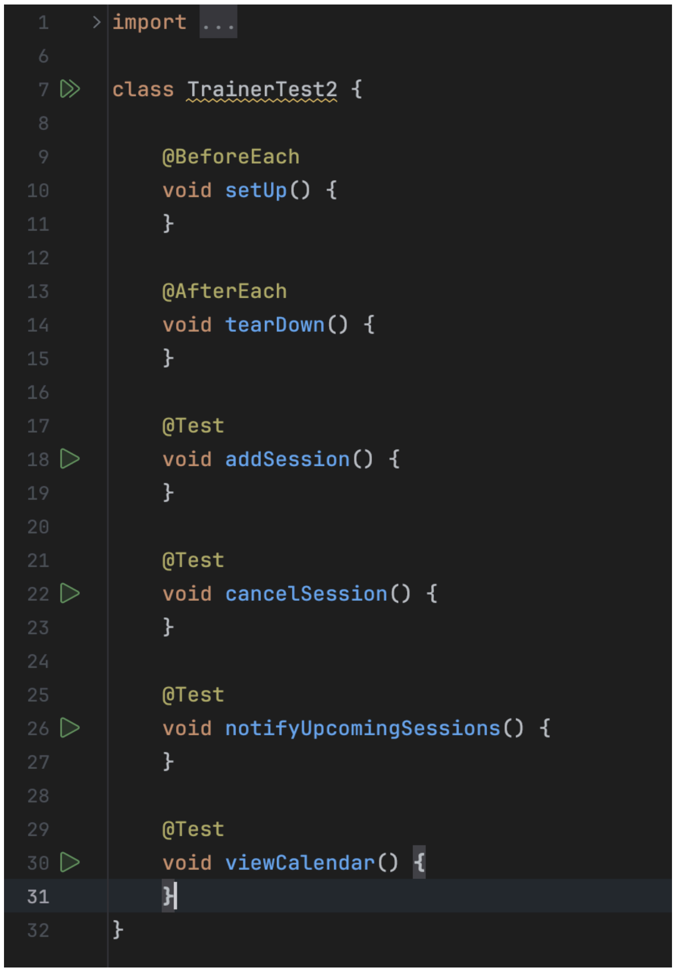
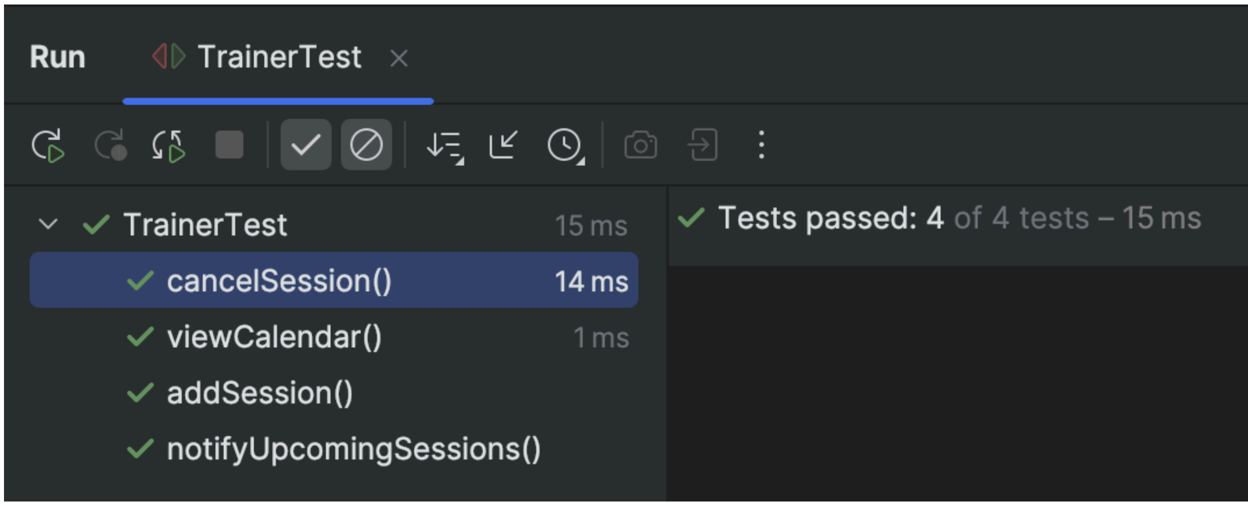
1st Trainer Tests

- What scenarios did you use to evaluate the code? Take screenshots
    - We used Intellij to first create the framework for the JUnit tests. Then based on the source code and the framework had perplexity create the unit tests. The scenarios were:
    - Adding a session
    - Cancelling a session
    - Notifying
    - Viewing Calendar
    - 

```java
import ...

class TrainerTest2 {

    @BeforeEach
    void setUp() {
    }

    @AfterEach
    void tearDown() {
    }

    @Test
    void addSession() {
    }

    @Test
    void cancelSession() {
    }

    @Test
    void notifyUpcomingSessions() {
    }

    @Test
    void viewCalendar() {
    }
}
```

- What was the outcome for each tested scenario? Take screenshots
    - The tests ran were: It sets up a `Trainer` object and two `Session` objects in the `setUp()` method.
    - It uses `ByteArrayOutputStream` to capture the console output for testing.
    - The `tearDown()` method restores the original `System.out`.

- Each test method (`addSession()`, `cancelSession()`, `notifyUpcomingSessions()`, and `viewCalendar()`) checks if the corresponding method in the `Trainer` class produces the expected output.
- The tests use assertions to verify that the output contains expected strings.

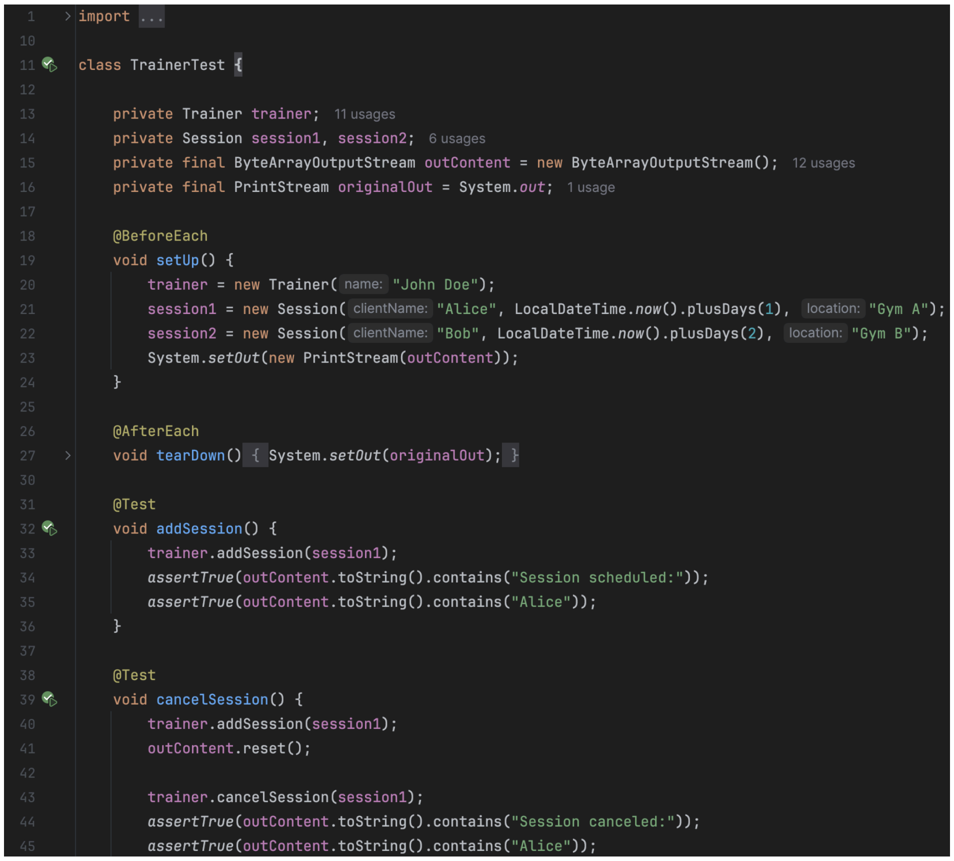


Run TrainerTest

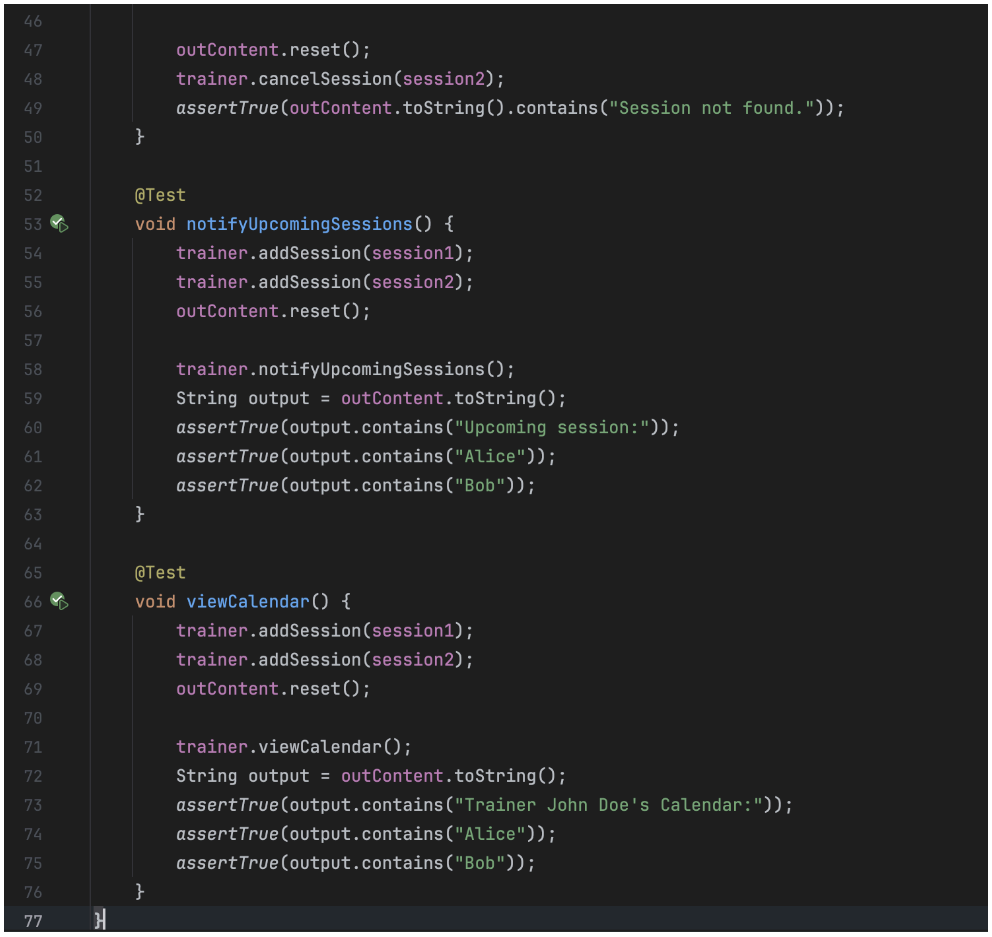
```
✓ TrainerTest                 15 ms    ✓ Tests passed: 4 of 4 tests – 15 ms
    ✓ cancelSession()         14 ms
    ✓ viewCalendar()           1 ms
    ✓ addSession()
    ✓ notifyUpcomingSessions()
```

-

* In terms of coverage, much code did you evaluate with your white-box tests? Take a screenshots



```java
import ...

class TrainerTest {

    private Trainer trainer;  11 usages
    private Session session1, session2;  6 usages
    private final ByteArrayOutputStream outContent = new ByteArrayOutputStream();  12 usages
    private final PrintStream originalOut = System.out;  1 usage

    @BeforeEach
    void setUp() {
        trainer = new Trainer( name: "John Doe");
        session1 = new Session( clientName: "Alice", LocalDateTime.now().plusDays(1),  location: "Gym A");
        session2 = new Session( clientName: "Bob", LocalDateTime.now().plusDays(2),  location: "Gym B");
        System.setOut(new PrintStream(outContent));
    }

    @AfterEach
    void tearDown() { System.setOut(originalOut); }

    @Test
    void addSession() {
        trainer.addSession(session1);
        assertTrue(outContent.toString().contains("Session scheduled:"));
        assertTrue(outContent.toString().contains("Alice"));
    }

    @Test
    void cancelSession() {
        trainer.addSession(session1);
        outContent.reset();

        trainer.cancelSession(session1);
        assertTrue(outContent.toString().contains("Session canceled:"));
        assertTrue(outContent.toString().contains("Alice"));
```



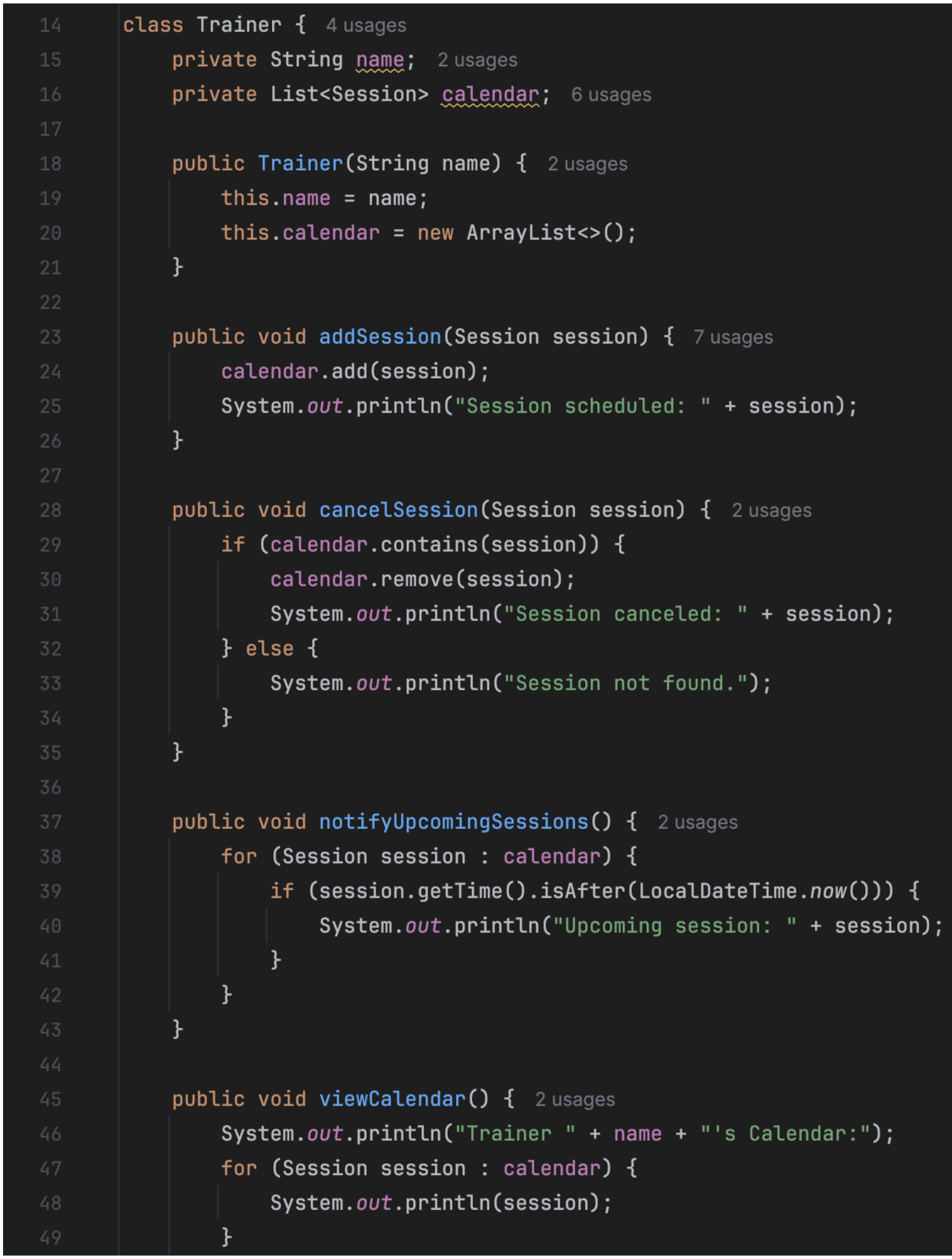
```
        outContent.reset();
        trainer.cancelSession(session2);
        assertTrue(outContent.toString().contains("Session not found."));
    }

    @Test
    void notifyUpcomingSessions() {
        trainer.addSession(session1);
        trainer.addSession(session2);
        outContent.reset();

        trainer.notifyUpcomingSessions();
        String output = outContent.toString();
        assertTrue(output.contains("Upcoming session:"));
        assertTrue(output.contains("Alice"));
        assertTrue(output.contains("Bob"));
    }

    @Test
    void viewCalendar() {
        trainer.addSession(session1);
        trainer.addSession(session2);
        outContent.reset();

        trainer.viewCalendar();
        String output = outContent.toString();
        assertTrue(output.contains("Trainer John Doe's Calendar:"));
        assertTrue(output.contains("Alice"));
        assertTrue(output.contains("Bob"));
    }
}
```

  - This code only tested the major class which was approximately 70% of the code.
- If you used ChatGPT, what prompts did you use?
  - Used Perplexity with the prompt below:
  - Create a JUnit unit test for this java class: source code provided. Use the following framework: framework code created by Intellij provided
  - create a junit unit test for this class: class Trainer {
  -     private String name;
  -     private List<Session> calendar;
  -
  -     public Trainer(String name) {
  -         this.name = name;
  -         this.calendar = new ArrayList<>();
  -     }
  -
  -     public void addSession(Session session) {

```
○           calendar.add(session);
○           System.out.println("Session scheduled: " + session);
○       }
○
○       public void cancelSession(Session session) {
○           if (calendar.contains(session)) {
○               calendar.remove(session);
○               System.out.println("Session canceled: " + session);
○           } else {
○               System.out.println("Session not found.");
○           }
○       }
○
○       public void notifyUpcomingSessions() {
○           for (Session session : calendar) {
○               if (session.getTime().isAfter(LocalDateTime.now())) {
○                   System.out.println("Upcoming session: " + session);
○               }
○           }
○       }
○
○       public void viewCalendar() {
○           System.out.println("Trainer " + name + "'s Calendar:");
○           for (Session session : calendar) {
○               System.out.println(session);
○           }
○       }
○   } Use this format: import org.junit.jupiter.api.AfterEach;
○   import org.junit.jupiter.api.BeforeEach;
○   import org.junit.jupiter.api.Test;
○
○   import static org.junit.jupiter.api.Assertions.*;
○
○   class TrainerTest {
○
○       @BeforeEach
○       void setUp() {
○       }
○
```

- @AfterEach
- void tearDown() {
- }
-
- @Test
- void addSession() {
- }
-
- @Test
- void cancelSession() {
- }
-
- @Test
- void notifyUpcomingSessions() {
- }
-
- @Test
- void viewCalendar() {
- }

- }



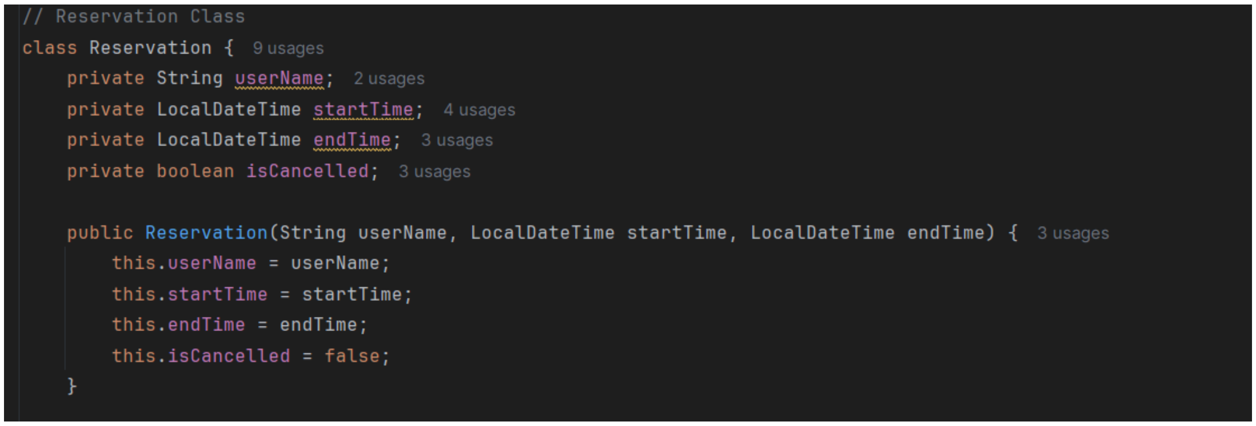
```
class Trainer {  4 usages
    private String name;  2 usages
    private List<Session> calendar;  6 usages

    public Trainer(String name) {  2 usages
        this.name = name;
        this.calendar = new ArrayList<>();
    }

    public void addSession(Session session) {  7 usages
        calendar.add(session);
        System.out.println("Session scheduled: " + session);
    }

    public void cancelSession(Session session) {  2 usages
        if (calendar.contains(session)) {
            calendar.remove(session);
            System.out.println("Session canceled: " + session);
        } else {
            System.out.println("Session not found.");
        }
    }

    public void notifyUpcomingSessions() {  2 usages
        for (Session session : calendar) {
            if (session.getTime().isAfter(LocalDateTime.now())) {
                System.out.println("Upcoming session: " + session);
            }
        }
    }

    public void viewCalendar() {  2 usages
        System.out.println("Trainer " + name + "'s Calendar:");
        for (Session session : calendar) {
            System.out.println(session);
        }
```

2nd Trainer Tests

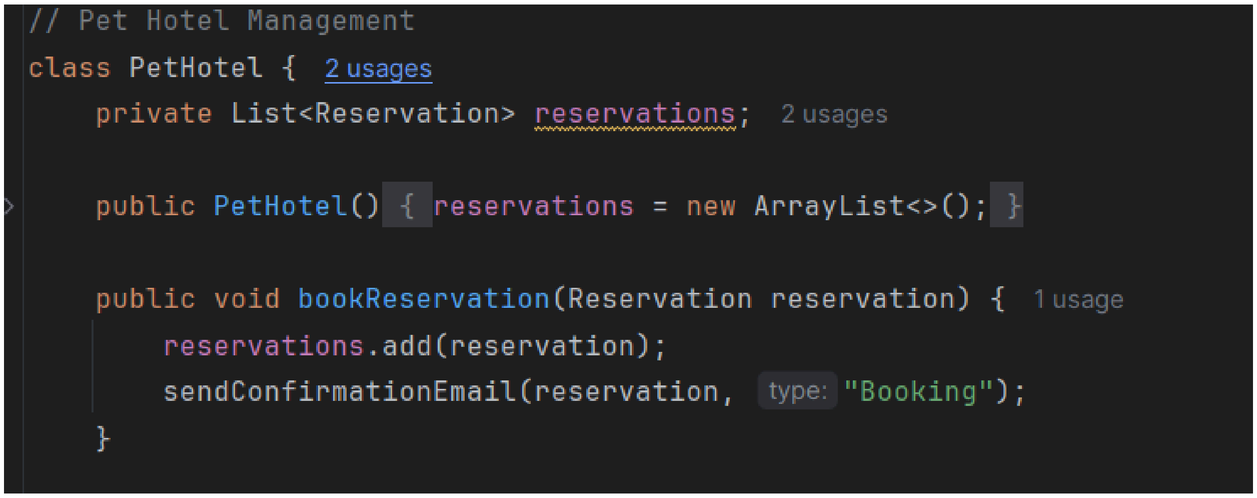
- What scenarios did you use to evaluate the code? Take screenshots

The scenarios were:

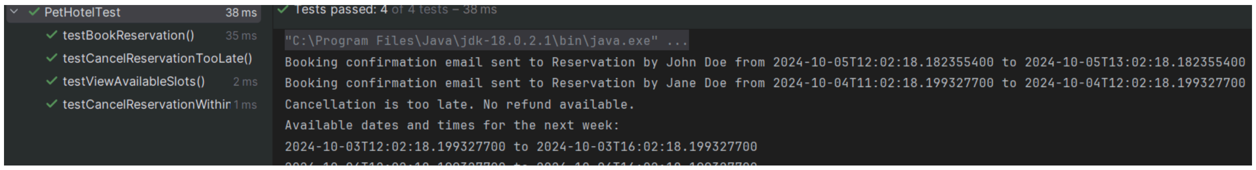
Reservation and the Pet Hotel



```java
// Reservation Class
class Reservation {  9 usages
    private String userName;  2 usages
    private LocalDateTime startTime;  4 usages
    private LocalDateTime endTime;  3 usages
    private boolean isCancelled;  3 usages

    public Reservation(String userName, LocalDateTime startTime, LocalDateTime endTime) {  3 usages
        this.userName = userName;
        this.startTime = startTime;
        this.endTime = endTime;
        this.isCancelled = false;
    }
```



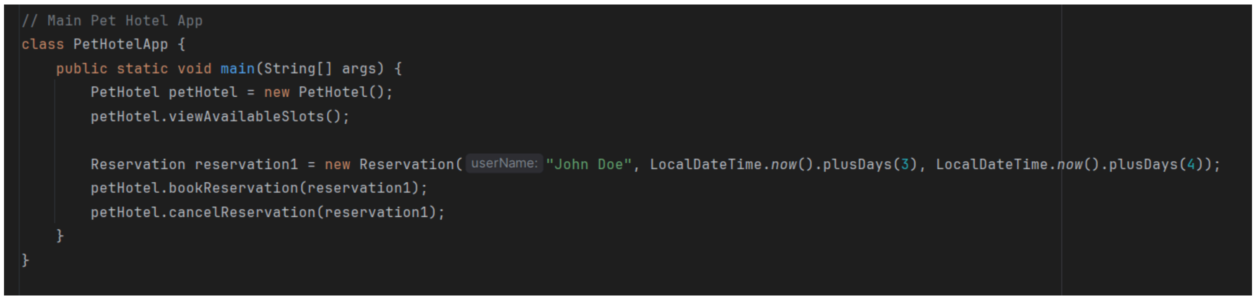
```java
// Pet Hotel Management
class PetHotel {  2 usages
    private List<Reservation> reservations;  2 usages

    public PetHotel() { reservations = new ArrayList<>(); }

    public void bookReservation(Reservation reservation) {  1 usage
        reservations.add(reservation);
        sendConfirmationEmail(reservation, type: "Booking");
    }
```

- What was the outcome for each tested scenario? Take screenshots
- The outcome was that ¾ tests passed, but the testReservationTool failed as the name outputted was incorrect.
- 

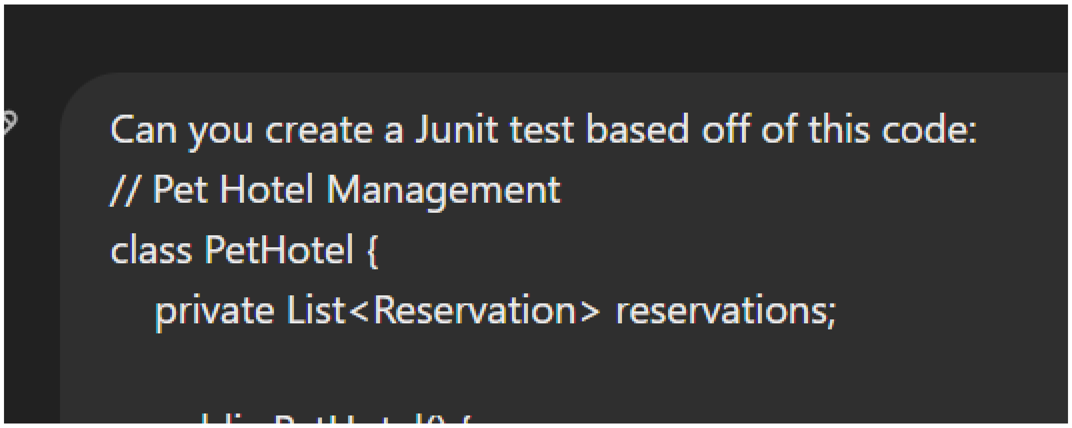
```
✓ PetHotelTest                         38 ms
    ✓ testBookReservation()            35 ms
    ✓ testCancelReservationTooLate()
    ✓ testViewAvailableSlots()          2 ms
    ✓ testCancelReservationWithir       1 ms
```

```
✓ Tests passed: 4 of 4 tests – 38 ms

"C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe" ...
Booking confirmation email sent to Reservation by John Doe from 2024-10-05T12:02:18.182355400 to 2024-10-05T13:02:18.182355400
Booking confirmation email sent to Reservation by Jane Doe from 2024-10-04T11:02:18.199327700 to 2024-10-04T12:02:18.199327700
Cancellation is too late. No refund available.
Available dates and times for the next week:
2024-10-03T12:02:18.199327700 to 2024-10-03T16:02:18.199327700
```

- For the second one all tested passed.

- In terms of coverage, much code did you evaluate with your white-box tests? Take a screenshots
- In terms of coverage, we covered all the classes and methods except the PetHotelApp class and methods.
- 

```
// Main Pet Hotel App
class PetHotelApp {
    public static void main(String[] args) {
        PetHotel petHotel = new PetHotel();
        petHotel.viewAvailableSlots();

        Reservation reservation1 = new Reservation( userName: "John Doe", LocalDateTime.now().plusDays(3), LocalDateTime.now().plusDays(4));
        petHotel.bookReservation(reservation1);
        petHotel.cancelReservation(reservation1);
    }
}
```

- If you used ChatGPT, what prompts did you use?
- For the prompts I just inputted for it to make a Junit test based off of the code provided, which was the full classes.
- 

Can you create a Junit test based off of this code:
// Pet Hotel Management
class PetHotel {
    private List<Reservation> reservations;