Contents

Program UML

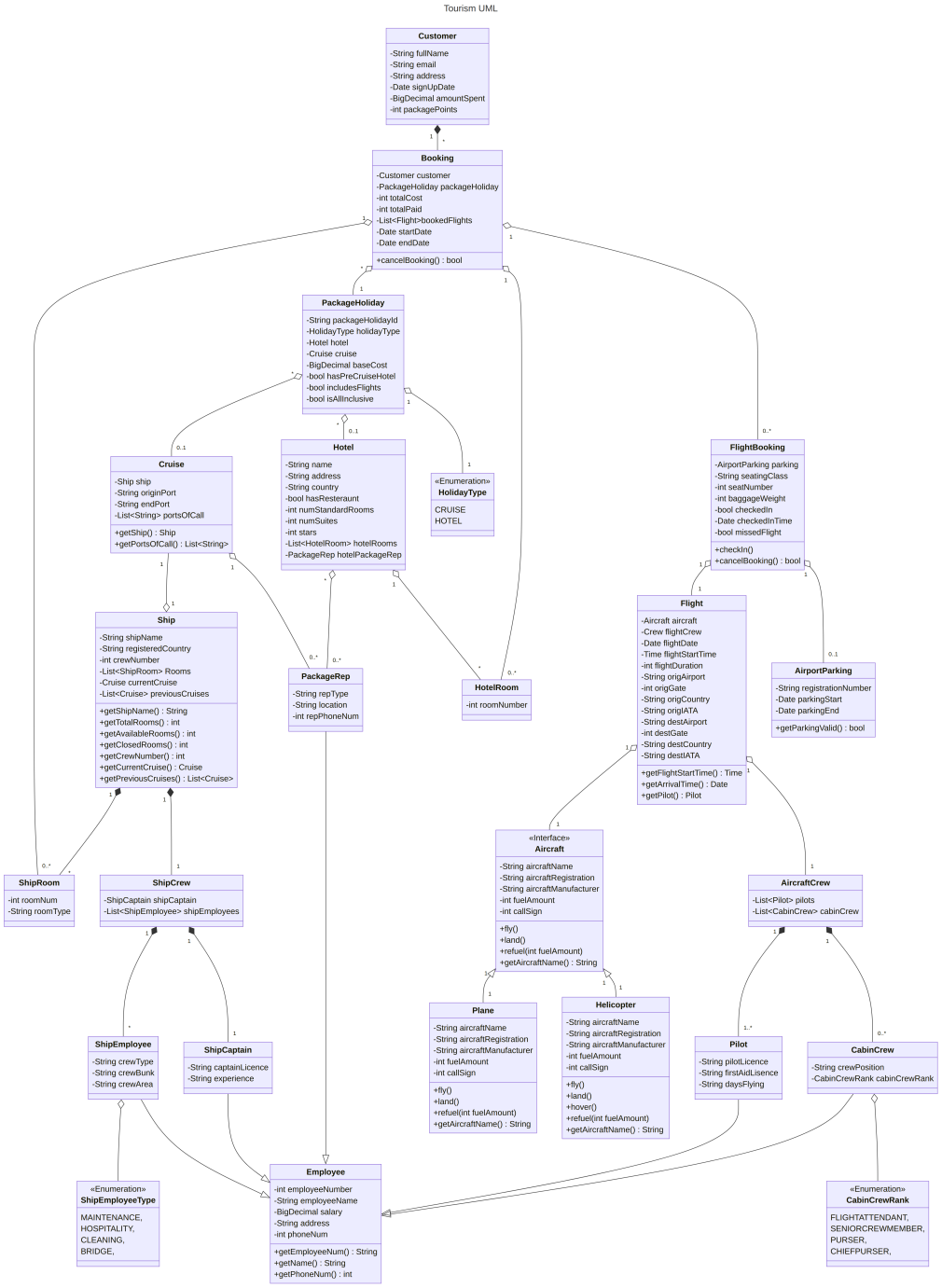

Figure 1: UML Representation of Program

First areas of testing.

The first stages of testing will ocurr during the development and programming process. The project will make extensive use of unit tests; these tests involve writing isolated automated tests that target small sections of the program, known as units.

Creation of the unit test involves developing a criteria, refered to the test case.

The unit returns output that is compared to the test's criteria that is known to be correct by the developers.

Ensuring that these small units of the program correct

Testing lots of small units that integrate to form a complex program reduces the amount of uncertain variables compared to a large monolithic test of the end result.

Furthermore,

Within this project, the Java unit testing framework JUnit will be used.

Building on top of unit testing, continuous integration ensures that all developers within the organisation:

- feature flags
- ui testing
- test lab
- testing pyramid
- e2e testing - operational - max interaction - require running services - automated ui testing server
- integration testing - testing api code inteactions, database connection
- black box testing - less interaction -
- martin folwer testing articles

Should the service require any external API's. These could be tracking APIs offered by various airports, or apis offered by the FIA (British equiavalent) to ensure that planes are correctly en route. Then the implementation of contact testing can ensure that updates to API returns and intefaces are quickly recognised and corrected. To ensure that the service does not suffer for too long.

End to end user testing.

`JUnit5` test for checking in process in `FlightBooking.java`,

```java
public boolean checkIn() {
    if (LocalDateTime.now().isAfter(flight.getFlightStart())) {
        hasMissedFlight = true;
        return false;
    }
    hasCheckedIn = true;
```

```
7       checkInTime = LocalDateTime.now();
8       return true;
9   }
```

The unit test created to test the `checkIn` method in the `FlightBooking` class.

```
1   package usw.holidays;
2
3   import org.junit.jupiter.api.DisplayName;
4   import org.junit.jupiter.api.Test;
5   import static org.junit.jupiter.api.Assertions.assertEquals;
6
7   class FlightBooking {
8       private Flight testFlight;
9       private FlightBooking testFlightBooking;
10
11      @Test
12      @DisplayName("Check-in process with an on-time flight booking")
13      public void checkIn() {
14          LocalDateTime earlyDateTime = new LocalDateTime();
15          testFlight = mock(Flight.class);
16          when(testFlight.getFlightStart()).thenReturn(earlyDateTime);
17
18          testFlightBooking = new FlightBooking(
19              testFlight, "economy", 12, 200
20          );
21
22          assertTrue(testFlightBooking.checkIn());
23      }
24
25      @Test
26      @DisplayName("Check-in process with a late flight booking")
27      public void checkInLate() {
28          LocalDateTime lateDateTime = new LocalDateTime();
29          testFlight = mock(Flight.class);
30
31          when(testFlight.getFlightStart()).thenReturn(lateDateTime);
32
33          testFlightBooking = new FlightBooking(
34              testFlight, "economy", 12, 200
35          );
36
37          assertFalse(testFlightBooking.checkIn());
38      }
39  }
```