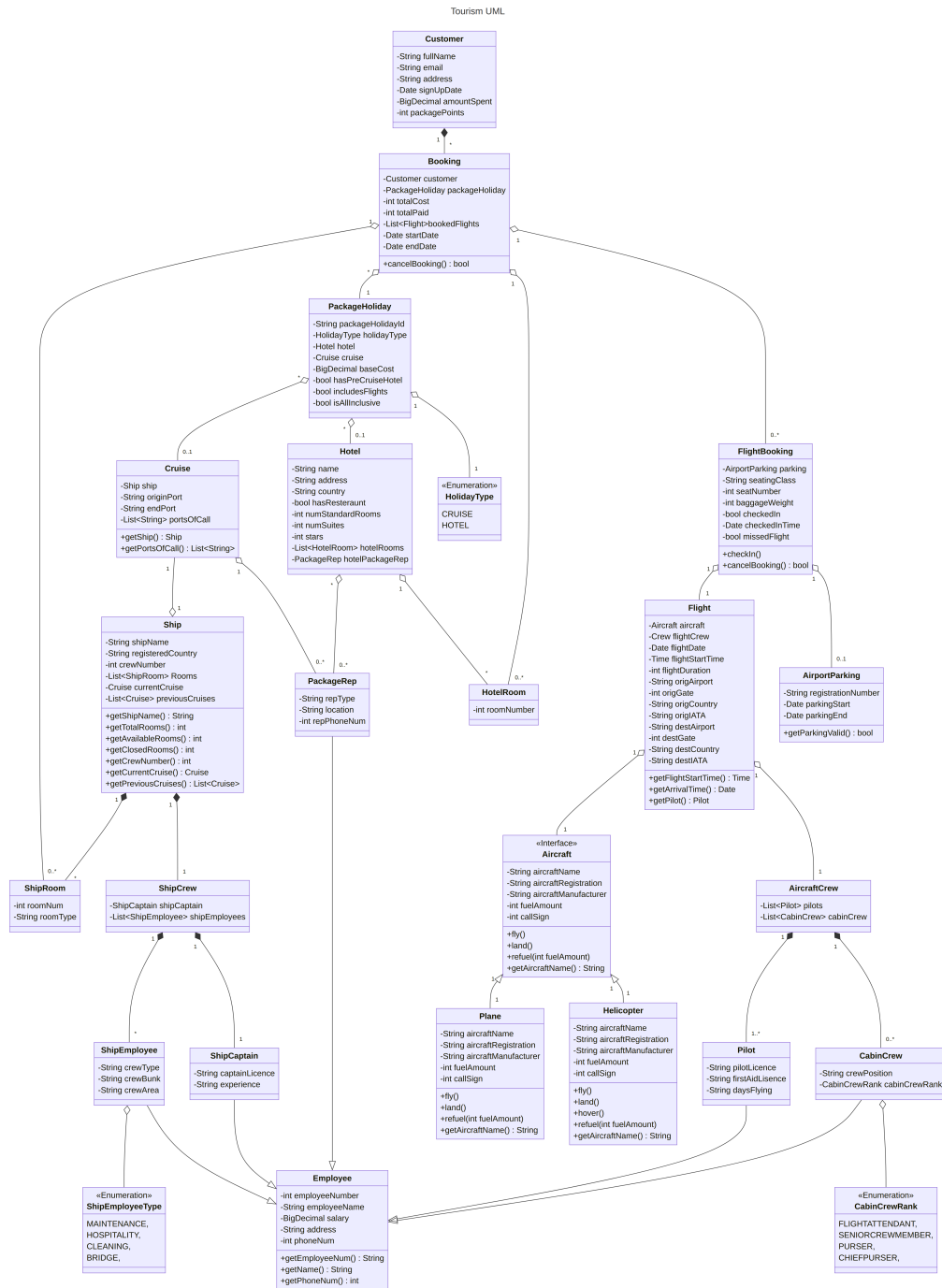


## **Contents**

<b>Program UML</b>	<b>2</b>
--------------------	----------

## Program UML



**Figure 1:** UML Representation of Program

First areas of testing.

The first stages of testing will occur 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, referred to as 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 are 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 interactions, 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 equivalent) to ensure that planes are correctly en route. Then the implementation of contract testing can ensure that updates to API returns and interfaces 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`,

```
1 public boolean checkIn() {
2     if (LocalDateTime.now().isAfter(flight.getFlightStart())) {
3         hasMissedFlight = true;
4         return false;
5     }
6     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.pop;
2
3  import org.junit.jupiter.api.DisplayName;
4  import org.junit.jupiter.api.Test;
5  import org.mockito.Mockito;
6
7  import java.time.LocalDateTime;
8
9  import static org.junit.jupiter.api.Assertions.assertFalse;
10 import static org.junit.jupiter.api.Assertions.assertTrue;
11
12 class FlightBookingTest {
13     private Flight testFlight;
14     private FlightBooking testFlightBooking;
15
16
17     @Test
18     @DisplayName("Check-in process with an on-time flight booking")
19     public void checkIn() {
20         LocalDateTime earlyDateTime = LocalDateTime.now().plusYears(1);
21         /*
22          * Create a mocked class of testFlight testFlight could use
23          * an API for plane tracking, so creating a stub ensures that
24          * the test is independent
25          */
26         testFlight = Mockito.mock(Flight.class);
27         // Set what to return when getFlightStart() is called
28         Mockito.when(testFlight.getFlightStart()).thenReturn(
29             earlyDateTime
30         );
31
32         // Call the check-in process
33         testFlightBooking = new FlightBooking(
34             testFlight, "economy", 12, 200
35         );
36         assertTrue(testFlightBooking.checkIn());
37     }
38
39     @Test
40     @DisplayName("Check-in process with a late flight booking")
41     public void checkInLate() {
42         LocalDateTime lateDateTime = LocalDateTime.now().minusYears(1);
43
44         // Create another mocked class
45         testFlight = Mockito.mock(Flight.class);
```

```
46      // Set return time to the future to fail check-in
47      Mockito.when(testFlight.getFlightStart()).thenReturn(
48          lateDateTime
49      );
50
51      // Call the check-in process
52      testFlightBooking = new FlightBooking(
53          testFlight, "economy", 12, 200
54      );
55      assertFalse(testFlightBooking.checkIn());
56  }
57 }
```