Mæli með að skoða git repoið frekar: https://github.com/KristinnRoach/Junit-testing-4T
Var bara hægt að skila pdf þannig hér er það:

```
package Hotel;

import java.time.LocalDate;

public interface searchInterface {
   HotelRoom[] searchHotels(String location, LocalDate arrival, LocalDate
departure, Integer maxPrice);
}
```

```
package Hotel;

import java.time.LocalDate;
import java.util.ArrayList;
import java.util.List;

// Mock object for HotelController.getSearchResults
// returns search results matching the input criteria for location and
dates

public class Mock implements searchInterface {
   @Override
   public HotelRoom[] searchHotels(String location, LocalDate arrival,
LocalDate departure, Integer maxPrice) {
       LocalDate[] dates = getTimeframe(arrival, departure);
       HotelRoom[] rooms = {
               new HotelRoom("Nonna Hótel", location, 3, dates, 333),
               new HotelRoom("Palla Pleis", location, 1, dates, 222),
               new HotelRoom("Tótu Mótel", location, 7, dates, 333)
       };
       return rooms;
   }

   public static LocalDate[] getTimeframe(LocalDate start, LocalDate end)
{
       long numOfDays = start.until(end.plusDays(1)).getDays();

       List<LocalDate> dateList = new ArrayList<>();

       for (int i = 0; i < numOfDays; i++) {
```

```
            LocalDate currentDate = start.plusDays(i);
            dateList.add(currentDate);
        }
        return dateList.toArray(new LocalDate[0]);
    }
}
```

```
package Hotel;

import java.time.LocalDate;
// Mock object for HotelController.getSearchResults
// returns an empty array
public class MockEmpty implements searchInterface {

    @Override
    public HotelRoom[] searchHotels(String location, LocalDate arrival,
LocalDate departure, Integer maxPrice) {
        return new HotelRoom[0];
    }
}
```

```
package Hotel;

import java.time.LocalDate;
// Mock object for HotelController.getSearchResults
// returns null
public class MockNull implements searchInterface {

    @Override
    public HotelRoom[] searchHotels(String location, LocalDate arrival,
LocalDate departure, Integer maxPrice) {
        return null;
    }
}
```

```java
import Hotel.*;
import org.junit.Test;
import java.time.LocalDate;
import static org.junit.jupiter.api.Assertions.*;

public class HotelControllerTest {

    @Test
    public void testGetSearchResultsArrivalBeforeDeparture() {
        searchInterface mock = new Mock();
        HotelController hotelController = new HotelController(mock);

        // Test data with incorrect dates (departure before arrival)
        LocalDate arrival = LocalDate.of(2024, 4, 7);
        LocalDate departure = LocalDate.of(2024, 4, 1);
        String location = "New York";
        Integer maxPrice = 3333;

        assertThrows(IllegalArgumentException.class, () ->
hotelController.getSearchResults(location, arrival, departure, maxPrice));
    }

    @Test
    public void testGetSearchResultsMissingData() {
        searchInterface mock = new Mock();
        HotelController hotelController = new HotelController(mock);

        // Test data with null value for departure
        LocalDate arrival = LocalDate.of(2024, 4, 7);
        LocalDate departure = null;
        String location = "New York";
        Integer maxPrice = 3333;

        assertThrows(IllegalArgumentException.class, () ->
hotelController.getSearchResults(location, arrival, departure, maxPrice));
    }

    @Test
    public void testGetSearchResultsWithEmptyResults() {
        // Mock object returning empty array
        searchInterface mockEmpty = new MockEmpty();
        HotelController hotelController = new HotelController(mockEmpty);

        // Test data
        LocalDate arrival = LocalDate.of(2024, 4, 1);
        LocalDate departure = LocalDate.of(2024, 4, 7);
        String location = "New York";
```

```
        Integer maxPrice = 200;

        // Call the method under test
        HotelRoom[] searchResults =
hotelController.getSearchResults(location, arrival, departure, maxPrice);

        // Assertions
        assertNotNull(searchResults);
        assertEquals(0, searchResults.length);
    }

    @Test
    public void testGetSearchResultsWithNullResults() {
        // Mock object returning null
        searchInterface mockNull = new MockNull();
        HotelController hotelController = new HotelController(mockNull);

        // Test data
        LocalDate arrival = LocalDate.of(2024, 4, 1);
        LocalDate departure = LocalDate.of(2024, 4, 7);
        String location = "New York";
        Integer maxPrice = 200;

        // Call the method under test
        HotelRoom[] searchResults =
hotelController.getSearchResults(location, arrival, departure, maxPrice);

        // Assertions
        assertNotNull(searchResults);
        assertEquals(0, searchResults.length);
    }
    @Test
    public void testGetSearchResultsWithValidDataSorting() {
        searchInterface mockHotelInterface = new Mock();
        HotelController hotelController = new
HotelController(mockHotelInterface);

        // Test data
        LocalDate arrival = LocalDate.of(2024, 4, 1);
        LocalDate departure = LocalDate.of(2024, 4, 7);
        String location = "New York";
        Integer maxPrice = 7777;

        // Expected search results
        HotelRoom[] expectedSearchResults = {
                new HotelRoom("Palla Pleis", location, 1,
Mock.getTimeframe(arrival, departure), 222),
```

```
                new HotelRoom("Nonna Hótel", location, 3,
Mock.getTimeframe(arrival, departure), 333),
                new HotelRoom("Tótu Mótel", location, 7,
Mock.getTimeframe(arrival, departure), 333)
        };

        // Call the method under test
        HotelRoom[] searchResults =
hotelController.getSearchResults(location, arrival, departure, maxPrice);

        // Assertions
        assertEquals(expectedSearchResults.length, searchResults.length);
        for (int i = 0; i < expectedSearchResults.length; i++) {
            assertEquals(expectedSearchResults[i].getHotelName(),
searchResults[i].getHotelName());
            assertEquals(expectedSearchResults[i].getLocation(),
searchResults[i].getLocation());
            assertEquals(expectedSearchResults[i].getNrOfBeds(),
searchResults[i].getNrOfBeds());
            assertEquals(expectedSearchResults[i].getPricePerNight(),
searchResults[i].getPricePerNight());

assertEquals(expectedSearchResults[i].getAvailableDates().length,
searchResults[i].getAvailableDates().length);
            for (int j = 0; j <
expectedSearchResults[i].getAvailableDates().length; j++) {

assertEquals(expectedSearchResults[i].getAvailableDates()[j],
searchResults[i].getAvailableDates()[j]);
            }
        }
    }
}
```

```java
package Hotel;
import java.time.LocalDate;
import java.util.Arrays;
import java.util.Comparator;

public class HotelController {
   private final LocalDate currentDate = LocalDate.now();
   private final searchInterface hotelSearchObj;

   public HotelController(searchInterface hotelSearchObj) {
       this.hotelSearchObj = hotelSearchObj;
   }

   private boolean validateSearchData(LocalDate arrival, LocalDate
departure, String location) {
       if (arrival == null || departure == null || location == null) {
           throw new IllegalArgumentException("Valid arrival date,
departure date or location missing.");
       }
       if (arrival.isBefore(currentDate) || arrival.isAfter(departure) ||
arrival.isEqual(departure)) {
           throw new IllegalArgumentException("Arrival date should be
before departure date and the current date");
       }
       return true;
   }
// Returns an array of hotel rooms that match the search criteria, an
empty array if no results are found
   public HotelRoom[] getSearchResults(String location, LocalDate arrival,
LocalDate departure, Integer maxPrice) {
       HotelRoom[] searchResults = null;
       if (validateSearchData(arrival, departure, location)) {
           searchResults = hotelSearchObj.searchHotels(location, arrival,
departure, maxPrice);
       }
       if (searchResults == null || searchResults.length == 0) {
           System.out.println("No available hotels found");
           return new HotelRoom[0];
       }
       // Sorts first by price, then hotel name alphabetically
       Arrays.sort(searchResults,
Comparator.nullsLast(Comparator.comparing(HotelRoom::getPricePerNight))
               .thenComparing(HotelRoom::getHotelName));

       System.out.println(Arrays.toString(searchResults)); // henda
       return searchResults;
   }
```

```
}
package Hotel;

import java.time.LocalDate;
import java.util.Arrays;

public class HotelRoom {

/* GETTERS */

public String getLocation() {
   return location;
}

public Integer getPricePerNight() {
   return pricePerNight;
}

public String getHotelName() {
   return HotelName;
}

public int getNrOfBeds() {
   return nrOfBeds;
}

public LocalDate[] getAvailableDates() {
   return availableDates;
}

/* FIELDS */
   private String HotelName;
   private String location;
   private int nrOfBeds;
   private Integer pricePerNight;  // Integer (ekki integer) leyfir null
values
   private LocalDate[] availableDates;

   /* Constructor */

   public HotelRoom(String HotelName, String location, int nrOfBeds,
LocalDate[] availableDates, Integer pricePerNight) {
       this.HotelName = HotelName;
       this.location = location;
       this.nrOfBeds = nrOfBeds;
       this.pricePerNight = pricePerNight;
       this.availableDates = availableDates;
```

```
    }

    @Override
    public String toString() {
        return "HotelRoom{" +
                "HotelName='" + HotelName + '\'' +
                ", location='" + location + '\'' +
                ", nrOfBeds=" + nrOfBeds +
                ", pricePerNight=" + pricePerNight +
                ", availableDates=" + Arrays.toString(availableDates) +
                '}';
    }
}
```