SYSC 2004 Object-0riented Software Development

Lab 7

Lab 7:

Background Reading

- *Objects First with Java*, Chapters 1-4, 5.4, and 6-8.

Objective

The objective of this lab is to have you become more familiar with `JUnit` testing.

Some of you may have noticed in Lab 4 and Lab 5 that the unit tests passed when your code had errors. In this lab, you are going to improve the *train-v3* project by making the unit tests more complete.

Getting Started

1. Download file `train-v3.zip` from cuLearn. Save the file to the desktop.
2. Locate your train-v2 project folder from Lab #5 and save that folder on the desktop.
3. Right-click on the `train-v3.zip` folder and select Extract All... to extract all the files into a folder called `train-v3`.
4. Launch BlueJ.
5. Open the *train-v3* project in BlueJ.
6. You need to include your work from Lab 5 in this project:
    1. The cars in the train are modelled by class `Car`, which you implemented in Lab 4, and then copied to Lab 5. Here is how to add a copy of this class to the *train-v3* project:
        - From the BlueJ main menu, select Edit > Add Class from File... A dialogue box will open.
        - Browse to the `train-v2` folder that contains your work from Lab 5. Select class `Car`, and click the Add button. A copy of the class will be placed in the *train-v3* project. (This will not delete the `Car` class from your *train-v2* project, and any changes that you make to `Car` in *train-v3* will not affect the original class in *train-v2*.)
    2. Repeat the above steps for class `Train`, which you implemented in Lab 5.

Part 1: Improving Class `SeatTest`

1. Change `testCreateSeat()` to ensure that all prices are within 50c accuracy (i.e. no more than 50c higher or lower than the expected value).
2. In `testBooking()` perform the same tests on `seat2` that were performed on `seat1`. Ensure that `seat2` has a different seat number and cost from `seat1`.
3. Make the same changes to `testCancelBooking()`.
4. Check that your code is nicely formatted and well commented. Generate your javadoc to check that it is complete and correct.
5. Get Part 1 checked by a TA.

Part 2: Improving Class `CarTest`

1. Add to `testBookNextSeat()` so that all the current tests are done for an economy car (as well as the current business car).
2. Make the same changes to `testCancelSeat()`.
3. Also modify `testCancelSeat()` to try to cancel a booking of seat 50 in each of the cars. (Note: There is no seat 50 in either car, so the bookings should be unsuccessful.)
4. Check that your code is nicely formatted and well commented. Generate your javadoc to check that it is complete and correct.
5. Get Part 2 checked by a TA.

Part 3: Improving Class `TrainTest`

1. Modify `testAddCar()`:
    - Add a fourth car (a business class car) to `testAddCar()` and ensure that it functions properly.
    - Rewrite the code to remove the variable "aCar" as it is not needed.
2. Modify `testIssueTicket()` so that it also uses a fourth car:
    - Business seat booking:
        - Fill all the seats in both business cars (instead of just one).
        - As before, check that another business seat cannot be booked.
        - Check that all seats in the 2nd business car are also booked.
    - Economy seat booking:
        - At the end of the test check that the second business car seats are still all booked.
    - Now rewrite the code so that you can remove the boolean "result" as it's not needed.

3. Modify `testCancelTicket` so that it also uses a fourth car:
    - Create a train with four cars (two business and two economy) as in the previous methods.
    - Fill the seats in the first business car and half the second business car, but just one of the economy cars.
    - In addition to the other existing checks:
        - cancel one of the booked seats near the front half of the second business car and ensure that functions properly, and
        - attempt to cancel one of the unbooked seats near the back of the second business car and ensure that functions correctly.
    - Now rewrite the code so that you can remove the boolean "result" as it's not needed.
4. Add a new test method `testBookCancelTicket` which is to be used to check that we can properly book a ticket after it has been cancelled:
    - Ensure that you avoid using a boolean "result".
    - Create a train with four cars (two business and two economy) as in the previous methods.
    - As in `testCancelTicket` fill the seats in the first business car and half the second business car, but just one of the economy cars.
    - Cancel any three tickets in the economy car.
    - Book four economy tickets and check that you get each of the three cancelled (starting with the one nearest the front of the car), and then, for the fourth, the first seat in the second economy car.
    - Cancel three business car tickets: one in the first business car and two near the front of the second.
    - Book four business tickets and check that you get each of the three cancelled (starting with the one in the first business car), and then, for the fourth, seat 16 of the second business car.
    - Ensure that you avoided using a boolean "result".
5. You should have noticed by now that the first part most methods is identical.
    - Move the repetitive code to `TrainTest`'s constructor. Note that you will need to add five fields: aTrain (type `Train`) and car1, car2, car3, and car4 (type `Car`). The `TrainTest` constructor will create the train and the four cars, and add the four cars to the train.
    - Method `testCreateEmptyTrain` is unchanged, although renaming the `Train` object to emptyTrain makes the code a little clearer.
    - The other methods can now be shortened.

6. Check that your code is nicely formatted and well commented. Generate your javadoc to check that it is complete and correct.
7. Get Part 3 checked by a TA.
