COMP311 Assignment 2: Debugging exercise

The purpose of this assignment is to experience using debugging tools that are built into an IDE. Any IDE worth using has a debugger. Every debugger is a bit different but all have some common features and most work in roughly similar ways.

You are given some Java code that is part of an application to sell tickets for an airline flight. You use the Debug and Java perspectives of Eclipse to locate and fix problems in the code. Work in groups of two, preferably one stronger in java coding and the other weaker (each one on its own eclipse workspace; the stronger has to help the weaker with concepts etc, but should not write and complete steps for him/her). Work together with answers of the questions and submit one answer per group.

Due

Complete this assignment in class. Answer the questions on questions page, sign the questions page and hand it in before you leave the lab session.

Marks

The assignment is marked out of 10.

Description of the application

The code is a slice of a larger application that sells tickets for airline fights. The first slice developed includes only the classes required to perform the very specific tasks of issuing a ticket, assigning a seat to a passenger, and building up the manifest (or list of passengers) on a specific flight.

Your starting position is a prototype that has a very simple command line interface so that the complications of a Web application or graphical user interface are eliminated. Most of the classes and associations between classes are shown in the class diagrams on page 3.

Business rules

- There are two classes of seats: Economy and Business.
- A business class ticket costs $750 and an economy seat costs $500.
- Passengers may be frequent flyers, airline employees (but not both) or neither.
- Airline employees get a 50% discount on economy seats, but not on business class seats.
- Frequent flyers get no discount but have privileges outside the scope of this slice of the application.

Notes

- For this slice of the application, passengers have already selected a specific flight and day.
- Invoicing and payment processing are handled by a separate subsystem.
- For simplicity, the test airplane has very limited seating of only 14 seats.
  Row 1 is business class and has one seat on either side of the aisle: 1A and 1B. Rows 2, 3 and 4 are economy class and have two seats on either side of the aisle: A, B, C, and D.
- No persistent data (databases) is used.
- All output goes to the console (command line I/O). The placeholder classes **Manifest** and **UserPrompter** will be replaced by a GUI user interface at a later time in development.

COMP311 Assignment 2: Debugging exercise

What the provided code should do (requirements):

- The **Manifest** class is the main program for this slice of the application.
- User interface asks each passenger whether (s)he want to purchase a ticket. If the answer is "no" the program stops as this situation simulates end of ticket sales.
- The UI first asks whether the passenger wants business class and then, if no, economy class
- The UI asks for the passenger name and whether the passenger is a frequent flyer or member of the airline staff.
- At this point, the UI invokes the core business logic to sell a ticket.
- The application should check whether a suitable seat is available. Passengers cannot choose their seat: the application assigns one. It issues a ticket and calculates the ticket price. Then it prints out the ticket details.
- Before ending, the **Manifest** class prints a list of all seats followed by the name of the passenger who owns a ticket for that seat or the word "Available" if the seat was not sold.

Notes:
The seating plan is generated by the constructor of the **SeatingPlan** class and can be modified easily by changing the enumeration type **SeatingClass**. You can assume these two classes have already been tested and have no errors. In other words, don't worry if you don't understand their code.

Instructions

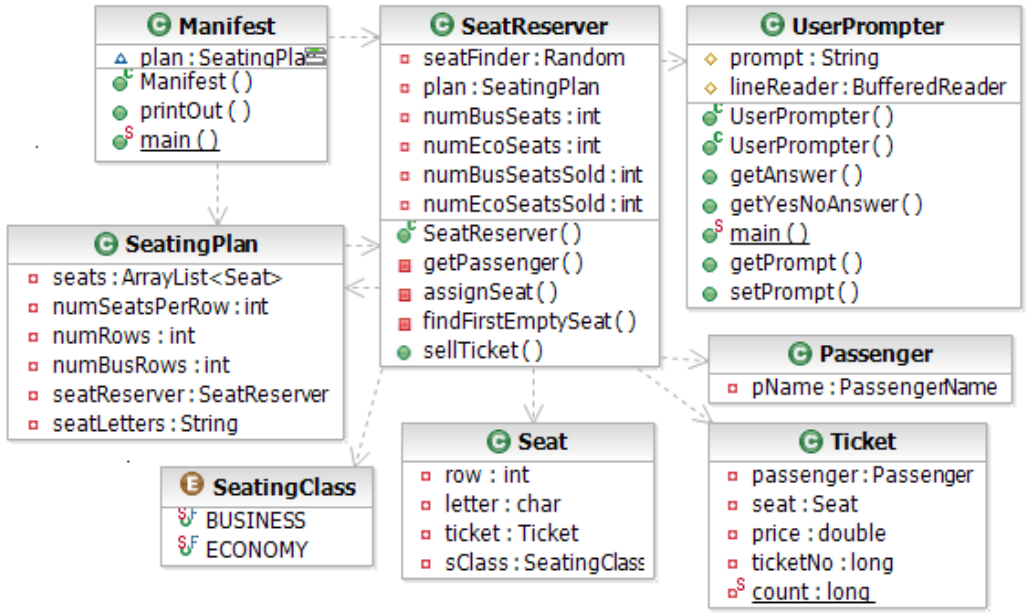
You are given a JAR that contains source code.

1. Import the Java source into Eclipse. Follow instructions from the instructor if you are not familiar with Eclipse
2. Run the program a few times to see how it works. You should see clear evidence of bugs.
3. Use the debugger to trace what is going wrong. Resist the temptation to just read the code looking for bugs. The instructor will walk you through using the debugger to determine what code causes at least one of the problems.
4. Continue to debug and fix bugs as you go until you are satisfied you have done all that you can with this code.
5. Answer the questions on the questions page.
6. Hand in your answers when you are done (or submit to eCentennial dropbox folder).

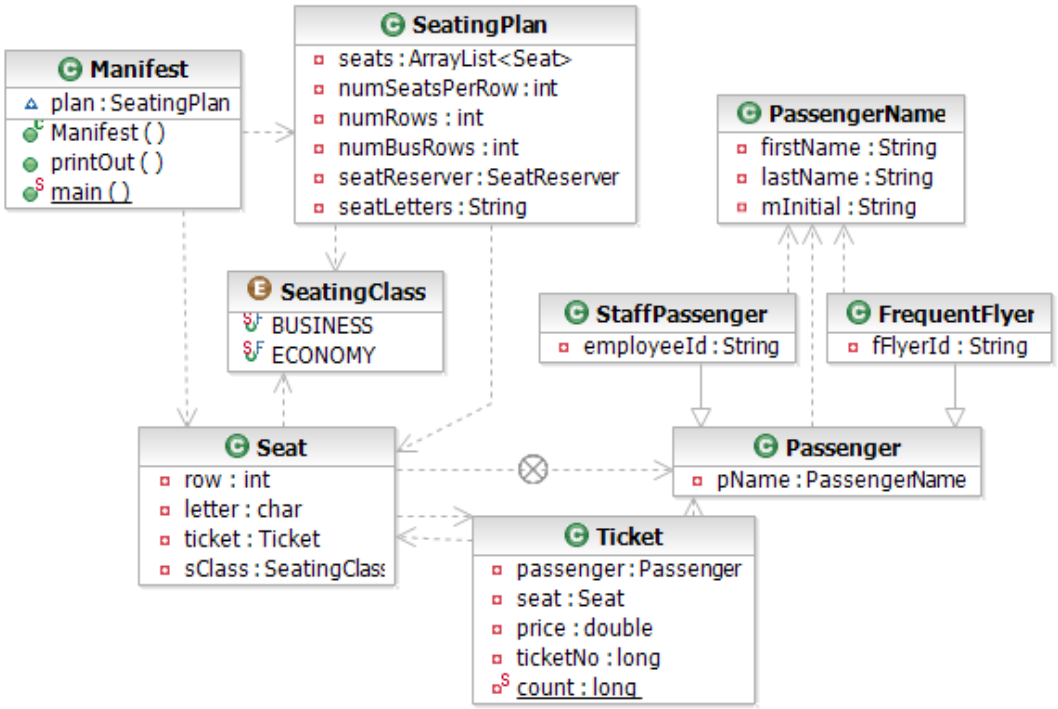
COMP311 Assignment 2: Debugging exercise

The JAR file contains 12 types, as shown in the following partial UML class diagrams.

UML class diagram 1 of air ticketing example:



UML class diagram 2 of air ticketing example:



Not shown in these diagrams is the interface **Discountable**, implemented by the class **StaffPassenger**.