Flight Scheduler Initial Phase - Programming Assignment 6

You have been asked to develop a Flight Scheduling application for the Fly-By-Night Airline. The Airline has one or more flights per day but they are not by time. Every Flight will be available for each day the airline flies. The customer gets booked on a specific flight for a specific day. Each Flight has a name and a number of seats on the flight. Each date is just a specific day. The Customer is identified by a first and last names.

This application should have a very nice GUI interface and will be a database driven application. The database used will be Derby. This application must use good Object Oriented Design and Programming. The database must use good Object-Oriented Design and Programming. There is a very close correlation between Object-Oriented Design and Database Design. Your application design should include at least four classes besides the main GUI class, e.g. Customer class, Flight class, Day class..., Bookings class, etc. Your database accesses should be in the classes that correlate with the database tables.

This assignment is the first half of the final project and will be submitted as Programming Assignment 6. This phase of the project will implement the following user commands:

Add Customer

The customer is added to the database. The customer is identified by first and last name.

Book Customer Day Flight

The customer will be assigned the flight for the requested day, if there are seats available. If seats are not available, the customer will be put on the wait list for that flight. The waiting list must be maintained in the order the customers tried to book their flights.

Status Flight Day

The Status command for flight and day will display the customers that have been booked for that flight on that day.

Status Waiting List by Day

The Status command for the Waitlist will display all the customers waiting for flights on the specified day.

Database considerations:

The Flight Table should be preloaded with several flights such as F101, G102 and the number of seats on the flight. Two seats per flight would be suggested to make testing easier.

The Day Table should be preloaded with several days of your choice.

The database tables should not contain redundant data, i.e. relevant data should only appear in one table.

GUI Guidelines:

The user should be required to enter only unknown data. Drop down lists of known data such as Customer names, Flight names, or Days should be displayed for the user to select from. Combo Boxes should be used for the drop down lists on the form. When information is requested to be displayed e.g. for a Status command, all of the requested information must be displayed. When a command is performed, the results of that command should be displayed to the user on the same display without the user needing to check Status to see what was done.

Submission Guidelines:

Don't forget to submit your PROJECT files and your DATABASE files. The database folder is probably located under "Users/USERNAME/.netbeans-derby/NAME_OF_DATABASE".
Zip the ENTIRE database folder and the ENTIRE project folder and submit the two zipped files in the dropbox under one submission.

Note:
Your project must be created with the name 'FlightSchedulerNameID', where Name is your name and ID is your psu account id, e.g. xxx1234. The database must be created with the name 'FlightSchedulerDBNameID', and a username and password of java and java. The database when submitted for PA 6, should contain data for the flights and days tables. All other tables should be empty.

Grading Criteria:

In this project I will be looking for good OO design practices and this includes:

- **Use of getter and setter methods for class variables**
- **Good naming of your classes, methods and variables**
- **Correct use of static and non-static methods**
- **The way you split this project into classes.**
- **All of your updates to the database must be done using SQL statements, do not use ResultSetTableModels to update the database.**

- **If a SQL statement to update the database needs to contain a variable, then you must use PreparedStatements, do not use concatenation of strings to create the SQL statement.**

Note: A good example to use for how to program this assignment is contained in the programming examples in Figures 24.30, 24.31 and 24.32 in Chapter 24 from our book.

Flight Scheduler Final Phase

You have been asked to develop a Flight Scheduling application for the Fly-By-Night Airline. The Airline has one or more flights per day but they are not by time. Every Flight will be available for each day the airline flies. The customer gets booked on a specific flight for a specific day. Each Flight has a name and a number of seats on the flight. Each date is just a specific day. The Customer is identified by first and last names.

This application should have a very nice GUI interface and will be a database driven application. The database used will be Derby. This application must use good Object Oriented Design and Programming. The database must use good Object-Oriented Design and Programming. There is a very close correlation between Object-Oriented Design and Database Design. Your application design should include at least four classes besides the main GUI class, e.g. Customer class, Flight class, Day class..., Bookings class etc. Your database accesses should be in the classes that correlate with the database tables.

This assignment is the final half of the project. The final project is a continuation of Programming Assignment 6. This phase of the project will additionally implement the following user commands:

Add Flight Seats

Add a new flight to the system. The Flight name is a string and Seats is the number of seats in the flight.

Cancel Customer Day

The booking for that Customer and Day must be removed from the flight's bookings or the waiting list. If the booked entry is removed from a flights bookings, the waiting list must be checked to determine if another customer can be booked with that flight for that day.

Add Day

Add a new Day for flights to the system.

Status Customer

The Status command for a customer will display the flight and day for each flight the customer has booked and/or is waitlisted for.

Drop Flight

The Drop command must remove a flight from the application. Any customers that have been booked for this flight on any day must be rebooked with another flight for that day if possible in priority sequence and the rebooking reported to the user. If the

customer cannot be rebooked, the user is informed that the customer could not be rebooked. Any customers on the waitlist for that flight must also be deleted

Database considerations:

The Flight and Day Tables no longer need to be preloaded with values. When submitted, all database tables should be empty.
The database tables should not contain redundant data, i.e. relevant data should only appear in one table.

GUI Guidelines:

The user should be required to enter only unknown data. Drop down lists of known data such as Flight names or Days should be displayed for the user to select from. Combo Boxes should be used to categorize data on the form. When information is requested to be displayed e.g. for a Status command, all of the requested information must be displayed. When a command is performed, the results of that command should be displayed to the user on the same display without the user needing to check Status to see what was done.

Submission Guidelines:

Don't forget to submit your PROJECT files and your DATABASE files. The database folder is probably located under "Users/USERNAME/.netbeans-derby/NAME_OF_DATABASE".
Zip the ENTIRE database folder and the ENTIRE project folder and submit the two zipped files in the dropbox under one submission.

Note:
Your project must be created with the name 'FlightSchedulerNameID', where Name is your name and ID is your psu account id, e.g. xxx1234. The database must be created with the name 'FlightSchedulerDBNameID', and a username and password of java and java. When your database is submitted, all tables should be empty.

Grading Criteria:

In this project I will be looking for good OO design practices and this includes:

- **Use of getter and setter methods for data**
- **Good naming of your classes, methods and variables**
- **Correct use of static and non-static methods**
- **The way you split this project into classes**
- **All of your updates to the database must be done using SQL statements, do not use ResultSetTableModels to update the database.**

- **If a SQL statement to update the database needs to contain a variable, then you must use PreparedStatements, do not use concatenation of strings to create the SQL statement.**