W. P. CAREY
SCHOOL *of* BUSINESS
ARIZONA STATE UNIVERSITY

CIS 494 – Business Systems Development with Java – Spring 2015

Assignment 5

Due on Blackboard: Friday, April 8 by 11:59 PM

Skills Developed: Data structures, priority queues, implementing interfaces, Comparators

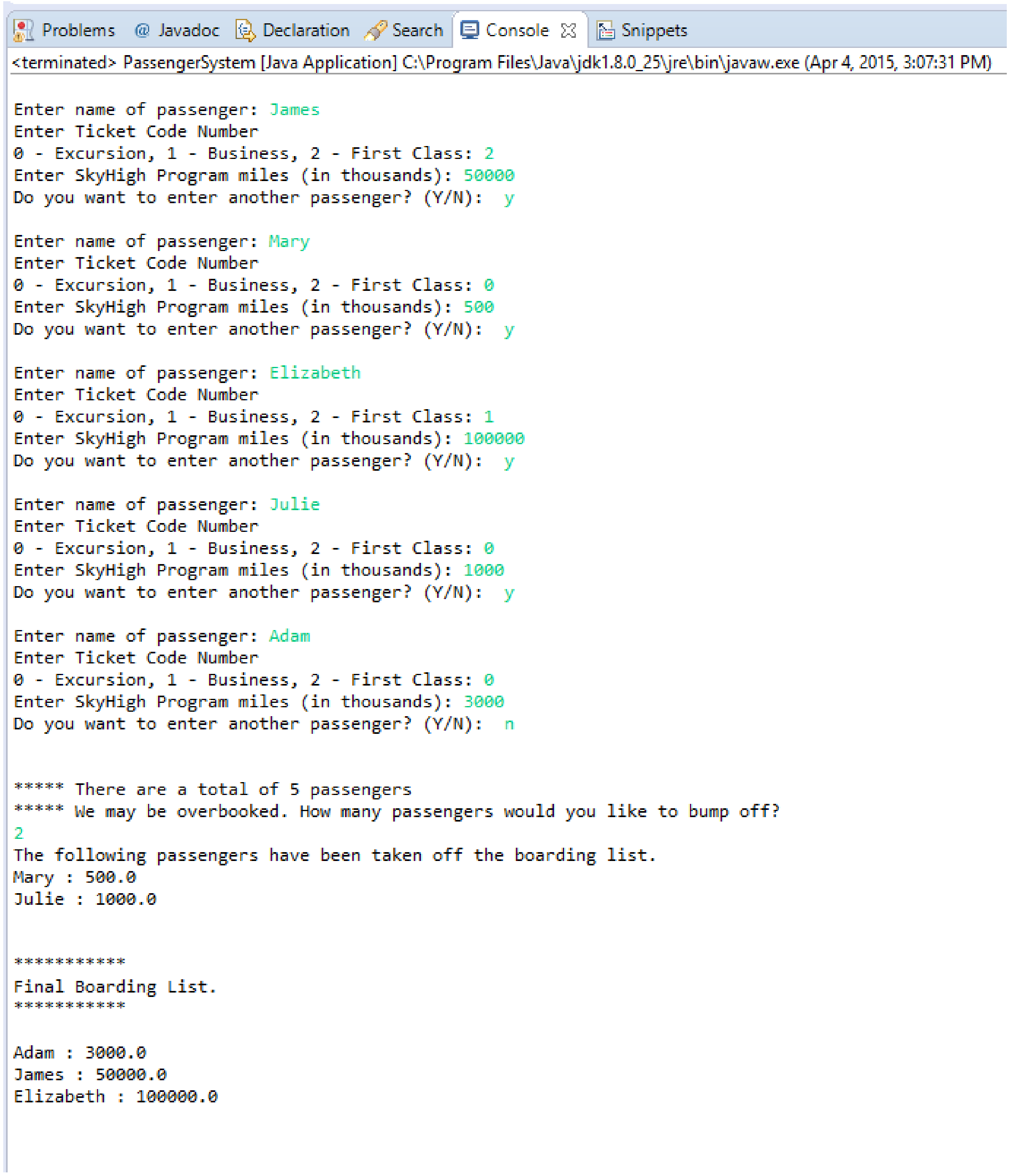
Design a Passenger Management System that allows users to add Passengers to a flight Manifest. Passengers have a name, ticket class (excursion, business, first class), and accumulated miles. The Manifest should be implemented as a Priority Queue. A Comparator object should be able to compare two given passengers and determine which passenger has lower boarding priority.

Boarding priority is determined by comparing the ticket classes. First class has a higher boarding priority than business class. Business class has a higher boarding priority than the excursion class (popularly known as "economy"). If passenger ticket classes are equal, the passenger with higher miles has the higher boarding priority.

Ask the users how many people should be "bumped off." The priority queue should automatically select the right people to be bumped off. Display the final passenger list to the screen.

This application can be either a console application or a JavaFX GUI application.

Sample Output



```
<terminated> PassengerSystem [Java Application] C:\Program Files\Java\jdk1.8.0_25\jre\bin\javaw.exe (Apr 4, 2015, 3:07:31 PM)

Enter name of passenger: James
Enter Ticket Code Number
0 - Excursion, 1 - Business, 2 - First Class: 2
Enter SkyHigh Program miles (in thousands): 50000
Do you want to enter another passenger? (Y/N):  y

Enter name of passenger: Mary
Enter Ticket Code Number
0 - Excursion, 1 - Business, 2 - First Class: 0
Enter SkyHigh Program miles (in thousands): 500
Do you want to enter another passenger? (Y/N):  y

Enter name of passenger: Elizabeth
Enter Ticket Code Number
0 - Excursion, 1 - Business, 2 - First Class: 1
Enter SkyHigh Program miles (in thousands): 100000
Do you want to enter another passenger? (Y/N):  y

Enter name of passenger: Julie
Enter Ticket Code Number
0 - Excursion, 1 - Business, 2 - First Class: 0
Enter SkyHigh Program miles (in thousands): 1000
Do you want to enter another passenger? (Y/N):  y

Enter name of passenger: Adam
Enter Ticket Code Number
0 - Excursion, 1 - Business, 2 - First Class: 0
Enter SkyHigh Program miles (in thousands): 3000
Do you want to enter another passenger? (Y/N):  n


***** There are a total of 5 passengers
***** We may be overbooked. How many passengers would you like to bump off?
2
The following passengers have been taken off the boarding list.
Mary : 500.0
Julie : 1000.0


***********
Final Boarding List.
***********

Adam : 3000.0
James : 50000.0
Elizabeth : 100000.0
```

1. Create a Passenger class with 3 attributes: name, miles, and a ticketCode (integer). The ticketCode will have values from 0 to 2 indicating ticket class (excursion, business, and first class). Implement constructors as well as get/set methods.

2. Create a PassengerComparator class that compares two Passenger objects. It should implement the Comparator interface. The PassengerComparator compare method should implement the following rules:

Passenger1 is less than Passenger2 if Passenger1 is flying on a lower class ticket.

Passenger1 is greater than Passenger2 if Passenger1 is flying on a higher class ticket.

If both Passenger1 and Passenger2 are flying on tickets of equal class, then Passenger passengers will be compared based on accumulated miles. Again, if Passenger1's miles are lower, Passenger one is considered "lesser" in status.

If both Passenger1 and Passenger2 are traveling in the same ticket classes and have the same number of accumulated miles, then both are equal.

3. Implement a PassengerSystem class. PassengerSystem should have in it a PriorityQueue of Passengers called the passengerManifest.

- Create a method to add a Passenger.
  - This method should ask the end user about passenger details (name, miles, ticket code). It should instantiate a new Passenger and add it to the manifest.
- Create a method, RunPassengerSytem.
  - This method should keep calling the AddPassenger method to add new passengers so long as the user wants to add new passengers.
  - It should then tell the user that the flight may be overbooked and ask how many passengers should be bumped off.
  - Bump off as many passengers as specified by calling the appropriate methods of the PriorityQueue, passengerManifest.
  - Finally, print out a list of the final manifest.

4. Call RunPassengerSystem from your main method!

<u>Submitting Files</u>

Submission should be made using a zip file that contains the entire Eclipse project folder. You will need to ***zip the entire project folder***. The folder will automatically contain the class source files as well as the compiled .class files.

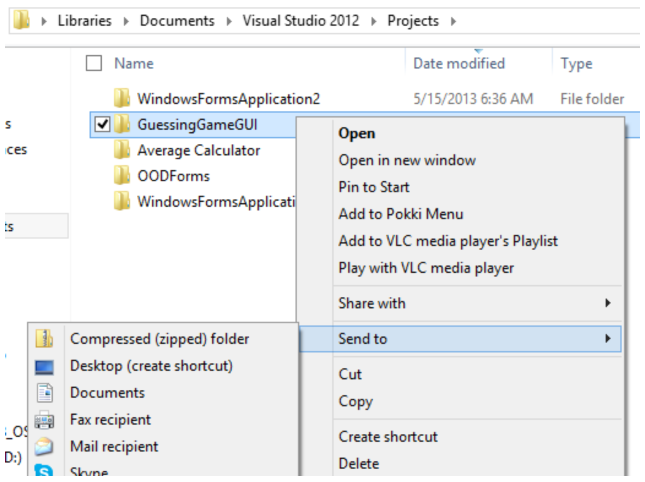
ZIP file should be named: **AX.zip or IC.zip**

> where X is the in-class assignment number, e.g., A1.zip is the submission file for Assignment 1 and IC1.zip is the submission file for InClass1.

Note: *The ZIP filename is <u>independent</u> of the Project name. Do not name your project A1.*

How to Properly ZIP and Submit Your Project Files:

- Go to the folder within {Eclipse Workspace}\
- ZIP the entire top-level folder for your project by right-clicking your project folder and selecting Send to | Compressed (zipped) folder.
- Finally, submit the ZIP file using the submission link on Blackboard by the due-date and time listed on the assignment. Upload the ZIP file.



Using built-in windows zip tools: http://windows.microsoft.com/en-us/windows/compress-uncompress-files-zip-files

Verify your files BEFORE and AFTER submission:

- Check for actual class files being present in the folder before you zip it.
- ***Ensure that you are not zipping a short-cut to the folder.***
- After zipping, check file size. A file size under 4K likely does not contain all the files.
- Unzip, extract all files, and verify you see actual files, not a solitary short-cut.
- Uncompress your zip file before submitting and verify that files are present.
- Make sure you have ***submitted*** your file and not just saved a draft on Blackboard. ***A blue clock indicates a submission in progress, i.e. a draft, not a submission. The draft is accessible only to you. You will get a ZERO if you only ever save a draft on Blackboard and never submit your files.***

- Download your zip file after submitting, uncompress, and again verify that your files are present. Test your files in Visual Studio after uncompressing.

This takes an extra couple of minutes. Please do it if your grade is important to you. If you do this, you will not end up submitting a bad file. If you submit an empty file, or one containing only a shortcut, or a bad zip file, or a bad project file, you will receive a score of zero and your only recourse will be to do the makeup assignment at the end of the semester.