Assignment #2
COMP 303
Well-Formed Networks of Objects
Due: October 26th 2016

Question 1: UML Sequence Diagrams

Parts A

Computer networks use a Sender function to send packets from a source computer into the network. A Receiver function, at the destination computer, listens to the network for packets. The sender sends and the receiver receives, by this technique we have communication.

The Sender function works this way: 1- it receives a string to transmit as a parameter, 2- it listens to the network waiting for it to be quiet, 3- it transmit the string, 4- it sets a timer for n milliseconds, 5- it waits for a confirmation packet from the destination computer informing of successful delivery, 6- if confirmation arrives in time it returns true else it returns false, end.

The receiver function works this way: 1- it begins in a busy loop waiting for a packet, 2- when a packet arrives it saves it in a buffer, 3- it waits for the network to be quiet, 4- then it transmits a confirmation packet, 5- it does not wait for anything else it simply returns the packet and ends.

Draw two sequence diagrams, one for the Sender function with the Internet and the other for the Receiver function and the Internet. Even though these are two separate diagrams, if placed side by side they should agree with each other demonstrating how the network communicates.

Part B

Simply in words, no diagrams, describe how the above sequence diagrams would need to change to allow for the case that the Sender wants to transmit m packets to the destination computer.

What to hand in:

- PDF file with the diagrams
- PDF for Part B, or combined with the diagrams PDF

Question 2: Well-formed Java Software

Write a well-formed Java program and submit the source files with supporting documentation. Do not provide a pdf for the source files.

Keep in mind the following well-formed concerns: abstraction, simplicity, optimality, encapsulation, information hiding, polymorphism, inheritance, interfaces, abstract methods, overriding methods, programming by contract, etc. You will need to choose what is important for this question. **You will be graded for your design choices**.

To help you plan the solution: **create a UML State diagram** detailing what you need to implement before you start programming. Cover all the cases. Make this State Diagram from the point of view of the programmer answering what needs to be implemented.

Here is the problem statement: Write a well-formed program that solves The Airplane Seat Reservation problem.

An aircraft has 200 seats; each row has 4 seats with an isle in the middle. In other words, two seats on the left side of the airplane a space and then two seats on the right side of the airplane. There are 50 rows.

Seat reservation is problematic because multiple users attempt to reserve seats at the same time and they are not aware of one another. Obvious conflicts can occur, for example, two users attempting to reserve the same seat at the same time.

For this question you will implement **three threads** that will attempt seat reservations. These threads will represent brokers who provide this service to customers. Two threads will be automated producer functions that randomly attempt seat reservations. The third producer will not be automated; instead it is a GUI Swing form that you will use to make manual reservations, as the third broker. Each producer has an ID number: 1, 2, and 3. These ID numbers will be recorded with the seat reservation so that we can know which producer did what.

The GUI displays the airplane seats together with the producer ID number and if the seat is available. To do this you have two choices: choice 1- draw a white box for each seat. An empty white box means the seat is available. A successfully reserved seat will have the producer's ID number written within the box. Choice 2- print a string of underline characters (IE. _) for each empty seat. Overwrite the underline with the producer's ID number to indicate that it is taken (in other words, not a graphic based solution). Use the row and column number method to identify which seat you are interested in. To summarize, this swing GUI displays the reservation form and the airplane seating availability “diagram”.

The remaining design is up to you, but it must be optimal, well-formed, and its source code complexity should be minimized.

One last requirement, use Javadoc to both comment the source code and generate an HTML document describing your program.

Things your program must use:

- Commenting using javadoc
- Programming by contract

What to hand in:

- Java source code with Javadoc comments
- The Javadoc generated html files for your program
- UML State diagram, as a PDF.

Grading comments:

- Your design choices make up a large portion of this grade
- Optimal, well-formed, and its source code complexity should be minimized

Points Awarded:

Question 1 Part A : 6 points

- +2 transmitter
- +2 receiver
- +2 proper UML

Question 1 Part B : 4 points

- +1 Code reuse
- +1 Well-formed techniques
- +1 Good solution
- +1 Solution was expressed well

Question 2 : 10 points

- +1 UML State Diagram
- +1 Javadoc
- +1 Contract programming
- +1 Well-formed code
- +1 Follow instructions
- +1 Two automated threads
- +1 One manual thread
- +1 GUI of seats
- +1 Optimal
- +1 Simple / minimized code