CSE445/598 Project 2 (Assignments 3 and 4) – (50+50 Points)
Fall 2018

Project (both assignment 1 and assignment 3) due: Saturday, September 22, 2018, by 11:59pm (Arizona Time), plus a grace period of 24 hours.

Introduction

The purpose of this project is to make sure that you understand and are familiar with the concepts covered in the lectures, including distributed computing, multithreading, thread definition, creation, management, synchronization, cooperation, event-driven programming, client and server architecture, service execution model of creating a new thread for each request, the performance of parallel computing, and the impact of multi-core processors to multithreading programs with complex coordination and cooperation. Furthermore, you are able to apply these concepts in a programming project.

You can choose to do this project as an **individual project** or as a **team project** of two or three members. In the case of the team project, a declaration must be given at the end of the assignment, which identifies the parts of individual contributions and team efforts. An overall percentage of contribution of each member (e.g., 50% and 50%, or 30%, 35% and 35%) must be given, which will be used as a reference of assigning (scaling) grades. Only one copy of the project should be submitted by one of the team members.

A team-building document is given separately for assignments 3 and 5. We need to know the teams in advance, because we need to create a server site for each team.

Section I Preparation and Practice Exercises (No submission required)

No submission is required for this section of exercises. However, doing these exercises can help you better understand the concepts and thus help you in quizzes, exams, as well as the assignment questions.

1. Reading: Textbook Chapter 2.

2. Answer the multiple choice questions in text section 2.8. Studying the material covered in these questions can help you prepare for the lecture exercises, quizzes, and the exams.

3. Study for the questions 2 through 20 in text section 2.8. Make sure that you understand these questions and can briefly answer these questions. Study the material covered in these questions can help you prepare for the exams and understand the homework assignment.

4. Test the programs given in questions 24 and 25 in text section 2.8. Identify the problems in the program and give correct versions of the programs.

5. If you want solve a more challenging problem in multithreading, you can do question 26 in text section 2.8.

6. **Tutorial**. To help you complete the project in Section II, you may want go through the tutorial given in the textbook chapter 2, which consists of

   6.1 Reading the case study in text section 2.6.3.

   6.2 Implementing and testing the program given in the case study. The program can be used as the starting point for your project in Section II.

   6.3 Extending the program based on the requirement in Section II.

Section II Project (Submission required)

Warning: This is a long programming project designed for a study load for three weeks of estimated 3*8 = 24 hours. It is challenging at both the conceptual level and the implementation level. You must distribute the load in the given three weeks. You will not have enough time to complete it if you start the project only in the last week before the project is due.

Purpose of this project is to exercise the concepts learned in this chapter. It is not the purpose of this project to create realistic services and applications. We will create more realistic services and applications in the subsequent projects. In this project, you can use a console application or a simple GUI application to implement the user interface to your program. You do not need to create Web applications.

Description: Consider that you are creating an e-business: a book distribution system that involves bookstores and publishers. The system consists of multiple bookstores (clients) and multiple publishers (servers). The bookstores can buy in quantity of books from the publishers with lower prices. The required architecture and the major components of the system are shown in the diagram below.

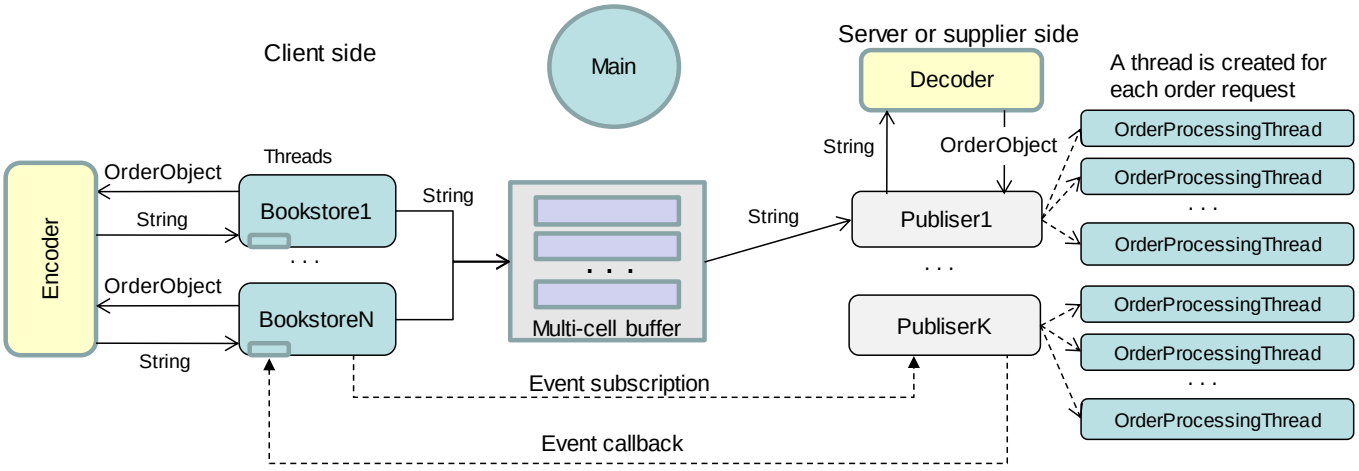

Figure 1 Architecture of a book distribution system

In this project, you will implement both clients and servers in one system in multithreading. You will further implement such systems in distributed web client and server systems in the following projects.

An **Operation Scenario** of the book distribution system is outlined as follows:

(1) A **Publisher** uses a pricing model to calculate dynamically the book price for the bookstores. If the new price is lower than the previous price, it emits a (promotional) event and calls the event handlers in the bookstores that have subscribed to the event.

(2) A **BookStore** evaluates the price, generates an OrderObject (consisting of multiple values), and sends the order to the Encoder to convert the order object into a plain string.
(3) The **Encoder** converts the object into a string.
(4) The Encoder sends the encoded string back to the caller.
(5) The BookStore sends the encoded string to one of the free cells in the **MultiCellBuffer**.
(6) The **Publisher** receives the encoded string from the MultiCellBuffer and sends the string to the Decoder for decoding.
(7) The **Decoder** sends the OrderObject to the Publisher. The decoded object must contain the same values generated by the BookStore.
(8) The Publisher creates a new thread, an OrderProcessingThread, to process the order;
(9) The **OrderProcessingThread** processes the order, e.g., checks the credit card number and calculates the total amount.
(10) The OrderProcessingThread sends a confirmation to the bookstore and prints the order (on screen).

Note, the encoder and decoder here do not perform encryption and decryption tasks.

Components in the diagram are explained in details as follows, with their grading scores (points) allocation, Components 1, 2, 3, 4, and 5 belong to Assignment 3, and the rest of the components belong to Assignment 4. You will submit both assignments together as a project. We will enter the scores under assignments 1 and 2, for the purpose of score management.

Assignment 3 Tasks

1. **Publisher1** through **PublisherK*** are the objects of a class on the server side: Each object has a method to be started as a thread by the Main method and will perform a number of functions. It uses a PricingModel to determine the book prices. It defines a price-cut event that can emit an event and call the event handlers in the BookStore if there is a price-cut according to the PricingModel. It receives the orders (in a string) from the MultiCellBuffer. It calls the Decoder to convert the string into the order object. For each order, you must start a new thread (resulting in multiple threads for processing multiple orders) from OrderProcessing class (or method) to process the order based on the current price. There is a counter p in the Publisher. After p (e.g., p = 20) price cuts have been made, a Publisher thread will terminate. The bookstores do not have to make an order after each price cut. [15 points]

*Note 1: For the individual project, the number of publishers K = 1. For a group project (both two-member and three-member group), K = 2 if you have a 2-member team, and K = 3 if you have a 3-member team.

2. **PricingModel**: It can be a class or a method in the Publisher class. It decides the price of books, which must be between 50 and 200. It can increase or decrease the price. You must define a mathematical model. The model can be a simple random function for individual projects. However, for the group projects, a more complex model must be developed, where the price must be a function with multiple parameters, such as the available number of the books and the number of orders received within a given time period. In other words, the function must take the amount of orders as input. You can use a hard-coded table of the prices, for example, in each weekday. You must make sure that your model will allow the price to go up some times and go down other times within your iterations of testing. [5 points]

3. **OrderProcessing** is a class or a method in a class on the server's side. Whenever an order needs to be processed, a new thread is instantiated from this class (or method) to process the order. It will check the validity of the credit card number. If you are doing an individual project or a two-member group project,

you can define your credit card format, for example, the credit card number from the bookstores must be a number registered to the Publisher, or a number between two given numbers (e.g., between 5000 and 7000). For the three-member group project, a bank service must be created. Each OrderProcessing thread will calculate the total amount of charge, e.g., unitPrice*NoOfBooks + Tax + LocationCharge. For the group (two and three members) projects, a confirmation must be sent back to the bookstore when an order is completed. You can implement the confirmation in different ways. For example, you can use another buffer for the confirmation, where you can use a buffer cell for each thread, so that you do not have to consider the conflict among the threads. However, you still need to coordinate the write and read between the producer and the consumer. [10 points]

4. **BookStore1** through **BookStoreN**, You can set N = 5 in your implementation. Each bookstore is a thread instantiated from the same class (or the same method in a class). The bookstore's actions are event-driven. Each bookstore contains a callback method (event handler) for the Publisher to call when a price-cut event occurs. The bookstore will calculate the number of books to order, for example, based on the need and the difference between the previous price and the current price. The thread will terminate after the Publisher thread has terminated. Each order is an OrderClass object. The object is sent to the Encoder for encoding. The encoded string is sent back to the bookstore. Then, the bookstore will send the order in string format to the MultiCellBuffer. For group project, before sending the order to the MultiCellBuffer, a time stamp must be saved. When the confirmation of order completion is received, the time of the order will be calculated and saved (or printed). [10 points]

The following components will be counted as assignment 3 tasks.

5. **OrderClass** is a class that contains at least the following private data members:

- senderId: the identity of the sender, you can use thread name or thread id.
- cardNo: an integer that represents a credit card number.
- receiverID: the identity of the receiver, you can use thread name or a unique name defined for a publisher. If you are doing an individual project, you do not need this field.
- amount: an integer that represents the number of books to order.
- unit price: a double that represents the unit price of the book received from the publisher.

You must use public methods to set and get the private data members. You must decide if these methods need to be synchronized. The instances created from this class are of the OrderObject. [10 points]

Assignment 4 Tasks

6. **MultiCellBuffer** class is used for the communication between the bookstores (clients) and the publishers (servers): This class has n data cells (for individual project, n = 2 and for group project, n = 3). The number of cells available must be less than (<) the max number of bookstores in your experiment. To write data into and to read data from one of the available cells, setOneCell and getOneCell methods can be defined. You must use a semaphore of value n to manage the availability of the cells. You must use an additional lock mechanism to provide read or write permissions for a cell. You cannot use a queue for the buffer, which is a different data structure. The semaphore allows a bookstore to see the availability of the cells, while the lock mechanism allows the agent to gain the right to write into one of the buffer cells. The Publisher can read buffer cells at the same time. Synchronization/monitor is required for read/write and write/write overlap. [20 points]

7. **Encoder** is a class or a method in a class: The Encoder will convert an OrderObject into a string. You can choose any way to encode the values into a string, as long as you can decode the string back to the original order object. [10 points]

8. **Decoder** is a class or a method in a class: The Decoder will convert the encoded string back into the OrderObject. Note: encoder and decoder here DO NOT perform encryption and decryption. [10 points]

9. **Bank service** is a class. This question is for the three-member group projects only. It allows the bookstore to apply for a credit card number. The bookstore will use this number to purchase books from the publishers. The publishers will send the credit card number and the amount to the bank. The bank charges the account and returns the message "valid" if the account exists and the funds are sufficient for the purchase, otherwise, it returns "not valid". When a publisher sends the credit card number and the amount to the bank, it will call the encryption service in the ASU repository to encrypt the credit card numbers in a string. If you use the .svc service, you must use "Add Service Reference". If you use the .asmx service, you must use "Add Web Reference". The recommendation is to use the .svc service, as it is the new standard. On the other hand, the bank will call the ASU decryption service to decrypt the number before comparing with the valid credit card numbers. The Bank class must also allow the account holder to deposit funds into the account. [20 points]

10. **Main**: The Main thread will perform necessary preparation, create the buffer classes, instantiate the objects, create threads, and start threads. [10 points]

Additional Tasks for group projects

Two-member Teams: If you are doing the project as a group of two members, you are required to complete the following additional tasks:

[1] You must implement multiple suppliers (publishers) with K = 2. You must deal with all the issues that are incurred because of the increase in the number of suppliers.

[2] You must develop a more complex price model.

[3] The supplier must send a confirmation back to the bookstore when an order is completed.

[4] The buffer size (number of cells in the buffer) is 3, instead of 2.

In this option, one member must write the code of the bookstores and the main program. The other member must write the code of the supplier. The two members must jointly write the code for the buffer class.

Three-member Teams: If you are doing the project as a group of three members, you are required to complete all the additional tasks for the two-member group projects and the following additional tasks:

[1] You must implement multiple suppliers (publishers) with K = 3. You must deal with all the issues that are incurred because of the increase in the number of suppliers.

[5] If you have a three-member team, you must also define the bank service. The third member must implement the Bank class and share some other tasks in the assignment.

For three member team, there are totally 70 points in Assignment 4 of the project. The 70 points will be scaled to 50 points proportionally. For example, 70 → 50, 60 → 43, and 50 → 36.

Notes:

1. It is the purpose of this course to enforce the knowledge of C# (Visual Studio), as we have multiple courses in our program that use Java. We encourage you to use C#. However, we allow the use of Java (on NetBeans) in this assignment. You must indicate the environment (e.g., VS 2015 or 2017 or NetBeans) that you use, so that the TA can use the same environment to grade the

project. We taught multithreading in both Java and C#. However, we did not teach event-driven programming in Java. If you choose to do the project in Java, you need to study this part on your own. I suggest that you do the project in Java only if you already know how to program events. The event handling in Java is not as easy as in C# for this project, and you need to take extra effort to implement the required functions.

2. You must follow what is defined in the assignment/project document. You have flexibility to choose your implementation details if they are not explicitly specified in the document. If you are not sure on any issue, ask the instructor or the TA by posting the question in the discussion board.

3. The program and each component of the program must be well commented.

4. You can choose to do this project as an individual project or a group project. I encourage you to do as a group project. Projects 3 and 5 will be required group projects and thus, you need to build a team anyway. The team building process is for assignments 3 and 5. The group used in assignment 2 does not have to be the same group for assignments 3 and 5. However, the group for assignment 3 and 5 must be the same, as the project in assignment 5 will be based on assignment 3.

5. Project 3 will be partly a team project. If you have formed a team in this project, you can use the same team for the project 3. Of course, you can also use a different team for the project 3.

Submission Requirement

All submissions must be electronically submitted to the assignment folder where you downloaded the assignment paper. All files must be zipped into a single file.

Submission preparation notice: The assignment consists of multiple **distributed** projects and components. They may be stored in different locations on your computer when you create them. You must choose your own location to store the project when you create the project. Then, you can copy these projects into a single folder for the blackboard submission. To make sure that you have all the files included in the zip file and they work together, you must **test** them before submission. You must also download your own submission from the blackboard. Unzip the file on a different machine, and test your assignment and see if you can run the solution in a different machine, because the TA will test your application on a different machine.

If you submitted an empty project folder, or an incomplete project folder, we cannot grade your resubmission after the due date! We grade only what you submitted before the submission due date. Please read FAQ document in the course Web page for more details.

Grading of Programming Assignment/Project

The TA will grade your program following these steps:

(1) The TA will read your program and give points based on the points allocated to each component, the readability of your code (organization of the code and comments), logic, inclusion of the required functions, and correctness of the implementations of each function.

(2) Compile the code. If it does not compile, 40% of the points given in (1) will be deducted. For example, if you are given 20 points in step (1), your points will become 12 if the program fails to compile.

(3) If the code passes the compilation, the TA will execute and test the code. If, for any reason, the program gives an incorrect output or crashes for any input, 20% of the points given in (1) will be deducted.

Please notice that the TA will not debug your program to figure out how big or how small the error is. You may lose 40% or 20% of your points for a small error such missing a comma or a space!

Late submission deduction policy:

- No penalty for late submissions that are received within 24 hours of the given due date;
- 1% grade deduction for every hour after the first 24 hours of the grace period!
- No submission will be allowed after Tuesday midnight.