Deadline: March 31, 2021 23:59:59.
Question#1: (4 points)

In a client-server system for a hypothetical banking application, the clients are ATM machines, and the server is the bank server. Suppose that the client has two operations: withdraw and deposit. Use the broker architectural pattern to document an initial architecture design for this system:
1-Draw a class diagram for this system. Use a client-side proxy to encrypt the data using an encrypt operation, and use a server side proxy to decrypt the data.
2- Draw a sequence diagram for this system.
3- Suppose that we want greater availability of the server, discuss what kind of tactics you should use to achieve that.

Question#2: (4 points)

You are given a class that processes the purchases for an online store. The class receives calls to retrieve the prices for items from a database, records the sold items, updates the database, and refreshes the webpage. What architectural pattern is suitable for this? Illustrate your answer by drawing a model for the solution, showing the method calls/events. Comment on how applying the pattern will impact the modifiability of the system.

Question#3: (4 points)

A "Personal Address Book" application program allows the user to add, delete, search, save and load her contact information. The program separates user (command-line) interface and internal processing subsystem. The internal processing system consists of the following classes: ContactManager (responsible for add and delete operations), ContactFinder (responsible for search operation), and DataManager (responsible for save and load operations).

1-What design pattern can be used to implement the user interface? Explain your answer using class diagram for the entire system.

2- Draw an UML sequence diagram to show the behavioural view of the personal address book program that demonstrates what happens when a user enter a new contact information.

Question#4: (4 points)

In a system comprising of 3 components: A, B, C. Calling A requires calling B, and calling B requires calling A. Component C is responsible for tasks T#1, T#2, and T#3.
Comment on the modifiability of this system. What are the problems that you see in this system, and how you solve them?

Suppose that T#1 is performed by both component A and C? What does it say about A and C? How you solve this problem?

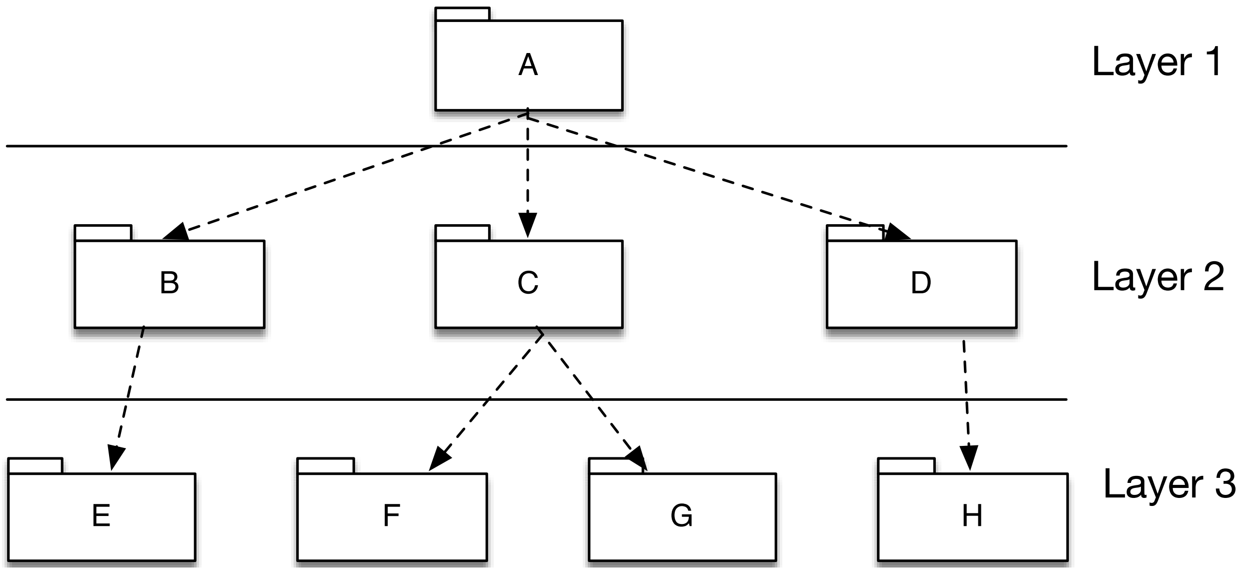
Question#5: (3 points)

Write a performance scenario for the Trent Course Registration System. Suppose that this system uses a client-server architecture. Think about whether your major concern is latency, throughput, or some other response measure. What tactics you will use to mitigate these issues. Elaborate your answer using examples.

Questions#6: (3 points)

Think about security scenarios for an automatic teller machine (ATM). How would you modify your design for ATM to satisfy one particular scenario? Choose one concrete security scenario to answer this question.

Question#7: (4 points)



Show how you will do the integration testing on the above layered system using bing bang, top-down, bottom up, sandwich and modified sandwich approaches?

Question#8: (4 points)

Write JUnit test cases for setLoanAmount (to test the Exception) and getMonthlyPayment.

```
public class Loan {
  private double annualInterestRate;
  private int numberOfYears;
  private double loanAmount;
  private java.util.Date loanDate;

  /** Default constructor */
  public Loan() {
    this(2.5, 1, 1000);
  }

  /** Construct a loan with specified annual interest rate,
      number of years, and loan amount
    */
  public Loan(double annualInterestRate, int numberOfYears,
      double loanAmount) {
    this.annualInterestRate = annualInterestRate;
    this.numberOfYears = numberOfYears;
    this.loanAmount = loanAmount;
    loanDate = new java.util.Date();
  }

  /** Return annualInterestRate */
  public double getAnnualInterestRate() {
    return annualInterestRate;
  }

  /** Set a new annualInterestRate */
  public void setAnnualInterestRate(double annualInterestRate) {
    this.annualInterestRate = annualInterestRate;
  }

  /** Return numberOfYears */
  public int getNumberOfYears() {
    return numberOfYears;
  }

  /** Set a new numberOfYears */
  public void setNumberOfYears(int numberOfYears) {
    this.numberOfYears = numberOfYears;
  }

  /** Return loanAmount */
  public double getLoanAmount() {
    return loanAmount;
  }
  /** Set a newloanAmount */
  public void setLoanAmount(double loanAmount) throws Exception{
       if (loanAmount < 0)
            throw new Exception("Money cannot be negative");
    this.loanAmount = loanAmount;
  }

  /** Find monthly payment */
  public double getMonthlyPayment() {
    double monthlyInterestRate = annualInterestRate / 1200;
    double monthlyPayment = loanAmount * monthlyInterestRate / (1 -
      (1 / Math.pow(1 + monthlyInterestRate, numberOfYears * 12)));
    return monthlyPayment;
  }

  /** Find total payment */
  public double getTotalPayment() {
    double totalPayment = getMonthlyPayment() * numberOfYears * 12;
    return totalPayment;
  }

  /** Return loan date */
  public java.util.Date getLoanDate() {
    return loanDate;
  }
}
```

Submit your:
1-Source code.
2-Screenshots of the test runs.

For all diagrams, use an appropriate modelling tool such as Microsoft Visio, draw.io. Submit your assignment on blackboard using a pdf file.