1. **Object Oriented Concepts Review:**

   **a. Overriding:**

   ```csharp
   class DrawingObject
   {
       public virtual void Draw()
       {
           Console.WriteLine("I'm just a generic drawing object.");
       }
   }

   class Line : DrawingObject
   {
       public override void Draw()
       {
           Console.WriteLine("I'm a Line.");
       }
   }

   class Circle : DrawingObject
   {
       public override void Draw()
       {
           Console.WriteLine("I'm a Circle.");
       }
   }

   class Square : DrawingObject
   {
       public override void Draw()
       {
           Console.WriteLine("I'm a Square.");
       }
   }

   class Program
   {
       static void Main(string[] args)
       {
           DrawingObject[] dObj = new DrawingObject[3];
           // More code...
       }
   }
   ```
dObj[0] = new Line();
dObj[1] = new Sequare();
dObj[2] = new DrawingObject();

foreach (DrawingObject drawObj in dObj)
{
  drawObj.Draw();
}

b. Overloading:

class Math
{
  public int Plus(int number1, int number2)
  {
    return number1 + number2;
  }

  public int Plus(int number1, int number2, int number3)
  {
    return number1 + number2 + number3;
  }
}

class Program
{
  static void Main(string[] args)
  {
    Math m = new Math();
    int x = m.Plus(10, 10);
    Console.WriteLine(x);

    int y = m.Plus(10, 15, 20);
    Console.WriteLine(y);
    Console.ReadLine();
  }
}
c. Abstract Class:

```java
public abstract class Shape2D {
    public abstract float getArea(int a, int b);
    public abstract float getPerimeterLength(int a, int b);
}

public abstract class EllipticalShape : Shape2D {
}

public class Circle : EllipticalShape {
    public override float getArea(int r, int b) {
        float x = (float) Math.Pow(r, 2f);
        float area = 3.14f * x;
        return area;
    }

    public override float getPerimeterLength(int a, int b) {
        float p = (float) 2 * 3.14f * a;
        return p;
    }
```
public static double[] getCenter()
{
    double[] center;
    center = new double[2];
    center[0] = 2.3;
    center[1] = 3.4;
    return (center);
}

public void printCenter()
{
    double[] a = getCenter();
    Console.WriteLine("The Center of the Circle is :");
    Console.Write("x= " + a[0] + " y= " + a[1]);
}

class Ellipse : EllipticalShape
{
    public override float getArea (int a, int b)
    {
        float area = 3.14f * a * b;
        return area;
    }

    public override float getPerimeterLength(int a, int b)
    {
        float x = (float) Math.Sqrt(Math.Pow(a, 2) + Math.Pow(b, 2));
        float p = (float) 2 * 3.14f * (x/2);
        return p;
    }
}

class Program
{
    static void Main(string[] args)
    {
        Circle c = new Circle();
        Ellipse e = new Ellipse();
        Console.WriteLine("Circle Area = " + c.getArea(10,10));
        Console.WriteLine("Ellipse Area = " + e.getArea(10, 5));
        Console.WriteLine("Circle Perimeter = " + c.getPerimeterLength(10, 10));
        Console.WriteLine("Ellipse Perimeter = " + e.getPerimeterLength(10, 5));
        c.printCenter();
        Console.ReadLine();
    }
}
**Note:** *Rectangle class implementation is a home work.*

**Example:**

```csharp
class BankAccount
{
    public virtual int accNo { get; set; }
}

class SavingAccount: BankAccount
{
    public override int accNo { get; set; }
    public double balance { get; set; }
    Customer c = new Customer();

    public double updateBalance(double balance)
    {
        return balance + balance*0.04 ;
    }
}

class Customer
{
    public List<SavingAccount> GetSetAccount { get; set; }
    public string fname { get; set; }
    public string lname { get; set; }
    public string address { get; set; }

    public void print()
    {
    }
}
```
```csharp
Console.WriteLine("Customer Basic Information : ");
Console.WriteLine("Name   :- "+this.fname);
Console.WriteLine("Surname:- " + this.lname);
Console.WriteLine("Address:- " + this.address);
    foreach (SavingAccount b in this.GetSetAccount )
    {
        Console.WriteLine("Account Number :- "+b.accNo);
        Console .WriteLine ("Balance: -"+b.updateBalance(b.balance));
}
}

class Program
{
    static void Main(string[] args)
    {
        List<SavingAccount> List = new List<SavingAccount>();
        SavingAccount a = new SavingAccount();
        a.accNo = 1234;
        a.balance = 1000;
        a.updateBalance(a.balance);
        List.Add(a);

        SavingAccount b = new SavingAccount();
        b.accNo = 5678;
        b.balance = 3000;
        b.updateBalance(b.balance);
        List.Add(b);

        Customer c = new Customer();
        c.fname = "Ahmed";
        c.lname = "mohamed";
        c.address = "Baghdad";
        c.GetSetAccount = List;
        c.print();

        Console.ReadLine();
    }
}
```

2. **Use case Diagram:**

A book is written by an author, published by a publisher, sold by a book store, and read by a reader. Moreover, for a reader to read a book, she must buy it from a book store that is selling it. Draw a use case diagram for this scenario, showing relationships between different use cases.
ii. Draw a use case diagram for a ticket distributor for a train system. The system includes two actors: a traveler, who purchases different types of tickets, and a central computer system, which maintains a reference database for the tariff. Use cases should include: BuyOneWayTicket, BuyWeeklyCard, BuyMonthlyCard, UpdateTariff. Also include the following exceptional cases: Time-Out (i.e., traveler took too long to insert the right amount), TransactionAborted (i.e., traveler selected the cancel button without completing the transaction), DistributorOutOfChange, and DistributorOutOfPaper.
iii. Draw a use case diagram for courses. In particular, assume that courses are taught by instructors, while registrars can enroll or remove students from a course. Students take a course, provided they are enrolled in it.

![Use Case Diagram for Courses]

- **System**
- **Instructor**
- **Registrar**
- **Students**

- **TeachCourse**
- **EnrollStudentInCourse**
- **RemoveStudentFromCourse**
- **TakeCourse**

iv. A midterm exam is prepared by the teacher and taken by each of the students in her class. In special cases, where the student misses the test, the student has to take a makeup test.

![Use Case Diagram for Tests]

- **System**
- **Student**
- **Teacher**

- **TakeTest**
- **MakeTest**
- **TakeMakeupTest**
3. **Class Diagrams:**

i. A hotel has a number of rooms that can be rent by guests. There are also a number of bathrooms, which are either connected to a specific room or are used to service multiple rooms on the floor. The rooms are classified into three types: single rooms, double rooms and family rooms. Each single room can only be rent to at most one guest. Each double room can be rent to at most two guests. Each family room can be rent to a family of up to two adults and two children.

![Class Diagram](image)

ii. A university offers a number of degree programmes, which are classified into BSc (Hons) degree programme, MSc degree program and PhD degree programme. To teach students in various programmes, the university runs a number of course modules. A particular module could be acceptable to a program, or compulsory to a program, or not acceptable at all to the programme. Each BSc (Hons) degree programme contains a number of modules as acceptable or compulsory, which are classified into stage I modules, advanced modules and honours modules. For a student who studies a BSc (Hons) degree programme, in order to obtain the degree he/she must complete a study plan that consists of at least 8 stage I modules, 16 advanced/honours modules and 4 honours modules that are acceptable to the programme.
iii. In a university there are different classrooms, offices and departments. A department has a name and it contains many offices. A person working at the university has a unique ID and can be a professor or an employee. A professor can be a full, associate or assistant professor and he/she is enrolled in one department. Offices and classrooms have a number ID, and a classroom has a number of seats. Every employee works in an office.
iv. Consider the world of companies: Companies employ employees (who can only work for one company), and consist of one or more departments. Each company has a single president, who is an employee. Departments have employees as members and run projects (one or more.) Employees can work in 1 to 3 projects, while a project can have 2 to 50 assigned employees. You may assume that companies have a name and address, while employees have a emp# and a salary.

4. Sequence Diagrams:

The following are description of systems. Draw a UML Sequence diagram to represent the Dynamic model for each of them:

i. To buy a book electronically from chapters.com, a customer needs to select the book from a list provided by Chapters’ eCommerce system, provide credit card information to the system, and then the system gets authorization from the bank for the payment, and -- if positive -- confirms the sale. The order is then sent to the orders department and when the book becomes available, it is shipped to the customer. Also, the order department charges the customer’s credit card by informing the bank of the amount.

*Draw a sequence diagram that models this process. Make sure to model all relevant actors and the interactions between them. Do show explicitly the time intervals when different actors actively participate in the process you are modeling.*
ii. Draw sequence diagram based on the following interactions between a video store clerk and objects in a video rental system. The scenario is named rent video to existing member:

a. Actor sends <<create>> message to create new Rental object named aRental, and the create message includes arguments for memberID and videoID.
b. aRental sends addMemberToRental message to Member object based on memberID, named aMember, which returns member details.
c. aRental sends rentVideo message to aVideo object based on the videoID, named aVideo, which returns video details.
d. aRental returns all rental details to the actor.
iii. A final exam for a given course is prepared by the instructor of the course and submitted to the DCS undergraduate office by March 20. The undergraduate office sends all exams to the Faculty of Arts & Science office where copies are made. On the day of the exam, copies are given to the presiding officer who takes them to the room where the exam is given. The students write the exam on an exam copy, and return to the presiding officer at the end of the exam. The written exam copies are given to the instructor for marking. Draw a sequence diagram that captures this description.
iv. The College Street Red Cross Blood Donor Centre operates as follows: On the day of a blood donation, the Donation Unit receives blood donations from donors and sends them to the Testing Unit which tests each blood donation for blood type and potential viral agents. The Testing Unit then sends the blood donation along with test results to the Processing Office (another unit of the Centre) which fills a form for each tested blood unit where the tests are OK, and sends the blood units and forms to the Distribution Office. If tests indicate that a blood unit may be contaminated with a viral agent, the Processing Office destroys that unit. Draw a sequence diagram for this Description.