	COLLEGE OF COMPUTING AND INFORMATION SCIENCES		
	Final Assessment Fall 2020 Semester		
Class Id	105150,105151,105152	**Course Title**	OOAD
Program	BSCS	**Campus / Shift**	North Campus / Morning
Date	10th Dec 2020	**Total Marks**	40
Duration	04 hours	**Faculty Name**	Mohammad Ayub Latif/Aleenah Khan
Student Id	63758	**Student Name**	Ali Salman Hassan

Instructions:

- Filling out Student-ID and Student-Name on exam header is mandatory.
- Do not remove or change any part of exam header or question paper.
- Write down your answers in given space or at the end of exam paper with proper title "Answer for Question# _ _".
- Answers should be formatted correctly (font size, alignment, etc.)
- Handwritten text or image should be on A4 size page with clear visibility of contents.
- Only PDF format is accepted (Student are advised to install necessary software)
- In case of CHEATING, COPIED material or any unfair means would result in negative marking or ZERO.
- A mandatory recorded viva session will be conducted to ascertain the quality of answer scripts where deemed necessary.

- **Caution:** Duration to perform Final Assessment is **03 hours only**. Extra 01 hour is given to cater all kinds of odds in submission of Answer-sheet. **Therefore, if you failed to upload answer sheet on LMS (in PDF format) within 04 hours limit, you would be considered as ABSENT/FAILED.**

Note: You are required to provide neat diagrams and neatly written code. In case the instructor fails to understand your handwriting, the instructor has the right to give a straight zero. You can also type your answers. Please note that you cannot provide the examples that are mentioned in the book. You have to come up with something which is a result of your own thought process.

All questions carry equal marks. All parts of a question also carry equal marks.

Question 01:

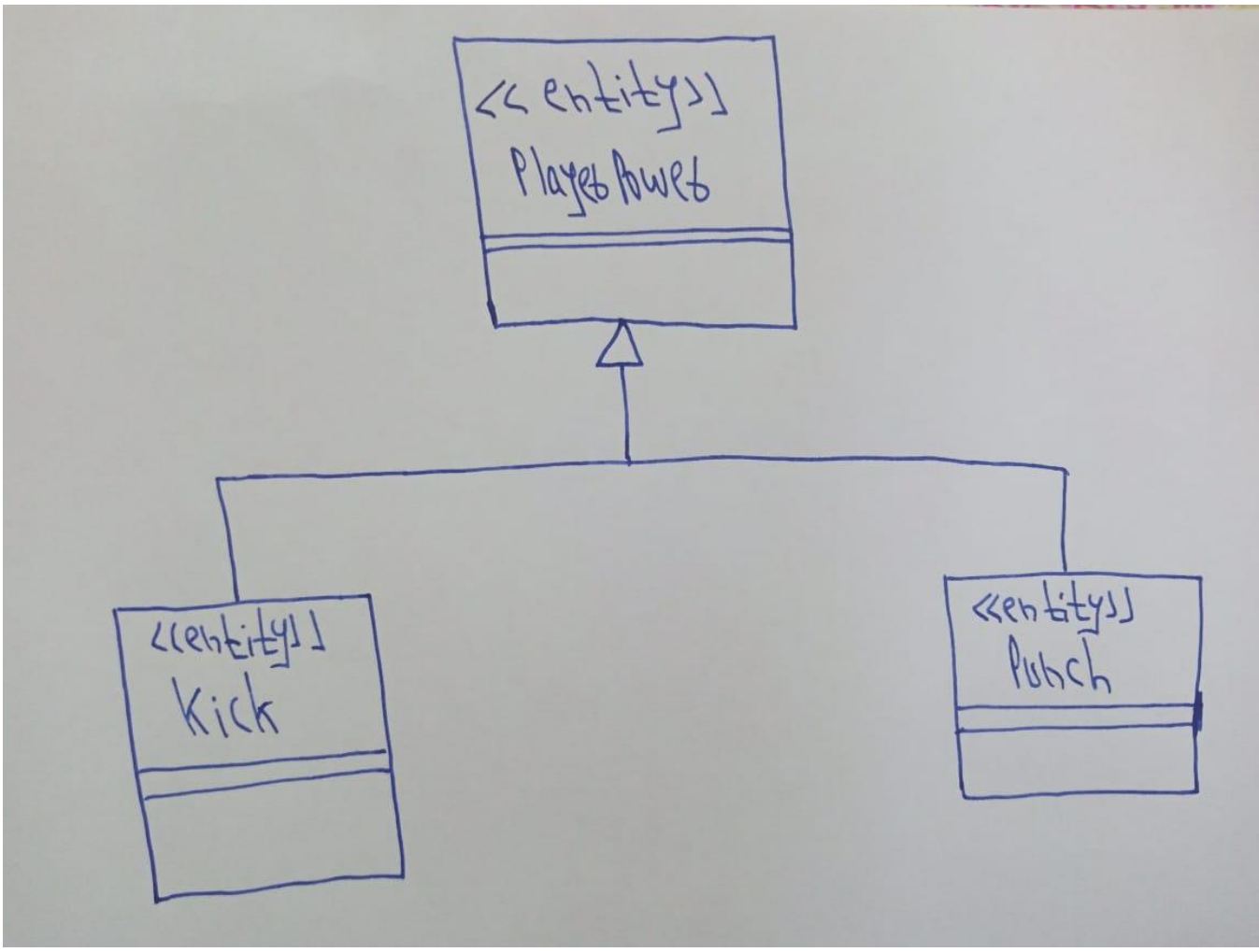
Answer the following questions:

1. What is an inheritance tree? Give an example.

   ANSWER:

   The process of calling the properties of one class into another class is known as Inheritance. There are two classes used in inheritance mainly known as the base class and derived class. The base (parent class) is used for the properties or methods of this class we want to inherit to another class whereas the derived class (child class) is used to inherit the properties and methods of the base class or parent class.

   Inheritance tree is where the classes are organized into a singly rooted tree structure. Information associated with level one of abstraction in a inheritance tree is automatically applied to the another levels of the inheritance tree.



2. In an interaction diagram, what are the first two objects? Explain their nature and reason for their existence. Provide an example.

   ANSWER:

   The 1st 2 objects are the actor & a boundary class. It is an interface which communicates with the actor

   & passes the data along the control class. An actor specifies the role by user or external system which interacts with the boundary.

   System cannot work without an actor because actor's actor can talk and interact with boundary classes. Boundary class creates a bridge between the Actor`s info and control class.

   EXAMPLE:

By taking an example of course registration system, the actors are student or professor or registration_Person whereas there can be multiple boundary classes like “Add_a _ourse” , “login”, “register_course”, etc.

3. Define Reflexive Relations and provide an example. Compare reflexive relations with a related concept in databases.

   ANSWER:

   Multiple objects belonging to the same class have to communicate with each other for data sharing. The relation between that objects is called as Reflexive relationships.

   Example:

   ```
   Class item{

   Int item_id;

   String item_name;

   Item I;

   Public void additem(item i){

   }

   }
   ```

   There is a method containing the parameters of item which is representing the reflexive relation.

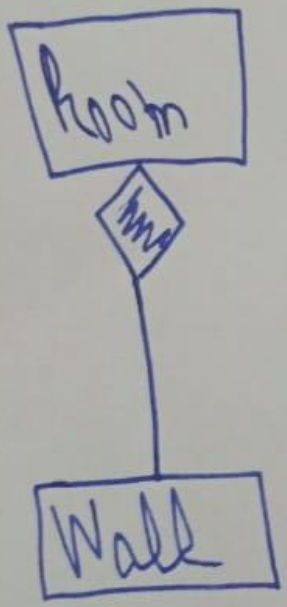
   It is representing the self-join which allows to join a table with itself. So, we can say that self-join is a concept of reflexive relation in database.

4. Differentiate between Aggregation by Value & Aggregation by Reference. Provide an example for each.

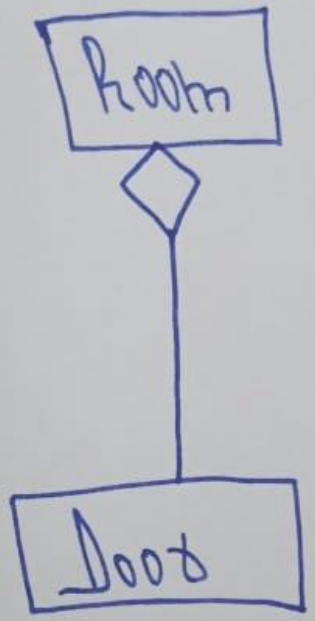
   ANSWER:

Aggregation by value	Aggregation by reference
The life cycle of the part of object is dependent on the object of whole class.	The life cycle of the part of object is not dependent on the object of whole class.
It is also called composition or containment.	It is like giving the address of the variable where the original data is stored.
Part of object is instantiated inside the constructor of the whole class.	Part of object is not instantiated inside the constructor of the whole class.

Aggregation by Value:



Aggregation by Reference:



5. Describe the term Homogenize? Explain the three possibilities associated with it. Provide an example for each of the three possibilities.

   ANSWER:

   The blending and cleaning of the project is called homogenization which is performed at the end of the cycle. We combine, delete or splits the classes in this process if it needed.

   - Combining class:

     Resolving the naming conflicts.

     If the behavior is same of 2 control classes.

   - Splitting class:

     Single class doing multiple functions, split the class.

     If an attribute is as important as like class, make it a class.

   - Eliminate class:

     If a class is not containing any structure or behavior, eliminate it.

     If control classes is only passing the data, eliminate it.

Question 02:

What is a State Chart diagram? For what purpose do we need a State Chart diagram? Explain ALL the components of a State Chart Diagram in detail. Draw a complete State Chart Diagram of a scenario of your choice.

ANSWER:

The diagram which consists of state, transitions, entry points, exit points, guard condition if occurring in scenario and action occurring in that scenario is known as state chart diagram.

We need state chart diagram to completely and clearly understanding our scenario to easily cop up with other team members.

The state chart diagram is describing the different states in the scenario.

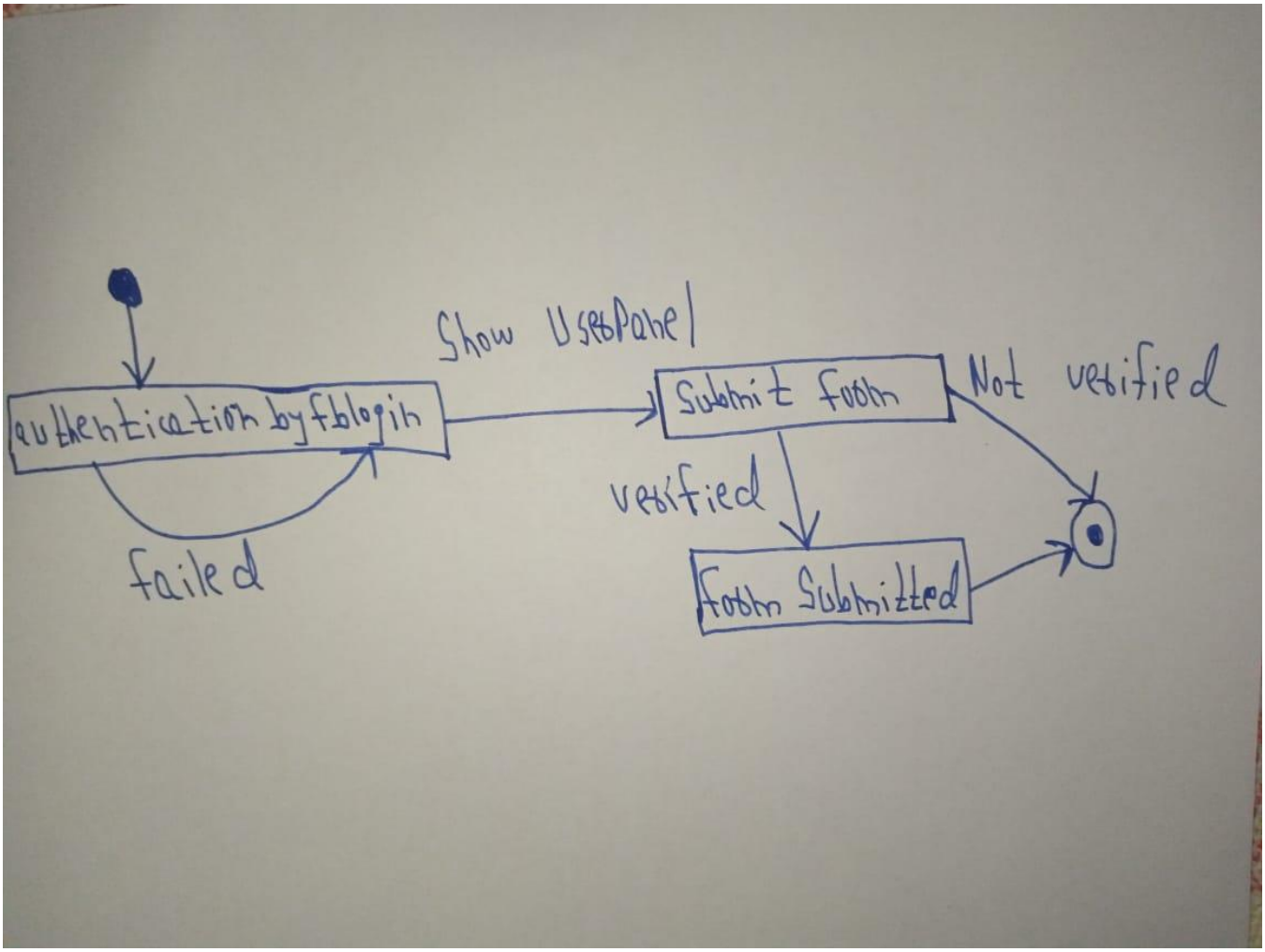
Components of state chart diagram:

- Action is denoted by square (round edges).
- Entry state is denoted by filled circle.
- Exit state is denoted by bull`s eye.
- State transition can or cannot trigger an event.

Scenario example:

We are authenticating the user by Facebook login and after that we are verifying the submitted form, if form is verified successfully submit the form.

Diagram:



Question 03:

Define generalization and specialization using a diagrammatic example. Explain Inheritance and differentiate it from aggregation using examples. What problems can arise if you use inheritance unnecessarily? Give an example scenario where Inheritance is misused and correct it by using Aggregation. Provide diagram to represent the scenario along with complete code.

ANSWER:

Generalization & Specialization:

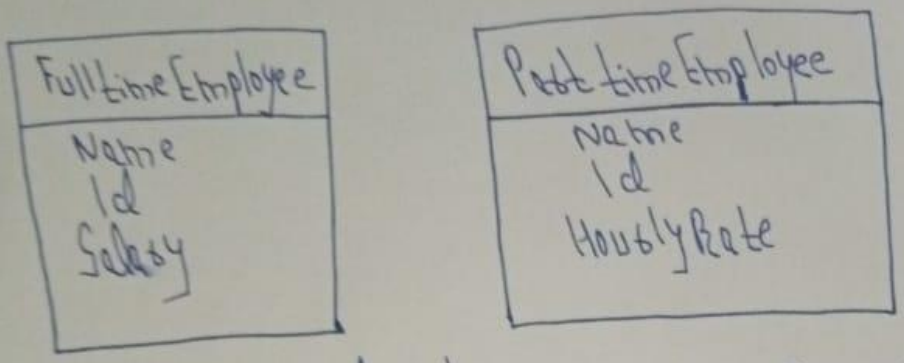
Generalization is a bottom to top approach. Generalization provides the capability to create

super classes that encapsulate structure and behavior common to several classes.

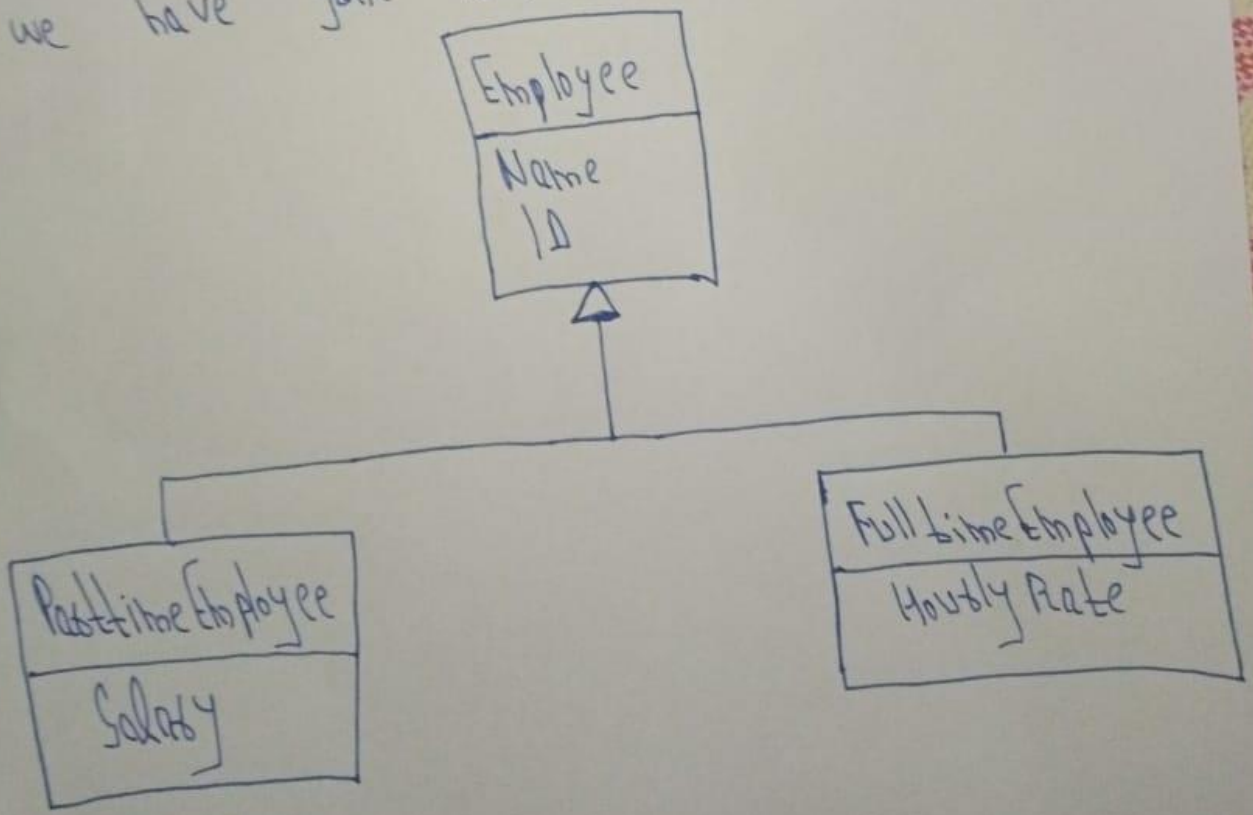
Specialization is a top to bottom approach. Specialization provides the ability to create

subclasses that represent refinements to the superclass—typically, structure and behavior are added to the new subclass.

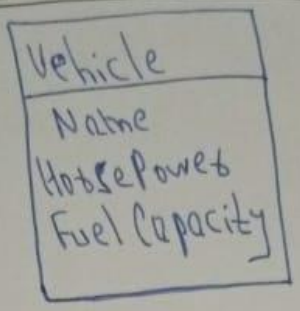
Generalization:



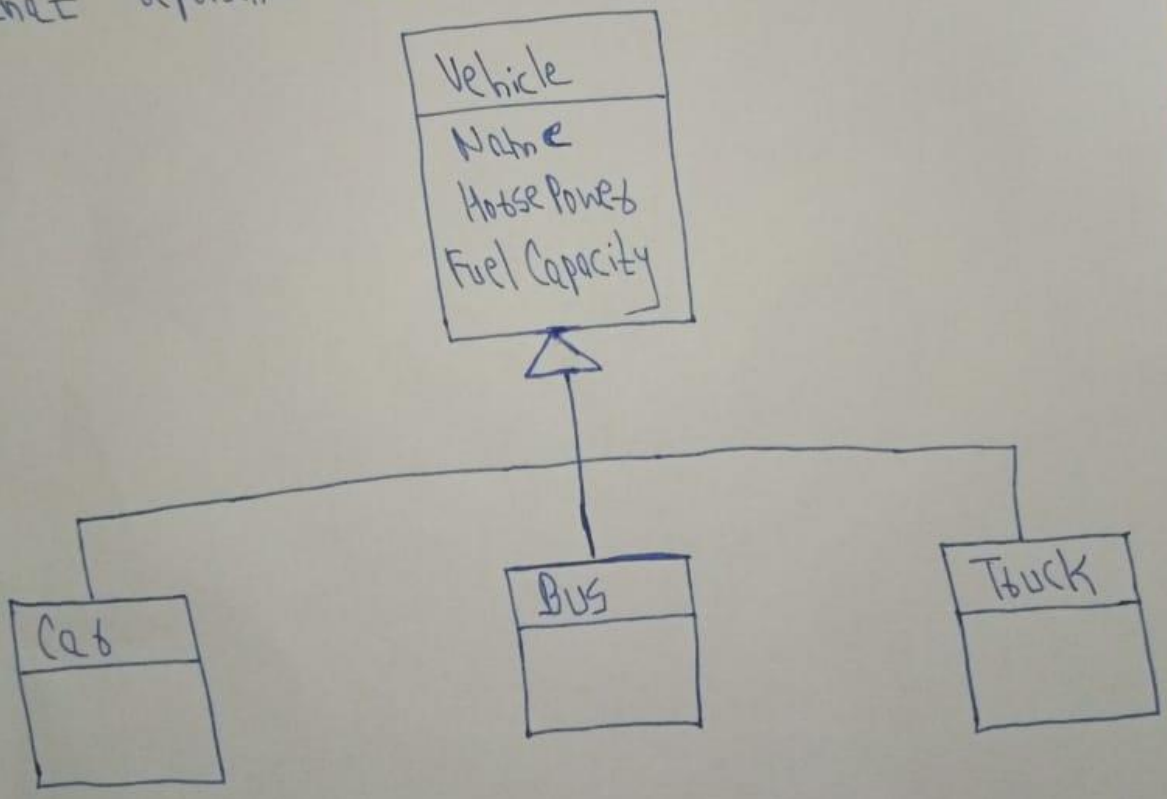
By using generalization, we can encapsulate common attribute in FullTimeEmployee class and PartTimeEmployee class into a Superclass called Employee thus we have generalized the design we have gone from bottom to top.



Specialization :



By using specialization, we can create subclasses that represent refinments to the superclass.



We would treat a class relation as a parent relation with other classes if a class is added as a superclass i.e., Generalization.

We would treat a class relation as a child relation with other classes if a class is added as a subclass i.e., Specialization.

Inheritance vs Aggregation:

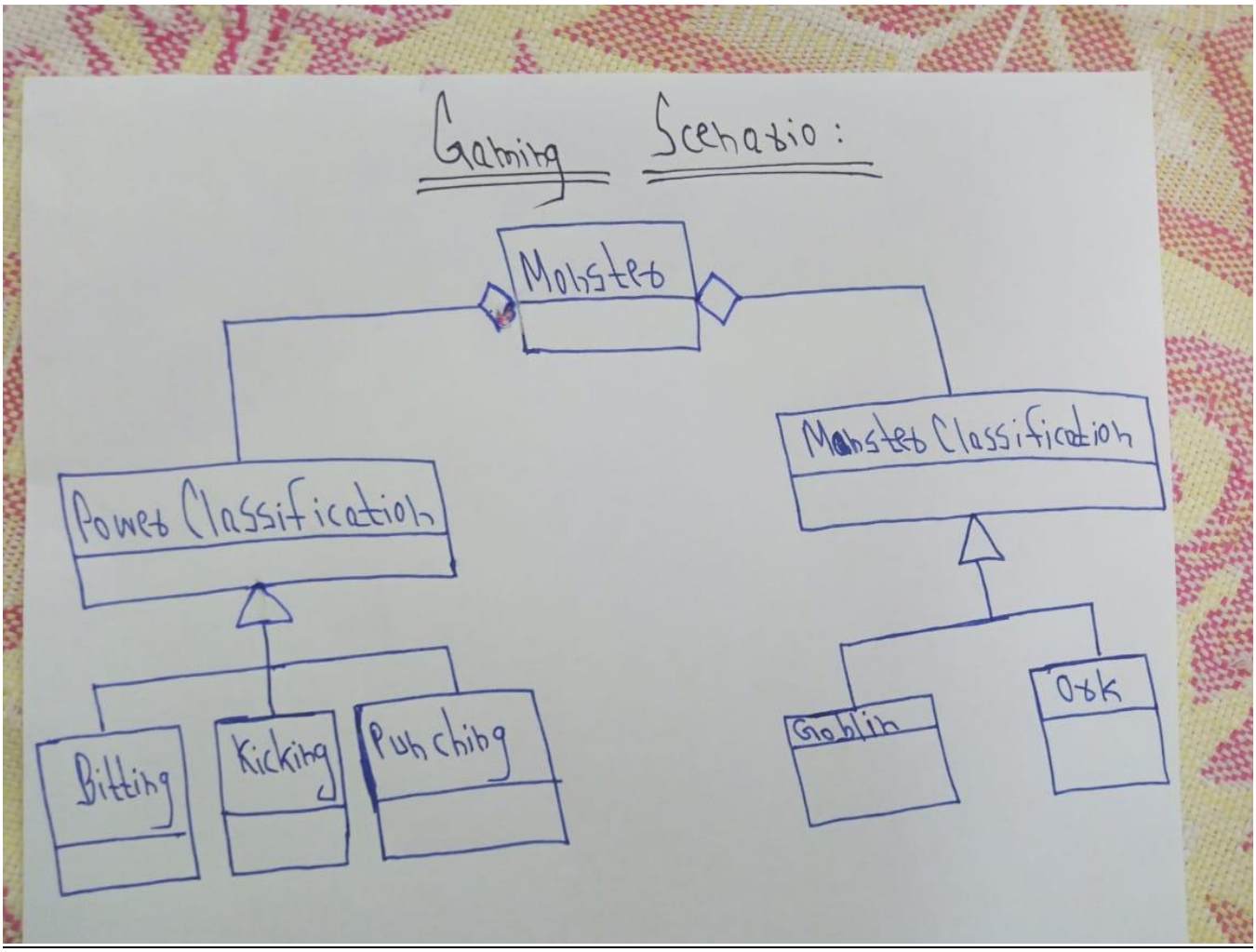
Inheritance defines a relationship among classes where one class shares the structure and/or behavior of one or more classes. A hierarchy of abstractions is created in which a subclass inherits from one or more super classes. Inheritance is also called an "is-a" or "kind-of" hierarchy. A subclass will inherit all attributes, operations, and relationships defined in any of its super classes. Thus, attributes and operations are defined at the highest level in the hierarchy at which they are applicable, which allows all lower classes in the hierarchy to inherit them. Subclasses may be augmented with additional attributes and operations that apply only to that level of the hierarchy. A subclass may supply its own implementation of an operation. Inheritance is a class relationship. There are two ways to find inheritance—generalization and specialization. Both methods typically are used for any system under development.

An aggregation relationship is a specialized form of association in which a whole is related to its part(s). Aggregation is known as a "part-of" or containment relationship. aggregation is an object relationship

Inheritance should be used to separate commonality from specifics. Aggregation should be used to show composite relationships. Often the two types of relationships are used together.

Example Scenario:

Diagram:



Explanation:

In this gaming scenario we have different types of monsters and different types of powers that these monsters possess. We have designed this scenario using the delegation pattern i.e mixture of aggregation and inheritance because by using delegation pattern we have eliminated the complexity of multiple inheritance because different monsters can have different powers also we are able to change the type of monster dynamically because Monster class has a reference to MonsterClassification and PowerClassification and we know by the rule of polymorphism an object variable of parent type can hold the reference of any of its child type.

Code

```
class Program
  {
     static List<Monster> SampleData()
     {
        return new List<Monster>
```

```
        {
            new Monster(new Goblin("Tolkien"), new Bitting(21, 10, 3)),
            new Monster(new Goblin("Azog"), new Kicking(11, 13, 4)),
            new Monster(new Ork("Uruk"), new Punching(14, 11, 3)),
            new Monster(new Ork("Gundabad"), new Kicking(16, 20, 7)),
            new Monster(new Goblin("Gnomish"), new Punching(16, 21, 2)),
            new Monster(new Ork("Lexicon"), new Bitting(22, 32, 5))
        };
    }

    static void Main(string[] args)
    {
        List<Monster> monsters = SampleData();

        foreach (Monster monster in monsters)
        {
            Console.WriteLine($"{monster.GetMonsterDetail()} {monster.GetPowerDetails()}");
            Console.WriteLine();
        }

        Console.ReadLine();
    }
}

public class Monster
{
    private MonsterClassification MonsterType { get; set; }
    private PowerClassification PowerType { get; set; }

    public Monster(MonsterClassification monsterType, PowerClassification powerType)
    {
        MonsterType = monsterType;
```

```
        PowerType = powerType;
    }



    public string GetPowerDetails()
    {
        return $"{MonsterType.Name} has attack damage of {PowerType.GetDamage()} ";
    }

    public string GetMonsterDetail()
    {
        return $"{MonsterType.GetMonster()}";
    }

    public void ChangeMonsterType(string name)
    {
        if (MonsterType.GetType() == typeof(Goblin))
            MonsterType = new Ork(name);
        else if (MonsterType.GetType() == typeof(Ork))
            MonsterType = new Goblin(name);
    }

    public void ChangePowerType(PowerClassification powerType)
    {
        if (powerType != null)
            PowerType = powerType;
        else
            throw new NullReferenceException("Power Type cannot be null.");
    }

}
```

```
public abstract class MonsterClassification
{
    public string Name { get; set; }

    public MonsterClassification(string name)
    {
        Name = name;
    }

    public abstract string GetMonster();
}

public class Goblin : MonsterClassification
{
    public Goblin(string name) : base(name)
    {
    }

    public override string GetMonster()
    {
        return $"{Name} is a Goblin";
    }
}

public class Ork : MonsterClassification
{
    public Ork(string name) : base(name)
    {

    }

    public override string GetMonster()
```

```
    {
        return $"{Name} is an Ork";
    }
}

public abstract class PowerClassification
{
    public int HitPoints { get; set; }
    public int AttackDamage { get; set; }

    public PowerClassification(int hitPoints, int attackDamage)
    {
        HitPoints = hitPoints;
        AttackDamage = attackDamage;
    }

    public abstract int GetDamage();
}

public class Bitting : PowerClassification
{
    public int BitegDamage { get; set; }

    public Bitting(int hitpoints, int attackDamage, int biteDamage) : base(hitpoints, attackDamage)
    {
        BitegDamage = biteDamage;
    }

    public override int GetDamage()
    {
        return AttackDamage + BitegDamage;
    }
```

```
}

public class Kicking : PowerClassification
{
    public int KickDamage { get; set; }

    public Kicking(int hitpoints, int attackDamage, int kickDamage) : base(hitpoints, attackDamage)
    {
        KickDamage = kickDamage;
    }

    public override int GetDamage()
    {
        return AttackDamage + KickDamage;
    }
}

public class Punching : PowerClassification
{
    public int PunchDamage { get; set; }

    public Punching(int hitpoints, int attackDamage, int punchDamage) : base(hitpoints, attackDamage)
    {
        PunchDamage = punchDamage;
    }

    public override int GetDamage()
    {
        return AttackDamage + PunchDamage;
    }
}
```

Question 04:

Explain the concept of an Association Class and differentiate it from the Association Relationship. Provide an example scenario that involves an Association Class and write complete functional code for it.

ANSWER:

ASSOCIATION CLASS:

The class that is part of an association relationship between the other classes is known as association class. Association class is same as the classes that contains operators, attributes, as well as other associations.

DIFFERENCE FROM ASSOCIATION RELATIONSHIP:

An association relationship is represented as many like one-to-one, one-to-many, or many-to-many. Also, an association relationship (between two or more objects) shows a path of communication (link) between themselves so that an object can send a message to another object.

EXAMPLE SCENARIO:

CODE:

```
public class SwipeCard{
    public void Swipe(Manager obj){}
    public string MakeofSwipeCard(){
        return "637758";}}

public class Manager{
    public void login(SwipeCard obj){
        obj.Swipe(this);}}

public class Main Class{
    Manager objManager=new Manager();
    objManager.WhoisManager(true);
    SwipeCard objSwipe=new SwipeCard();
    string StoMake=objSwipe.MakeofSwipeCard();
```

EXPLANATION:
This relationship is called the Association relationship.

The scenario I created is showing how the "SwipeCard" class uses the "Manager" class and the "Manager" class uses the "SwipeCard" class. We can also see the object of the "Manager" class and "SwipeCard" independently and they can have their own objects lifecycle.

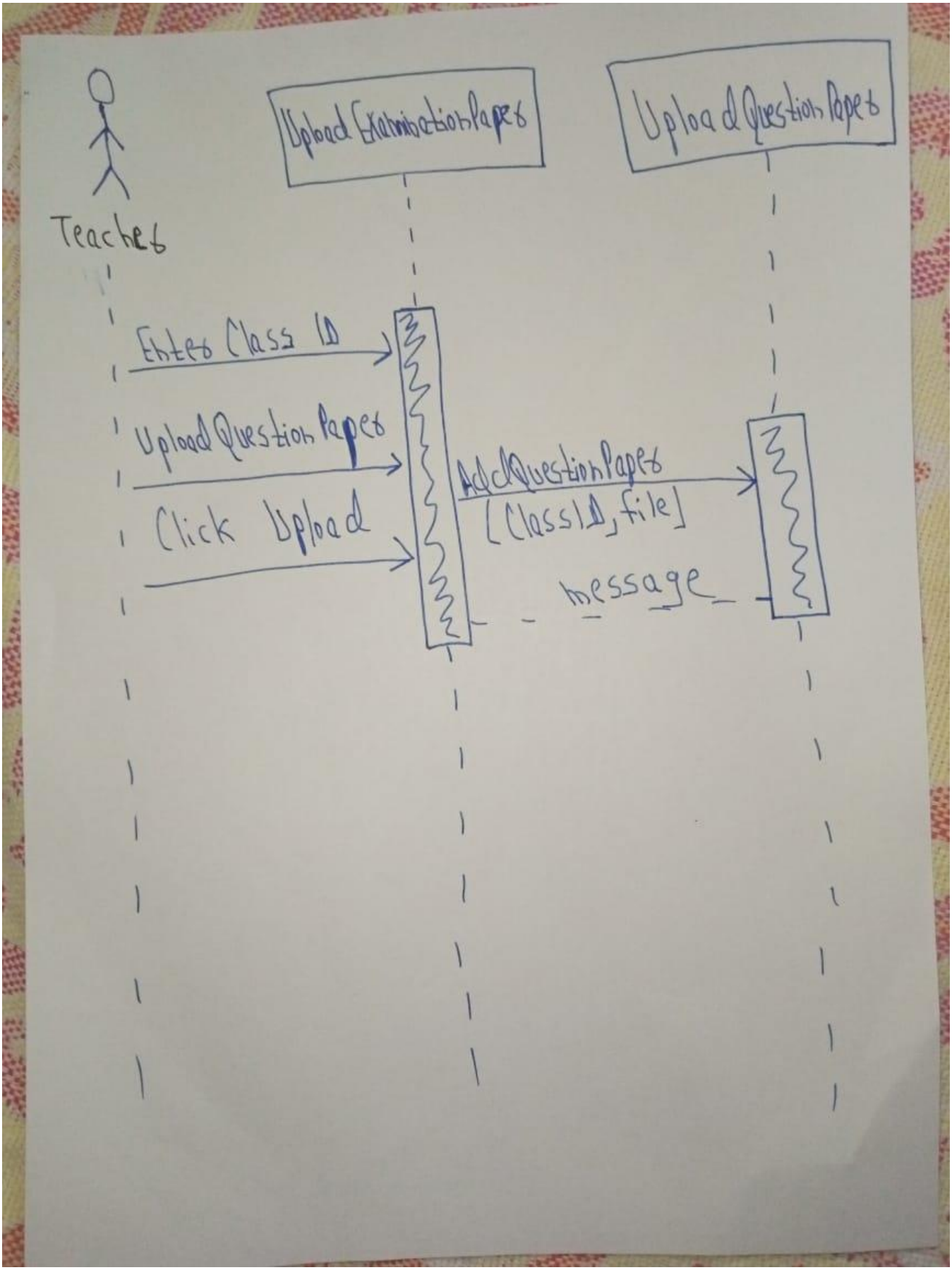
Question 05:

For the use case defined below, create its sequence diagram and finally write down the complete functional code of it.

To conduct an exam of a specific subject, a teacher first creates the question paper. She submits the question paper in the software system. She receives the message from the system that her question paper has been submitted and the exam will start on the given date.

ANSWER:

DIAGRAM:


Teacher
Upload Examination Papers
Upload Question Papers
Enters Class ID
Upload Question Papers
Click Upload
Add Question Papers
[ClassID, file]
message

CODE:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class UploadExaminationPaper : System.Web.UI.Page
{
   protected void Page_Load(object sender, EventArgs e)
   {

   }
   protected void btnUpload_Click(object sender, EventArgs e)
   {
      uploadQuestionPaper uqp = new uploadQuestionPaper ();
      lblMessage.Text= uqp.AddQuestionPaper(txtClassID.Text,FileUpload1);
   }
}
class uploadQuestionPaper
{
   public string AddQuestionPaper(string classid, System.Web.UI.WebControls.FileUpload FileUpload1)
   {
      string message = "";
      if (FileUpload1.HasFile)
         try
         {
            FileUpload1.SaveAs("C:\\ExamQuestionPaper\\"+ classid+"\\" + FileUpload1.FileName);
            message="Question paper has been submitted and the exam will start on the given date"+DateTime.Today.AddDays(2);
         }
         catch (Exception ex)
```

```
        {
            message = "File Not Uploaded Successfully" + ex.Message.ToString();
        }
      else
      {
        message = "Please Select the File & Upload It Again";
      }
      return message;
   }
}
```

Best of Luck 😊