

# ASSIGNMENT 1

## Advanced Programming

Name : Võ Viết Long  
Student Id : GCD18601  
Class : 0704

## **P1 Examine the characteristics of the object-orientated paradigm as well as the various class relationships**

---

- **Define**
- Object Oriented Programming (OOP) is a programming model based on the concept of "object", which can contain data and code: data in the form of fields (often called properties or properties), and code , in procedural form (commonly referred to as methods).
- One characteristic of objects is that the object's own procedures can access and often modify its own data fields (objects that have a concept of "this" or "self"). In OOP, computer programs are designed by turning them out of objects that interact with each other. OOP languages are very diverse, but the most common ones are class-based, meaning that objects that are instances of classes also define their types.
- Many of the most widely used programming languages (such as C ++, Java, Python, etc.) are multi-paradigm and they support object-oriented programming to a greater or lesser extent, usually combined with procedural and imperative programming.

## 1. Encapsulation :

Encapsulation is a way to hide the internal processing properties of an object, other objects that cannot directly change the state can only be accessed through the public methods of that object.

---

```


1    using System;
2
3    namespace oop
4    {
5        class Rectangle
6        {
7            private double length;
8            private double width;
9
10           public void Acceptdetails()
11           {
12               Console.WriteLine("Enter Length: ");
13               length = Convert.ToDouble(Console.ReadLine());
14               Console.WriteLine("Enter Width: ");
15               width = Convert.ToDouble(Console.ReadLine());
16           }
17           public double GetArea()
18           {
19               return length * width;
20           }
21           public void Display()
22           {
23               Console.WriteLine("Length: {0}", length);
24               Console.WriteLine("Width: {0}", width);
25               Console.WriteLine("Area: {0}", GetArea());
26           }
27       }
28       class ExecuteRectangle
29       {
30           static void Main(string[] args)
31           {
32               Rectangle r = new Rectangle();
33               r.Acceptdetails();
34               r.Display();
35               Console.ReadLine();
36           }
37       }
38   }
39

```

we have 2 fields. we have acceptdetails.  
we enter 2 fields. Next we have the get  
area methods to calculate the area of the  
rectangle. We have a display method to  
show all fields to the screen.

## 2. Abstraction :

The process of picking out (*abstracting*) common features of objects and procedures. A programmer would use abstraction, for example, to note that two functions perform almost the same task and can be combined into a single function. Abstraction is one of the most important techniques in software engineering and is closely related to two other important techniques -- *encapsulation* and *information hiding*. All three techniques are used to reduce complexity.



```

1  using System;
2  namespace oop
3  {
4      abstract class Animal
5      {
6          protected string name;
7          protected int age;
8          public Animal(string name, int age)
9          {
10             this.name = name;
11             this.age = age;
12          }
13          public Animal()
14          {
15             this.name = null;
16             this.age = 0;
17          }
18
19          public abstract void eat();
20      }
21      class Dog : Animal
22      {
23
24          public Dog(string name, int age) : base(name, age)
25          {
26             this.name = name;
27             this.age = age;
28          }
29          public Dog() : base()
30          {
31             this.name = null;
32             this.age = 0;
33          }
34          public override void eat()
35          {
36             Console.WriteLine("Mlem Mlem");
37          }
38          public string Bark()
39          {
40             return "Gruuu!";
41          }

```

```

41  class Program
42  {
43      static void Main(string[] args)
44      {
45          Dog dog = new Dog("Lulu", 8);
46          dog.eat();
47          Console.WriteLine(dog.Bark());
48      }
49  }
50

```

Microsoft Visual Studio Debug Console

```

Mlem Mlem
Gruuu!

```

```

1  using System;
2  namespace oop
3  {
4      interface animal
5      {
6          void eat();
7      }
8      class dog : animal
9      {
10         public void eat()
11         {
12             Console.WriteLine(" mlem mlem");
13         }
14     }
15
16

```

### 3. Inheritance

---

- In object-oriented programming (OOP) inheritance is a feature that represents the "is a" relationship between different classes. Inheritance allows a class to have the same behavior as another class and extend or tailor that behavior to provide special action for specific needs.

```
1  using System;
2
3  namespace oop
4  {
5      class Animal
6      {
7          public void Eat()
8          {
9              Console.WriteLine("Everyanimaleatssomething.");
10         }
11         public void DoSomething()
12         {
13             Console.WriteLine("Everyanimaldoessomething.");
14         }
15     }
16     class Cat : Animal
17     {
18         static void Main(String[] args)
19         {
20             Cat objCat = new Cat();
21             objCat.Eat();
22             objCat.DoSomething();
23         }
24     }
25 }
26
```

Everyanimaleatssomething.  
Everyanimaldoessomething.



## 4. Polymorphism :

---

- In [object-oriented programming](#), *polymorphism* refers to a programming language's ability to process objects differently depending on their data type or [class](#). More specifically, it is the ability to redefine *methods* for *derived classes*. For example, given a base class *shape*, polymorphism enables the programmer to define different *area* methods for any number of derived classes, such as circles, rectangles and triangles. No matter what shape an object is, applying the *area* method to it will return the correct results. Polymorphism is considered to be a requirement of any true object-oriented programming language (OOPL).
- a. OverLoading
  - If we create two or more members with the same name but differ in the number of parameters or the type of parameters, it is called member overloading.
  - Method Overload means two or more methods with the same name but with different number of parameters or data type of each parameter.
  - The purpose of method overload is to increase the readability of the program because we don't need to name many different names for the same action

```

1  using System;
2  namespace oop
3  {
4      class PhepTinh
5      {
6          public int Cong(int a, int b)
7          {
8              Console.WriteLine("Cong(" + a + ", " + b + ") --Goi phuong thuc Cong(int a, int b)");
9              return a + b;
10         }
11
12         public int Cong(int a, int b, int c)
13         {
14             Console.WriteLine("Cong(" + a + ", " + b + ", " + c + ") --Goi phuong thuc Cong(int a, int b, int c)");
15             return a + b + c;
16         }
17
18         public float Cong(float a, float b)
19         {
20             Console.WriteLine("Cong(" + a + ", " + b + ") --Goi phuong thuc Cong(float a, float b)");
21             return a + b;
22         }
23
24         public float Cong(int a, float b)
25         {
26             Console.WriteLine("Cong(" + a + ", " + b + ") --Goi phuong thuc Cong(int a, float b)");
27             return a + b;
28         }
29     }
30     class Program
31     {
32         public static void Main(string[] args)
33         {
34             PhepTinh p = new PhepTinh();
35             p.Cong(1, 2);
36             p.Cong(1, 2, 3);
37             p.Cong(1.2f, 1.2f);
38             p.Cong(2, 1.2f);
39             Console.ReadKey();
40         }
41     }

```

In a class we can have many methods with the same name but different numbers of parameters. Then, when we call the method name, the program will base on the number of arguments we passed to choose which method of the class to call.

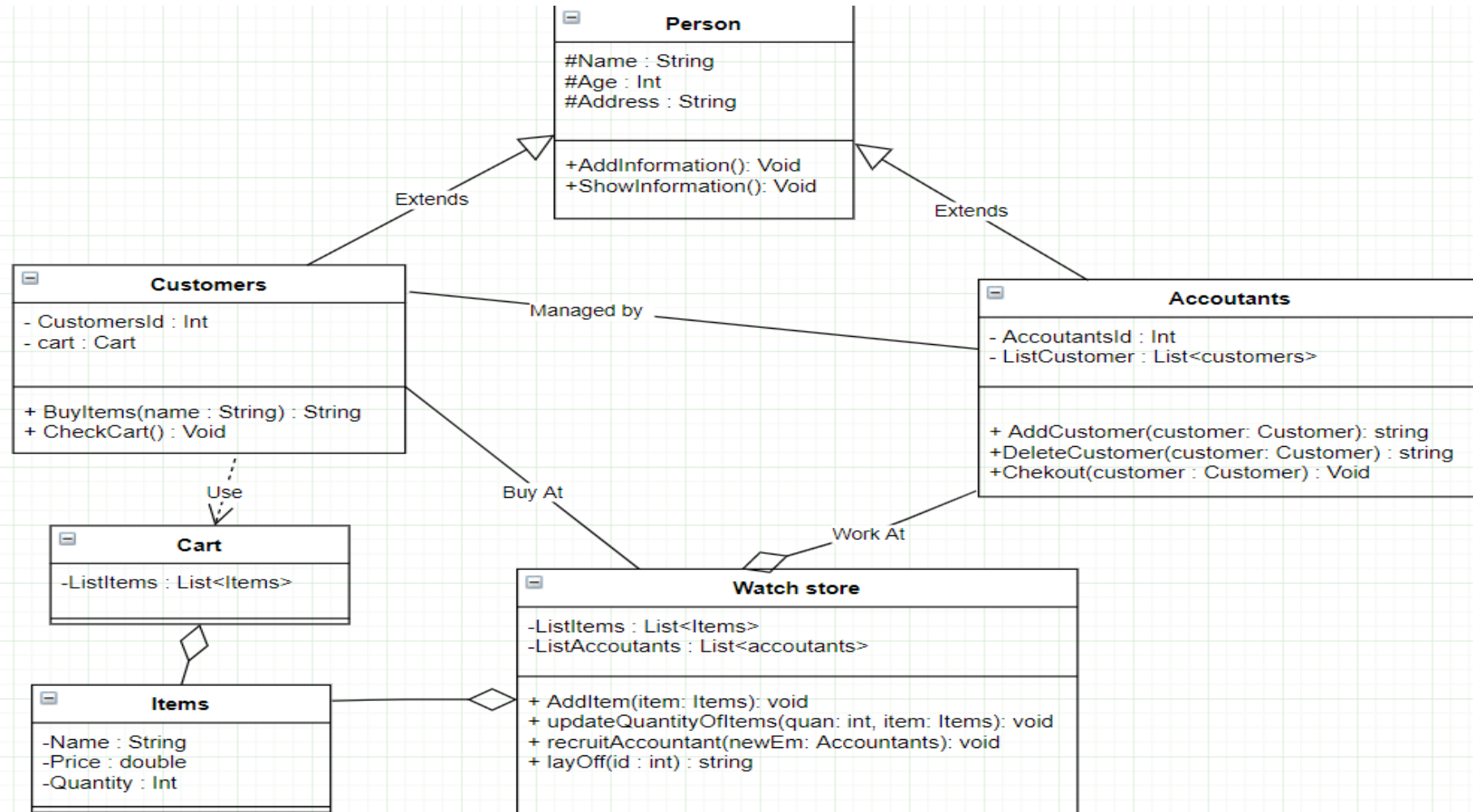
#### Overriding

If a derived class (also called a subclass) defines the same method as defined in its base class (also known as a parent class), then it is called method overriding. In C#, to override methods in C#, we need to use the virtual keyword with the method in the base class (superclass) and the override keyword with the method in the derived class (subclass).

```
1  using System;
2  namespace oop
3  {
4      class DongVat
5      {
6          public virtual void keu()
7          {
8              Console.WriteLine("dong vat keu");
9          }
10     }
11
12     class Cho : DongVat
13     {
14         public override void keu()
15         {
16             base.keu();
17             Console.WriteLine("cho keu gau gau");
18         }
19     }
20     class Program
21     {
22         public static void Main(string[] args)
23         {
24             Cho c = new Cho();
25             c.keu();
26             Console.ReadKey();
27         }
28     }
29 }
```

Dog class inherits from the animal class. But the animal class is different from the dog class, it uses overriding.

P2 Design and build class diagrams using a UML tool.



Everyone in the store is required to provide Name, Age and Address information.

Customers can purchase watches in the store and can check out their shopping cart

Accountants can check the list of customers, add new customers. Check out customer information.

Stores can add new items, update quantities, hire more salespeople, and fire poor employees.

Shopping cart has the function of storing items that customers have selected to buy.

Items must have name, price and quantity

---

Thank For  
Listening

