**Addis Ababa University**

**Addis Ababa Institute of Technology School of Information Technology and Engineering**

# OOP II Assignment

**Mohammed sheihussein ate/4972/11**

1. what is object?

Objects are the first things that come to mind while building a program in object-oriented programming (OOP), and they are also the units of code that emerge from the process. In the interim, each object is turned into a generic object class, with even more generic classes generated to allow objects to share models and reuse class definitions in their code. Each object is an instance of a specific class or subclass, with its own set of methods and data fields. The actual program that executes in the computer is called an object.

2. what is encapsulation?

In object-oriented programming languages, the idea of encapsulation (also known as OOP Encapsulation) refers to the grouping of data and the functions that operate on it into a single unit. In numerous programming languages, encapsulation is widely utilized in the form of classes.

3. what is abstraction?

One of the major principles in object-oriented programming (OOP) languages is abstraction. Its major purpose is to manage complexity by obfuscating irrelevant information from the user. This allows the user to build more complicated logic on top of the offered abstraction without having to understand or even consider all of the hidden complexity.

Abstraction is one of the key concepts of object-oriented programming (OOP) languages. Its main goal is to handle complexity by hiding unnecessary details from the user.

4. which are access specifiers?

Access modifiers are used to implement an important aspect of Object-Oriented Programming known as Data Hiding.

5. what is inheritance?

OOP is all about real objects, and inheritance is one way to represent real relationships. Here are some examples-cars, buses, bicycles-all of which fall into a broader category called vehicles. This means that it inherits the characteristics of a class vehicle, which means that everything is used for transportation. With the help of inheritance, this relationship can be represented in code.

Another attractive aspect of inheritance is that it is transitive in nature. But what does that mean? More on this  later in this article. Python also supports several types of inheritance. This will be discussed in more detail in this article.

6. how can you implement multiple inheritance in c#?

In Multiple inheritance, one class can have more than one superclass and inherit features from all its parent classes.

e.g:-  // C# program to illustrate

```
// multiple class inheritance
using System;
using System.Collections;


// Parent class 1
class Geeks1 {


 // Providing the implementation
 // of languages() method
 public void languages()
 {


  // Creating ArrayList
  ArrayList My_list = new ArrayList();


  // Adding elements in the
  // My_list ArrayList
  My_list.Add("C");
  My_list.Add("C++");
  My_list.Add("C#");
  My_list.Add("Java");

  Console.WriteLine("Languages provided by GeeksforGeeks:");
  foreach(var elements in My_list)
  {
   Console.WriteLine(elements);
  }
 }
}

// Parent class 2
class Geeks2 {

 // Providing the implementation
 // of courses() method
 public void courses()
 {

  // Creating ArrayList
  ArrayList My_list = new ArrayList();

  // Adding elements in the
  // My_list ArrayList
  My_list.Add("System Design");
  My_list.Add("Fork Python");
  My_list.Add("Geeks Classes DSA");
  My_list.Add("Fork Java");

  Console.WriteLine("\nCourses provided by GeeksforGeeks:");
  foreach(var elements in My_list)
  {
   Console.WriteLine(elements);
  }
 }
}

// Child class
class GeeksforGeeks : Geeks1, Geeks2 {
}

public class GFG {

 // Main method
 static public void Main()
 {

  // Creating object of GeeksforGeeks class
  GeeksforGeeks obj = new GeeksforGeeks();
  obj.languages();
  obj.courses();
```

}

}

7. are private class members inherited to the derived class?

Derived classes do not "inherit" them because they do not "inherit" in a way that the private members of the base class are inaccessible. An instance of a

derived class contains an instance of a private member of the base class

ss fo

r obvious reasons.

8. what is polymorphism?

Polymorphism is one of the fundamental concepts of object-oriented programming (OOP) and describes situations in which something occurs in many different forms. In computer science, it describes the concept that you can access objects of different types through the same interface.

9. what is method overloading?

Overloading occurs when you have two methods with the same name but different signatures (or arguments). In a class, we can implement two or more methods with the same name.

10. when and why to use method overloading ?

Overloading Methods

It is very easy to create a class with overloaded methods, just define methods with the same name but  different argument lists.

* Method overloading can be achieved as follows:

* By changing the number of parameters in a method

* By changing the order of parameters in a method

11. what is method overriding?

Method Overriding is a technique that allows the invoking of functions from another class (base class) in the derived class. Creating a method in the derived class with the same signature as a method in the base class is called as method overriding.

12. what is constructor?

A constructor is a special type of member function of a class which initializes objects of a class. In C++, Constructor is automatically called when object(instance of class) create. It is special member function of the class because it does not have any return type.

13. describe some of the key points regarding the constructor?

Some of the key points regarding constructor are

* A class can have any number of constructors.

* A constructor doesn't have any return type, not even void.

* A static constructor can not be a parametrized constructor.

* Within a class, you can create one static constructor only.

14. what is the use of private constructor in c#?

Private constructors are used to prevent creating instances of a class when there are no instance fields or methods, such as the Math class, or when a method is called to obtain an instance of a class

15. can you create object of class with private constructor in c#?

Yes, we can declare a constructor as private. If we declare a constructor as private we are not able to create an object of a class. We can use this private constructor in the Singleton Design Pattern.

16.  what is the use of private constructor in c#?

The use of private constructor is to serve singleton classes. A singleton class is a class that limits the number of objects that can be created into one. By using private constructor we can ensure that  more than one object cannot be created at the same time. By providing a private constructor, you prevent the creation of class instances elsewhere than within the class. We will see in the below example how to use private constructor to limit the number of objects for a singleton class.

17. what is the use of static constructor in c#?

A static constructor is used to initialize any static data, or to perform a particular action that needs to be performed once only. It is called automatically before the first instance is created or any static members are referenced.

Static constructors are useful when creating wrapper classes for unmanaged code, when the constructor can call the LoadLibrary method. Static constructors are also a convenient place to enforce run-time checks on the type parameter that cannot be checked at compile time via constraints.

18. what is Destructors?

estructors in C# are methods inside the class used to destroy instances of that class when they are no longer needed. The Destructor is called implicitly by the .NET Framework’s Garbage collector and therefore programmer has no control as when to invoke the destructor.

19. what is namespace in c#?

Namespaces are used in C# to organize and provide a level of separation of codes. They can be considered as a container which consists of other namespaces, classes, etc.

20. what are virtual override and new keywords in c#?

The virtual keyword is used to modify a method, property, indexer, or event d

eclared in the base class and allow it to be overridden in the derived class.

The override keyword is used to extend or modify a virtual/abstract method, property, indexer, or event of base class into derived class.

The new keyword is used to hide a method, property, indexer, or event of base class into derived class.

21. what is the difference between struct and class in c#?

* Structs are value types, allocated either on the stack or inline in containing types. Classes are reference types, allocated on the heap and garbage-collected.

* Allocations and de-allocations of value types are in general cheaper than allocations and de-allocations of reference types. Assignments of large reference types are cheaper than assignments of large value types.

* In structs, each variable contains its own copy of the data (except in the case of the ref and out parameter variables), and an operation on one variable does not affect another variable.

In classes, two variables can contain the reference of the same object and any operation on one variable can affect another variable.

22. what is interface?

Interface in C# is a blueprint of a class. It is like abstract class because all the methods which are declared inside the interface are abstract methods. It cannot have method body and cannot be instantiated. It is used to achieve multiple inheritance which can't be achieved by class.

23. why to use interface in c#?

By using interfaces, you can, for example, include behavior from multiple sources in a class. That capability is important in C# because the language doesn't support multiple inheritance of classes. In addition, you must use an interface if you want to simulate inheritance for structs, because they can't actually inherit from another struct or class.

24. what is implicity interface implementation?

Implicit implementations tend to be more common and more convenient for usage. They are less verbose and any usage of the concrete type will have the implementations of the members exposed. Implicit implementations don't include the name of the interface being implemented before the member name, so the compiler infers this. The members will be exposed as public and will be accessible when the object is cast as the concrete type.

25. what is explicit interface implementation?

Explicitly telling the compiler that a particular member belongs to that particular interface is called Explicit interface implementation.

If a class implements from more than one interface that has methods with the same signature then the call to such a method will implement the same method and not the interface-specific methods. This will defeat the whole purpose of using different Interfaces.

26. what is abstract class?

An abstract class is a template definition of methods and variables of a class (category of objects) that contains one or more abstracted methods.

27. describe abstract class in detail?

Data abstraction is the process of hiding certain details and showing only essential information to the user.

Abstraction can be achieved with either abstract classes or interfaces (which you will learn more about in the next chapter).

The abstract keyword is used for classes and methods:

Abstract class: is a restricted class that cannot be used to create objects (to access it, it must be inherited from another class).

Abstract method: can only be used in an abstract class, and it does not have a body. The body is provided by the derived class (inherited from).

28. what is the difference between abstraction and encapsulation?

Encapsulation hides variables or some implementation that may be changed so often in a class to prevent outsiders access it directly. They must access it via getter and setter methods.

Abstraction is used to hide something too, but in a higher degree (class, interface). Clients who use an abstract class (or interface) do not care about what it was, they just need to know what it can do.

29. can abstract class be sealed in in c#?

an abstract method or class cannot be declared  sealed.

a subclass of

an abstract class can  be instantiated only if it implements all  the abstract methods of its superclass. Such classes are called concrete classes to distinguish them from abstract classes.

30. can an abstract class have a constructor? If so what is the use?

Yes, an abstract class can have a constructor. In general, a class constructor is used to initialize fields. Along the same lines, an abstract class constructor is used to initialize fields of the abstract class.

31. can you declare abstract methods as private in c#?

if a method of a class is private, you cannot access it outside the current class, not even from its subclasses.

however, in case of  abstract method, you cannot use it from the same class, you must override it from the subclass and use it.

Therefore, an abstract method cannot be private.

32. can abstract class have static methods in c#?y

yes, abstract class can have static methods. The reason  is that static methods don't work on the class instance, they are directly associated with the class itself. So if you write a static method in the class and compile it, and when you try to expose the IL, it will be like any other class accessing the static member.

33. does abstract class support multiple inheritance in c#?

* an abstract class cannot be inherited by structs.

It can contain constructors or destructors.

It can implement functions with non-abstract methods.

It cannot support multiple inheritance.

34. abstract class must have only abstract methods. true or false?

false

35. when do you use abstract class?

It is not possible to define an abstract class and an interface system without distinguishing between them. Once you understand both, you'll be able to decide which one to choose and how to use them. So, let us define the concepts and explain their differences.

36. why can abstract class not be instantiated?

The abstract modifier indicates that the thing being modified has a missing or incomplete implementation. The abstract modifier can be used with classes, methods, properties, indexers, and events. Use the abstract modifier in a class declaration to indicate that a class is intended only to be a base class of other classes, not instantiated on its own. Members marked as abstract must be implemented by non-abstract classes that derive from the abstract class.

37. which type of members can you define in an abstract class?

you can add any type of members in abstract class. In general, the data members of a class should be initialized and assigned to only within the constructor and other member functions of that class.

38. what is operator overloading?

polymorphism: Polymorphism (or operator overloading) is a manner in which OO systems allow the same operator name or symbol to be used for multiple operations. That is, it allows the operator symbol or name to be bound to more than one implementation of the operator.

39. is it possible to restrict object creation in c#?

We can limit the number of object creation of class in C# using the static variable.

Static variable is used to share the value to all instance of that class.

40. can you inherit enum in c#?

 * Enum can not inherit in derived class because by default Enum is sealed.

41. is it possible to achieve method extension using interface?

you can use extension methods to extend a class or interface, but you must not override them. An extension method with the same name and same signature as an interface or class method will never be called. At compile time, extension methods always have lower precedence than instance methods defined in the type itself.

42. is it possible that a method can return multiple values at a time?

no, you cannot return multiple values from a function in C# (for versions lower than C# 7), at least not  the way you can  in Python.

there are some alternatives, however:

 you can return an array of objects of type  with any value you want.

43. what is constant?

constants are immutable values that are known at compile time and do not change throughout the life of the p

rogram. Constants are declared with the const modification. Only built-in types in C# (except System.Object) can be declared as const.

44. what is readonly?

the readonly keyword is a modifier that can be used in four contexts:

in a field declaration, readonly indicates that an assignment to the field can occur only  as part of the declaration or in a constructor of the same one class. Read-only fields can be assigned and reassigned multiple times in the field's declaration and constructor.

cannot assign a read-only field  after exiting the constructor.

45. what is static?

In C#, static means something which cannot be instantiated. You cannot create an object of a static class and cannot access static members using an object.

C# classes, variables, methods, properties, operators, events, and constructors can be defined as static using the static modifier keyword.

46 what is static readonly?

readonly is the keyword whose value we can change during runtime or we can assign it at run time but only through the non-static constructor. Not even a method.