AAiT
ADDIS ABABA INSTITUTE OF TECHNOLOGY
አዲስ አበባ ቴክኖሎጂ ኢንስቲትዩት
ADDIS ABABA UNIVERSITY
አዲስ አበባ ዩኒቨርሲቲ

School of Information Technology and Engineering

OOP II Assignment1

Name:- Tadesse Ageru

ID:- ATE/3174/11

1. **What is an Object?**

Objects are key to understanding *object-oriented* technology. Look around right now and you'll find many examples of real-world objects: your dog, your desk, your television set, your bicycle.

2. **What is Encapsulation?**

In c# Encapsulation is a process of binding the data members and member functions into a single unit. In c# The class is the real-time example for encapsulation because it will combine various types of data members and member functions into a single unit.

Generally, in c# The encapsulation is used to prevent alteration of code (data) accidentally from the outside functions. In c#, by defining the class fields with properties, we can protect the data from accidental corruption

3. **What is Abstraction?**

Abstraction is an important part of object oriented programming. It means that only the required information is visible to the user and the rest of the information is hidden. Abstraction can be implemented using abstract classes in C#. Abstract classes are base classes with partial implementation.

4. **Which are Access Specifiers?**

Access modifiers (or access specifiers) are keywords in object-oriented languages that set the accessibility of classes, methods, and other members. Access modifiers are a specific part of programming language syntax used to facilitate the encapsulation of components.

5. **What is Inheritance?**

In object-oriented programming, inheritance is the mechanism of basing an object or class upon another object or class, retaining similar implementation. Also defined as deriving new classes from existing ones such as super class or base class and then forming them into a hierarchy of classes

6. **How can you implement multiple inheritance in C#?**

C# does support multiple class inheritance with the help of the interface. Using Interface the inherited class (child) can get the features of his parent class.

7. **Are private class members inherited to the derived class?**

No private members of the base-class are accessible within the derived-class and to the instances of derived-class.

8. **What is Polymorphism?**

In programming languages and type theory, polymorphism is the provision of a single interface to entities of different types, or the use of a single symbol to represent multiple different types.

9. **What is method Overloading?**

Method Overloading is the common way of implementing polymorphism. It is the ability to redefine a function in more than one form. A user can implement function overloading by defining two or more functions in a class sharing the same name. C# can distinguish the methods with different method signatures.

10. **When and why to use method Overloading?**

Overloading happens 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.

overloading is used to increase code readability and maintainability. Although it is possible to have methods with the same name that perform a totally different function, it is advised that overloaded methods must have similarities in the way they perform.

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 a Constructor?**

A constructor is a special method of the class which gets automatically invoked whenever an instance of the class is created. Like methods, a constructor also contains the collection of instructions that are executed at the time of Object creation.

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

14. **What is a Private Constructor?**

A private constructor is a special instance constructor. It is generally used in classes that contain static members only. If a class has one or more private constructors and no public constructors, other classes (except nested classes) cannot create instances of this class.

15. **Can you create object of class with private constructor in C#?**

No, because an object of a class having a private constructor cannot be instantiated from outside of the class.

16. **What is the use of private constructor in C#?**

The use of private constructor in c# are:

- It is used to stop object creation of a class.
- It is used to stop a class from being inherited.
- It is used in singleton design patterns, to make sure that the only one instance of a class can ever be created.

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 only once. It is called automatically before the first instance is created or any static members are referenced.

18. **What is a Destructor?**

Destructor is a special method of a class, and it is used in a class to destroy the object or instances of classes. The destructor in c# will invoke automatically whenever the class instances become unreachable. The destructor will invoke automatically whenever an instance of a class is no longer needed.

19. **What is Namespaces?**

A namespace is a declarative region that provides a scope to the identifiers (the names of types, functions, variables, etc) inside it. Namespaces are used to organize code into logical groups and to prevent name collisions that can occur especially when your code base includes multiple libraries

20. **What are Virtual, Override, and New keywords in C#?**

The ***virtual keyword*** is used to modify a method, property, indexer, or event declared 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.

- In case of struct, Allocations and de-allocations of value types are in general cheaper than allocations and de-allocations of reference types.

  In the case of class, 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 an abstract class because all the methods which are declared inside the interface are abstract methods. It cannot have a method body and cannot be instantiated. It is used to achieve multiple inheritance which can't be achieved by class.

23. **Why to use Interfaces in C#?**

An interface may not declare instance data such as fields, auto-implemented properties, or property-like events.

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 Implicit interface implementation?

Implicit Interface Implementations are implemented implicitly by declaring a public member in the class with the same signature of the method as defined in the interface and the same return type. This is how you normally implement interfaces.

25. What is Explicit interface implementation?

Explicit Interface Implementation are Implementing interfaces explicitly, the interface is no longer declared public with the implementation, and the interface member is prefixed with the name of the interface'

26. What is Abstract class?

An abstract class is a special type of class that cannot be instantiated. An abstract class is designed to be inherited by subclasses that either implement or override its methods. In other words, abstract classes are either partially implemented or not implemented at all.

27. Describe Abstract class in detail.

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). An abstract class can have both abstract and regular methods.

To access the abstract class, it must be inherited from another class. The abstract class is used to achieve security - hide certain details and only show the important details of an object. Abstraction can also be achieved with Interfaces.

28. What is the difference between Abstraction and Encapsulation?

Abstraction:- is a process. It is the act of identifying the relevant qualities and behaviors an object should possess.

Encapsulation:- is the mechanism by which the abstraction is implemented.

Abstraction	**Encapsulation**
Abstraction solves the problem at the design level.	Encapsulation solves the problem at the implementation level.
Abstraction is used for hiding the unwanted data and giving only relevant data.	Encapsulation is hiding the code and data into a single unit to protect the data from the outer world.
Abstraction is set focus on the object instead of how it does it.	Encapsulation means hiding the internal details or mechanics of how an object does something.
Abstraction is outer layout in terms of design.	Encapsulation is the inner layout in terms of implementation.

29. Can Abstract class be Sealed in C#?

The abstract method or class cannot be declared as sealed.

30. Can abstract class have Constructors in C#?

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 the child classes of it. But, incase of an abstract method, you cannot use it from the same class, you need to override it from subclass and use. Therefore, the abstract method cannot be private.

32. **Can abstract class have static methods in C#?**

Yes, abstract classes can have Static Methods. The reason for this is Static methods do not work on the instance of the class, they are directly associated with the class itself.

33. **Does Abstract class support multiple Inheritance?**

This is not allowed because you can do more than this with abstract classes. It wouldn't make sense to allow multiple inheritance, provided you only used an abstract class when you could have used an interface. So an abstract class cannot be inherited by structures. 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. Is it true or false?**

<u>True</u>

35. **When do you use Abstract Class?**

Generally, we use abstract classes at the time of inheritance. A user must use the override keyword before the method which is declared as abstract in child class, the abstract class is used to inherit in the child class. An abstract class cannot be inherited by structures. It can contains constructors or destructors

36. **Why can Abstract class not be Instantiated?**

An abstract class cannot be instantiated because it may contain members that are abstract and have no implementation.

37. **Which type of members can you define in an Abstract class?**

You can add any type of member in an abstract class. 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?**

In computer programming, operator overloading, sometimes termed operator ad hoc polymorphism, is a specific case of polymorphism, where different operators have different implementations depending on their arguments. Operator overloading is generally defined by a programming language, a programmer, or both.

Operator overloading gives the ability to use the same operator to do various operations. It provides additional capabilities to C# operators when they are applied to user-defined data types.

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. Note : In the above sample, we have created the static variable count, which will hold the incremented count value while creating the instance of that class.

40. **Can you inherit Enum in C#?**

No, it is not possible. Enum can not inherit from a derived class because by default Enum is sealed.

41. **Is it possible to achieve Method extension using Interface?**

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

42. **Is it possible that a Method can return multiple values at a time?**

No, we can't return multiple values from a function in C#

43. **What is Constant?**

A constant is a value or variable that can't be changed in the program.

44. **What is Readonly?**

Readonly is a keyword that used to define a variable which can be assigned once after declaration either during declaration or in 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. static is a key word used for both classes and attributes. If it is used with attributes in a class, it can be used with class. There is no need of instantiation of C# Objects. It can be used with C# definition.

46. **What is Static ReadOnly?**

A Static Readonly type variable's value can be assigned at runtime or assigned at compile time and changed at runtime. But this variable's value can only be changed in the static constructor. And cannot be changed further. It can change only once at runtime.