

**Addis Ababa University**

**Addis Ababa institute of Technology**

OOP II

Assignment

NAME – YABSRA MELAKU

ID – ATE/4107/11

1. **What is an Object?**

is an instance of a class that is created dynamically.Object is also a keyword that is an alias for the predefined type System.

2. **What is Encapsulation?**

The bundling up of data into a single unit is referred to as encapsulation. It's the structure that binds code and the data it manipulates together. Encapsulation, on the other hand, is a protective shield that prohibits data from being accessible by code outside of the shield.

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?**

C# has 5 access specifier or access modifier keywords; those are private, public, internal, protected and protected Internal

5. **What is Inheritance?**

Inheritance is an important pillar of OOP(Object Oriented Programming). It is the mechanism in C# by which one class is allowed to inherit the features(fields and methods) of another class.

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. But C# does not support multiple class inheritance. To overcome this problem we use interfaces to achieve multiple class inheritance.

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

The derived class doesn't "inherit" the private members of the base class in any way - it can't access them, so it doesn't "inherit" them. An instance of the derived class contains instances of the private members of the base class, for obvious reasons.

8. **What is Polymorphism?**

Polymorphism, in C#, is the ability of objects of different types to provide a unique interface for different implementations of methods. It is usually used in the context of late binding, where the behavior of an object to respond to a call to its method members is determined based on object type at run time

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?**

when you actually do need multiple methods with different parameters, but the methods do the same thing. That is, don't use overloading if the multiple methods perform different tasks.

The advantage of method overloading is that it increases the readability of the program because you don't need to use different names for same action.

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 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 cannot be a parametrized constructor. Within a class, you can create one static constructor only.

14. **What is 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, object of a class having private constructor cannot be instantiated from outside of the class.

16. **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

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 Destructor?**

Destructors 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 Namespaces?**

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. A namespace can have following types as its members: Namespaces (Nested Namespace) Classes.

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

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 while classes are reference types. Structs can be instantiated without using a new operator. A struct cannot inherit from another struct or class, and it cannot be the base of a class. All structs inherit directly from System.

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 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.

24. **What is Implicit interface implementation?**

Implicit interface implementation

This is the most regular or obvious way to implement members of an interface. Here we don't specify the interface name of the members and implement implicitly. The method can be declared at any interface (s) the class implements

25. **What is Explicit interface implementation?**

Interfaces are implemented implicit 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.

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.**

An abstract class is a template definition of methods and variables of a class (category of objects) that contains one or more abstracted  methods. Abstract classes are used in all object-oriented programming languages, including java (see Java abstract class, c++,c# and VB.NET. Objects or classes may be abstracted, which means that they are summarized into characteristics that are relevant to the current program’s operation.

Individual instances resulting from classes are objects. Declaring a class as abstract means that it cannot be directly instantiated  , which means that an object cannot be created from it. That protects the code from being used incorrectly. Abstract classes require subclasses to further define attributes necessary for individual instantiation. Abstract classes contrast with concrete classes, which are the default type. A concrete class has no abstracted methods and can be instantiated and used in code.

Abstract classes aren’t required in programming but the concept is provided to keep code cleaner than it would be otherwise and make programming more efficient because extraneous details are not constantly being referred to.

28. **What is the difference between Abstraction and Encapsulation?**

Abstraction is the method of hiding the unwanted information. Whereas encapsulation is a method to hide the data in a single entity or unit along with a method to protect information from outside. We can implement abstraction using abstract class and interfaces

29. **Can Abstract class be Sealed in C#?**

The abstract method or class cannot be declared as sealed. A subclass of an abstract class can only be instantiated if it implements all of the abstract methods of its superclass. Such classes are called concrete classes to differentiate them from abstract classes

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 class 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

34. **Abstract class must have only abstract methods. Is it true or false?**

An abstract class is a class that is declared abstract —it may or may not include abstract methods. Abstract classes cannot be instantiated, but they can be subclassed. When an abstract class is subclassed, the subclass usually provides implementations for all of the abstract methods in its parent class.

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

An abstract class is used if you want to provide a common, implemented functionality among all the implementations of the component. Abstract classes will allow you to partially implement your class, whereas interfaces would have no implementation for any members

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

An abstract class cannot be instantiated. ... The sealed modifier prevents a class from being inherited and the abstract modifier requires a class to be inherited. A non-abstract class derived from an abstract class must include actual implementations of all inherited abstract methods and accessors.

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?**

Operator Overloading is the method by which we can change the function of some specific operators to do some different task

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#?**

Enums cannot inherit from other enums. In fact all enums must actually inherit from System

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

You 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, you can't return multiple values from a function in C# (for versions lower than C# 7), at least not in the way you can do it in Python. However, there are a couple alternatives: You can return an array of type object with the multiple values you want in it

43. **What is Constant?**

Constants are immutable values which are known at compile time and do not change for the life of the program. Constants are declared with the const modifier

44. **What is Readonly?**

The readonly keyword is a modifier that can be used in four contexts: In a field declaration, readonly indicates that assignment to the field can only occur as part of the declaration or in a constructor in the same class. ... A readonly field can't be assigned after the constructor exits.

45**. What is Static?**

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?**

A Static Readonly type variable's value can be assigned at runtime or assigned at compile time and changed at runtime.