

**OOP II Assignment**

[**Yosef Sahle**](mailto:yosefsahle48@gmail.com) **ATE/7950/11**

**Submitted to** [**Kabila Haile**](mailto:%20kabilahaile@gmail.com)

**1. what is Object?**

It is a basic unit of object-oriented programing and represents the real-life entities. And it is consists state, behavior and identity.

**2. What is Encapsulation?**

Encapsulation in OOP Meaning: In object-oriented computer programming languages, the notion of encapsulation (or OOP Encapsulation) refers to the bundling of data, along with the methods that operate on that data, into a single unit. Many programming languages use encapsulation frequently in the form of classes.

**3. What is Abstraction?**

Abstraction occurs when a programmer hides any irrelevant data about an object or an instantiated class to reduce complexity and help users interact with a program more efficiently.

**4. Which are Access Specifiers?**

Access modifiers and specifiers are keywords (private, public, internal, protected and protected internal) to specify the accessibility of a type and its members.

**5. What is Inheritance?**

Inheritance is the procedure in which one class inherits the attributes and methods of another class

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

multiple inheritance can be achieved in C# using interfaces. This is the simple mathematical operation programing demonstrating how multiple inheritance can be achieved in C# using interface concept.

**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.

**8. What is Polymorphism?**

Polymorphism is one of the core concepts of object-oriented programming (OOP) and describes situations in which something occurs in several 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?**

Method overloading is a form of polymorphism in OOP. 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.

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

If we need to do the same kind of the operation in different ways i.e. for different inputs. In the example described below, we are doing the addition operation for different inputs. It is hard to find many different meaningful names for single action.

Method overloading can be done by changing:

1. The number of parameters in two methods.
2. The data types of the parameters of methods.
3. The Order of the parameters of methods.

**11. What is method Overriding?**

Method overriding, in object-oriented programming, is a language feature that allows a subclass or child class to provide a specific implementation of a method that is already provided by one of its super classes or parent classes.

**12. What is Constructor?**

In [class-based](https://en.wikipedia.org/wiki/Class-based_programming) [object-oriented programming](https://en.wikipedia.org/wiki/Object-oriented_programming), a constructor (abbreviation: ctor) is a special type of [subroutine](https://en.wikipedia.org/wiki/Subroutine) called to [create an object](https://en.wikipedia.org/wiki/Object_creation). It prepares the new object for use, often accepting [arguments](https://en.wikipedia.org/wiki/Parameter_(computer_programming)) that the constructor uses to set required [member variables](https://en.wikipedia.org/wiki/Member_variable).

A constructor resembles an [instance method](https://en.wikipedia.org/wiki/Method_(computer_science)), but it differs from a method in that it has no explicit [return type](https://en.wikipedia.org/wiki/Return_type), it is not implicitly [inherited](https://en.wikipedia.org/wiki/Inheritance_(object-oriented_programming)) and it usually has different rules for scope modifiers. Constructors often have the same name as the declaring [class](https://en.wikipedia.org/wiki/Class_(computer_science)). They have the task of [initializing](https://en.wikipedia.org/wiki/Initialization_(computing)) the object's [data members](https://en.wikipedia.org/wiki/Data_member) and of establishing the [invariant of the class](https://en.wikipedia.org/wiki/Class_invariant), failing if the invariant is invalid. A properly written constructor leaves the resulting [object](https://en.wikipedia.org/wiki/Object_(computer_science)) in a valid state. [Immutable objects](https://en.wikipedia.org/wiki/Immutable_object) must be initialized in a constructor.

**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 Private Constructor?**

When a constructor is created with a private specifier, it is not possible for other classes to derive from this class, neither is it possible to create an instance of this class. They are usually used in classes that contain static members only. Some key points of a private constructor are:

**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#?**

* It is used to stop object creation of a class.
* It is used to stop a class to be 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](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/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?**

In object-oriented programming, a destructor (sometimes abbreviated dtor) is a method which is invoked mechanically just before the memory of the object is released.

**19. What is Namespaces?**

A namespace is designed for providing a way to keep one set of names separate from another. The class names declared in one namespace does not conflict with the same class names declared in another.

**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

1. Structs are value types, allocated either on the stack or inline in containing types.
2. Allocations and de-allocations of value types are in general cheaper than allocations and de-allocations of reference types.
3. 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.

classes

1. Classes are reference types, allocated on the heap and garbage-collected.
2. Assignments of large reference types are cheaper than assignments of large value types.
3. 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?**

An interface contains definitions for a group of related functionalities that a non-abstract [class](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/class) or a [struct](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/struct) must implement. An interface may define static methods, which must have an implementation.

**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?**

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.

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

An explicit interface implementation is a class member that is only called through the specified interface. Name the class member by prefixing it with the name of the interface and a period.

**26. What is Abstract class?**

**Abstraction** is the concept of object-oriented programming that “shows” only essential attributes and “hides” unnecessary information. The main purpose of abstraction is hiding the unnecessary details from the users. Abstraction is selecting data from a larger pool to show only relevant details of the object to the user. It helps in reducing programming complexity and efforts. It is one of the most important concepts of OOPs.

**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 (OOP) 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. To be platform-agnostic, Java code is compiled into class files that can be interpreted by any Java VM. The resulting class file can run on different machines once a compatible Java VM has been downloaded and installed for the OS platform.

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

Difference between Abstraction and Encapsulation:

| Abstraction | Encapsulation |
| --- | --- |
| Abstraction is the process or method of gaining the information. | While encapsulation is the process or method to contain the information. |
| In abstraction, problems are solved at the design or interface level. | While in encapsulation, problems are solved at the implementation level. |
| 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. | Whereas encapsulation can be implemented using by access modifier i.e. private, protected and public. |
| In abstraction, implementation complexities are hidden using abstract classes and interfaces. | While in encapsulation, the data is hidden using methods of getters and setters. |
| The objects that help to perform abstraction are encapsulated. | Whereas the objects that result in encapsulation need not be abstracted. |

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

When a class is declared sealed, it cannot be inherited, abstract classes cannot be declared sealed.

To prevent being overridden, use the sealed in C#. When you use sealed modifiers in C# on a method, then the method loses its capabilities of overriding. The sealed method should be part of a derived class and the method must be an overridden method.

**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.

**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. So if you write a static method in the class and compile it, and when you try to view the IL, it will be same as any other class accessing the static member.

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

* An abstract class cannot be inherited by structures.
* It can contains 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?**

TRUE

**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 whatsoever.

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

Abstract classes cannot be instantiated, but they can be subclasses. When an abstract class is sub classed, the subclass usually provides implementations for all of the abstract methods in its parent class. However, if it does not, then the subclass must also be declared abstract.

**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. To do otherwise breaks encapsulation, thereby making maintenance and modification of the class more difficult.

**38. What is Operator Overloading?**

Operator overloading is a technique by which operators used in a programming language are implemented in user-defined types with customized logic that is based on the types of arguments passed.  
  
Operator overloading facilitates the specification of user-defined implementation for operations wherein one or both operands are of user-defined class or structure type. This helps user-defined types to behave much like the fundamental primitive data types. Operator overloading is helpful in cases where the operators used for certain types provide semantics related to the domain context and syntactic support as found in the programming language. It is used for syntactical convenience, readability and maintainability.

**39. Is it possible to restrict object creation in C#?**

YES! 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#?**

No, not possible. Enums are by default sealed. So, we can’t inherit.

**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?**

As per the Java Language Specification, the methods in Java can return only one value at a time. So returning multiple values from a method is theoretically not possible in Java. But the beauty of Java lies in the fact that we can do desired things with some smart workarounds. This post provides an overview of some of the available alternatives to accomplish this.

**43. What is Constant?**

A **constant** is a value that cannot be altered by the program during normal execution, i.e., the value is constant. When associated with an identifier, a constant is said to be “named,” although the terms “constant” and “named constant” are often used interchangeably. This is contrasted with a **variable**, which is an identifier with a value that can be changed during normal execution, i.e., the value is variable.

**44. What is Read-only?**

Read-only 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.

A read-only object is an object whose data fields can be viewed but cannot be modified. ... For example, the statement “public final int x = 5;” makes the variable x viewable by other outside objects but non-modifiable. Once the variable x is set to a value, it cannot be modified again.

**45. What is Static?**

In object-oriented programming, there is also the concept of a static member variable, which is a "class variable" of a statically defined class, i.e., a member variable of a given class which is shared across all instances (objects), and is accessible as a member variable of these objects.

**46. What is Static Read-only?**

A Static Read-only 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. Let's understand it practically.