1/14/2022

Object Oriented Programming II

Assignment 1: Conceptual Questions

Abel Tadesse

ATE/1349/11

Submitted to Kabila Haile

1 What is an Object?

An object in the real world is a thing, such as a car or a person, whereas an object in programming often represents something in the real world, such as a product or bank account, but this can also be something more abstract.

2 What is Encapsulation?

Encapsulation is the combination of the data and actions that are related to an object. For example, a Bank Account type might have data, such as Balance and Account Name, as well as actions, such as Deposit and Withdraw. When encapsulating, you often want to control what can access those actions and the data, for example, restricting how the internal state of an object can be accessed or modified from the outside.

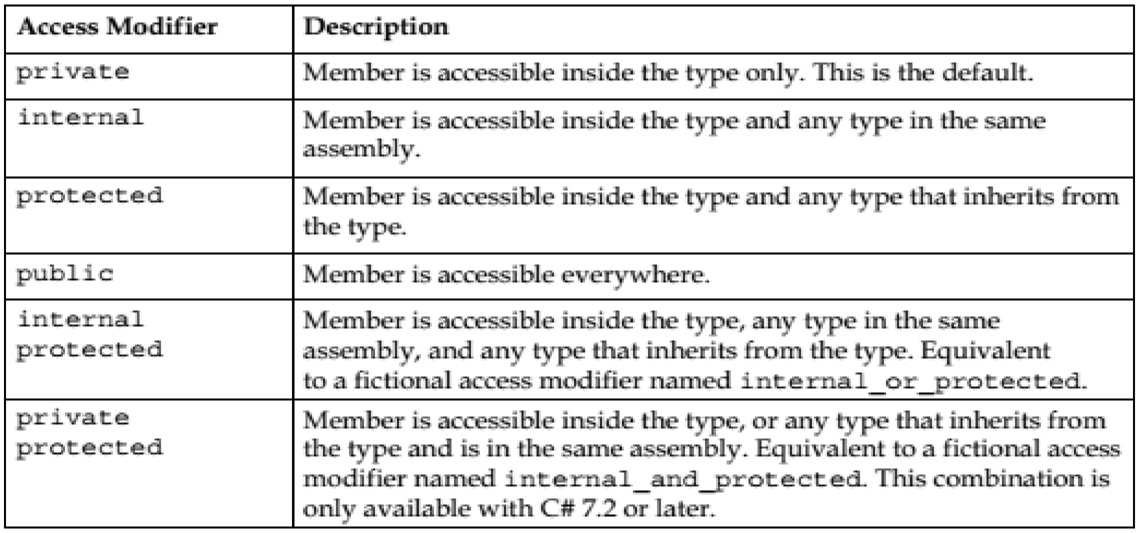
Encapsulation may also refer to a mechanism of restricting the direct access to some components of an object, such that users cannot access state values for all of the variables of a particular object. Encapsulation can be used to hide both data members and data functions or methods associated with an instantiated class or object.

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. The term abstraction vs encapsulation can be used to describe the process of hiding some of the information contained in an object or class, but it may also refer to the object itself. An abstraction is any named entity that contains a selection of data and behaviors specific to a particular usage of the originating entity.

4 Which are Access Specifiers?

we explicitly applied the public keyword to these fields. If we hadn't, then they would be implicitly private to the class, which means they are accessible only inside the class. There are four access modifier keywords, and two combinations of access modifier keywords that you can apply to a class member, such as a field or method, as shown in the following table:



Access Modifier	Description
`private`	Member is accessible inside the type only. This is the default.
`internal`	Member is accessible inside the type and any type in the same assembly.
`protected`	Member is accessible inside the type and any type that inherits from the type.
`public`	Member is accessible everywhere.
`internal protected`	Member is accessible inside the type, any type in the same assembly, and any type that inherits from the type. Equivalent to a fictional access modifier named `internal_or_protected`.
`private protected`	Member is accessible inside the type, or any type that inherits from the type and is in the same assembly. Equivalent to a fictional access modifier named `internal_and_protected`. This combination is only available with C# 7.2 or later.

5 What is Inheritance?

Inheritance is about reusing code by having a subclass derive from a base or super class. All functionality in the base class is inherited by and becomes available in the derived class. For example, the base or super Exception class has some members that have the same implementation across all exceptions, and the sub or derived SqlException class inherits those members and has extra members only relevant to when an SQL database exception occurs like a property for the database connection

Different Types of Inheritance

Single inheritance: In this inheritance, a derived class is created from a single base class.

Multi-level inheritance: In this inheritance, a derived class is created from another derived class.

Multiple inheritance: In this inheritance, a derived class is created from more than one base class.

Multipath inheritance: In this inheritance, a derived class is created from another derived classes and the same base class of another derived classes.

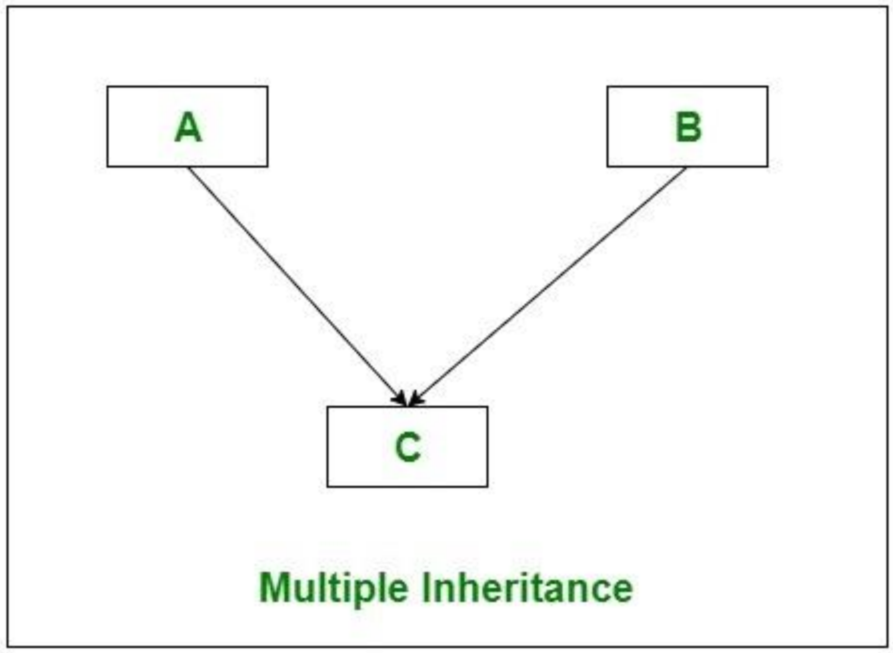
Hierarchical Inheritance: In this inheritance, more than one derived classes are created from a single base class and father child classes act as parent classes for more than one child classes.

Hybrid inheritance: This is combination of more than one inheritance. Hence, it may be a combination of Multilevel and Multiple inheritance or Hierarchical and

Multilevel inheritance or Hierarchical and Multipath inheritance or Hierarchical, Multilevel and Multiple inheritance.

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. As shown in the below diagram, class C inherits the features of class A and B.



But C# does not support multiple class inheritance. To overcome this problem we use interfaces to achieve multiple class inheritance. With the help of the interface, class C (as shown in the above diagram) can get the features of class A and B.

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 is about allowing a derived class to override an inherited action to provide custom behavior.

The object-oriented programming language processes classes and objects by a single interface. It implements the concepts of function overloading, overriding, and virtual functions. Also, it is typically used for instrumenting inheritance in programming.

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**. i.e. the methods can have the same name but with different parameters list (i.e. the number of the parameters, order of the parameters, and data types of the parameters) within the same class.

- Overloaded methods are differentiated based on the number and type of the parameters passed as arguments to the methods.
- You cannot define more than one method with the same name, Order and the type of the arguments. It would be compiler error.
- The compiler does not consider the return type while differentiating the overloaded method. But you cannot declare two methods with the same signature and different return type. It will throw a compile-time error. If both methods have the same parameter types, but different return type, then it is not possible.

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.

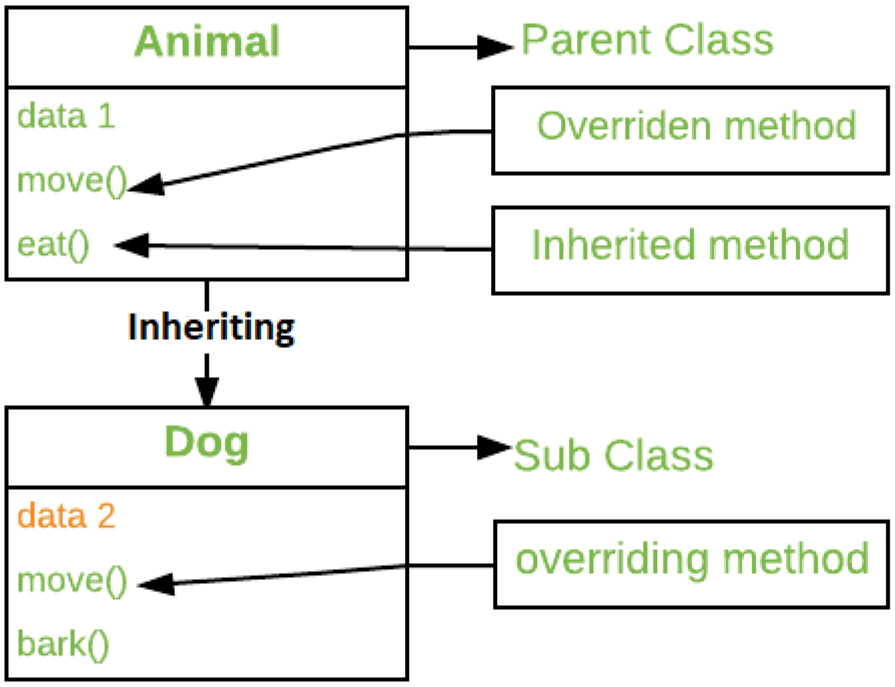
Method overloading can be done by changing:

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 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

The method that is overridden by an override declaration is called the overridden base method. An override method is a new implementation of a member that is inherited from a base class. The overridden base method must be virtual, abstract, or override.



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.

13Describe 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.

16What 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. An instance variable or an object is eligible for destruction when it is no longer reachable.

- A Destructor is unique to its class i.e. there cannot be more than one destructor in a class.
- A Destructor has no return type and has exactly the same name as the class name (Including the same case).
- It is distinguished apart from a constructor because of the *Tilde symbol (~)* prefixed to its name.
- A Destructor does not accept any parameters and modifiers.

- It cannot be defined in Structures. It is only used with classes.
- It cannot be overloaded or inherited.
- It is called when the program exits.
- Internally, Destructor called the Finalize method on the base class of object.

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 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#?

S.N	Struct	Classes
1	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.
2	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.
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	In classes, two variables can contain the reference of the same object and any operation on one

	operation on one variable does not affect another variable.	variable can affect another variable.

In this way, struct should be used only when you are sure that,

- It logically represents a single value, like primitive types (int, double, etc.).
- It is immutable.
- It should not be boxed and un-boxed frequently.

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.

23Why 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

25What 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 (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#?

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

31Can 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, in case 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 Abstract class have static methods in C#?Can

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

35 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.

36 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.

37 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.

Java does not support operator overloading, except for string concatenation for which it overloads the + operator internally.

38 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.

39 Can you inherit Enum in C#?

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

40 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.

41 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.

42 43What 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.

43 What is 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.

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.

44 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.

45 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. Let's understand it practically.