ADDIS ABABA UNIVERSITY
ADDIS ABABA INSTITUTE OF TECHNOLOGY
SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING
OOP II ASSIGNMENT 1

ATE/6954/11 Kidus Girma

Submitted to:- Kabila Haile

Conceptual Questions

1. What is an Object?
   - **An object is a block of memory that has been allocated and configured according to the blueprint which can be stored in either a named variable or in an array or collection.**
2. What is Encapsulation?
   - **It is a process of binding the data members and member functions into a single unit and it is used to prevent alteration of code accidentally from the outside functions.**
3. What is Abstraction?
   - **It process of identifying the relevant qualities and behaviors an object should possess. Also,it is a process of hiding work style of an object and showing only those information required to understand the object.**

4. Which are Access Specifiers?
   - There are 5 types,
     - **private:** limits the accessibility of a member to within the defined type, for example if a variable or a functions is being created in a ClassA and declared as private then another ClassB can't access that.
     - **public:** has no limits, any members or types defined as public can be accessed within the class, assembly even outside the assembly.
     - **protected:** plays a role only when inheritance is used. In other words any protected type or member becomes accessible when a child is inherited by the parent. In other cases (when no inheritance) protected members and types are not visible.

- **internal:** internal plays an important role when we want our class members to be accessible within the assembly. An assembly is the produced .dll or .exe from your .NET Language code (C#). therefore, if you have a C# project that has ClassA, ClassB and ClassC then any internal type and members will become accessible across the classes with in the assembly.

- **Protected internal:** is a combination of protected and internal both. A protected internal will be accessible within the assembly due to its internal flavor and also via inheritance due to its protected flavor.

5. What is Inheritance?
   - **is the ability to create a class that inherits attributes and behaviors from an existing class. The newly created class is the derived (or child) class and the existing class is the base (or parent) class.**
6. How can you implement multiple inheritance in C#?
   - **In multiple inheritance one class can have more that one superclass and inherit features from all its parent classes. For example,we have two interface called class A and class B. But C# doesn't support multiple class inheritance.So, to solve this problem we use interfaces to achieve multiple class inheritance. From an other class called C which inherits both class A & B.**

7. Are private class members inherited to the derived class?
   - **The derived class can't access private class members so it doesn't inherit them.**
8. What is Polymorphism?
   - **Is the ability of objects of different types to provide a unique interface for different implementations of methods.**
9. What is 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.**
10. When and why to use method Overloading?
    - **Why we use it is because it increases the readability of the program because you don't need to use different names for same action.**
11. What is method Overriding?
    - **is a technique that allows the invoking of functions from another class (base class) in the derived class.**
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. It contains the collection of instructions that are executed at the time of Object creation.**

13. Describe some of the key points regarding the Constructor.

- **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?
    - **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#?
    - **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 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.**
18. What is Destructor?
    - **Is a method inside the class used to destroy instances of that class when they are no longer needed.**
19. What is Namespaces?
    - **They are used to organize too many classes so that it can be easy to handle the application.**
20. What are Virtual, Override, and New keywords in C#?
    - **The Virtual keyword is used for generating a virtual path for its derived classes on implementing method overriding. The Virtual keyword is used within a set with an override keyword.**
    - **The Override keyword is used in the derived class of the base class in order to override the base class method. The Override keyword is used with the virtual keyword.**
    - **The New keyword is used to create an object or instantiate an object.**

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?
    - **An interface defines a contract. Any class or struct that implements that contract must provide an implementation of the members defined in the interface. It is a collection of method and property declarations.**

23. Why to use Interfaces in C#?
    - **To include behaviour from multiple sources in a class.**
24. What is Implicit interface implementation?
    - **Is making the members of the interface public in the class.**
25. What is Explicit interface implementation?
    - **in the class the interface members are not declared as public members and cannot be directly accessed using an instance of the class, but a cast to the interface allows accessing the members**
26. What is Abstract class?
    - **An abstract class is a special type of class that cannot be instantiated. It is designed to be inherited by subclasses that either implement or override its methods.**
27. Describe Abstract class in detail.
    - **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. You can have functionality in your abstract class—the methods in an abstract class can be both abstract and concrete. An abstract class can have constructors—this is one major difference between an abstract class and an interface. You can take advantage of abstract classes to design components and specify some level of common functionality that must be implemented by derived classes.**

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 solves the problem in the design level.while Encapsulation solves the problem in the implementation level.**
    - **Abstraction is used for hiding the unwanted data and giving only relevant data.while Encapsulation is hiding the code and data into a single unit to protect the data from outer world.**

29. Can Abstract class be Sealed in C#?
    - **When a class is declared sealed, it cannot be inherited, therefore,abstract classes cannot be declared sealed.**
30. Can abstract class have Constructors in C#?
    - **Yes, an abstract class can have a constructor. a class constructor is used to initialize fields.**
31. Can you declare abstract methods as private in C#?
    - **If a method of a class is private, we cannot access it outside the current class, not even from the child classes of it. But, incase of an abstract method, we 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?
    - **No it doesn't. 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?
    - **False. we can have an abstract class without Abstract Methods.**

35. When do you use Abstract Class?
    - **It is used if we want to provide a common, implemented functionality among all the implementations of the component. Abstract classes will allow us to partially implement our class.**
36. Why can Abstract class not 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 members in 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?
    - **Gives the ability to use the same operator to do various operations. It provides additional capabilities to operators when they are applied to user-defined data types**
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. it is not possible. Enum can’t inherit in derived class because by default Enum is sealed.**
41. Is it possible to achieve Method extension using Interface?
    - **Yes ,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.**
42. Is it possible that a Method can return multiple values at a time?
    - **No, you can't return multiple values from a function**.
43. What is Constant?
    - **Constants are immutable values which are known at compile time and do not change for the life of the program**.
44. What is Readonly?
    - **Readonly keyword is a modifier that can be used 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. We cannot create an object of a static class and cannot access static members using an object.**

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.**