Object-Oriented Programming Questions
C-CAT Specific Previous Year & Sample Questions

1. Which of the following is **not** a core part of Object-Oriented Programming (OOP) concepts?
    1. **Multitasking**
    2. Information Hiding
    3. Polymorphism
    4. Type Checking
2. Which one of the following is considered the first *fully* Object-Oriented Language (a 'pure' OOP language)?
    1. **Simula**
    2. Java
    3. C++
    4. Smalltalk
3. Programming without inheritance is often referred to as:
    1. **Programming with abstract data types**
    2. Not object-oriented
    3. Object-oriented
    4. None of the above
4. To be accessed from a member function of a derived class, data in the base class must be declared as:
    1. **`public`**
    2. `default` (Not a C++ keyword)
    3. `protected`
    4. **`public` or `protected`**
5. What do you mean by **passive objects**?
    1. Passive objects do not take part in the interaction between objects.
    2. Passive objects are ones which passively wait for a message to be processed, waiting for another object that requires its services.
    3. Both 1 and 2
    4. None of the above
6. What is a **non-persistent object**?
    1. A non-persistent object is said to be transient or ephemeral, existing only during program execution. By default, objects are considered as non-persistent.
    2. It has no existence.
    3. Both 1 and 2
    4. None of the above
7. A genuine demerit of the `friend` function is that it violates encapsulation. Which of the following, if true, would be an additional demerit?
    1. **Objects will occupy maximum size of the memory.**
    2. **We can't use run-time polymorphism concepts in those members.**
    3. **Both 1 and 2**
    4. None of the above
8. Runtime polymorphism is achieved only when a virtual function is accessed through a ___________ to the base class.

    1. dot operator
    2. **`this` operator**
    3. pointer
    4. None of the above
9. Which operator **cannot** be overloaded as a non-member (friend) function in C++?
    1. `==`
    2. **`=` (Assignment operator)**
    3. `+`
    4. `>>`
10. Data duplication in diamond inheritance is avoided by making the base class ___________.
    1. pure virtual
    2. virtual
    3. friend
    4. generic
11. Object-oriented technology's ___________ feature means that a small change in user requirements should not require large changes to be made to the system.
    1. **Modularity**
    2. Encapsulation
    3. Modelling
    4. Abstraction
12. A complex object containing other objects where the contained objects cannot exist independently is specifically called _______?
    1. Aggregation
    2. Composition
    3. Association
    4. None of these options
13. Inheritance (Generalization/Specialization) in UML can be represented by:
    1. A line with a hollow arrowhead pointing in the direction of the parent or superclass.
    2. Only with a line between base class & derived classes.
    3. Diamond shape between classes.
    4. Cannot be represented in UML.
14. Which of the following is **true**? (Equivalent to "Which of the following is not false?")
    1. In an interface, methods do not have any implementation.
    2. Any class that implements an interface must provide the implementation.
    3. Both are true.
    4. Both are false.
15. Encapsulation is the combining of data and the methods that operate on the data. It is often associated with:
    1. Information hiding
    2. Data Binding
    3. Information hiding and Data Binding
    4. None of these options
16. ___________ is the property of an object that distinguishes it from all other objects.
    1. Identity
    2. State

    3. Behaviour
    4. Polymorphism
17. What is the term used to describe the situation when a derived class provides a new implementation for a function already provided in the base class (same name and signature)?
    1. Inheritance
    2. Overloading
    3. Both Inheritance & Overloading
    4. Overriding
18. A class that contains at least one pure virtual function is an Abstract class and __________.
    1. Cannot have zero direct instances.
    2. Can have multiple direct instances.
    3. Can have both zero and multiple direct instances.
    4. Cannot be directly instantiated (cannot create an object of its type).

II. Conceptual & C++ Specific OOP Questions

1. The OOP concept in C++, exposing only necessary information to users or clients while hiding the complex implementation details, is known as:
    1. Abstraction
    2. Encapsulation
    3. Data Hiding
    4. Hiding Complexity
2. Which of the following **cannot** be used to distinguish between overloaded methods (function overloading)?
    1. Number of parameters
    2. Type of parameters
    3. Return type
    4. Order of parameters
3. Which feature is **not** available in C++ object-oriented programming?
    1. Virtual destructor
    2. Virtual constructor
    3. Virtual function
    4. All are unavailable
4. If you want to write multiple functions in a class with the same name, but they perform different tasks based on the arguments, what C++ feature will you use?
    1. Function overriding
    2. Encapsulation
    3. Function overloading
    4. None
5. Not using a virtual destructor feature in a C++ base class can cause:
    1. Memory leak
    2. An issue in creating an object of the class
    3. An issue in calling the base class destructor
    4. Nothing
6. Which operator is used to allocate an object dynamically (on the heap) of a class in C++?

    1. Scope resolution operator (`::`)
    2. Conditional operator (`?:`)
    3. New operator
    4. Membership access (`.` or `->`)
7. When you create an object of a derived class in C++:
    1. Derived class constructor is called first, then the base class constructor.
    2. Base class constructor is called first, then derived class constructor.
    3. Base class constructor will not be called.
    4. None of the above.
8. Which of the following functions are provided by the compiler by default if we don't write them in a C++ class?
    1. Copy constructor
    2. Assignment operator
    3. Default constructor
    4. All the above
9. A static method belongs to:
    1. An instance of a class
    2. The base class only
    3. The class itself
    4. All derived objects
10. Which of the following represents the correct signature of a default constructor?
    1. `-classname()`
    2. `classname()`
    3. `()classname`
    4. `~classname()`

I. C-CAT Specific Previous Year & Sample Questions

1. **1** - Multitasking/Threading is a concept related to operating systems or concurrent programming, not a core principle of Object-Oriented Programming, whose pillars are Abstraction, Encapsulation, Inheritance, and Polymorphism.
2. **4** - **Smalltalk** is recognized as the first *pure* object-oriented programming language, as everything in the language is treated as an object. Simula is considered the first object-oriented language.
3. **2** - In the context of a strict OOP definition (the four pillars), a language that lacks inheritance is often classified as an *object-based* language, hence it is **Not object oriented** by the most rigid definition.
4. **4** - In C++, a member function of a derived class can access members of its base class only if they are declared as `public` or `protected`. `private` members are inaccessible.
5. **2** - A **passive object** (or server object) is one that is acted upon by another object (client object). It waits passively for a message or request for its services.
6. **1** - **Non-persistent objects** are temporary, or transient/ephemeral. They exist in memory only while the program is running and are destroyed when the program terminates.
7. **4** - The primary demerit is that `friend` functions **violate encapsulation** by being able to access a class's private and protected members. Options 1 and 2 are incorrect statements regarding friend functions.
8. **3** - Runtime polymorphism (dynamic method dispatch) in C++ is achieved when a call to a `virtual` function is made through a **pointer** or **reference** to the base class that is pointing to a derived class object.
9. **2** - The **assignment operator (=)**, the subscript operator (`[]`), the function call operator (`()`), and the member access operator (`->`) must be overloaded as **member functions**. They cannot be overloaded as friend functions.
10. **2** - The **virtual** keyword is used during inheritance to ensure that only a single copy of the common base class exists in the final derived class, which solves the ambiguity/duplication of the *diamond problem*.
11. **1** - **Modularity** is the feature of breaking down a large system into smaller, independent, and interchangeable parts, which limits the impact of changes to a small portion of the system.
12. **2** - **Composition** is a strong form of aggregation where the contained objects (parts) are an integral part of the composite object and cannot exist without the whole.
13. **1** - Inheritance (Generalization) in UML is represented by a solid line with an **unfilled (hollow) arrowhead** pointing from the derived class to the base class.
14. **3** - In general OOP terms (like Java/C# interfaces), both statements are true: interface methods are abstract (no body), and an implementing class must provide the body.
15. **3** - Encapsulation is the mechanism of bundling data and methods, which facilitates **Information Hiding** (by controlling access) and is a form of **Data Binding**.
16. **1** - **Identity** is the unique property of an object (distinct from its state and behavior) that allows a system to distinguish it from all other objects.
17. **4** - **Method Overriding** is the mechanism where a derived class redefines (provides a new implementation for) a method that is already inherited from its base class,

using the same name and signature.
18. **4** - An abstract class **cannot be directly instantiated** (i.e., you cannot create an object of the abstract class type).

II. Conceptual & C++ Specific OOP Questions

1. **1** - **Abstraction** is the principle of showing only the relevant information to the user and hiding the background details and implementation.
2. **3** - Function Overloading relies on the function signature being unique. The function signature includes the name and the type/number/order of its parameters. The **Return Type** is *not* considered part of the signature for overloading in C++.
3. **2** - C++ supports `virtual` functions and `virtual` destructors. A **Virtual constructor** is not supported because the virtual mechanism (using the v-table) is only available *after* an object is constructed.
4. **3** - **Function overloading** allows multiple functions in the same scope to share the same name, as long as they have different parameter lists (different number, type, or order of arguments).
5. **1** - If a base class destructor is not `virtual` and a derived class object is deleted via a base class pointer, the derived class's destructor will not be called, which often results in a **Memory leak** (as derived class resources are not cleaned up).
6. **3** - The **`new` operator** in C++ is used to dynamically allocate memory for an object on the heap, and it returns a pointer to the newly created object.
7. **2** - In an inheritance hierarchy, constructors are always called from the base class downward to the derived class. The **Base class constructor is called first**.
8. **4** - The C++ compiler provides four main default functions if the user does not define them: the default constructor, the destructor, the copy constructor, and the copy assignment operator.
9. **3** - A **static method** (or static member function) belongs to the **class itself**, not to any specific object (instance) of the class. It can be called without creating an object.
10. **2** - A constructor has the same name as the class and no return type. **`classname()`** is the correct signature for the default (no-argument) constructor.