CS304 - Object Oriented Programming FAQs By www.virtualians.pk

Question: What is OOP?

Answer: The object oriented programming is commonly known as OOP. Most of the languages are developed using OOP concept. Object-oriented programming (OOP) is a programming concept that uses "objects" to develop a system. An object hides the implementation details and exposes only the functionalities and parameters it requires to its client. Here also an object shares the same concept as that of a bike. While driving a motor bike, we are unaware of its implementation details such as how it is developed, internal working of gears etc.? We know only the functions or actions it can perform.

Question: What are the various elements of OOP?

Answer: Various elements of OOP are: Object Class Method Encapsulation Information Hiding Inheritance Polymorphism

Question: What are the characteristics of Object Oriented programming language?

Answer: Some key features of the Object Oriented programming are: Emphasis on data rather than procedure Programs are divided into entities known as objects Data Structures are designed such that they characterize objects Functions that operate on data of an object are tied together in data structures Data is hidden and cannot be accessed by external functions Objects communicate with each other through functions New data and functions can be easily added whenever necessary Follows bottom up design in program design

Question: What are the basic Concepts used in the Object-Oriented Programming language?

Answer: Object Class Data Abstraction and Encapsulation Polymorphism Inheritance Message passing Dynamic binding

Question: What Is An Object? (Object-Oriented Technology)

Answer: There are many definitions of an object, such as found in [Booch 91, p77]: "An object has state, behavior, and identity; the structure and behavior of similar objects are defined in their common class; the terms instance and object are interchangeable". This is a "classical languages" definition, as defined in [Coplien 92, p280], where "classes play a central role in the object model", since they do not in prototyping/delegation languages. "The term object was first formally applied in the Simula language, and objects typically existed in Simula programs to simulate some aspect of reality" [Booch 91, p77]. Other definitions referenced by Booch include Smith and Tockey: "an object represents an individual, identifiable item, unit, or entity, either real or abstract, with a well-defined role in the problem domain." and [Cox 91]: "anything with a crisply defined boundary" (in context, this is "outside the computer domain". A more conventional definition appears on pg 54). Booch goes on to describe these definitions in depth. [Martin 92, p 241] defines: "An "object" is anything to which a concept applies", and "A concept is an idea or notion we share that applies to certain objects in our awareness". [Rumbaugh 91] defines: "We define an object as a concept, abstraction or thing with crisp boundaries and meaning for the problem at hand." [Shlaer 88, p 14] defines: "An object is an abstraction of a set of real-world things such that:

Question: What Is Object Encapsulation (Or Protection)?

Answer: [Booch 91, p. 45] defines: "Encapsulation is the process of hiding all of the details of an object that do not contribute to its essential characteristics." [Coad 91, 1.1.2] defines: "Encapsulation (Information Hiding). A principle, used when developing an overall program structure, that each component of a program should encapsulate or hide a single design decision... The interface to each module is defined in such a way as to reveal as little as possible about its inner workings. [Oxford, 1986]" Some languages permit arbitrary access to objects and allow methods to be defined outside of a class as in conventional programming. Simula and Object Pascal provide no protection for objects, meaning instance variables may be accessed wherever visible. CLOS and Ada allow methods to be defined outside of a class, providing functions and procedures. While both CLOS and Ada have packages for encapsulation, CLOS's are optional while Ada's methodology clearly specifies class-like encapsulation (Adts). However most object-oriented languages provide a well defined interface to their objects thru classes. C++ has a very general encapsulation/protection mechanism with public, private and protected members. Public members (member data and member functions) may be accessed from anywhere. A Stack's Push and Pop methods will be public. Private members are only accessible from within a class. A Stack's representation, such as a list or array, will usually be private. Protected members are accessible from within a class and also from within subclasses (also called derived classes). A Stack's representation could be declared protected allowing subclass

access. C++ also allows a class to specify friends (other (sub)classes and functions), that can access all members (its representation). Eiffel 3.0 allows exporting access to specific classes.

Question: What Is A Class?

Answer: A class is a general term denoting classification and also has a new meaning in object-oriented methods. Within the OO context, a class is a specification of structure (instance variables), behavior (methods), and inheritance (parents, or recursive structure and behavior) for objects. As pointed out above, classes can also specify access permissions for clients and derived classes, visibility and member lookup resolution. This is a feature-based or intensional definition, emphasizing a class as a descriptor/constructor of objects (as opposed to a collection of objects, as with the more classical extensional view, which may begin the analysis process). Original Aristotlean classification defines a "class" as a generalization of objects: [Booch 91, p93] "a group, set, or kind marked by common attributes or a common attribute; a group division, distinction, or rating based on quality, degree of competence, or condition".

Question: What Is A Meta-Class?

Answer: Meta-Class is a class' class. If a class is an object, then that object must have a class (in classical OO anyway). Compilers provide an easy way to picture Meta-Classes. Classes must be implemented in some way; perhaps with dictionaries for methods, instances, and parents and methods to perform all the work of being a class. This can be declared in a class named "Meta-Class". The Meta-Class can also provide services to application programs, such as returning a set of all methods, instances or parents for review (or even modification). [Booch 91, p 119] provides another example in Smalltalk with timers. In Smalltalk, the situation is more complex

Question: What Is Inheritance?

Answer: Inheritance provides a natural classification for kinds of objects and allowsfor the commonality of objects to be explicitly taken advantage of in modeling and constructing object systems. Natural means we use concepts, classification, and generalization to understand and deal with the complexities of the real world. See the example below using computers. Inheritance is a relationship between classes where one class is the parent base/superclass/ancestor/etc.) class of another. Inheritance provides programming by extension (as opposed to programming by reinvention [LaLonde 90]) and can be used as an is-a-kind-of (or is-a) relationship or for differential programming. Inheritance can also double for assignment

Question: What Is The Difference Between Object-Based And Object-Oriented?

Answer: Object-Based Programming usually refers to objects without inheritance [Cardelli 85] and hence without polymorphism, as in '83 Ada and Modula-2. These languages support abstract data types (Adts) and not classes, which provide inheritance and polymorphism. Ada95 and Modula-3; however, support both inheritance and polymorphism and are object-oriented. [Cardelli 85, p481] state "that a language is object-oriented if and only if it satisfies the following requirements: - It supports objects that are data abstractions with an interface of named operations and a hidden local state. - Objects have an associated type. - Types may inherit attributes from supertypes. object-oriented = data abstractions + object types + type inheritance These definitions are also found in [Booch 91, Ch2 and Wegner 87]. [Coad 91] provides another model: Object-Oriented = Classes and Objects + Inheritance + Communication with messages

Question: What is Abstraction?

Answer: The importance of abstraction is derived from its ability to hide irrelevant details and from the use of names to reference objects. Abstraction is essential in the construction of programs. It places the emphasis on what an object is or does rather than how it is represented or how it works. Thus, it is the primary means of managing complexity in large programs.

Question: What is a Class Diagram?

Answer: A class diagrams are widely used to describe the types of objects in a system and their relationships. Class diagrams model class structure and contents using design elements such as classes, packages and objects.

Question: What is Method Overriding?

Answer: Method overriding is a language feature that allows a subclass to override a specific implementation of a method that is already provided by one of its super-classes. A subclass can give its own definition of methods but need to have the same signature as the method in its super-class. This means that when overriding a method the subclass's method has to have the same name and parameter list as the super-class's overridden method.

Question: What is Operator Overloading?

Answer: The operator overloading is a specific case of polymorphisms in which some or all of operators like +, - or == are treated as polymorphic (multi) functions and as such have different behaviors depending on the types of its arguments.

Question: What is Method Overloading?

Answer: The method overloading is the ability to define several methods (in same class) all with the same name but different on the basis of i) number of parameters ii) types of parameters.

Question: What is Polymorphisms?

Answer: Polymorphism is a generic term that means 'many shapes'. More precisely Polymorphism means the ability to request that the same operations be performed by a wide range of different types of things.

Question: What is Inheritance?

Answer: Ability of a new class to be created, from an existing class by extending it, is called inheritance.

Question: What is a base class?

Answer: When inheritance is used to create a new class from another, the new class is called the subclass or derived class, and the class from which it was derived is called the base class.

Question: What is a concrete class?

Answer: A concrete class is one that can be used to directly create, or instantiate objects, unlike an abstract base class which can only be used as a base class for other classes which eventually lead to concrete classes

Question: What are data members?

Answer: Objects are miniature programs, consisting of both code and data. The code consists of a series of member functions. The data items are called data members.

Question: What is a constructor?

Answer: Objects are complete, miniature programs and, like any good programs, have well defined initialization and termination phases. They have special routines (i.e. member functions) to look after this. The initialization routine is called the constructor, and C++ ensures that every object is properly initialized by calling its constructor. The designer of the object can have more than one constructor, a situation called overloading and then the compiler will select between them depending on exactly what arguments are passed to the constructor function. However there must always be a default constructor, to be used when no information is supplied.

Question: What is a destructor?

Answer: The termination routine is called the destructor, and C++ will provide a default if none is supplied. If, during the lifetime of the object, it uses heap memory then the designer of the object must provide a destructor function to release such memory to avoid a memory leak.

Question: What is global variable?

Answer: Global variables can be accessed throughout a program. Another way to put this is to say they have global scope.

Question: What is local variable?

Answer: Local variables can only be accessed within the function, or more specifically the compound statement in which they are declared. Another way to put this is to say they have local scope.

Question: What is a null pointer?

Answer: A null pointer is a pointer that is currently pointing to nothing. Often pointers are set to zero to make them null pointers or tested against zero to see if they are null or not.

Question: What is a pointer?

Answer: A pointer is a variable that holds the address of another variable or object.

Question: What is meant by protected?

Answer: The protected keyword in the class statement means that the following members of the class are not available to users of the objects of the class, but can be used by any subclass that inherits from it, and consequently forms part of its implementation.