1. **What is a class?**

A class in C++ is a user defined type or data structure declared with keyword *class* that has data and functions (also called methods) as its members whose access is governed by the three access specifiers *private*, *protected* or *public* (by default access to members of a class is *private*). A class (declared with keyword *class*) in C++ differs from a structure (declared with keyword *struct*) as by default, members are *private* in a class while they are *public* in a structure. The private members are not accessible outside the class; they can be accessed only through methods of the class. The public members form an interface to the class and are accessible outside the class. Instances of these data types are known as [objects](https://en.wikipedia.org/wiki/Object_(computer_science)) and can contain [member variables](https://en.wikipedia.org/wiki/Variable_(programming)), [constants](https://en.wikipedia.org/wiki/Variable_(programming)), [member functions](https://en.wikipedia.org/wiki/Method_(computer_science)), and [overloaded operators](https://en.wikipedia.org/wiki/Operator_overloading) defined by the programmer.

**2. How does a class accomplish abstraction, encapsulation, and data hiding?**

Abstraction and encapsulation are complementary concepts: abstraction focuses on the observable behavior of an object... encapsulation focuses upon the implementation that gives rise to this behavior... encapsulation is most often achieved through information hiding, which is the process of hiding all of the secrets of object that do not contribute to its essential characteristics.  
"Encapsulation is then the technique for packaging the information in such a way as to hide what should be hidden, and make visible what is intended to be visible."; remembering that encapsulation is achieved through information hiding.

**3. What is the relationship between an object and a class?**

Usually means “an instance of a class.” Thus a class defines the behavior of possibly many objects (instances).

**4. In what way, aside from being functions, are class function members different from class data members?**

Member functions can (and should) be used to interact with data contained within user defined types. User defined types provide flexibility in the *"*[*divide and conquer*](https://en.wikipedia.org/wiki/divide_and_conquer)*"* scheme in program writing. In other words, one programmer can write a user defined type and guarantee an interface. Another programmer can write the main program with that expected interface. The two pieces are put together and compiled for usage. User defined types provide *encapsulation* defined in the Object Oriented Programming (OOP) paradigm.

Within classes, to protect the data members, the programmer can define functions to perform the operations on those data members. Member functions and functions are names used interchangeably in reference to classes. Function prototypes are declared within the class definition. These prototypes can take the form of non-class functions as well as class suitable prototypes. Functions can be declared and defined within the class definition. However, most functions can have very large definitions and make the class very unreadable. Therefore it is possible to define the function outside of the class definition using the scope resolution operator "**::**". This scope resolution operator allows a programmer to define the functions somewhere else. This can allow the programmer to provide a header file *.h* defining the class and a *.obj* file built from the compiled *.cpp* file which contains the function definitions. This can hide the implementation and prevent tampering. The user would have to define every function again to change the implementation. Functions within classes can access and modify (unless the function is constant) data members without declaring them, because the data members are already declared in the class.

**6. When are class constructors called? When are class destructors called?**

**Constructors at the beginning of using or initializing class .** A constructor is a kind of member function that initializes an instance of its class. A constructor has the same name as the class and no return value. A constructor can have any number of parameters and a class may have any number of overloaded constructors. Constructors may have any accessibility, public, protected or private. If you don't define any constructors, the compiler will generate a default constructor that takes no parameters; you can override this behavior by declaring a default constructor as deleted.

**Destructors are used to destruct instances of classes.** Destructors cannot be inherited or overloaded.Destructors cannot be called. They are invoked automatically. A destructor does not take modifiers or have parameter**s.**

**8. What is a default constructor? What is the advantage of having one?**

A default constructor is a constructor which can be called with no arguments (either defined with an empty parameter list, or with default arguments provided for every parameter).Advantage is initialization .

**10. What are this and \*this?**

The short answer is that this is a special keyword that identifies **"this"** object - the one on which you are currently operating. The slightly longer, more complex answer is this:

When you have a class, it can have member functions of two types: static and non-static. The non-static member functions must operate on a particular *instance* of the class, and they need to know where that instance is. To help them, the language defines an implicit variable (i.e. one that is declared automatically for you when it is needed without you having to do anything) which is called this and which will automatically be made to point to the particular instance of the class on which the member function is operating.