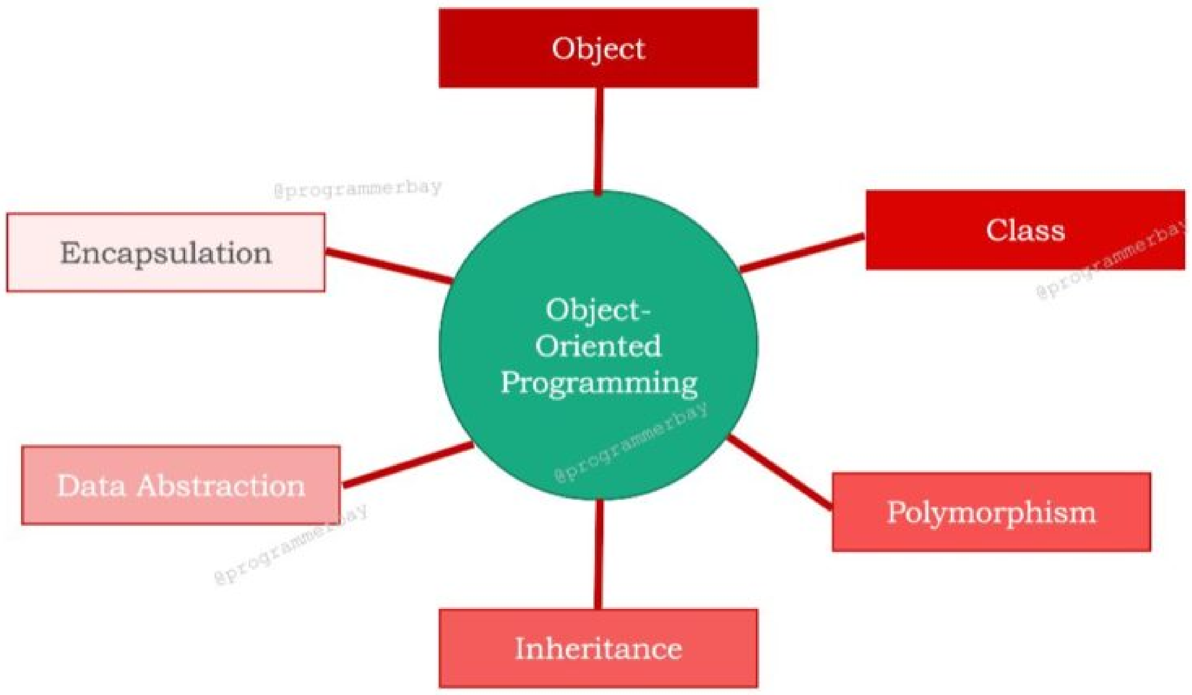
**1. Discuss the features of Object-Oriented Programming. Differentiate between Object Oriented Programming and any other programming language that you know.**
**[ asked in 2075 ]**



Object-Oriented Programming or OOPs is a programming paradigm that revolves around the concept of object, which contains properties and methods. It combines a group of related attributes and behaviour within a single unit named class, that enhances program design structure and code readability. Further, it also resolves drawbacks of Procedural programming i.e  code complexity, unusable code.

Object-Oriented programming focuses on binding attributes and behavior of a real-world entity represented using an object and supports features like abstraction, encapsulation, inheritance, and polymorphism. These are the following OOPs features:

- Classes
- Objects
- Inheritance
- Polymorphism
- Data Abstraction and Encapsulation
-

Other OOPs Features:

- Dynamic Binding
- Message Passing

**Classes:**

A class is the user-defined data type and the main building block of object-oriented programming. It is an identifiable entity that can have some descriptive properties. It is a user-defined data type that holds data members and member functions in a single unit. It is like a blueprint of an object.

For example, consider an entity “Laptop” , what attributes, you can think of? RAM, OS, memory, manufacturer name, model name and so on.

```
public class Laptop
{
String modelName;
String manufacturerName;
String ram;
String memory;
String os;

void boot()
} }
```

The above code represents, how a laptop’s attributes and its behavior are put together in a single place.

**Objects:**

Objects are like a signature of a class. An object is an instance of a class without an object, no memory is allocated to a class’s data members or member function. With an object of a class, we can access the data members and member functions that can be accessed (as per the private, public, protected accessibility scope).

An object represents a real-world entity, having a set of attributes and behavior. However, an object can’t be simply declared as same as primitive types.

**Inheritance:**

The ability to inherit the properties of one class to another, or inherit the properties from a base class to an inherited class is known as the concept of Inheritance. With the help of inheritance, we can use the data members and member functions of a class to another.

It enables a class to acquire or get the properties from another class
It makes code reusable

It makes easier to add new features or methods to a class
It provides an overriding feature which allows a child class to have a specific implementation of a method defined in the parent class

A class that is inherited by other classes is termed as super class or parent class or base class, whereas a class that extends another class is termed as sub-class or child class

**Features of Inheritance :**

- code Reusability
- Ease to add new feature
- Overriding

Inheritance can be classified into majorly 5 types : Single Inheritance, Multilevel inheritance, Hierarchical Inheritance, Multiple Inheritance and Hybrid Inheritance

```
class Laptop extends Electronics
{


}
```

we can also control or check the data members and member functions to be accessed in the child classes by the means of access specifiers, types of inheritance, access specifiers, and their respective access will be discussed in later articles.

**Polymorphism:**

Polymorphism is best defined in one statement i.e., “Polymorphism means one name many forms”.
Polymorphism can be best explained with an example with a comparison to C language; in C language there was one limitation that we can not use the already used function name, but C++ provides us a new feature of Polymorphism, by which we can use the same function name again and again with different signatures.

Further, it can be classified into Static and Dynamic polymorphism ( Runtime Polymorphism ) .

**Data Abstraction and Encapsulation:**

Data Abstraction means hiding the background details and provide only essential details. one of the most common examples regarding this feature is Television, in television we control it with help of a remote and, know its external or the features to be used most whereas we don’t know the internal working of Television.

Encapsulation means binding up data under a single unit. A class is one of the main implementations of this concept. It can be viewed as a wrapper that restricts or prevents properties and methods from outside access. With the help of access modifiers such as private, protected, and public keywords, one can control the access of data members and member functions.

**Other OOPs Features:**

**Dynamic Binding:**
Dynamic Binding which is also known as Late binding or run-time binding, is a process of executing the part of the code at runtime.
Sometimes we need to access some part at the runtime if we need then we can use this approach. We use virtual functions to achieve Dynamic Binding.

**Message Passing:**
In this, objects pass the message to each other in order to contact each other. Like when we want 2 or multiple objects to contact each other it is possible with the OOP.

**Problem with Procedural Programming?**

Procedural Programming follows top-down approach, meaning a program is viewed as a series of sequential steps.
In procedural programming, a program is divided into various procedures or functions which operate on data, the issue with this approach is, when the program grows larger, the code redundancy, maintainability, and complexity increases.

Both procedural and object-oriented are *imperative programming*.

**Difference between Procedural Programming and Object Oriented Programming:**

| Procedural Programming | Object Oriented Programming |
|---|---|
| In procedural programming, program is divided into small parts called functions. | In object oriented programming, program is divided into small parts called objects. |
| Procedural programming follows top down approach. | top down approach. Object oriented programming follows bottom up approach. |
| There is no access specifier in procedural programming. | Object oriented programming have access specifiers like private, public, protected etc. |
| Adding new data and function is not easy. | Adding new data and function is easy. |
| Procedural programming does not have any proper way for hiding data so it is less secure. | Object oriented programming provides data hiding so it is more secure. |
| In procedural programming, overloading is not possible. | Overloading is possible in object oriented programming. |
| In procedural programming, function is more important than data. | In object oriented programming, data is more important than function. |
| Procedural programming is based on unreal world. | Object oriented programming is based on real world. |
| Examples: C, FORTRAN, Pascal, Basic etc. | Examples: C++, Java, Python, C# etc. |

**Procedural Programming:**

Procedural Programming can be defined as a programming model which is derived from structured programming, based upon the concept of calling procedure. Procedures, also known as routines, subroutines or functions, simply consist of a series of computational steps to be carried out. During a program's execution, any given procedure might be called at any point, including by other procedures or itself.

Languages used in Procedural Programming:

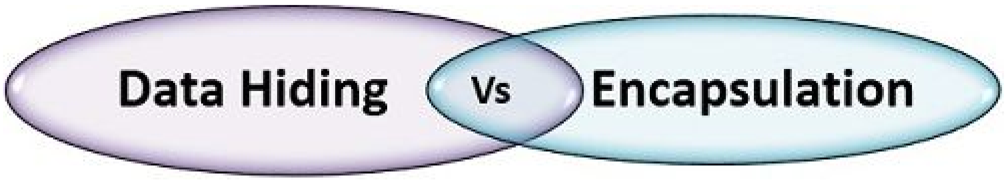
FORTRAN, ALGOL, COBOL,
BASIC, Pascal and C.

**Object Oriented Programming:**

Object oriented programming can be defined as a programming model which is based upon the concept of objects. Objects contain data in the form of attributes and code in the form of methods. In object oriented programming, computer programs are designed using the concept of objects that interact with real world. Object oriented programming languages are various but the most popular ones are class-based, meaning that objects are instances of classes, which also determine their types.

**1. Explain in detail the following principles of Object-Oriented Programming.**
**i. Data encapsulation and data hiding.   ii. Inheritance and polymorphism.**
**iii. Abstraction**
**[ asked in 2066 ]**

**I.**



Data hiding and encapsulation both are the important concept of object oriented programming. Encapsulation means wrapping the implementation of data member and methods inside a class. When implementation of all the data member and methods inside a class are encapsulated, the method name can only describe what action it can perform on an object of that class. Data Hiding means protecting the members of a class from an illegal or unauthorized access.

The main difference between data hiding and encapsulation is that data hiding focus more on data security and encapsulation focuses more on hiding the complexity of the system. There some other differences between data hiding and encapsulation they are described in the comparison chart shown below.

| BASIS FOR COMPARISON | DATA HIDING | ENCAPSULATION |
|---|---|---|
| Basic | Data hiding concern about data security along with hiding complexity. | Encapsulation concerns about wrapping data to hide the complexity of a system. |
| Focus | Data Hiding focuses on restricting or permitting the use of data inside the capsule. | Encapsulation focuses on enveloping or wrapping the complex data. |
| Access Specifier | The data under data hiding is always private and inaccessible. | The data under encapsulation may be private or public. |
| Process | Data hiding is a process as well as technique. | Encapsulation is a sub-process in data hiding. |

**Definition of Data Hiding**

Data hiding is a concept in object-oriented programming which confirms the security of members of a class from unauthorized access. Data hiding is a technique of protecting the data members from being manipulated or hacked from any other source. Data is the most sensitive and volatile content of a program which if manipulated can result in an incorrect output and it also harms the integrity of the data.

Data hiding is controlled in Java with the help of access modifiers (private, public and protected). The data which is public is accessible from outside the class hence if you want to hide your data or restrict it from being accessed from outside, declare your data private. Private data is only accessible to the objects of that class only.

Let us understand data hiding with the help of an example. Suppose you declared a CheckAccount class and you have a data member balance inside that class. Here, the balance of an account is sensitive information. You may allow an outside application to check balance inside of an account but,  you won't allow an outside application to alter the balance attribute. Thus declaring the balance attribute private you would restrict the access to balance from an outside application.

Data hiding also reduces some complexity of the system. Data hiding can be achieved through the encapsulation, as encapsulation is a subprocess of data hiding.

**Definition of Encapsulation**

Encapsulation is binding the code and data together in a capsule to hide the complexity of a class. Encapsulation has less to do with access specifiers (private, public and protected).  In encapsulation members inside a class can be private, public or protected.

The private members of a class are only accessible to the objects of that class only, and the public members are accessible to the objects of the class as well as they are accessible from outside the class. Encapsulation helps the end user of a system to learn what to do with the system instead of how it must do.

Let us understand encapsulation with the help of an example of a car. If a driver of a car wants to change the gear of car, what he needs is to just change the position of the liver operating gears of the car and it thus changes the gear of a car. A driver does not need to understand the complexity of, what is the mechanism behind changing the gear. This is how encapsulation reduces the complexity of a system. Encapsulation makes the system easier to operate by the end user.

**Inheritance and polymorphism**

Inheritance is one in which a new class is created that inherits the properties of the already exist class. It supports the concept of code reusability and reduces the length of the code in object-oriented programming.
Types of Inheritance are:

- Single inheritance
- Multi-level inheritance
- Multiple inheritance
- Hybrid inheritance
- Hierarchical inheritance

**Polymorphism:**

Polymorphism is that in which we can perform a task in multiple forms or ways. It is applied to the functions or methods. Polymorphism allows the object to decide which form of the function to implement at compile-time as well as run-time.
Types of Polymorphism are:

Compile-time polymorphism (Method overloading)
Run-time polymorphism (Method Overriding)

Difference between Inheritance and Polymorphism:

| Inheritance | Polymorphism |
|---|---|
| Inheritance is one in which a new class is created (derived class) that inherits the features from the already existing class (Base class). | Whereas polymorphism is that which can be defined in multiple forms. |
| It is basically applied to classes. | Whereas it is basically applied to functions or methods. |
| Inheritance supports the concept of reusability and reduces code length in object-oriented programming. | Polymorphism allows the object to decide which form of the function to implement at compile-time (overloading) as well as run-time (overriding). |
| Inheritance can be single, hybrid, multiple, hierarchical and multilevel inheritance. | Whereas it can be compiled-time polymorphism (overload) as well as run-time polymorphism (overriding). |
| It is used in pattern designing. | While it is also used in pattern designing. |

**Abstraction:**

Abstraction is one of the key concepts of object-oriented programming (OOP) languages. Its main goal is to handle complexity by hiding unnecessary details from the user. That enables the user to implement more complex logic on top of the provided abstraction without understanding or even thinking about all the hidden complexity.

That's a very generic concept that's not limited to object-oriented programming. You can find it everywhere in the real world.

Objects in an OOP language provide an abstraction that hides the internal implementation details. We just need to know which methods of the object are available to call and which input parameters are needed to trigger a specific operation. But you don't need to understand how this method is implemented and which kinds of actions it has to perform to create the expected result.

**3. What is object-oriented approach?**
**How is it different from structured programming approach?**
**Discuss the features of object-oriented languages in detail.**

In the object-oriented approach, the focus is on capturing the structure and behavior of information systems into small modules that combines both data and process. The main aim of Object Oriented Design (OOD) is to improve the quality and productivity of system analysis and design by making it more usable.

In analysis phase, OO models are used to fill the gap between problem and solution. It performs well in situation where systems are undergoing continuous design, adaption, and maintenance. It identifies the objects in problem domain, classifying them in terms of data and behavior.

The OO model is beneficial in the following ways −

It facilitates changes in the system at low cost.

It promotes the reuse of components.

It simplifies the problem of integrating components to configure large system.

It simplifies the design of distributed systems.

**Differentiate between operator overloading and function overloading.**

| Operator overloading | Function overloading |
|---|---|
| Operator overloading is a programming concept where different operators have different implementations depending on their arguments. It is the mechanism of giving a special meaning to an operator.<br>( overloading means the use of the same thing for different purposes. ) | Two or more functions can have the same name but different parameters; such functions are called function overloading. |
| Operator overloading allows operators to have an extended meaning beyond their predefined operational meaning. | Function overloading (method overloading) allows us to define a method in such a way that there are multiple ways to call it. |
| Overloaded operators are functions with special names: the keyword "operator" followed by the symbol for the operator being defined. Like any other function, an overloaded operator has a return type and a parameter list. | We cannot overload function declarations that differ only by return type. |
| SYNTAX:<br><br>`Box operator+(const Box&);`<br>`Box operator+(const Box&, const Box&);` | TYPES:<br><br>Unary Operators Overloading<br>Binary Operators Overloading<br>Relational Operators Overloading<br>Input/Output Operators Overloading |

It is possible to do a declaration with the same name as a previously declared declaration in the same scope, which is called an overloaded declaration. This type of declaration can have different arguments and obviously different implementations.

When we call an overloaded declaration, the compiler finds out the exactly appropriate function to use, based on the argument types you have used in the function call. The process of selecting the suitable overloaded function or operator is called overload resolution. This is how overloading works in most programming languages.

**Explain the object oriented programming with its advantages. What are the features of object oriented languages? Explain.**
**asked in 2072**

**Benefits of OOP**

1. We can build the programs from standard working modules that communicate with one another, rather than having to start writing the code from scratch which leads to saving of development time and higher productivity,
2. OOP language allows to break the program into the bit-sized problems that can be solved easily (one object at a time).
3. The new technology promises greater programmer productivity, better quality of software and lesser maintenance cost.
4. OOP systems can be easily upgraded from small to large systems.
5. It is possible that multiple instances of objects co-exist without any interference,
6. It is very easy to partition the work in a project based on objects.
7. It is possible to map the objects in problem domain to those in the program.
8. The principle of data hiding helps the programmer to build secure programs which cannot be invaded by the code in other parts of the program.
9. By using inheritance, we can eliminate redundant code and extend the use of existing classes.
10. Message passing techniques is used for communication between objects which makes the interface descriptions with external systems much simpler.
11. The data-centered design approach enables us to capture more details of model in an implementable form.

**Disadvantages of OOP**

1. The length of the programmes developed using OOP language is much larger than the procedural approach. Since the programme becomes larger in size, it requires more time to be executed that leads to slower execution of the programme.
2. We can not apply OOP everywhere as it is not a universal language. It is applied only when it is required. It is not suitable for all types of problems.
3. Programmers need to have brilliant designing skill and programming skill along with proper planning because using OOP is little bit tricky.
4. OOPs take time to get used to it. The thought process involved in object-oriented programming may not be natural for some people.
5. Everything is treated as object in OOP so before applying it we need to have excellent thinking in terms of objects.

**What is meant by type conversion?**
**Define two way of converting one user defined data type (object) to another user defined object? { not completed and unknown topic !!!! }**

Type conversion is the process that converts the predefined data type of one variable into an appropriate data type. The main idea behind type conversion is to convert two different data type variables into a single data type to solve mathematical and logical expressions easily without any data loss.

For example, we are adding two numbers, where one variable is of int type and another of float type; we need to convert or typecast the int variable into a float to make them both float data types to add them.

Type conversion can be done in two ways in C++, one is implicit type conversion, and the second is explicit type conversion. Those conversions are done by the compiler itself, called the implicit type or automatic type conversion. The conversion, which is done by the user or requires user interferences called the explicit or user define type conversion. Let's discuss the implicit and explicit type conversion in C++.

**Implicit Type Conversion**
The implicit type conversion is the type of conversion done automatically by the compiler without any human effort. It means an implicit conversion automatically converts one data type into another type based on some predefined rules of the C++ compiler. Hence, it is also known as the automatic type conversion.

```
int x = 20;
short int y = 5;
int z = x + y;     ( implicit type )
```

**Explicit type conversion**

Conversions that require user intervention to change the data type of one variable to another, is called the explicit type conversion. In other words, an explicit conversion allows the programmer to manually changes or typecasts the data type from one variable to another type. Hence, it is also known as typecasting. Generally, we force the explicit type conversion to convert data from one type to another because it does not follow the implicit conversion rule.

The explicit type conversion is divided into two ways:

1. Explicit conversion using the cast operator
2. Explicit conversion using the assignment operator

**Second Part:**
A user-defined data types are designed by the user to suit their requirements, the compiler does not support automatic type conversions for such data types therefore, the user needs to design the conversion routines by themselves if required.

Conversion of one class type to another class type: In this type, one class type is converted into another class type. It can be done  in 2 ways :

1.Using constructor
2.Using Overloading casting operator

**Using constructor :**

In the Destination  class we use the constructor method

```
//Objects of different types
ObjectX=ObjectY;
```

Here ObjectX is Destination object and ObjectY is source object

**What is the difference between an object-oriented programming language and object-based programming language?**

Object-Oriented Languages (OOP) follow all the concepts of OOPs whereas, Object-based languages don't follow all the concepts of OOPs like inheritance and polymorphism.

Object-oriented languages do not have the inbuilt objects whereas Object-based languages have the inbuilt objects, for example, JavaScript has window object.
Examples for Object Oriented Languages include Java, C# whereas Object-based languages include VB etc.

**What is the use of get and getline functions? Explain with suitable example.**
The getline() function reads a whole line, and using the newline character transmitted by the Enter key to mark the end of input. The get() function is much like getline() but rather than read and discard the newline character, get() leaves that character in the input queue.

**What is meant by pass by reference?**
**How can we pass arguments by reference by using reference variable? Illustrate with example. { to do }**

Pass-by-reference means to pass the reference of an argument in the calling function to the corresponding formal parameter of the called function. The called function can modify the value of the argument by using its reference passed in.

The reference parameters are initialized with the actual arguments when the function is called.

The following example shows how arguments are passed by reference.
```
#include <stdio.h>

void swapnum(int &i, int &j) {
  int temp = i;
  i = j;
  j = temp;
}

int main(void) {
  int a = 10;
  int b = 20;

  swapnum(a, b);
  printf("A is %d and B is %d\n", a, b);
  return 0;
}
```

## What is the concept of friend function? How it violates the data hiding principle? Justify with example. { to do }

A friend function is a function that is specified outside a class but has the ability to access the class members' protected and private data. A friend can be a member's function, function template, or function, or a class or class template, in which case the entire class and all of its members are friends.

In special cases when a class's private data needs to be accessed directly without using objects of that class, we need friend functions. For instance, let's consider two classes: Director and Doctor. We may want the function gross_salary to operate the objects of both these classes. The function does not need to be a member of either of the classes.

They are also used in operator overloading because they are more intuitive. The binary arithmetic operator that is commonly used can be overloaded the friend function way.

**Special features of friend functions:**

- A friend function does not fall within the scope of the class for which it was declared as a friend. Hence, functionality is not limited to one class.
- The friend function can be a member of another class or a function that is outside the scope of the class.
- A friend function can be declared in the private or public part of a class without changing its meaning.
- Friend functions are not called using objects of the class because they are not within the class's scope.
- Without the help of any object, the friend function can be invoked like a normal member function.
- Friend functions can use objects of the class as arguments.
- A friend function cannot explicitly access member names directly. Every member name has to use the object's name and dot operator.. For example, Doctor.pay where pay is the object name.

## What is meant by stream?

Stream in C++ means a stream of characters that gets transferred between the program thread and input or output. There are a number of C++ stream classes eligible and defined which is related to the files and streams for providing input-output operations. All the classes and structures maintaining the file and folders with hierarchies are defined within the file with iostream.h standard library. Classes associated with the C++ stream include ios class, istream class, and ostream class. Class ios is indirectly inherited from the base class involving iostream class using istream class and ostream class which is declared virtually.

## Manipulators

Manipulators are special functions that can be included in the I/O statement to alter the format parameters of a stream. To access manipulators, the file iomanip.h should be included in the program.
manipulators are simply an instruction to the output stream that modify the output in various ways. In other words, we can say that Manipulators are operators that are used to format the data display.

The manipulators in C++ are stream functions that change the properties of an input or output stream. It's used to format the input and output streams by modifying the stream's format flags and values.

The header file iomanip.h> contains a set of manipulator functions.

To utilize the manipulator functions in your program, you must include this header.

**Advantages and Purpose of Manipulators**

- It is mainly used to make up the program structure.
- Manipulators functions are special stream function that changes certain format and characteristics of the input and output.
- To carry out the operations of the manipulators <iomanip.h> must be included.
- Manipulators functions are specially designed to be used in conjunction with insertion (<<) and extraction (>>) operator on stream objects.
- Manipulators are used to changing the format of parameters on streams and to insert or extract certain special characters.

**Here is the list of standard input/output Manipulators and their Functions in C++**

setw (int n) – To set field width to n
Setbase – To set the base of the number system
stprecision (int p) – The precision is fixed to p
setfill (Char f) – To set the character to be filled
setiosflags (long l) – Format flag is set to l
resetiosflags (long l) – Removes the flags indicated by l
endl – Gives a new line
skipws – Omits white space in input
noskipws – Does not omit white space in the input
ends – Adds null character to close an output string
flush – Flushes the buffer stream
lock – Locks the file associated with the file handle
ws – Omits the leading white spaces present before the first field
hex, oct, dec – Displays the number in hexadecimal or octal or in decimal format

## Protected Access Specifier

The protected keyword specifies access to class members in the member-list up to the next access specifier (public or private) or the end of the class definition. Class members declared as protected can be used only by the following:

- Member functions of the class that originally declared these members.
- 
- Friends of the class that originally declared these members.
- 
- Classes derived with public or protected access from the class that originally declared these members.
- 
- Direct privately derived classes that also have private access to protected members.

When preceding the name of a base class, the protected keyword specifies that the public and protected members of the base class are protected members of its derived classes.

Protected members are not as private as private members, which are accessible only to members of the class in which they are declared, but they are not as public as public members, which are accessible in any function.

Protected members that are also declared as static are accessible to any friend or member function of a derived class. Protected members that are not declared as static are accessible to friends and member functions in a derived class only through a pointer to, reference to, or object of the derived class.