COMPUTER SCIENCE

OBJECT ORIENTED PROGRAMMING CONCEPTS

Object Oriented Programming follows bottom up approach in program design and emphasizes on
safety and security of data..

FEATURES OF OBJECT ORIENTED PROGRAMMING:

Inheritance:

- Inheritance is the process of forming a new class from an existing class or base class. The
base class is also known as parent class or super class.
- Derived class is also known as a child class or sub class. Inheritance helps in reusability of code , thus reducing the overall size of the program

Data Abstraction:

- It refers to the act of representing essential features without including the background
details .Example : For driving , only accelerator, clutch and brake controls need to be
learnt rather than working of engine and other details.

Data Encapsulation:

- It means wrapping up data and associated functions into one single unit called class..
- A class groups its members into three sections :public, private and protected, where
private and protected members remain hidden from outside world and thereby helps in
implementing data hiding.

Modularity :

- The act of partitioning a complex program into simpler fragments called modules is
called as modularity.
- It reduces the complexity to some degree and

• It creates a number of well defined boundaries within the program .

Polymorphism:

• **Poly** means many and **morphs** mean form, so polymorphism means one name multiple
forms.
• It is the ability for a message or data to be processed in more than one form.
• C++ implements Polymorhism through Function Overloading , Operator overloading and

Objects and Classes :

The major components of Object Oriented Programming are . **Classes & Objects**
A **Class i**s a group of similar objects . **Objects** share two characteristics: They all have *state* and *behavior*. For example : Dogs have state name, color, breed, hungry and behavior barking, fetching, wagging tail. Bicycles also have state current gear, current pedal cadence, current speed) and behavior changing gear, applying brakes). Identifying the state and be havior for realworld
objects is a great way to begin thinking in terms of object-oriented programming. These real-world observations all translate into the world of object-oriented programming.
Software objects are conceptually similar to real-world objects: they too consist of state and related behavior. An object stores its state in *fields* variables in some programming languages) and exposes its behavior through functions

Classes in Programming :

- **It is a collection of variables, often of different types and its associated functions.**
- **Class just binds data and its associated functions under one unit there by enforcing encapsulation.**
- Classes define types of data structures and the functions that operate on those data structures.
- A class defines a blueprint for a data type.

Declaration/Definition :

A class definition starts with the keyword **class** followed by the class name; and the class body, enclosed by a pair of curly braces. A class definition must be followed either by a semicolon or a list of declarations.

```
class class_name {
access_specifier_1:
member1;
access_specifier_2:
member2;
...
} object_names;
```

Where class_name is a valid identifier for the class, object_names is an optional list of names for objects of this class. The body of the declaration can contain members that can either be data or function declarations, and optionally access specifiers.
[Note: the default access specifier is private.
Example : class Box { int a;
public:
double length; // Length of a box
double breadth; // Breadth of a box
double height; // Height of a box
};
18

Access specifiers in Classes:

Access specifiers are used to identify access rights for the data and member functions of the class.
There are three main types of access specifiers in C++ programming language:

- private
- public
- protected

Member-Access Control

Type of Access Meaning
Private Class members declared as **private** can be used only by member functions and friends classes or function s) of the class.
Protected Class members declared as **protected** can be used by member functions and friends (classes or functions) of the class. Additionally, they can be used by classes derived from the class.
Public Class members declared as **public** can be used by any function.

➢ Importance of Access Specifiers

Access control helps prevent you from using objects in ways they were not intended to be used. Thus it helps in implementing data hiding and data abstraction.

OBJECTS in C++:

Objects represent instances of a class. Objects are basic run time entities in an object oriented system.
Creating object / defining the object of a class:
The general syntax of defining the object of a class is:-
Class_name object_name;
In C++, a class variable is known as an object. The declaration of an object is similar to that of a variable of any data type. The members of a class are accessed or referenced using object of a class.

```
Box Box1; // Declare Box1 of type Box
Box Box2; // Declare Box2 of type Box
```

Both of the objects Box1 and Box2 will have their own copy of data members.

Accessing / calling members of a class*All member of a class are private by default*.

Private member can be accessed only by the function of the class itself.Public member of a class can be accessed through any object of the class. They are accessed or called using object of that class with the help of dot operator (..

The general syntax for accessing data member of a class is:-

Object_name.Data_member=value;

The general syntax for accessing member function of a class is:-

Object_name. Function_name (actual arguments);

The dot ('. ' used above is called the **dot operator or class member access operator**. The dot operator is used to connect the object and the member function. The private data of a class can be accessed only through the member function of that class.

Class methods definitions Defining the member functions

Member functions can be defined in two places:-

- ## Outside the class definition

The member functions of a class can be defined outside the class definitions. It is only declared inside the class but defined outside the class. The general form of member function definition outside the class definition is:

Return_type Class_name:: function_name argument list

{

Function body

}

Where symbol :: is a scope resolution operator.

The scope resolution operator :: specifies the class to which the member being declared belongs, granting exactly the same scope properties as if this function definition was directly

included within the class definition

```
class sum
{
int A, B, Total;
public:
void getdata (;
void display (;
};
void sum:: getdata ( // Function definition outside class definition Use of :: operator
{
cout<<" n enter the value of A and B";
cin>>A>>B;
}
void sum:: display // Function definition outside class definition Use of :: operator
{
Total =A+B;
cout<<" n the sum of A and B="<<Total;
}
```

- ## Inside the class definition

The member function of a class can be declared and defined inside the class definition.

```
class sum
{
int A, B, Total;
public:
void getdata (
{ cout< ”
n
enter the value of A
and B”;
cin>>A>>B;
}
void display (
{
total = A+B;
cout<<” n the sum of A and B=”<<total;
}
};
```

Differences between struct and classes in C++

In C++, a *structure* is a class defined with the `struct` keyword.Its members and base classes are
public by default. A class defined with the `class` keyword has private members and base classes
by default. This is the only difference between structs and classes in C++.

INLINE FUNCTIONS

- **Inline functions definition starts with keyword inline**
- **The compiler replaces the function call statement with the function code itselfexpansion and then compiles the entire code.**
- **They run little faster than normal functions as function calling overheads are saved.**
- **A function can be declared inline by placing the keyword inline before it.**

Example

```
inline void Square (int a
{ cout<<a*a;}
void main(
{.
Square4); { cout <<4*4;}
Square8) ; { cout <<8*8; }
}
```

In place of function call , function body is substituted because Square (is inline function