**QUE 1 : Explain the key concepts of Object-Oriented Programming (OOP).**

**1. Classes and Objects**

* **Class**: A blueprint or template that defines the properties (attributes) and behaviors (methods) of an object.
* **Object**: An instance of a class. It represents a real-world entity based on the class blueprint.

**Example**: A Car class might have attributes like color, make, and methods like drive(). An object would be a specific car like a red Toyota.

**2. Encapsulation**

* Hiding the internal details of an object and exposing only what is necessary.
* This helps protect the data by restricting direct access and allows controlled access via public methods.

**Example**: A bank account might hide its balance and only allow deposits or withdrawals through deposit() and withdraw() methods.

**3. Inheritance**

* A mechanism where one class (derived class) inherits the properties and methods of another class (base class).
* It allows code reuse and creates a hierarchy of classes.

**Example**: A Student class might inherit from a Person class, so it gets attributes like name and age.

**4. Polymorphism**

* The ability of different classes to provide a unique implementation of the same method.
* This allows you to use the same method name in different contexts.

**Example**: A Shape class might have a draw() method, and different classes like Circle and Rectangle would implement draw() in their own way.

**5. Abstraction**

* Hiding complex implementation details and showing only the essential features.
* Helps to reduce complexity and focus on the important aspects of an object.

**Example**: A Car class might have a start() method. The user doesn't need to know the internal mechanics, just that calling start() turns on the car.

**QUE 2 : What are classes and objects in C++? Provide an example.**

**Classes and Objects in C++**

In **C++**, a **class** is a user-defined blueprint or prototype from which objects are created. It encapsulates data for the object and the methods (functions) that operate on that data.

An **object** is an instance of a class. When a class is defined, no memory is allocated until an object of that class is created.

**Key Concepts:**

* **Class**: A blueprint for creating objects. It defines the attributes (data members) and behaviors (functions or methods) that the objects of the class will have.
* **Object**: A specific instance of a class. It has actual values for the attributes and can call the methods defined in the class.

**Example of Classes and Objects in C++**

#include <iostream>

using namespace std;

// Define the class "Car"

class Car {

public:

// Data members (attributes)

string make;

string model;

int year;

// Member function (method) to display car details

void displayCarDetails() {

cout << "Car Details:" << endl;

cout << "Make: " << make << endl;

cout << "Model: " << model << endl;

cout << "Year: " << year << endl;

}

};

int main() {

// Create an object (instance) of the class Car

Car myCar;

// Setting values for the object myCar

myCar.make = "Toyota";

myCar.model = "Corolla";

myCar.year = 2020;

// Calling the method to display details of myCar

myCar.displayCarDetails();

return 0;

}

**Explanation:**

1. **Class Definition:**
   * We define a class called Car with **attributes**: make, model, and year (these are data members), and a **method**: displayCarDetails() (which displays the car's details).
2. **Creating an Object:**
   * In the main() function, we create an object myCar of the Car class. This object is an instance of the class with specific values.
3. **Setting Attributes:**
   * We assign values to the attributes of the myCar object: "Toyota", "Corolla", and 2020.
4. **Calling the Method:**
   * We call the displayCarDetails() method for the myCar object, which prints out the details of the car.

**QUE 3 :What is inheritance in C++? Explain with an example.**

**Inheritance** is a concept in **C++** where a new class (derived class) gets the properties and behaviors of an existing class (base class). It allows us to **reuse code** and create a hierarchy between classes.

* **Base Class**: The class whose properties and methods are inherited.
* **Derived Class**: The class that inherits from the base class.

**Example:**

#include <iostream>

using namespace std;

// Base class

class Animal {

public:

void makeSound() {

cout << "Some animal sound!" << endl;

}

};

// Derived class

class Dog : public Animal {

public:

void makeSound() {

cout << "Woof Woof!" << endl; // Overridden method

}

};

int main() {

Dog myDog;

myDog.makeSound(); // Calls the Dog's version of makeSound()

return 0;

}

**Explanation:**

* Dog **inherits** from Animal using public.
* The Dog class **overrides** the makeSound() method to provide its own behavior.
* The derived class **reuses** the code from the base class and can also add or change functionality.

**QUE 4 : What is encapsulation in C++? How is it achieved in classes?**

**Encapsulation in C++**

**Encapsulation** is the concept of **hiding** the internal details of an object and only exposing the necessary parts through public methods. This helps protect data from being directly accessed or modified, ensuring better control over the object’s state.

**How Encapsulation Works in C++:**

1. **Private data members**: Data (attributes) are made **private** so they can't be accessed directly from outside the class.
2. **Public methods**: **Getters** (to access data) and **setters** (to modify data) are provided to control how the data is accessed or changed.

**Example:**

cpp

Copy code

#include <iostream>

using namespace std;

class BankAccount {

private:

double balance; // Private data member

public:

// Constructor to initialize balance

BankAccount(double initialBalance) {

balance = (initialBalance >= 0) ? initialBalance : 0;

}

// Getter method to access balance

double getBalance() {

return balance;

}

// Setter method to deposit money

void deposit(double amount) {

if (amount > 0) balance += amount;

}

};

int main() {

BankAccount account(100); // Create object with initial balance of 100

cout << "Balance: " << account.getBalance() << endl; // Access balance

account.deposit(50); // Deposit money

cout << "Updated Balance: " << account.getBalance() << endl; // Access balance

return 0;

}