Assignment-3

1.What is an object in C++?

Explain the concept of encapsulation with an example.

Ans-In C++, an object is an instance of a class. A class is like a blueprint, while an object is a specific realization of that blueprint. Objects have attributes (data members) and behaviors (member functions) that define their properties and actions.

cpp

#include <iostream>

using namespace std;// Define a class

class Car {

public:

string brand;

int speed;

// Member function

void showInfo() {

cout << "Brand: " << brand << ", Speed: " << speed << " km/h" << endl;

}

};

int main() { // Create an object of Car class

Car myCar; // Assign values to object properties

myCar.speed = 120;

// Call member function

myCar.showInfo();

return 0;

}

Output:

Brand: Toyota, Speed: 120 km/h

2.What is a class in C++ and how does it differ from an object?

Ans-In C++, a class is a blueprint or template for creating objects, defining their data members (attributes) and member functions (methods), while an object is a specific instance or realization of that class.

3.Explain the concept of encapsulation with an example.

Ans-Encapsulation is one of the fundamental principles of Object-Oriented Programming (OOP). It refers to the bundling of data (variables) and methods (functions) that operate on that data into a single unit, typically a class. It also involves restricting direct access to some of the object's components, which helps prevent unintended interference and enhances security.

Example:-

cpp

4.How do you define a class in C++?

Ans-In C++, a **class** is defined using the class keyword. A class is a blueprint for creating objects and encapsulates data members (variables) and member functions (methods) that operate on the data.

#include <iostream>

using namespace std;

class MyClass { // Class definition

public: // Access specifier

int num; // Data member (variable)

void display() { // Member function

cout << "Number: " << num << endl; }

};

int main() {

MyClass obj; // Creating an object of MyClass

obj.num = 10; // Assigning value

obj.display(); // Calling the member function

return 0;}

4.Describe the syntax for creating an object of a class.

Ans-An **object** is an instance of a class. Once a class is defined, we can create objects using the following syntax:

Example-

#include <iostream>

using namespace std;

class Car { // Class definition

public:

string brand;

int year;

void display() { // Member function

cout << "Brand: " << brand << ", Year: " << year << endl;

}

};

int main() {

Car myCar; // Creating an object of the Car class

// Assigning values to object members

myCar.brand = "Toyota";

myCar.year = 2022;

// Calling a member function

myCar.display();

return 0;

}

6.What are private members in a class and how are they accessed?

Ans-**Private members** in a class are variables or methods that cannot be accessed directly from outside the class. They can only be accessed within the class itself.

class ClassName {

private:

int privateVar; // Private data member

void privateMethod() { // Private member function

// Code here

}

};

How to Access Private Members?

Since private members cannot be accessed directly outside the class, they must be accessed using public member functions (getter/setter methods).

#include <iostream>

using namespace std;

class BankAccount {

private:

double balance; // Private data member

public:

// Constructor to initialize balance

BankAccount(double initialBalance) {

balance = initialBalance; } // Public method to deposit money

void deposit(double amount) {

balance += amount;

cout << "Deposited: $" << amount << ", New Balance: $" << balance << endl;

}

// Public method to get the balance (Getter function)

double getBalance() {

return balance;

}

};

int main() {

BankAccount myAccount(1000); // Creating an object

myAccount.deposit(500); // Depositing money

cout << "Final Balance: $" << myAccount.getBalance() << endl; // Accessing private data via getter

// Direct access to private member is not allowed:

// myAccount.balance = 2000; // Error: balance is private

return 0;

}

7.What are public members in a class and how are they accessed?

Ans-**Public members** of a class are variables (data members) or methods (member functions) that can be accessed from outside the class using an object.

class ClassName {

public:

int publicVar; // Public data member

void publicMethod() { // Public member function

// Code here

}

};

Public members are accessed using the **dot (.) operator** with an object of the class.

**Example: Accessing Public Members**

#include <iostream>

using namespace std;

class Car {

public:

string brand; // Public data member

int year;

// Public member function

void display() {

cout << "Brand: " << brand << ", Year: " << year << endl;

}

};

int main() {

Car myCar; // Creating an object

// Accessing public data members

myCar.brand = "Toyota";

myCar.year = 2022;

// Calling a public member function

myCar.display();

return 0;

}

8**.** Explain the significance of access specifiers in a class**.**

**Access Specifiers in a Class (C++)**

Access specifiers in C++ determine the visibility and accessibility of class members (variables and functions). They help in data hiding,security, and encapsulation.

**Types of Access Specifiers:**

| **Access Specifier** | **Accessibility** | **Usage** |
| --- | --- | --- |
| **public** | Accessible from anywhere (inside & outside the class). | Used for functions or data that should be accessible freely. |
| **private** | Accessible only within the class. | Used to protect data from direct modification. |
| **protected** | Accessible within the class and its derived (child) classes. | Used in inheritance to allow controlled access to derived classes. |

9. Provide an example of a class with both private and public members.

Ans-Example-

#include <iostream>

using namespace std;

class BankAccount {

private:

double balance; // Private member: Cannot be accessed directly

public:

string accountHolder; // Public member: Can be accessed directly

// Constructor to initialize balance

BankAccount(string name, double initialBalance) {

accountHolder = name;

balance = initialBalance;

}

// Public method to deposit money (modifies private data)

void deposit(double amount) {

if (amount > 0) {

balance += amount;

cout << "Deposited: $" << amount << ", New Balance: $" << balance << endl;

} else {

cout << "Deposit amount must be positive!" << endl;

}

}

// Public method to withdraw money (modifies private data)

void withdraw(double amount) {

if (amount > 0 && amount <= balance) {

balance -= amount;

cout << "Withdrawn: $" << amount << ", Remaining Balance: $" << balance << endl;

} else {

cout << "Insufficient funds or invalid amount!" << endl;

}

}

// Public method to get balance (access private data)

double getBalance() {

return balance;

}

};

int main() {

// Creating an object of BankAccount

BankAccount myAccount("Pooja", 5000);

// Accessing public member directly

cout << "Account Holder: " << myAccount.accountHolder << endl;

// Accessing private member through public methods

myAccount.deposit(1500);

myAccount.withdraw(2000);

// Using getter method to access private balance

cout << "Final Balance: $" << myAccount.getBalance() << endl; return 0;

}

10. How does data hiding work in C++?

Ans-Data Hiding means restricting direct access to a class’s data members to protect them from accidental modification.

Private members (variables/functions) cannot be accessed directly from outside the class.

They can only be accessed using public functions (getter & setter methods).

11. What is a static data member in C++?

Ans-A static data member in C++ is a variable that is shared amongall objects of a class. It is declared using the static keyword inside the class.

-Belongs to the class, not to individual objects.  
 -Memory is allocated only once, not separately for each object.  
 -All objects share the same value of the static variable.

#include <iostream>

using namespace std;

class Student {

private:

string name;

int rollNumber;

public:

static int totalStudents; // Static data member

// Constructor to initialize student details

Student(string n, int roll) {

name = n;

rollNumber = roll;

totalStudents++; // Increment static variable

}

// Function to display student details

void display() {

cout << "Name: " << name << ", Roll No: " << rollNumber << endl;

}

};

// Initialize static data member outside the class

int Student::totalStudents = 0;

int main() {

Student s1("Pooja", 101);

Student s2("Amit", 102);

Student s3("Raj", 103);

s1.display();

s2.display();

s3.display(); // Accessing static member using class name

cout << "Total Students: " << Student::totalStudents << endl;

return 0;}

12.How do you declare and initialize a static data member?

Ans-A static data member is a variable that belongs to the class, notto individual objects. It is shared among all objects of the class.

#include <iostream>

using namespace std;

class Counter {

public:

static int count; // Declaration of static member

Counter() {

count++; // Increment count when an object is created

}

};

// Initialization of static data member outside the class

int Counter::count = 0;

int main() {

Counter c1, c2, c3; // Creating three objects

// Accessing static variable using class name

cout << "Total Objects Created: " << Counter::count << endl;

return 0;

}

13.What is a static function member in C++?

Ans-A static member function in C++ is a function that belongs to the class, not to any specific object. It can only access static datamembers and can be called without creating an object.

#include <iostream>

using namespace std;

class Counter {

private:

static int count; // Static data member

public:

// Constructor to increment count

Counter() {

count++; }

// Static function to get count

static int getCount() {

return count;

}

};

// Initialize static data member outside the class

int Counter::count = 0;

int main() {

Counter c1, c2, c3; // Creating objects

// Accessing static function without an object

cout << "Total Objects Created: " << Counter::getCount() << endl; return 0;s

}

14. . How do static function members differ from regular function members?

Ans-

| **Feature** | **Static Member Function** | **Regular function member** |
| --- | --- | --- |
| Belongs to | Class (shared among all objects) | Individual objects |
| Access Method | Can be called using ClassName::FunctionName() | Needs an object to be called (obj.FunctionName()) |
| Access to Non-Static Members | Cannot access non-static variables or functions | Can access both static and non-static members |
| Access to Static Members | Can access only static data members | Can access static and non-static members |
| Usage | Used for operations related to the class as a whole (e.g., counting objects) | Used for object-specific operations (e.g., modifying instance variables) |

15. Provide an example of a class with static data and function members.

#include <iostream>

using namespace std;

class Counter {

private:

static int count; // Static data member (shared by all objects)

public:

// Constructor increments count when an object is created

Counter() {

count++;

}

// Static function to get the count value

static int getCount() {

return count;

}

};

// Initialize static data member outside the class

int Counter::count = 0;

int main() {

Counter obj1, obj2, obj3; // Creating objects

// Accessing static function without an object

cout << "Total Objects Created: " << Counter::getCount() << endl;

return 0;

}