TOPIC: CLASSES AND OBJECTS

Explanation:

- A **class** is a blueprint for creating objects. It defines data members and member functions.
- An **object** is an instance of a class, with its own copies of data members.

Important Pointers:

- A class is declared using the class keyword.
- Objects are created from a class using the class name.
- Classes can have public, private, and protected access specifiers

MEMBER OF THE CLASS

MEMBER OF THE CLASS

1. Static Members
   - Static Variables Member
   - Static Functions Member

2. Non-Static Members
   - Non-Static Variables Member
   - Non-Static Functions Member

Example:

```cpp
#include <iostream>
using namespace std;

class Car {
public:
    string brand;
    void showBrand() {
        cout << "Car Brand: " << brand << endl;
    }
};

int main() {
    Car car1;
    car1.brand = "Toyota";
    car1.showBrand();
    return 0;
}
```

TOPIC: STATIC MEMBERS

Explanation:

- Static members belong to the class itself, not individual objects.
- They are shared across all objects of the class.
- Static members are declared using the static keyword.

Important Pointers:

- A **static variable** retains its value throughout program execution.
- A **static function** can only access static variables.

TOPIC: STATIC VARIABLES

Explanation:

- A static variable inside a class is shared among all the objects of the class.
- It is initialized only once and retains its value throughout the program.
- A static variable belongs to the class, not to any specific object.

Important Pointers:

- Declared using **static** keyword inside the class.
- It must be defined outside the class separately.
- Can be accessed using the class name.

Example:

```cpp
#include <iostream>
using namespace std;

class Counter {
    static int count;
public:
    Counter() { count++; }
    static void showCount() {
        cout << "Count: " << count << endl;
    }
};
int Counter::count = 0;

int main() {
    Counter c1, c2, c3;
    Counter::showCount();
    return 0;
}
```

STUDENT TASK FOR STATIC VARIABLES

Question**:**
Create a Company class with:

- A static variable employeeCount to track the number of employees.
- A constructor that increments employeeCount whenever a new employee is added.
- A static function showTotalEmployees() to display the total number of employees.
- Create multiple objects and observe how the static variable retains its value across objects.

TOPIC: STATIC FUNCTIONS

Explanation:

- A static function is a member function of a class that can be called without created an object.
- It can be declared using **static** keyword.
- It can only access the static members of the class where they can’t access the non-static members of the class.

Important Pointers:

- Declared using **static** keyword inside the class.
- It can called using the class name.
- It can’t access the non-static members because it doesn’t operate on a specific object.
- Often used as utility function that don’t require an object of the class.

Example:

```
#include <iostream>
using namespace std;

class Counter {
    static int count;  // Static data member
public:
    static void showCount() {  // Static function
        cout << "Total Objects Created: " << count << endl;
    }
    Counter() {
        count++;
    }
};

int Counter::count = 0;  // Initialize static variable

int main() {
    Counter c1, c2;
    Counter::showCount();  // Calling static function without an object
    return 0;
}
```

STUDENT TASK FOR STATIC FUNCTIONS

Question**:**

Create a Bank class that:

- Has a static variable interestRate (shared across all the objects)
- Includes the static function setInterestRate(float rate) to modify the rate
- Has a static function to getInterest() to display the current interest rate.
- Call setInterestRate() and getInterest() from the main() method without creating the object.

TOPIC: STATIC OBJECTS

Explanation:

- Static objects retain their values between function calls.
- They are initialized only once and exist throughout the program.

Important Pointers:

- Declared inside a function using static keyword.
- Memory for a static object is allocated only once.

Example:

```cpp
#include <iostream>
using namespace std;

class Test {
public:
    Test() { cout << "Constructor Called" << endl; }
};

void demo() {
    static Test t; // Only initialized once
}

int main() {
    demo();
    demo();
    return 0;
}
```

STUDENT TASK FOR STATIC OBJECTS

Question**:**
Create a function createInstance() that contains a **static object** of a class Counter.

- The constructor should print "Object Created".
- Call createInstance() multiple times from main() and observe how many times the constructor executes.

STUDENT TASK FOR STATIC MEMBERS

Question**:**
Create a Library class with:

- A **static variable** totalBooks to count the total number of books issued.
- A function issueBook() that increments totalBooks.
- A **static function** showTotalBooks() to display the total books issued.
- Create multiple objects and observe the shared totalBooks value.

TOPIC: CONSTANT MEMBER FUNCTIONS

Explanation:

- A constant member function cannot modify any data members of the class.
- It is declared using the const keyword at the end of the function declaration.

Important Pointers:

- Helps in maintaining immutability for specific operations.
- Can only call other constant member functions.

Example:

```cpp
#include <iostream>
using namespace std;

class Demo {
public:
    void show() const {
        cout << "Constant Function" << endl;
    }
};

int main() {
    Demo obj;
    obj.show();
    return 0;
}
```

STUDENT TASK FOR CONSTANT MEMBER FUNCTIONS

Question**:**
Create a Rectangle class with

- Data Members **length** and **width**
- A constant member function **getArea**() that returns the area of the rectangle.
- Try modifying length inside **getArea**() and observe the error.

TOPIC: CONSTANT OBJECTS

Explanation:

- A constant object is an object whose values cannot be modified after initialization.
- Declared using the const keyword.

Important Pointers:

- Can only call **constant member functions**.
- Helps in preventing accidental modifications.

Example:

```cpp
#include <iostream>
using namespace std;

class Test {
public:
    void show() const {
        cout << "Constant Object Example" << endl;
    }
};

int main() {
    const Test t;
    t.show(); // Allowed, as show() is a constant function
    return 0;
}
```

STUDENT TASK FOR CONSTANT OBJECTS

Question**:**

Create a Car class with a **constant object** in main().

- The class should have a function showDetails() marked as const.
- Try calling a **non-constant function** using the constant object and observe the error.

TOPIC: FRIEND FUNCTIONS

Explanation:

- A **friend function** is a function that is not a member of a class but has access to its private members.
- It is declared using the friend keyword inside the class.

Important Pointers:

- Friend functions can access **private** and **protected** members.
- They are defined outside the class but can access its members.

Example:

```cpp
#include <iostream>
using namespace std;

class A {
private:
    int x;
public:
    A() { x = 10; }
    friend void show(A);
};

void show(A obj) {
    cout << "Value of x: " << obj.x << endl;
}

int main() {
    A obj;
    show(obj);
    return 0;
}
```

STUDENT TASK FOR FRIEND FUNCTIONS

Question**:**
Create two classes, ClassA and ClassB.

- Make a **friend function** that accesses private data of both classes and displays their values.
- Demonstrate how a function outside the class can still access private members.

TOPIC: FRIEND CLASS

Explanation:

- A **friend class** can access private and protected members of another class.
- It is declared using the friend keyword inside the class definition.

Important Pointers:

- Used when two classes need to work closely together.
- Helps in breaking encapsulation safely when needed.

Example:

Example:

```cpp
#include <iostream>
using namespace std;

class A {
private:
    int data;
public:
    A() { data = 10; }
    friend class B; // Declaring class B as a friend
};

class B {
public:
    void show(A obj) {
        cout << "Data: " << obj.data << endl;
    }
};

int main() {
    A obj;
    B objB;
    objB.show(obj);
    return 0;
}
```

STUDENT TASK FOR FRIEND CLASS

- Question**:**
  Create a class Person with a private member age.
- Create a **friend class** Doctor that has a function checkAge() to access and print age.
- Create objects of both classes and demonstrate how Doctor can access Person's private data.