INTERNATIONAL ISLAMIC UNIVERSITY, ISLAMABAD

SOFTWARE ENGINEERING DEPARTMENT

Total Marks: 05

Obtained Marks: _____

Assignment 01

Last date of Submission: 19 March 2025

Submitted To: **Shakeel Ahmad**

Student Name: **Iqbal Hassan Tariq**

Reg. Number: **004361/BSSE/F24**

Object-Oriented Programming (OOP) - Assignment 01

Instructions:

- Submit the assignment on Google Classroom by the due date.
- Follow the plagiarism policy strictly.

- Use tables, diagrams, and appropriate formatting to enhance clarity.
- Provide C++ code snippets where applicable.

Q1: Programming Paradigms

a. Comparison of Different Programming Paradigms:

Feature	Procedural Programming	Object-Oriented Programming	Functional Programming	Logical Programming
Definition	Based on procedures (functions) that operate on data.	Based on objects and classes.	Focuses on mathematical functions without side effects.	Uses logic and facts to derive conclusions.
Key Concept	Functions and control structures.	Objects, classes, inheritance, and polymorphism.	Pure functions and immutability.	Rules and inference mechanisms.
Data Handling	Uses variables and structured data types.	Encapsulates data inside objects.	Avoids mutable data.	Uses symbolic representations and rules.
Code Reuse	Low reusability; requires repetition.	High reusability via inheritance and polymorphism.	Reuses functions via composition.	Reuses rules for different queries.
Example Lang.	C, Pascal	C++, Java, Python	Haskell, Lisp	Prolog

b. **Advantages and Disadvantages of Object-Oriented Programming**

Advantages:

1. **Modularity:** Code is organized into classes and objects, enhancing maintainability.
2. **Reusability:** Inheritance allows code reuse, reducing redundancy.

Disadvantages:

1. **Performance Overhead:** OOP requires more memory and processing due to abstraction layers.
2. **Complexity:** The learning curve is higher compared to procedural programming.

Q2: Matching OOP Concepts

Concept	Definition
Encapsulation	(b) Combining data and methods that operate on that data.
Abstraction	(c) Hiding complex details and showing only essential features.
Inheritance	(a) Ability of a class to inherit properties of another class.
Polymorphism	(d) Ability of an object to take multiple forms.

Q3: Classes and Objects

a. C++ Class Definition:

```
#include <iostream>
using namespace std;
class Student {
private:
    string name;
    int rollNumber;
    double GPA;
public:
    void setDetails(string n, int r, double g) {
        name = n;
        rollNumber = r;
        GPA = g;
    }
    void displayDetails() {
        cout << "Name: " << name << "\nRoll Number: " << rollNumber << "\nGPA: " << GPA << endl;
    }
};
```

b. Creating and using an object:

```
int main() {
    Student s1;
    s1.setDetails("John Doe", 101, 3.9);
    s1.displayDetails();
    return 0;
}
```

Q4: Access Specifiers

a. Completing the table:

Access Specifier	Inside Class	Outside Class
Public	Accessible	Accessible
Private	Accessible	Not Accessible
Protected	Accessible	Accessible Only in Derived Classes

b. Why keep data members private?

1. **Encapsulation:** Prevents unauthorized access and ensures data integrity.
2. **Security:** Direct modification of attributes can lead to unintended behavior.

Q5: Structures in C++

a. Given the struct:

```
struct Employee {
   int empID;
   string name;
   double salary;
};
```

1. Creating an object:

```
Employee e1;
```

2. Accessing name attribute:

```
e1.name = "Alice";
cout << e1.name;
```

3. Converting Structure to Class

```
class Employee {
private:
    int empID;
    string name;
    double salary;
public:
    void setDetails(int id, string n, double s) {
        empID = id;
        name = n;
        salary = s;
    }
    void displayDetails() {
        cout << "ID: " << empID << "\nName: " << name << "\nSalary: "
<< salary << endl;
    }
};
```

Q6: Constant and Static Members

a. Difference between const and static members:

1. const members cannot be modified after initialization.
2. static members belong to the class, not instances, and are shared among all objects.

```
class Example {
private:
   static int count;
   const int id;
public:
   Example(int i) : id(i) { count++; }
   static void showCount() { cout << count; }
};
```

b. Error correction in given code:

Errors:

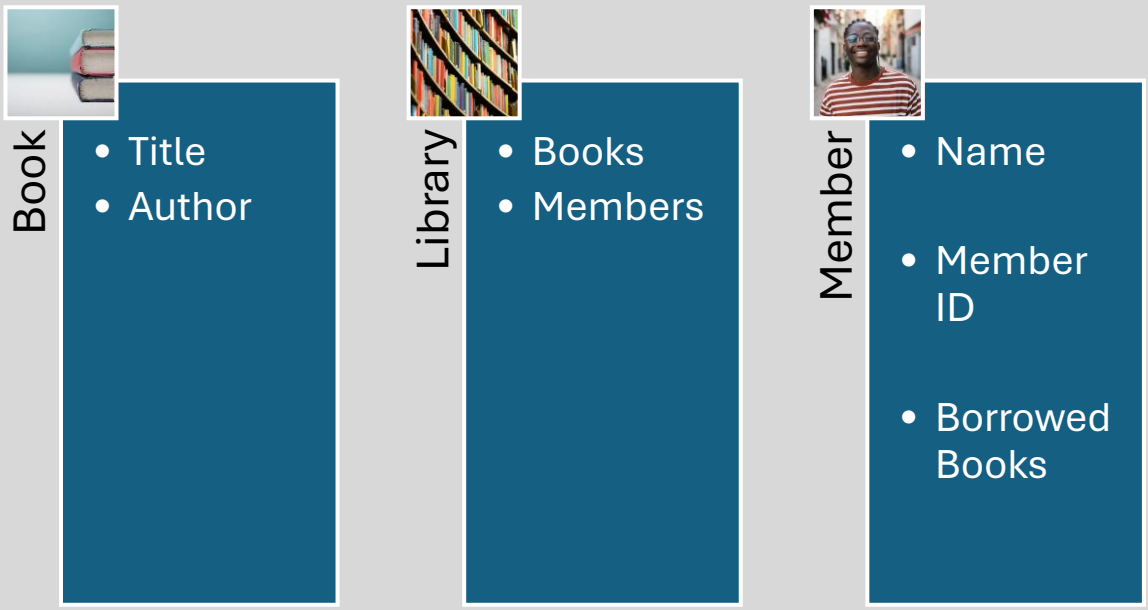
1. const int value; must be initialized in the constructor.
2. static int count; needs an external definition.
3. setValue() cannot modify const member.

Corrected Code:

```
class Demo {
private:
   const int value;
   static int count;
public:
   Demo(int v) : value(v) {}
   static void showCount() { cout << count; }
};   int Demo::count = 0;
```

Q7: OOP in Practice

Library Management System Class Diagram:



b. OOP Principles Used:

1. **Encapsulation:** Data is hidden within classes (e.g., book details are private in Book).
2. **Inheritance:** LibraryItem base class could be used for Book and Magazine