Practical 1:-

Employee Management System

Scenario:

You are tasked with designing a system to manage employees in a company. Each employee has a unique ID, name, department, and salary. The system should support operations like adding, removing, and updating employee details.

Class Design:

- Create a base class Employee with the following:
- Properties: Id, Name, Department, Salary.
- Methods: DisplayDetails() (to print employee details).
- Implement encapsulation to ensure data is validated:
- Salary cannot be negative.
- Department name must not be empty.

Derived Class:

- Add specialized classes like Manager and Developer, inheriting from Employee.
- Include additional properties for these classes:
- Manager: TeamSize (number of team members).
- Developer: ProgrammingLanguages (list of programming languages).
- Use of Collections:
- Use a List<Employee> to store employee data.
- Add methods to perform the following:
- Add an Employee: Add a new employee (manager or developer) to the list.
- Remove an Employee: Remove an employee by ID.
- Update an Employee: Update an employee’s details by ID.

Polymorphism:

Use polymorphism for DisplayDetails() so that Manager and Developer can show their specialized details.

Static Members:

Add a static property to track the total number of employees in the system.

Practical 2:-

Product Management System :

1. Add, update, delete, and display products.
2. Categorize products as Electronics, Clothing, and Groceries with different pricing rules.
3. Store products in a collection (List) for efficient management.
4. Follow SOLID principles for scalability and maintainability.

Practical 3:-

The scenario involves a Clock class that raises an event every second, and a Display class that listens to this event to update the time displayed on the screen. This example illustrates the core concepts of defining, subscribing to, and raising events in C#.
Explanation
TimeEventArgs: This class defines the event data, which includes the current time.
Clock: This class has an event SecondChanged that is triggered every second. The Start method runs an infinite loop, raising the event every second.
Display: This class subscribes to the Clock's SecondChanged event and updates the display every time the event is raised.
Program: This is the main entry point. It creates instances of Clock and Display, subscribes the Display to the Clock's events, and starts the Clock.
When you run this program, the Display class will print the current time to the console every second, demonstrating the use of events in C#.