Experiment – 8 (Improvised)

1. Imagine you're designing a **Notification System** for a mobile application. You have different types of notifications **(e.g., SMS, Email, Push Notification)**. A base class Notification is created with an **abstract send() method**, which is overridden in the derived classes (SMSNotification, EmailNotification). The derived class will define the notification content.

   Task:

   - Create a base class Notification with an abstract send() method.
   - The derived class SMSNotification should override the send() method to print a message, which contains the text message being sent.
   - In the base class constructor, call the send() method.
   - In the main() method, create an object of SMSNotification and call its send() method.

2. Imagine you are developing a **Smart Home System** where various home appliances like lights, fans, and air conditioners are controlled remotely. The operations to control these devices (e.g., turn on/off) must be publicly accessible so that the user can control them from a smartphone app or voice assistant.

   Task:

   - Create an interface **ApplianceControl** with two methods: **turnOn()** and **turnOff()**.
   - Write a program that implements this interface in classes such as **Light, Fan,** and **AirConditioner,** where each class defines how the appliances are turned on or off.

- Demonstrate that the **turnOn()** and **turnOff()** methods are automatically public by calling them in the main() method for different appliance objects (Light, Fan, and AirConditioner).
- Write a program that demonstrates that all methods in an interface are automatically public, as all devices should publicly expose their start/stop functionality.

3. Design a smart car system where various functionalities like engine control, air conditioning (AC), and music system are managed. These functionalities should be controlled through interfaces, and the car should also keep track of its speed and overall status.

Problem Statement:

i) Create Three Interfaces:

a) EngineControl: Methods **startEngine()** and **stopEngine().**

b) ACControl: Methods **turnOnAC()** and **turnOffAC().**

c) MusicControl: Methods **playMusic()** and **stopMusic().**

ii) Create a New Interface:

CarControl: This interface extends **EngineControl, ACControl, and MusicControl,** and adds a method **checkCarStatus()** to display the status of the engine, AC, music, and speed.

iii) Concrete Class:

Vehicle: A class with a **speed** attribute and a method **getSpeed()** to return the car's current speed.

iv) Implementing Class:

SmartCar: Implements **CarControl** and inherits from **Vehicle**. Implement all methods from the interfaces, ensuring each method prints a message (e.g., "Engine started", "AC turned off"). The **checkCarStatus()** method should print the current state of all components (engine, AC, music) and the speed.

v) Four Methods:

Create a class with four methods, each accepting one of the following interfaces as an argument: EngineControl, ACControl, MusicControl, and CarControl. These methods will demonstrate control over the car's systems.

vi) Main Method:

In main(), create an object of SmartCar. Use it to call each of the four methods, showing control of the engine, AC, and music, and finally check the overall status using CarControl.