**Exercise 1:**

Create an abstract class **Vehicle (modelId,** **nameOfTheModel, typeOfVehicle)**  with abstract methods **startEngine()** and **stopEngine()**. Create two subclasses, **Car** and **Motorcycle**, that inherit from **Vehicle**. Implement the methods in each subclass to print out appropriate messages.

Some of the examples of cars

1. **Sedans:**
   * Model ID: SD-001
   * Model Name: Acclaim Deluxe
2. **SUVs:**
   * Model ID: SUV-002
   * Model Name: Expedition Xplorer
3. **Trucks:**
   * Model ID: TRK-003
   * Model Name: Titan Hauler
4. **Sports Cars:**
   * Model ID: SC-004
   * Model Name: Velocity Viper
5. **Electric Cars:**
   * Model ID: EC-005
   * Model Name: Electron Evolve
6. **Motorcycles:**
   * Model ID: MC-006
   * Model Name: Blaze Raptor
7. **Vans:**
   * Model ID: VN-007
   * Model Name: Transit Voyager
8. **Crossover Vehicles:**
   * Model ID: CV-008
   * Model Name: Crosswind Cruiser
9. **Luxury Cars:**
   * Model ID: LC-009
   * Model Name: Elegance Elite
10. **Hybrid Cars:**
    * Model ID: HC-010
    * Model Name: EcoHarmony Hybrid

Some of the examples of motorcycles

1. **Model ID: MC-101**
   * Model Name: Thunderbolt Titan
   * Type of Vehicle: Cruiser
2. **Model ID: MC-202**
   * Model Name: Vortex Viper
   * Type of Vehicle: Sportbike
3. **Model ID: MC-303**
   * Model Name: Shadow Sabre
   * Type of Vehicle: Cruiser
4. **Model ID: MC-404**
   * Model Name: Blaze Breaker
   * Type of Vehicle: Off-Road/Adventure
5. **Model ID: MC-505**
   * Model Name: Phantom Fury
   * Type of Vehicle: Touring
6. **Model ID: MC-606**
   * Model Name: Zenith Zephyr
   * Type of Vehicle: Standard/Street
7. **Model ID: MC-707**
   * Model Name: Inferno Ignitor
   * Type of Vehicle: Sportbike
8. **Model ID: MC-808**
   * Model Name: Stealth Stinger
   * Type of Vehicle: Cruiser
9. **Model ID: MC-909**
   * Model Name: Nova Nighthawk
   * Type of Vehicle: Adventure
10. **Model ID: MC-1010**
    * Model Name: Eclipse Enigma
    * Type of Vehicle: Sport Touring

**Exercise 2:**

**Create an abstract class Shape with an abstract method calculateArea(). Create two subclasses, Triangle and Square, that inherit from Shape. Implement the calculateArea() method to calculate the area of each shape.**

**Exercise 3:**

**Create an abstract class Person with attributes name and age, and an abstract method displayDetails(). Create two subclasses, Student and Teacher, that inherit from Person. Implement the displayDetails() method to display the details of each person.**

**Exercise 4:**

**Create an abstract class BankAccount with attributes accountNumber and balance, and abstract methods deposit() and withdraw(). Create two subclasses, SavingsAccount and CheckingAccount, that inherit from BankAccount. Implement the methods in each subclass to handle deposits and withdrawals appropriately.**

**Exercise 5:**

**Create an abstract class Shape with attributes color and filled, and an abstract method draw(). Create two subclasses, Circle and Rectangle, that inherit from Shape. Implement the draw() method to print out drawing instructions for each shape.**

**Exercise 6:**

**Create an abstract class Employee with attributes name, salary, and a constructor to initialize them. Create two subclasses, Manager and Worker, that inherit from Employee. Implement a method displayInfo() in each subclass to display the details of each employee**.

**Exercise 7:**

**Create an abstract class Animal with attributes name and sound, and a constructor to initialize them. Create two subclasses, Dog and Cat, that inherit from Animal. Implement a method makeSound() in each subclass to print the sound of the animal.**

**Exercise 8:**

**Create an abstract class Employee with attributes name and salary, and abstract methods calculatePay() and displayInfo(). Create two subclasses, Manager and Worker, that inherit from Employee. Implement the methods in each subclass to calculate pay and display employee information.**

**Exercise 9:**

**Create an abstract class BankAccount with attributes accountNumber and balance, and abstract methods deposit() and withdraw(). Create two subclasses, SavingsAccount and CheckingAccount, that inherit from BankAccount. Implement the methods in each subclass to handle deposits and withdrawals, with a minimum balance requirement for the checking account.**

**Exercise 10:**

**Create an abstract class Shape with an abstract method calculateArea() and a concrete method displayInfo() to display the shape's name. Create two subclasses, Rectangle and Circle, that inherit from Shape. Implement the calculateArea() method in each subclass to calculate the area of each shape.**