1. Shape Hierarchy: Create an abstract class Shape with an abstract method calculateArea() and a concrete method display().

The display() method should print a message like "This is a shape." Create two subclasses, Circle and Rectangle, that extend

Shape and provide their own implementations for calculateArea(). The Circle class should have a radius property, and the

Rectangle class should have length and width properties.

2. Animal Sounds: Design an abstract class Animal with an abstract method makeSound() and a concrete method sleep().

The sleep() method can simply print "Zzz...". Create subclasses Dog and Cat that extend Animal and implement makeSound()

to print "Woof" and "Meow" respectively.

3. Bank Account System: Create an abstract class BankAccount with abstract methods deposit() and withdraw(). It should also

have a concrete method getBalance() that returns the account balance. Create two subclasses, SavingsAccount and

CurrentAccount, that extend BankAccount and implement the abstract methods according to the rules of each account type

(e.g., a SavingsAccount might have a transaction limit).

4. Vehicles: Design an abstract class Vehicle with abstract methods startEngine() and stopEngine(). Implement subclasses

Car and Motorcycle that extend Vehicle and provide a concrete implementation for each method.

5. Musical Instruments: Create an abstract class Instrument with abstract methods play() and tune(). Implement subclasses

Glockenspiel and Violin that extend Instrument and provide their own unique implementation for each method.

6. Drawable Interface: Define an interface Drawable with a single abstract method draw(). Then, create classes Circle,

Rectangle, and Triangle that all implement the Drawable interface and provide a concrete draw() method that prints a

message indicating it's drawing that shape.

7. Sortable Interface: Define an interface Sortable with a method sort() that takes an array of integers. Create two separate

classes, BubbleSort and SelectionSort, that implement the Sortable interface and provide their own sorting algorithm

implementation in the sort() method.

8. Playable Interface: Create an interface Playable with an abstract method play(). Create classes Football, Volleyball,

and Basketball that implement the Playable interface and override the play() method to describe how to play the

respective sport.

9. Encryptable Interface: Create an interface Encryptable with two methods: encrypt(String data) and decrypt(String encryptedData).

Then, create a class AESCipher that implements this interface, providing a simple encryption and decryption logic.

10. Person Activity: Design an abstract class Person with abstract methods eat() and exercise(). Create subclasses Athlete

and LazyPerson that extend the Person class and implement the eat() and exercise() methods differently based on their

specific behavior.