****

**Programming Fundamentals - OOP Assignment (v2)**

**Task 1 (Code Already Provided): Design a Zoo Management System Using OOP**

**Objective:** The goal of this assignment is to design a Zoo Management System using the advanced principles of Object-Oriented Programming (OOP) that you have learned. This will help you understand and apply the concepts of classes, objects, properties, methods, along with the advanced concepts of OOP, which are encapsulation, abstraction, inheritance, polymorphism and interfaces.

**Task Details:**

The task is to enhance the previous task by creating an abstract class for animals, interface for sound behavior, encapsulation of animal attributes, overloading the 'Eat' method, and displaying polymorphism through various methods.

**Instructions:**

1. ISoundBehaviour Interface: Create an interface 'ISoundBehaviour' with a method 'MakeSound()' which will be implemented by the specific animal classes.
2. Abstract Animal Class: Modify the 'Animal' class to make it abstract. This class should have 'Name', 'Age', and 'Species' as protected attributes (encapsulation), and it should declare an abstract 'Eat()' method.
3. Specific Animal Classes: Modify the subclasses (Lion, Elephant, Monkey) to inherit from the abstract 'Animal' class and implement 'ISoundBehaviour' interface.

· Override the abstract 'Eat()' method in each specific animal class. Additionally, overload the 'Eat()' method to accept food as a parameter.

· Implement the 'MakeSound()' method for each class.

1. Zoo Class: No changes required.
2. Main Program: Create an instance of the 'Zoo' class. Create instances of your specific animal classes and add them to the zoo. Call the 'FeedAllAnimals()' method and 'MakeSound()' for each animal.

The code for the above assignment is already given. Your goal is to understand the problem statement, analyze and debug the given code.

**Task 2: Create a Bank Account Management System**

**Objective:** The goal of this assignment is to design a Bank Account Management System using the principles of Object-Oriented Programming (OOP). You should demonstrate the concepts of classes, objects, inheritance, encapsulation, polymorphism, abstraction, and interfaces.

Task Details: You are tasked to create classes and objects for different types of bank accounts in a bank. Each bank account has common attributes like account number, account holder's name, and balance. There are two types of accounts: Savings Account and Checking Account.

**Instructions:**

1. IBankAccount Interface: Create an interface 'IBankAccount' with methods 'Deposit()' and 'Withdraw()'.
2. BankAccount Class: Create an abstract class 'BankAccount' that implements the 'IBankAccount' interface. It should have protected attributes for account number, account holder's name, and balance (encapsulation). Include an abstract method 'DisplayAccountInfo()' to display the account's information.
3. SavingsAccount Class: Create a subclass 'SavingsAccount' that inherits from the 'BankAccount' class and has an additional attribute 'interestRate'. Override the 'Deposit()' and 'DisplayAccountInfo()' methods. The 'Deposit()' method should add interest to the account balance whenever a deposit is made.
4. CheckingAccount Class: Create another subclass 'CheckingAccount' that also inherits from the 'BankAccount' class. Override the 'Withdraw()' and 'DisplayAccountInfo()' methods. The 'Withdraw()' method should display an error message if the withdrawal amount exceeds the account balance.
5. Bank Class: Create a class 'Bank' that has a list of BankAccounts. This class should have methods to 'AddAccount()', 'DepositToAccount()', and 'WithdrawFromAccount()'.
6. Main Program: In your main program, create an instance of the 'Bank' class. Create instances of your 'SavingsAccount' and 'CheckingAccount' classes and add them to the bank. Then make some deposits and withdrawals to demonstrate the polymorphic behavior of 'Deposit()' and 'Withdraw()' methods.

In this modified task, you should demonstrate:

* Abstraction: 'BankAccount' is an abstract class.
* Encapsulation: 'account number', 'account holder's name', and 'balance' are encapsulated within the 'BankAccount' class.
* Polymorphism: 'Deposit()', 'Withdraw()', and 'DisplayAccountInfo()' methods are overridden in subclasses.
* Inheritance: 'SavingsAccount' and 'CheckingAccount' classes inherit from the 'BankAccount' class.
* Interfaces: 'IBankAccount' interface is implemented by 'BankAccount' class.