Shared Chat • Default (GPT-3.5)

Python Programming Fundamentals Assignment

September 25, 2023

Explain encapsulation in basic terms 10 basic examples including the bank example to explain this

Encapsulation is a fundamental concept in object-oriented programming that involves bundling data (attributes) and the methods (functions) that operate on that data into a single unit called a class. The data within a class is typically kept private or protected, and access to it is controlled through well-defined interfaces (public methods). Here are 10 basic examples, including one related to a bank, to help explain encapsulation:

1. **Person Class**: Imagine a class representing a person with attributes like name, age, and address. The class encapsulates this data and provides methods like `getAge()` and `changeAddress()` to control access and modifications.
2. **Car Class**: A car class might encapsulate data like make, model, and year, with methods for starting the engine and changing the car's speed.
3. **Temperature Converter**: A class that encapsulates methods for converting temperatures between Celsius and Fahrenheit, keeping the conversion formula hidden from the user.
4. **Email Client**: An email client class might encapsulate the functionality for sending, receiving, and organizing emails.
5. **Inventory System**: In a store's inventory system, a class could represent products with attributes like name, price, and quantity, and provide methods to update these values.
6. **Student Record**: A student record class could encapsulate student information such as name, ID, and GPA, with methods to add courses and calculate averages.
7. **Library Catalog**: In a library catalog, a class might encapsulate book details like title, author, and publication year, along with methods for checking out and returning books.
8. **Smartphone**: A smartphone class could encapsulate features like screen size, battery life, and camera specifications, offering methods for making calls, taking photos, and installing apps.

9. **Online Shopping Cart**: A shopping cart class could encapsulate information about items added to the cart, their prices, and quantities, with methods for adding, removing, and calculating the total cost.
10. **Bank Account**: In a bank example, a bank account class might encapsulate details such as account number, balance, and account holder's name, along with methods for depositing, withdrawing, and checking the balance. This encapsulation helps ensure that sensitive financial data is kept secure and is only accessible through controlled methods.
    In each of these examples, encapsulation allows you to hide the internal implementation details of the class while providing well-defined interfaces for interacting with the data and functionality. This helps improve code organization, maintainability, and security by preventing unauthorized access or unintended modifications to the underlying data.

give code in each of all the 10 examples with their usage and explanation

Certainly! Here are code examples for each of the 10 scenarios mentioned earlier, along with explanations and sample usages:

1. **Person Class**:

python Copy code

```python
class Person:
    def __init__(self, name, age, address):
        self.name = name
        self.age = age
        self.address = address

    def getAge(self):
        return self.age

    def changeAddress(self, new_address):
        self.address = new_address

# Usage:
person1 = Person("Alice", 25, "123 Main St")
print(person1.getAge())  # Output: 25
person1.changeAddress("456 Elm St")
print(person1.address)   # Output: 456 Elm St
```

1. **Car Class:**

python Copy code

```python
class Car:
    def __init__(self, make, model, year):
        self.make = make
        self.model = model
        self.year = year
        self.is_engine_on = False
        self.speed = 0

    def startEngine(self):
        self.is_engine_on = True

    def accelerate(self, amount):
        if self.is_engine_on:
            self.speed += amount

# Usage:
my_car = Car("Toyota", "Camry", 2022)
my_car.startEngine()
my_car.accelerate(30)
print(my_car.speed)  # Output: 30
```

1. **Temperature Converter** (No need for usage as it's a simple function encapsulation example).

python Copy code

```python
class TemperatureConverter:
    @staticmethod
    def celsius_to_fahrenheit(celsius):
        return (celsius * 9/5) + 32

    @staticmethod
    def fahrenheit_to_celsius(fahrenheit):
        return (fahrenheit - 32) * 5/9
```

1. **Email Client** (No need for usage as it's a placeholder for email-related methods).

python Copy code

```python
class EmailClient:
    def __init__(self, email_address, password):
        # Initialize email client with user's credentials.
        pass

    def sendEmail(self, recipient, subject, body):
        # Code to send an email.
        pass

    def receiveEmail(self):
        # Code to receive emails.
        pass
```

1. **Inventory System**:

python Copy code

```python
class Product:
    def __init__(self, name, price, quantity):
        self.name = name
        self.price = price
        self.quantity = quantity

    def updatePrice(self, new_price):
        self.price = new_price

    def updateQuantity(self, new_quantity):
        self.quantity = new_quantity

# Usage:
product1 = Product("Widget", 10.99, 100)
product1.updatePrice(11.99)
product1.updateQuantity(90)
print(product1.price)     # Output: 11.99
print(product1.quantity)  # Output: 90
```

1. **Student Record:**

python Copy code

```python
class Student:
    def __init__(self, name, student_id):
        self.name = name
        self.student_id = student_id
        self.courses = []

    def addCourse(self, course_name):
        self.courses.append(course_name)

    def calculateGPA(self):
        # Code to calculate GPA.
        pass

# Usage:
student1 = Student("Bob", "12345")
student1.addCourse("Math")
student1.addCourse("History")
```

1. **Library Catalog** (No need for usage as it's a placeholder for library-related methods).

python Copy code

```python
class LibraryCatalog:
    def __init__(self):
        # Initialize library catalog.
        pass

    def checkoutBook(self, book_title):
        # Code to check out a book.
        pass

    def returnBook(self, book_title):
        # Code to return a book.
        pass
```

1. **Smartphone** (No need for usage as it's a placeholder for smartphone-related methods).

python Copy code

```python
class Smartphone:
    def __init__(self, make, model):
        # Initialize smartphone with make and model.
        pass

    def makeCall(self, recipient):
        # Code to make a call.
        pass

    def takePhoto(self):
        # Code to take a photo.
        pass
```

1. **Online Shopping Cart**:

python Copy code

```python
class ShoppingCart:
    def __init__(self):
        self.items = []

    def addItem(self, item_name, price, quantity):
        self.items.append({"name": item_name, "price": price, "quar

    def calculateTotal(self):
        total = sum(item["price"] * item["quantity"] for item in se
        return total

# Usage:
cart = ShoppingCart()
cart.addItem("Item A", 10.99, 3)
cart.addItem("Item B", 5.99, 2)
print(cart.calculateTotal())  # Output: 47.95
```

1. **Bank Account:**

python Copy code

```python
class BankAccount:
    def __init__(self, account_number, account_holder, balance):
        self.account_number = account_number
        self.account_holder = account_holder
        self.balance = balance

    def deposit(self, amount):
        self.balance += amount

    def withdraw(self, amount):
        if self.balance >= amount:
            self.balance -= amount
        else:
            print("Insufficient funds!")

# Usage:
account1 = BankAccount("12345", "Alice", 1000.0)
account1.deposit(500.0)
account1.withdraw(200.0)
print(account1.balance)  # Output: 1300.0
```

These code examples illustrate encapsulation by defining classes with private data attributes and methods that provide controlled access and modification of that data. They also showcase how objects of these classes can be created and used in various scenarios.

Continue this conversation

Report content|Terms of use|Privacy policy