Roteiro de Atividade Prática

Nome: GUILHERME HENRIQUE COSTA MOREIRA Turma: 2B

Atividade 1: Encapsulamento em uma Classe Pessoa

Descrição do exercício:
Crie uma classe Pessoa que encapsule atributos, como nome e idade. Esses atributos devem ser privados para evitar acesso direto de fora da classe.

Tempo estimado: 15 minutos

Lista de materiais

- Computador com internet;
- Caderno para anotações;
- 1 caneta.

Procedimento experimental

1. Analise o exemplo-base para a criação do código:

```
class Pessoa:
    def __init__(self, nome, idade):
        self.__nome = nome
        self.__idade = idade

# Uso da classe
pessoa = Pessoa("Ana", 30)
print(pessoa.get_nome())
print(pessoa.get_idade())
```

2. A partir da análise anterior, agora é a sua vez de criar as formas privadas de acesso das informações sobre a pessoa, evitando o acesso direto de fora da classe.

3. Anote o código desenvolvido nas linhas seguintes e envie por meio do AVA.

```
class Pessoa:
    def __init__(self, nome, idade):
        self.__nome = nome
        self.__idade = idade

    def get_nome(self):
        return self.__nome

    def get_idade(self):
        return self.__idade

    def set_nome(self, nome):
        self.__nome = nome

    def set_idade(self, idade):
        self.__idade = idade

# Exemplo de uso:
pessoa = Pessoa('Guilherme', 17)
print(pessoa.get_nome())
print(pessoa.get_idade())
```

Atividade 2: Classe ContaBancaria com Atributos Protegidos

Descrição do exercício:
Desenvolva uma classe ContaBancaria com um atributo protegido saldo. Implemente um método para depositar dinheiro na conta.

Tempo estimado: 15 minutos

Procedimento experimental

1. Analise o exemplo-base para a criação do código:

```
class ContaBancaria:
    def __init__(self, saldo_inicial):
        self._saldo = saldo_inicial  # Atributo protegido

    def get_saldo(self):
        return self._saldo

# Uso da classe
conta = ContaBancaria(1000)
conta.depositar(500)
print(conta.get_saldo())
```

2. Agora, a partir do código analisado, crie o método que permita o depósito do dinheiro na conta, conforme solicitado.

3. Anote o código desenvolvido nas linhas seguintes e envie por meio do AVA.

```python
class ContaBancaria:
    def __init__(self, saldo_inicial):
        self._saldo = saldo_inicial  # Atributo protegido

    def get_saldo(self):
        return self._saldo
```

```
    def depositar(self, valor):
        if valor > 0:
            self._saldo += valor
        else:
            print("Valor para depósito deve ser positivo.")

conta = ContaBancaria(1000)
while True:
    teste = float(input("Digite o quanto você quer depositar: "))
    if teste < 25000:
        conta.depositar(teste)
        print("Saldo atual:", conta.get_saldo())
    else:
        print("Valor máximo para depósito excedido (limite: 25000).")
```

Atividade 3: Classe SensorTemperatura com Getters e Setters

Descrição do exercício:
Crie uma classe SensorTemperatura com um atributo privado temperatura. Utilize getters e setters para acessar e modificar a temperatura.

Tempo estimado: 10 minutos

Procedimento experimental

1. Analise o exemplo-base para criação do código:

```
class SensorTemperatura:
    def __init__(self, temperatura=0):
        self.__temperatura = temperatura

    def set_temperatura(self, nova_temperatura):
        if -50 <= nova_temperatura <= 150:
            self.__temperatura = nova_temperatura
        else:
            print("Temperatura fora do intervalo permitido.")

# Uso da classe
sensor = SensorTemperatura()
sensor.set_temperatura(25)
print(sensor.get_temperatura())
```

2. Agora, a partir do código analisado, crie o método que permita obter o valor da temperatura fora da classe.

3. Anote o código desenvolvido nas linhas seguintes e envie por meio do AVA.

```
class SensorTemperatura:
    def __init__(self, temperatura=0):
        self.__temperatura = temperatura

    def set_temperatura(self, nova_temperatura):
        if -50 <= nova_temperatura <= 150:
            self.__temperatura = nova_temperatura
        else:
            print("Temperatura fora do intervalo permitido.")

    def get_temperatura(self):
        return self.__temperatura

# Uso da classe
```

```
sensor = SensorTemperatura()
sensor.set_temperatura(25)
print(sensor.get_temperatura())
```