UNIVERSIDADE
FEDERAL DO CEARÁ

UNIVERSIDADE FEDERAL DO CEARÁ
CURSO DE TECNÓLOGO EM SEGURANÇA DA INFORMAÇÃO

JOSÉ BASÍLIO BRANDÃO COELHO
RYAN GUILHERME ALVES DE MESQUITA

CRIAÇÃO DE UM COFRINHO DO TIPO PORQUINHO EM PROGRAMAÇÃO ORIENTADA A OBJETOS EM PYTHON.

ITAPAJÉ
2023

Matéria: Desenvolvimento de Sistemas
Curso de Segurança da Informação
Semestre 2023.2

Relatório de Trabalho - Cofre (porquinho)

Autores:
Ryan Guilherme
Basílio Brandão

Resumo:
Este relatório descreve o desenvolvimento de uma classe em Python para um cofrinho do tipo porquinho. O projeto envolve a implementação de métodos e atributos que permitem operações como escolher cor, tamanho, qual dentre algumas opções de moedas colocar, escolher se vai colocar algum item, quebrar o cofrinho, mostrar o volume livre, mostrar seu conteúdo e mostrar o valor total em moedas.

Introdução:
Nesse projeto criamos duas pastas para conseguirmos importar o método “Enum”, na pasta “cofrinho.py” criamos as classes “Moeda”, “Item” e “Cofrinho”. As classes têm os seus respectivos atributos, a class “Moeda”: o valor de suas moedas, M10, M25, M50, M100, a class “Item”: descrição, volume, a class “Cofrinho”: cor, volume_Max, volume_atual, moedas, itens, o projeto contém os métodos: volume_disponivel, adicionar_moeda, adicionar_item, quebrar_cofre, olhar conteudo, valor total.

Desenvolvimento:

A class “Moeda” foi implementada com os seguintes atributos:

- M10: representa sua moeda de valor 10
- M25: representa sua moeda de valor 25

- M50: representa sua moeda de valor 50
- M100: representa sua moeda de valor 100

A class “Item” foi implementada com os seguintes atributos:

- descrição: descreve qual item vai ser inserido no cofre
- volume: descreve o tamanho do item que foi inserido

A class “Cofrinho” foi implementada com os seguintes atributos:

- cor: define sua cor
- volume_Max: define seu volume máximo
- volume_atual: define seu volume atual/volume que está livre
- moedas: define as moedas que já estão no cofre
- itens: define os itens que já estão no cofre

O método construtor da classe “Cofrinho” inicia os valores iniciais dos atributos, volume_atual em 0, moedas em uma lista vazia e itens em uma lista vazia.

Métodos da classe “Cofrinho”

- Método “volume_disponivel”: Verifica seu volume disponível fazendo uma operação de subtrair os atributos “volume_Max” e “volume_atual”(self.volume_Max - self.volume_atual)

- Método “adicionar_moeda”: Verifica se o seu volume disponível é suficiente para o valor da moeda que vai ser escolhida utilizando o método “Enum” na pasta “menu.py”, você vai ter 4 escolhas de moedas, sendo elas(10, 25, 50, 100). Se volume_disponivel for maior ou igual o valor da sua moeda, vai ser adicionado a lista moedas usando o comando .append, e depois o valor da moeda vai ser somado ao atributo “volume_atual”, e depois vai aparecer a mensagem “Moeda de(valor da sua moeda) inserida no cofre.”, e se o volume disponível do cofre não for suficiente para o valor da sua moeda, vai aparecer a mensagem “Volume insuficiente para a moeda.”.

- Método “adicionar_item”: Verifica se volume_disponível é suficiente para o seu tipo de item, seu item vai ser escolhido com o método “Enum” na pasta “menu.py”, se o volume for maior ou igual ao seu item, o item vai ser adicionado à lista itens utilizando o comando .append e o valor desse item vai ser somado ao atributo “volume_atual”, depois vai aparecer a mensagem “Item (descrição do seu item) inserido no cofre”, caso o volume disponível do cofre não seja suficiente para seu item, a mensagem “Volume insuficiente para o item” vai aparecer na sua tela.

- Método “quebrar_cofre”: Caso você escolha a opção quebrar cofre, a mensagem “Cofre quebrado! Itens e Moedas obtidos."” vai aparecer na sua tela, e vai zerar o valor e listas dos atributos volume_atual, moedas e itens

- Método “olhar_conteudo”: Verifica o conteúdo do cofre, mostrando os valores das moedas na lista moedas, e mostrando as descrições dos itens adicionados na lista itens.

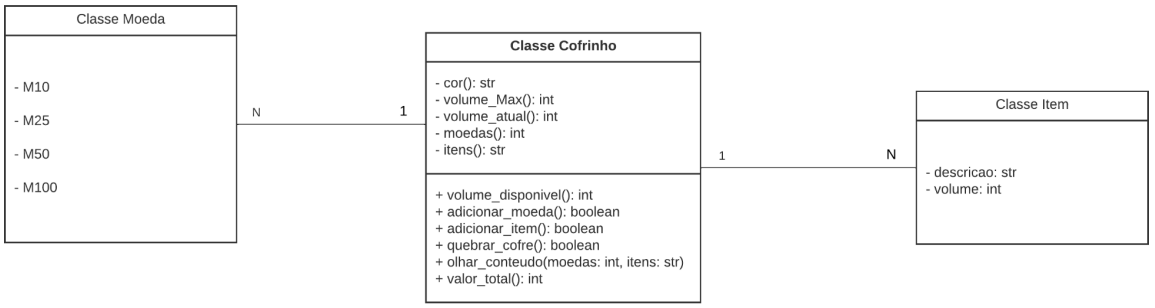
- Método “valor_total”: Criamos uma variável com o nome “valor_moedas” para atribuir a soma dos valores das moedas adicionadas na lista “moedas” utilizando o comando “sum”, e quando a soma for feita, a mensagem “Valor total de moedas no cofre: valor_moedas” na sua tela.

Código pasta cofrinho.py

```
from enum import Enum

class Cofrinho:
    def _init_(self, cor, volume_Max):
        self.cor = cor
        self.volume_Max = volume_Max
        self.volume_atual = 0
        self.moedas = []
        self.itens = []

    def volume_disponivel(self):
        return self.volume_Max - self.volume_atual
```

```
    def adicionar_moeda(self, moeda):
        if self.volume_disponivel() >= moeda.value:
            self.moedas.append(moeda)
            self.volume_atual += moeda.value
            print(f"Moeda de {moeda.value} inserida no cofre.")
        else:
            print("Volume insuficiente para a moeda.")

    def adicionar_item(self, item):
        if self.volume_disponivel() >= item.volume:
            self.itens.append(item)
            self.volume_atual += item.volume
            print(f"Item '{item.descricao}' inserido no cofre.")
        else:
            print("Volume insuficiente para o item.")

    def quebrar_cofre(self):
        print(f"Cofre quebrado! Itens e Moedas obtidos.")
        self.moedas = []
        self.itens = []
        self.volume_atual = 0

    def olhar_conteudo(self):
        print(f"\nConteúdo do cofre:")
        print(f"Moedas: {[moeda.value for moeda in self.moedas]}")
        print(f"Itens: {[item.descricao for item in self.itens]}")

    def valor_total(self):
        valor_moedas = sum([moeda.value for moeda in self.moedas])
        print(f"Valor total em moedas no cofre: {valor_moedas}")


class Item:
    def _init_(self, descricao, volume):
        self.descricao = descricao
        self.volume = volume


class Moeda(Enum):
    M10 = 10
    M25 = 25
    M50 = 50
    M100 = 100
```

Diagrama de Classe:



Relacionamentos entre as classes:

Cofrinho e Moeda (1 para muitos):
Um Cofrinho pode conter várias Moedas, indicando uma relação de 1 para muitos. Essa relação é expressa pelo método adicionar_moeda() na classe Cofrinho, que permite adicionar uma moeda ao cofrinho.

Cofrinho e Item (1 para muitos):
Um Cofrinho pode conter vários Itens, indicando uma relação de 1 para muitos. Essa relação é expressa pelo método adicionar_item() na classe Cofrinho, que permite adicionar um item ao cofrinho.

Moeda e Cofrinho (Muitos para 1):
Muitas Moedas podem estar associadas a um único Cofrinho. Isso é refletido na relação entre Moeda e Cofrinho, indicando que várias moedas podem estar em um cofrinho.

Item e Cofrinho (Muitos para 1):
Muitos Itens podem estar associados a um único Cofrinho. Isso é refletido na relação entre Item e Cofrinho, indicando que vários itens podem estar em um cofrinho.

Conclusão

O projeto de desenvolvimento de uma classe Cofrinho foi concluído com êxito. Todos os métodos foram implementados e testados. O código demonstra o funcionamento adequado das operações relacionadas a um cofrinho.

Referências

pythonando. O guia completo de POO com python. 2022. YouTube.https://youtu.be/jeLeW6q9Mo4?si=2vVFFcSyWt35I2_m

Hashtag Programação. Como sair do zero em Classes no python - self e init explicados.2022.Youtube.https://youtu.be/gomDSZaay3E?si=G9tDdPVsW-XtxXYn

pythonando. O guia completo de POO com python - staticmethods vs class methods. 2022. YouTube.https://youtu.be/gLaZQYZjcOE?si=hkGrpIe9CzJa0sty