Trabalho - Contas Bancárias

Um banco contratou você para renovar seu sistema bancário, de modo a separar a lógica de negócio das interfaces de usuário. Dessa forma, as diversas formas de interação com os clientes (caixa eletrônico, internet banking e aplicativo) poderão acessar o mesmo conjunto de operações bancárias de maneira uniforme.

1. Representação em memória

Sua primeira tarefa é definir como representar as contas bancárias e suas movimentações na memória, implementando as operações fundamentais sobre esse conjunto de dados.

Identifique os dados necessários para representar:

- uma conta bancária;
- uma movimentação financeira (depósito, saque, pagamento etc.).

Defina as abstrações principais:

- Conta
- Movimentação

Você pode usar classes, dicionários ou outras estruturas (em Python ou Java), mas deve justificar a escolha e apresentar a estrutura de dados graficamente.

Entregue um diagrama em arquivo contas.png ou contas.jpg que represente como as contas e suas movimentações estão armazenadas na memória.

2. Operações básicas

Implemente as operações bancárias em um módulo chamado contas.py (Python) ou Contas.java (Java). As operações mínimas são:

- Criar conta bancária – cada conta é identificada por um número único.
- Realizar movimentação financeira – depósito, saque ou pagamento.
    - Toda movimentação contém:

    - data (tupla ano, mês, dia ou equivalente);
    - valor (positivo ou negativo);
    - descrição (texto).
  - Restrições:
    - o saldo nunca pode ser negativo;
    - não é possível registrar movimentações retroativas.
- Consultar saldo atual de uma conta.
- Emitir extrato a partir de uma data.
- Fechar conta, apenas se o saldo for zero.

Mensagens de erro devem ser exibidas para:

- contas inexistentes;
- movimentações retroativas;
- saldo insuficiente;
- tentativa de fechar conta com saldo diferente de zero.

3. Interface de teste

Crie uma interface simples (interface.py ou Interface.java) de linha de comando que:

- leia operações da entrada padrão;
- execute os comandos chamando as funções do módulo principal;
- exiba os resultados.

Operações aceitas:

- abrir → cria conta;
- depositar → realiza depósito;
- sacar → realiza saque;
- saldo → mostra saldo;
- extrato → mostra movimentações desde uma data;
- fechar → tenta encerrar a conta;
- sair → encerra o programa.

4. Aplicação de Padrões de Projeto

Para estruturar melhor seu código, utilize pelo menos três padrões de projeto, escolhidos dentre os seguintes:

1. **MVC (Model–View–Controller)**
   Separe a lógica de negócio (*Model*), a interface (*View*) e o controle das operações (*Controller*).

2. **Factory Method**
   Use um método-fábrica para criar novas contas ou movimentações, garantindo flexibilidade na criação de diferentes tipos de contas no futuro.

3. **Observer**
   Permita que a interface ou módulos auxiliares "observem" mudanças nas contas (ex.: notificação de movimentação).

Outros padrões também podem ser usados (por exemplo, Strategy, Command, Singleton), desde que devidamente justificados no relatório.

O relatório deve conter uma breve explicação (5 a 10 linhas) de como cada padrão foi aplicado.

5. Entregáveis

1. Código-fonte completo (.py ou .java).

2. Arquivo de diagrama (contas.png ou .jpg).

3. Relatório curto (README.md ou .pdf) descrevendo:
    1. como o sistema foi estruturado;
    2. quais padrões de projeto foram usados e onde;
    3. como as operações podem ser testadas.

6. Exemplo de execução

```
abrir

111

depositar

111 500.00 10/10/2021 Primeiro depósito

saldo

111

sacar

111 200.00 11/10/2021 Pagamento

extrato

111 01/10/2021

fechar

111

sair
```