TRABALHO

Apresentação

Neste trabalho, você deve apresentar um problema que envolva o gerenciamento de um conjunto de entidades, como descrito nas seções a seguir. Você deve então propor e desenvolver uma solução para este problema usando a linguagem Java e os conceitos vistos na disciplina: Orientação a Objetos, Herança, Polimorfismo, Classe Abstrata, Interface e Exceção. **Todos** estes conceitos devem ser aplicados no código desenvolvido de maneira adequada para resolver o problema proposto.

Para auxiliar na descrição do trabalho, considere o exemplo de cadastro de contas em um banco, como foi desenvolvido em algumas tarefas da disciplina. Você **não deve** usar este exemplo no seu trabalho. Pense em algo semelhante a isto mas diferente. A seguir, eu descreverei os requisitos mínimos do trabalho usando este exemplo.

Grupos

Os grupos devem ser formados por 3, 4 ou 5 estudantes. Após o prazo de formação dos grupos, o professor fará os ajustes necessários naqueles que não conseguirem formar grupos respeitando estes limites. Mas estes estudantes não receberão a nota da entrega de grupos (ver Entregas no final deste documento). Se você não conseguiu formar um grupo, **faça a entrega do grupo**, mesmo que seja apenas com o seu nome. O professor fará ajustes considerando apenas os estudantes que realizarem a entrega de grupos.

Requisitos Mínimos

Abaixo, eu descrevo os requisitos mínimos que um trabalho deve atender.

- **Classe de Gerenciamento**. O problema deve incluir o gerenciamento de um conjunto de entidades. No nosso exemplo, o problema era gerenciar um conjunto de contas, ou seja, nossas entidades eram as contas. A classe responsável por armazenar estas entidades (que no nosso

exemplo era a classe `Banco`) deve prover, pelo menos, as operações de inserção, busca e remoção de entidades. Para armazenar as entidades, você deve usar alguma classe da biblioteca *Collections* do Java. **Não use apenas um vetor simples!**

- **Entidades**. O problema deve considerar, pelo menos, três tipos de entidades que são relacionadas mas possuem algumas diferenças de comportamento. No nosso exemplo, tínhamos contas de três tipos: corrente, poupança e investimento. Todas estas entidades são contas mas alguns comportamentos são específicos para cada tipo de conta. Por exemplo, o tratamento do saque é diferente em cada conta. Também tínhamos o método `fimDoMes()` cujo comportamento era diferente em cada entidade.

- **Chave Única**. As entidades devem possuir um atributo que as identifique de maneria única. Chamaremos este atributo de **chave** da entidade. No nosso exemplo, a chave seria o número da conta. As operações de busca e remoção na classe de gerenciamento devem receber uma chave para identificar a conta a ser buscada ou removida. A operação de inserção na classe de gerenciamento deve garantir que nunca existirão duas entidades com a mesma chave.

- **Relatórios**. A classe de gerenciamento deve incluir, pelo menos, dois métodos de relatório que exibem as entidades e suas informações relevantes.

    - Um destes métodos deve exibir as entidades em uma ordem específica. Você pode definir qual é esta ordem mas ela deve ser determinada por algum atributo das entidades. No nosso exemplo, poderíamos ter um relatório de contas ordenadas pelo nome do titular.

    - O outro método de relatório deve possuir uma opção de filtrar as entidades de acordo com algum atributo comum a todas as entidades. No nosso exemplo, poderíamos ter um relatório cujo filtro fosse baseado no valor do saldo. Por exemplo, um relatório que exibe todas as contas com saldo maior ou igual a um dado valor.

- **Comportamento Específico**. Você deve definir um comportamento que seja específico para alguns tipos de entidades mas não para todos. A classe de gerenciamento deve incluir um método baseado neste comportamento e invocará o método específico das entidades apropriadas, retornando ou exibindo algum resultado baseado nisto. No nosso exemplo, poderíamos definir um comportamento denominado **rentável**. Uma conta rentável possui algum tipo de rendimento, portanto, teríamos que as contas poupança e investimento são rentáveis. Uma conta-corrente, porém, não é rentável. Então poderíamos criar um método chamado `calculaRendimentos()` na classe `Banco` que deveria calcular o rendimento de cada conta rentável (invocando um método específico deste tipo de contas) e retornaria o total de rendimentos de todas as contas rentáveis do banco.

- **Programa com Interface para o Usuário**. Você deve implementar um programa que permita ao usuário acessar todas as funcionalidades descritas acima de maneira interativa. Deve haver um menu de opções para realizar cada operação e o usuário pode realizar quantas operações e na ordem que desejar. O menu também deve incluir uma opção para encerrar o programa. Esta interface com o usuário pode ser realizada de maneira textual (via linha de comando).

Entregas

O trabalho será dividido em três entregas, cada qual com sua pontuação, critérios de avaliação e prazo. Todas as entregas devem ser realizadas exclusivamente pelo AVA, respeitando os prazos definidos lá. Abaixo, as entregas são descritas em detalhes.

Grupos (1 ponto)

Definir os membros de cada grupo.

Proposta de Problema (2 pontos)

Entregar um documento descrevendo o problema proposto. Você deve descrever o contexto do problema, a classe de gerenciamento, quais são as entidades gerenciadas, assim como suas semelhanças e diferenças, qual a chave única das entidades, quais os relatórios e seus critérios (de ordenação e filtragem) e, por fim, qual é o *comportamento específico* e quais entidades implementam este comportamento. A descrição de cada tópico deve ser sucinta, com foco na descrição do problema e não na solução.

Solução com Programa (7 pontos)

A segunda entrega deve ser o programa completo incluindo as classes de entidade, a classe de gerenciamento, demais classes e interfaces, assim como a interface do programa com o usuário. Como dito acima, esta interface pode ser via linha de comando.