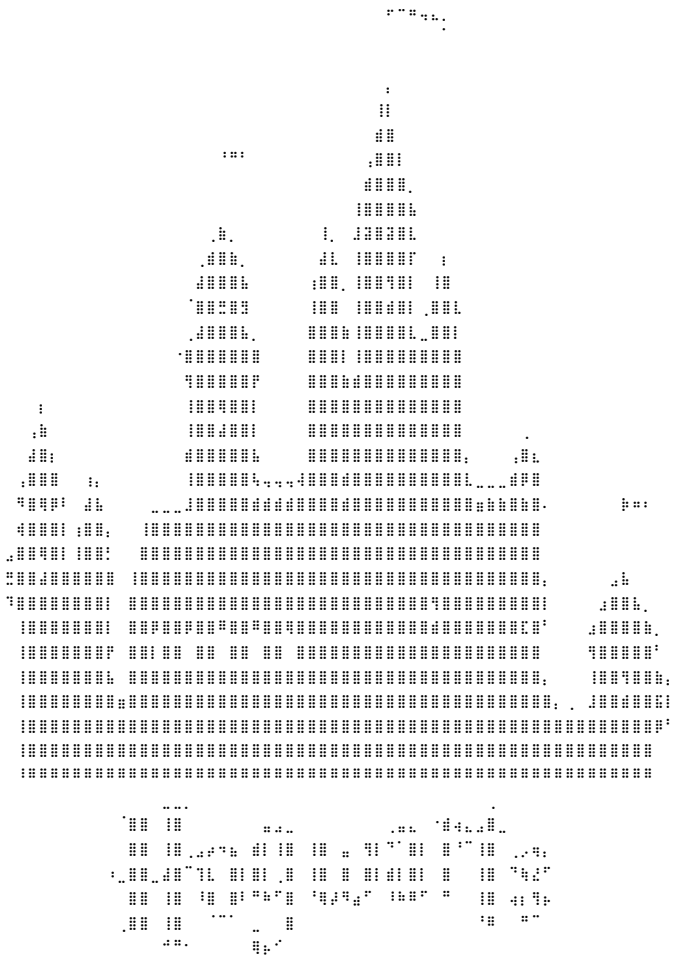
INSTITUTO FEDERAL DA PARAÍBA
CAMPUS CAMPINA GRANDE
BACHARELADO EM ENGENHARIA DA COMPUTAÇÃO
DISCIPLINA DE POO e LAB. POO
PROF. VICTOR ANDRÉ PINHO DE OLIVEIRA

Atividade Unidade III - 3 - Polimorfismo

Lista Única

Instruções

Responda às questões abaixo. Pode usar este próprio documento. Questões práticas devem ser anexadas separadamente.



Legenda:

Quirrell: Questões mais simples e diretas

Olho-Tonto: Questões de nível intermediário

Dolores: Questões de nível mais difícil, exigentes

Voldemort: Questões com nível de exigência altíssimo, desafiadoras

Questões

1. **Quirrell** O que é e como se dá o polimorfismo?

   R:

2. **Quirrell** Quais os mecanismos básicos para o polimorfismo em C++?

   R:

3. **Quirrell** Para que servem métodos virtuais?

   R:

4. **Quirrell** Qual a diferença entre método virtual e método virtual puro?

   R:

5. **Quirrell** Quando empregar um método virtual puro?

   R:

6. **Quirrell** O que é uma classe abstrata? Qual a diferença entre classe abstrata e concreta?

   R:

7. **Quirrell** Como podemos criar uma classe abstrata em C++?

   R:

8. **Quirrell** Quando usar destrutores virtuais?

   R:

9. **Quirrell** (Hierarquia Livro - *Hogwarts* - *Revisitada*) Aproveite a hierarquia Livro - Hogwarts criada na Atividade III - 2. Adapte as classes para que o polimorfismo funcione com o método ler. Mostre que o polimorfismo está funcionando corretamente por meio de exemplos na função main.

10. **Olho-Tonto** (*Hierarquia Forma*) Aproveite o código Aula11Ex4 desenvolvido em "sala" e faça o que se pede:

    a. Na classe abstrata FormaBidimensional, acrescente os métodos virtuais puros calcularArea e calcularPerimetro.

    b. Em cada uma das classes derivadas, adicione atributos específicos para que se possa calcular a área e o perímetro da forma (circulo: raio; quadrado: lado; e triângulo: base e altura).

c. Ajuste os construtores das classes derivadas para receber os devidos argumentos.

d. Implemente (sobrescreva) os métodos virtuais para que cada forma mostre sua própria área e perímetro.

e. Mostre que suas classes estão funcionando adequadamente.

11. Dolores (*Agenda de Contatos*) Crie uma agenda de contatos.

    Nessa agenda, contatos podem ser adicionados, removidos ou pesquisados por nome ou por CPF/CNPJ.

    - A adição de contatos deve ser realizada por meio da sobrecarga ao operador << entre Agenda e Contato ou Agenda.
    - A remoção de contatos deve ser realizada por meio da sobrecarga do operador >> entre Agenda e Contato ou Agenda e string.
    - A pesquisa pode ser implementada por meio de um método comum.

    Deve ser possível, também, visualizar todos os contatos da agenda - por meio da sobrecarga do operador << com o cout e Agenda. Essa visualização deve exibir todos os dados da Pessoa (quer seja física, quer seja jurídica).

    Dois tipos de contatos podem fazer parte da agenda: Pessoa Física e Pessoa Jurídica.

    - A Pessoa Física tem os seguintes atributos: CPF, nome, endereço, data de nascimento, email, estado civil, outros que você julgar necessários.
    - A Pessoa Jurídica possui: CNPJ, nome, endereço, email, inscrição estadual, razão social, outros que você julgar necessários.

    Com base no descrito acima, apresente uma solução Orientada a Objetos em C++ usando os conceitos de Herança e Polimorfismo (e outros que julgar necessários). Crie um código na main para ilustrar que sua solução está funcionando.

12. Dolores (Classe ExpressoHogwarts + Hierarquia Humano - *Hogwarts - revisitados*).

    O professor Snape, com toda sua frieza e rigidez natos, está desconfiando de que alguns Trouxas conseguiram se infiltrar no Expresso de Hogwarts. Diante disso, ele pretende visitar maldosamente alguns assentos para verificar isso pessoalmente, e questionar ao indivíduo sobre quem ele é.

    Para os propósitos desta questão, você deverá aproveitar (e adaptar) a classe ExpressoHogwarts criada na Atividade III - 1 e a hierarquia Humano criada na Atividade III - 2. Adapte a classe ExpressoHogwarts de modo que ela permita o embarque e o desembarque de Trouxas e Bruxos (e não mais somente Bruxos). Para simplificar, caso prefira, pode considerar a capacidade do Expresso fixa em 100. Você deve também sobrecarregar o operador de indexação ([]), para que seja possível acessar um assento qualquer. Adapte também a hierarquia Humano de modo a empregar um método polimórfico quemSou, para dizer se o Humano é um Trouxa ou um Bruxo, bem como o seu nome.

Lamento ter que te pedir isto: mas preciso que ajude Snape nessa cínica jornada. Escreva uma main que demonstre claramente a narrativa acima apresentada.

13. **Voldemort** (*Hierarquia Conta*) Um banco trabalha com três tipos de contas: (a) conta corrente comum; (b) conta corrente com limite; e (c) conta poupança. Em todos os casos é necessário guardar o número da conta, o nome do correntista e o saldo.

    Para a conta poupança é necessário guardar o dia do aniversário da conta (quando são creditados os juros).

    Já para a conta com limite é necessário guardar o valor do limite.

    As contas também armazenam uma lista de transações. Uma transação é definida por uma data, valor da transação e descrição. Se o valor for negativo, a transação é considerada um débito (crédito caso contrário).

    As operações possíveis são: depósito, retirada e impressão de extrato. Essas operações devem ser definidas numa classe abstrata denominada Conta.

    A operação de depósito é igual nos três tipos de conta.

    A retirada só é diferente na conta com limite, pois esta admite que o saldo fique negativo até o limite estabelecido.

    Finalmente o extrato é diferente para as três: (a) na conta comum exibe o número da conta, nome do cliente, transações e o saldo; (b) na conta limite imprime também o valor do limite; e (c) na conta poupança imprime também o dia do aniversário.

    Implemente a hierarquia de classes das contas explorando polimorfismo.

    Faça um programa em C++ que permita ao usuário fazer depósitos, retiradas e verificação de extrato nas suas contas a partir do número da conta. Utilize um array (de ponteiros para Conta) para armazenar todos os tipos de contas.