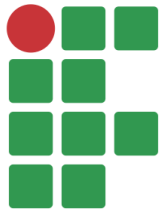
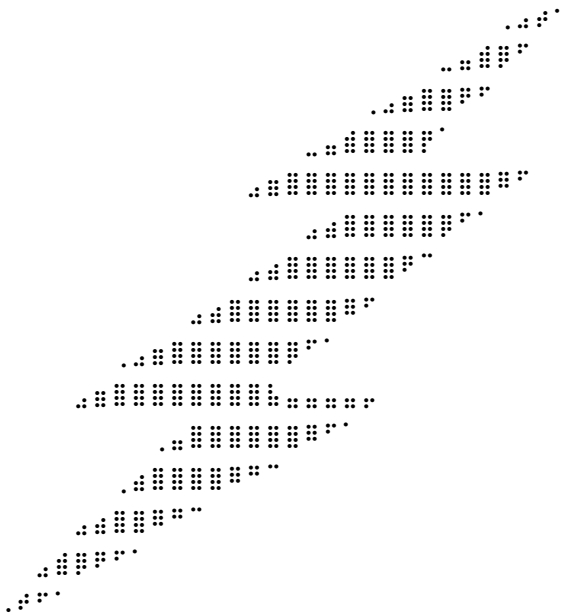


INSTITUTO FEDERAL DA PARAÍBA
CAMPUS CAMPINA GRANDE
BACHARELADO EM ENGENHARIA DA COMPUTAÇÃO
DISCIPLINA DE POO e LAB. POO
PROF. VICTOR ANDRÉ PINHO DE OLIVEIRA

Atividade Unidade II - 1 - Introdução à OO

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** Explique a diferença entre classes e objetos.

Classes são moldes; objetos são instâncias desses moldes.

2. **Quirrell** Considere o objeto Varinha (Harry Potter). Descreva 3 atributos e 3 comportamentos.

Varinha:

Atributos: material, comprimento, núcleo

Comportamentos: lançar feitiço, brilhar, vibrar

3. **Quirrell** Considere o objeto Joystick (periférico). Descreva 3 atributos e 2 comportamentos.

Joystick:

Atributos: número de botões, tipo de conexão, marca

Comportamentos: mover cursor, enviar comando

4. **Quirrell** Qual a unidade básica de programação de C++?

A unidade básica é a classe.

5. **Quirrell** O que é instanciar?

Instanciar é criar um objeto a partir de uma classe.

6. **Quirrell** De que maneira uma Classe oferece serviços para outrem?

Através de seus métodos públicos.

7. **Quirrell** O que é abstração? O que é encapsulamento?

Abstração é focar no essencial; encapsulamento é esconder detalhes internos.

8. **Quirrell** Como que um objeto envia uma mensagem para outro objeto?

Chamando um método público do outro objeto.

9. **Quirrell** Para que serve o especificador de acesso **public**? E o **private**?

public permite acesso externo; private restringe acesso ao escopo da classe.

10. **Quirrell** O que são métodos set? E métodos get?

Set altera atributos; get retorna atributos.

11. **Quirrell** Quando devemos empregar o qualificador const nos métodos?

Quando o método não altera o estado do objeto.

12. **Quirrell** O que é um construtor-padrão?

Construtor-padrão não tem parâmetros.

13. **Quirrell** Quando um construtor é invocado?

É invocado na criação do objeto.

14. **Quirrell** Por que é importante separar a Interface da Implementação? Como podemos fazer isso em C++?

Para manter código limpo e reutilizável; usando arquivos .h para interface e .cpp para implementação.

15. **Olho-Tonto** (Classe *Carro*) Modifique a última versão da **Classe Carro** desenvolvida na "sala" (presente no slide) de modo que:
    a. Inclua um novo membro de dados int que representa o ano do carro;
    b. Forneça uma função set para alterar o ano e uma função get para recuperá-lo;
    c. Modifique o construtor para aceitar o novo parâmetro;
    d. Adicione um método toString que retorna uma string contendo a marca, o modelo e o ano do carro.
16. **Olho-Tonto** (Classe *Empregado*) Crie uma Classe chamada **Empregado** que possui 3 **membros de dados** - um **nome**, um **sobrenome** e um **salário mensal**. Sua Classe deve ter um **construtor** que inicialize os 3 membros de dados. Forneça uma função set e uma função get para cada membro de dados. Se o salário mensal não for positivo, configure-o como 0. Escreva um programa de teste que demonstre as capacidades da classe Empregado. Crie dois objetos Empregado e exiba seu salário mensal. Em seguida, dê um aumento de 10% para cada um dos empregados e exiba novamente o salário mensal.
17. **Olho-Tonto** (Classe *Bruxo - Hogwarts*) Crie uma Classe **Bruxo** que possui 3 **membros de dados** - um **nome**, uma **casa** e um **feitiço predileto**. Sua Classe deve ter um **construtor** que inicialize os 3 membros de dados. Forneça uma função set e uma função get para cada membro de dados. Em especial, o método set para o atributo **casa** deve validar se a casa é uma das 4 casas de Hogwarts (Grifinória, Sonserina, Lufa-lufa ou Corvinal), pois, caso não seja, deverá deixar a casa vazia. Escreva um método chamado **lancarFeitico** que, quando invocado, apresente na

tela a mensagem “Lançando feitiço FEITIÇO”, substituindo FEITIÇO pelo valor do atributo **feitiço predileto**. Adicione um método **display** que apresenta na tela todos os atributos do Bruxo. Escreva um programa de teste que demonstre as capacidades da sua classe.

18. Dolores (Classe *ContaBanco*) Crie uma Classe chamada **ContaBanco** que um banco poderia utilizar para representar contas bancárias dos clientes. Sua classe deve incluir um membro de dados do tipo **double** para representar o **saldo da conta**. Sua Classe deve fornecer um **construtor** que recebe um saldo inicial e o utiliza para inicializar o membro de dados. O construtor deve validar o saldo inicial para assegurar que ele seja maior ou igual a zero. Caso seja menor que zero, o construtor simplesmente deverá setar o saldo para zero. A classe deve fornecer três **funções-membro**. A função-membro *creditar* deve adicionar uma quantia passada como argumento ao saldo atual. A função-membro *debitar* deve retirar a quantia passada como argumento da conta. Se houver tentativa de retirar um valor acima do saldo, então o saldo deverá permanecer inalterado e a função-membro deverá exibir uma mensagem de erro na tela. A função-membro *getSaldo* deve retornar o saldo atual. Escreva um programa que cria dois objetos **ContaBanco** e testa suas funções-membro.

19. Dolores (Classe *Data*) Crie uma Classe Data que inclua 3 partes de informações como atributos - dia (int), mes (int) e ano (int). Sua Classe deve ter um construtor com três parâmetros que os utiliza para inicializar os 3 membros de dados. Para o propósito deste exercício, assuma que os valores fornecidos para o dia e ano estão corretos, mas certifique-se de que o valor do mês esteja no intervalo 1-12; se não estiver, configure o mês como 1. Forneça uma função set e uma função get para cada membro de dados. Forneça também uma função-membro toString que retorna uma string com o dia, o mês e o ano separados por barras (/). Escreva um programa que demonstre as capacidades da sua Classe Data.

20. Dolores (Classe *Cupom*) Crie uma Classe chamada Cupom que uma loja de suprimentos de informática possa utilizar para representar um cupom de um item vendido na loja. Um Cupom deve incluir quatro membros de dados - id (string), descrição (string), quantidade (int) e o preço do item (float). Sua Classe deve ter um construtor que inicializa os 4 membros de dados. Forneça uma função set e uma função get para cada membro de dados. Além disso, forneça uma função-membro chamada de calculaCupom que calcula o valor total da nota e depois retorna esse valor. Se a quantidade não for positiva, ela deve ser configurada como 0. Se o preço do item não for positivo, ele deve ser configurado com 0. Escreva um programa para ilustrar as capacidades de sua Classe Cupom.

21. Voldemort (Classe *Cupom revisitada*) Aperfeiçoe a Classe Cupom da questão anterior de modo que o Cupom possa aceitar não somente um item, mas 20 itens - no máximo. Trate cada item como uma unidade atômica (como um registro - **struct**), em que seus campos sejam exatamente os atributos da Classe anterior. Ao invés de um método set para cada atributo, crie um método addItem para adicionar o item de uma só vez (passando o id, a descrição, a quantidade e o preço do item de uma só vez, como argumentos para o método). Forneça uma função get que receba o id

como argumento e retorne, caso exista, o item específico. Forneça uma função-membro chamada de calculaCupom que calcula o valor total da nota e depois retorna esse valor. Adicione um método chamado de “imprimir nota” que exiba na tela todos os itens presentes na nota, calculando o total por item e, no final, o total da nota. Escreva um programa para ilustrar as capacidades de sua nova Classe Cupom.