Universidade Federal de Viçosa
Campus Rio Paranaíba
Sistemas de Informação
Prof. Guilherme Pena

SIN141
Computação Orientada a Objetos
Lista de Exercícios 2
19/06/2018

Classes (Parte 2)

ATENÇÃO: Todas as respostas e programas serão enviados via **PVANet** e aceitarei apenas arquivos "txt" que conterão todos os programas do roteiro. **NÃO** aceitarei zip, doc, etc; apenas txt.

Exercícios (Entrega **individual** um dia antes da prova: 19/06/2018)

Descritivos (para Revisão apenas)

1) Preencha as seguintes lacunas:

a) Tratar um objeto de classe básica como um _____ pode causar erros.

b) Funções que podem ser sobrescritas são declaradas com a palavra-chave _____ .

c) Se uma classe contém pelo menos uma função virtual pura ela é uma classe _____.

d) O _____ envolve utilizar um ponteiro de classe básica para invocar funções virtual em objetos de classe básica e de classe derivada.

e) Classes _____ não permitem que objetos sejam instanciados.

f) Se uma classe derivada de uma classe _____ não implementar uma função _____ _____, ela continuará sendo uma classe _____ .

g) Uma classe pode ser derivada de mais de uma classe básica através da _____.

h) O uso de _____ representa uma reusabilidade de código em que uma classe pode se tornar genérica e seus trechos de códigos serem utilizados com diferentes _____ .

i) Para definir uma classe genérica deve-se usar a sintaxe _____ .

j) Na parte da implementação dos métodos de uma classe genérica, cada um deles possui a sintaxe _____ emcima do seu nome.

k) Ao usar _____, as classes ***não*** podem separadas em arquivos (.h) e (.cpp).

l) O tratamento de exceções possui basicamente as palavras-chave _____, _____ e _____.

m) A palavra _____ permite definir um bloco de prova.

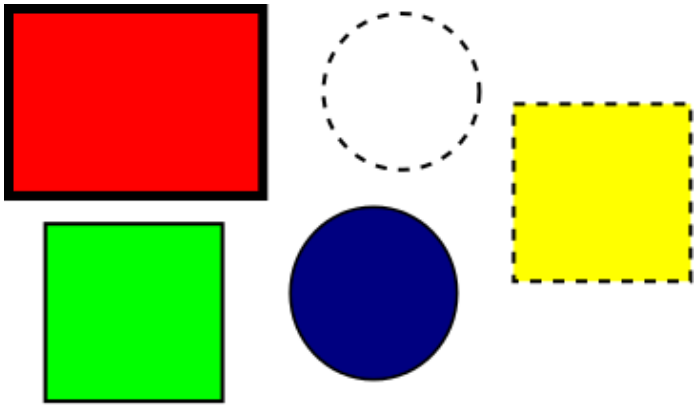
n) Se houver um teste de exceção, essa deverá ser lançada pela palavra _____ no próprio bloco de prova, ou numa função que foi invocada no bloco de prova.

o) Toda exceção lançada será tratada em um bloco _____ referente ao tipo do objeto lançado.

p) Quando uma _____ precisa ser tratada em dois pontos diferentes de um sistema, ela deve ser relançada para um chamador anterior. A isso denominamos _____ .

Implementação (Divida cada projeto em **.h**, **.cpp**, **main** e **makefile**)

1) (Projeto FormaBásica) Na lista de exercícios anteriro, criamos um projeto para programas de manipulação de imagens, em que figuras geométricas, que são desenhadas numa tela, podem ter suas características armazenadas por diferentes tipos de objetos.



Tais figuras geométricas podem manter características como **posição na tela (x, y)**, **cor de fundo**, **tipo de contorno**, **espessura do contorno** além de outras características relacionadas com o tipo da figura (círculo ou retângulo).

Sendo assim, definiu-se uma *hierarquia de herança* contendo uma classe básica **FormaBasica** com os atributos como "**private**" na sua implementação. Dessa forma, objetos como Circulo e Retangulo foram derivados da classe básica conforme o diagrama:

Refaça a implementação da classe mas agora defina e implemente qual o método que poderia ter uma característica ***Polimórfica***? Lembre-se que não basta apenas o método para testar a o polimorfismo.

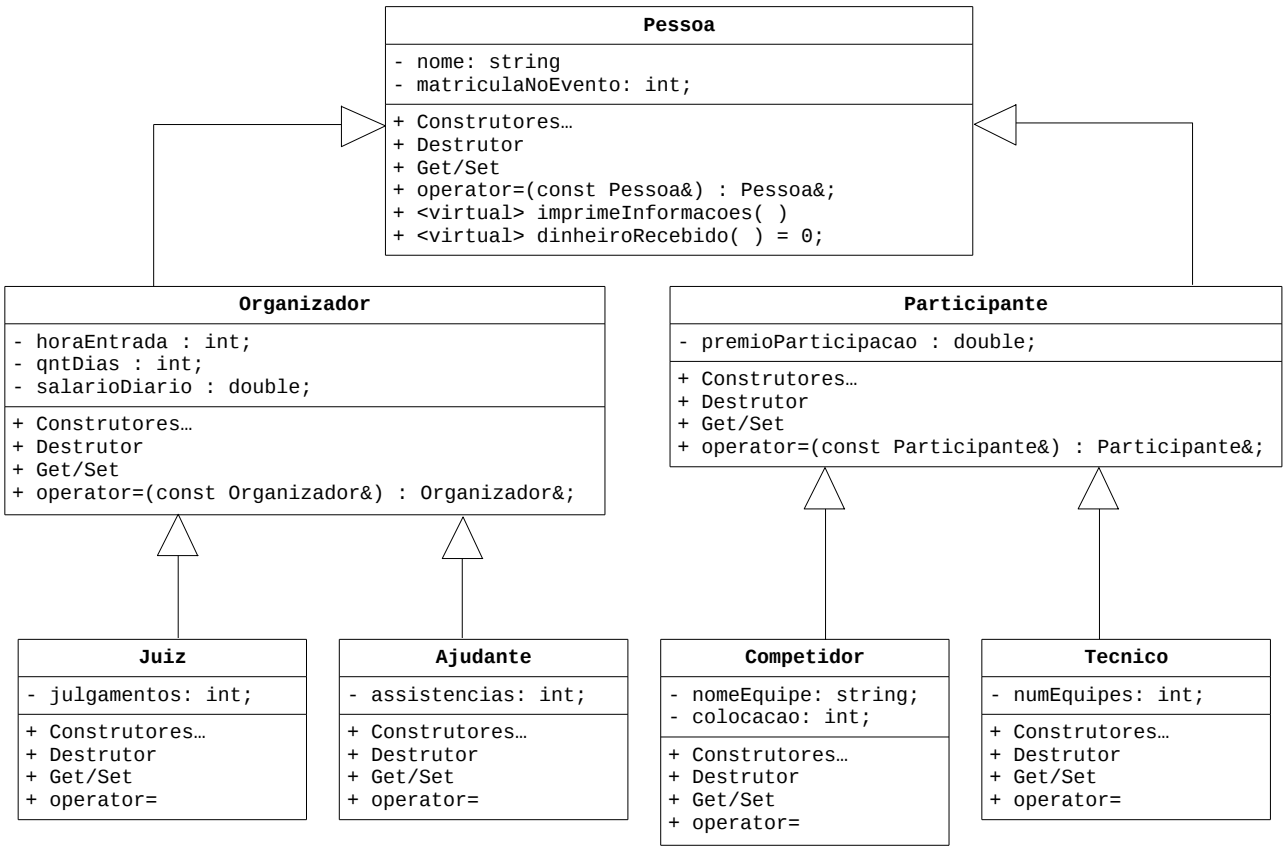
Na mesma idéia, qual método poderia dar uma característica de ***classe abstrata*** para a classe básica? Faça essa implementação também.

Inclua mais duas classes no projeto, uma para representar uma forma ***Triângulo*** e uma para representar uma forma ***Quadrado*** (Note que existe uma relação entre Quadrado e Retângulo).

Implemente uma função ***main*** que possa fazer os testes das modificações que você realizar no projeto.

2) (Projeto Competicao) Implemente um projeto orientado a objetos para organizar as informações de todos os perfis de pessoas que atuam em uma competição como organizadores ou participantes e ao final, poder gerar relatórios contendo as informações pertinentes das pessoas e também calcular o dinheiro ganho na competição.

O projeto deverá ser modelado conforme o esboço do diagrama UML abaixo:



Para o cálculo do dinheiro recebido na competição, seguem algumas instruções:
- **Juiz**: Recebe o número de dias vezes o salário diário + 15 reais por julgamento.
- **Ajudante**: Recebe o número de dias vezes o salário diário + 10 reais por assitência.
- **Competidor**: Recebe 100 reais de participação + 300 reais dividido pela colocação se ficar em primeiro, segundo ou terceiro.
- **Tecnico**: Recebe 100 reais de participação + 50 reais por equipe que lidera.

3) (Array Especial) (Templates não se dividem em .h e .cpp) Implemente uma classe genérica, através de template, que deverá simular um array com algumas características especiais:

ArrayEspecial<T>
- elementos : T*; - tam : int; - max : int; - ativos : bool*;
+ ArrayEspecial(); + ArrayEspecial(const ArrayEspecial<T>&); + ~ArrayEspecial(); + operator=(const ArrayEspecial<T>&) : ArrayEspecial<T>&; + insere(T e, int pos); + remove(int pos); + imprimeElementos();

- O armazenamento dos elementos será de forma dinâmica;

- O array possuirá um vetor auxiliar de *booleanos* para indicar quais elementos estão ativos ou não.

- O atributo tam guardará o número de elementos ativos.

- O atributo max representa o tamanho máximo do conjunto.

- O construtor-padrão deve alocar um espaço para 10 elementos apenas.

- O construtor de cópia e operador de atribuição entre objetos deve fazer a cópia correta das informações.

- O método insere recebe o elemento a inserir e uma posição, se a posição já possuir um elemento (estar ativa), a inserção não é feita.

- O método remove recebe uma posição para remover um elemento, e a operação deve ser concretizada apenas se tinha um elemento ativo nessa posição.

- O método imprime, mostra na tela apenas os elementos ativos.

Crie uma função main que seja capaz de testar cada uma das funcionalidades da sua classe, e teste-a com diferentes tipos.

4) (Tratamento de Exceção) Crie classes onde os objetos deverão representar exceções a serem usadas na estrutura do exercício anterior.

- Crie uma exceção para o método insere caso a posição esteja ocupada.
- Crie uma exceção para o método insere caso a posição de inserção seja inválida.
- Crie uma exceção para o método remove caso a posição de remoção seja inválida.
- Crie uma exceção para o método remove caso a posição já estivesse liberada.

Faça os lançamentos corretos (*throw*) em caso de “erros” que podem ocorrer nessas operações.

Modifique a função main anterior para fazer o tratamento correto das exceções lançadas (*try, catch*) e implemente na main instruções que forçarão as quatro exceções a serem lançadas.