CMP1056 - Técnicas de Programação II
Lista de Exercícios
Max Gontijo de Oliveira

- Todas as classes deverão ser testadas em um programa principal (`main`).

1. **[Construtores, métodos, autorreferência, sobrecarga de operador e exceções]** Crie uma classe chamada `MatrixInt`. Essa classe deverá ter os seguintes atributos privados:

   - O ponteiro para uma matriz de inteiros (`int **`)
   - Um inteiro que indica a quantidade de linhas da matriz (`int`)
   - Um inteiro que indica a quantidade de colunas da matriz (`int`)

   Crie métodos *get* APENAS para os atributos que indicam a quantidade de linhas e colunas.

   (a) **[Construtor]** Crie, na classe `MatrixInt`, um construtor que receba a quantidade de linhas e de colunas (dois inteiros) e inicialize os atributos respectivos com esses valores. O construtor deverá ainda criar uma matriz de inteiros dinamicamente com as dimensões recebidas por parâmetro e armazenar o ponteiro dessa matriz no respectivo atributo da classe. Todos os elementos da matriz criada deverão ser zerados. Caso a quantidade de linhas ou a quantidade de colunas seja zero ou negativo, o construtor deverá lançar uma exceção com uma mensagem e um código de erro (dica, crie uma classe para ser sua exceção).

   (b) Crie, na classe `MatrixInt`, um método chamado `getValor` que receba, por parâmetro, linha e coluna e retorne o valor (também inteiro) na respectiva posição da matriz. Caso a posição não exista na matriz, o método deverá lançar uma exceção com uma mensagem e um código de erro.

   (c) Crie, na classe `MatrixInt`, um método chamado `setValor` que receba por parâmetro, linha, coluna e um valor (todos inteiros) e atribua o valor passado por parâmetro à respectiva posição na matriz. Caso a posição não exista na matriz, o método deverá lançar uma exceção com uma mensagem e um código de erro.

   (d) **[Método]** Crie, na classe `MatrixInt`, um método chamado `imprimir` que seja capaz de imprimir os valores da matriz na tela, formatada de modo que,0 cada linha da matriz seja exibida em uma linha diferente na tela.

   (e) **[Sobrecarga de operador]**Crie uma sobrecarga do operador + na classe `MatrixInt`, de modo que o compilador consiga realizar a operação $C = A + B$, onde A, B e C são variáveis do tipo `MatrixInt`. Note que a operação deverá utilizar duas matrizes e criar uma nova matriz com a soma matricial das outras duas. Caso a soma não seja possível (matrizes com dimensões diferentes), deve ser lançada uma exceção com uma mensagem e um código de erro.

2. **[Herança]** Um vetor de tamanho N nada mais é que uma matriz com N linhas e apenas uma coluna. Assim, aproveite essa característica e crie uma classe chamada `Vector` que herde da classe `MatrixInt` implementada na questão 1. Nenhum atributo adicional será necessário.

   (a) **[Chamada a construtor da superclasse]** Crie, na classe `Vector`, um construtor que receba o tamanho do vetor. Note que será necessária uma chamada explícita ao construtor da superclasse. No caso, passe para esse construtor da superclasse `MatrixInt`, o tamanho do vetor como sendo a quantidade de linhas e o valor 1 (um) como a quantidade de colunas (uma vez que se trata de um vetor).

   (b) **[Chamada a método da superclasse]** Como a classe `Vector` herda de `MatrixInt`, ela possui métodos que, embora funcionem, não fazem sentido em um vetor. Assim, crie novos métodos `getValor` e `setValor` passando apenas um índice. Na implementação desses métodos, faça uso dos métodos `getValor` e `setValor` da superclasse, já que os atributos da superclasse são privados e, portanto, não podem ser acessados nas subclasses. Além disso, crie também um método chamado `getTamanho`, que irá retornar o tamanho do vetor. Note que o tamanho do vetor é simplesmente a quantidade de linhas dessa matriz de uma coluna.

3. **[Template]** Crie uma classe chamada `Matrix` que tenha os mesmos atributos privados que a classe `MatrixInt` implementada na questão 1 e que tenha também os mesmos métodos (a sobrecarga de operador não conta). A única diferença é que essa nova classe não irá armazenar uma matriz somente de inteiros, mas uma matriz de de qualquer tipo. Para isso, essa nova classe deverá utilizar template. Portanto, crie essa nova classe fazendo as adaptações necessárias para que a classe `Matrix` possa armazenar qualquer tipo de dado. Dica: no construtor, não será necessário inicializar os valores da matriz com 0, uma vez que a matriz pode nem ser de algum tipo numérico primitivo.

4. **[Sobrecarga de métodos e polimorfismo]** Crie uma classe chamada `Retangulo`. Essa classe deverá ter apenas dois atributos internos protegidos (`protected`): altura (quantidade de linhas na tela) e largura (quantidade de colunas na tela) do retângulo.

   (a) Crie um construtor padrão na classe `Retangulo` que inicialize o tamanho do desenho com altura igual a 5 e largura igual a 5.

   (b) Crie métodos *get*s e *set*'s para viabilizar acesso aos atributos da classe `Retangulo`.

   (c) Crie na classe `Retangulo`, um método público abstrato (`virtual`) sem parâmetros que se chamará `desenhar`.

(d) Crie uma classe chamada `RetanguloCheio` que herde de `Retangulo`. Essa classe deverá sobrescrever o método `desenhar` de modo que o método imprima na tela um retângulo cuja borda seja formada pelo caractere `*` (asterisco) e seja internamente preenchido com o caractere `@` (arroba). Esse retângulo terá o tamanho determinado pelos atributos respectivos da superclasse.

(e) Crie uma classe chamada `RetanguloVazio` que herde de `Retangulo`. Essa classe deverá sobrescrever o método `desenhar` de modo que o método imprima na tela um retângulo cuja borda seja formada pelo caractere `*` (asterisco) e não seja internamente preenchido (tecnicamente, será internamente preenchido por espaços em branco). Esse retângulo terá o tamanho determinado pelos atributos respectivos da superclasse.

(f) Crie um programa que realize os seguintes passos:

1. Programa cria uma vetor de ponteiros para `Retangulo` com tamanho 5.
2. Para cada posição nesse vetor:
   (a) Programa lê do teclado um número inteiro OP entre 1 e 2.
   (b) Se $OP = 1$, então o programa instancia dinamicamente um `RetanguloCheio` e armazena seu endereço no vetor de ponteiros para `Retangulo`.
   (c) Se $OP = 2$, então o programa instancia dinamicamente um `RetanguloVazio` e armazena seu endereço no vetor de ponteiros para `Retangulo`.
   (d) Programa lê do teclado dois números inteiros e configura altura e largura do retângulo instanciado por meio dos métodos *set*'s.
3. Após criar os 5 retângulos, programa chama o método `desenhar` de todos eles, exibindo os retângulos criados.

5. **[Membros estáticos]** Crie uma classe chamada `Funcionario` que tenha como atributos privados, o cpf (`long`), o nome (`string`) e o salário base (`double`).

   Crie um método na classe `Funcionario` que se chame `calcularSalarioLiquido`, que não tenha nenhum parâmetro e retorne o cálculo do salário líquido do funcionário.

   Esse cálculo deve ser realizado descontando do salário base os tributos de acordo com a seguinte tabela de tributos resumida:

Nº Faixa	**Faixa do salário base**	**Porcentagem de tributos sobre o salário base**
1	Até R$ 800,00	10.0%
2	Acima de R$ 800,00 e até R$ 1.500,00	13.5%
3	Acima de R$ 1.500,00 e até R$ 3.150,00	16.0%
4	Acima de R$ 3.150,00 e até R$ 5.000,00	20.0%
5	Acima de R$ 5.000,00	23.7%

   Para efetuar o cálculo, crie atributos estáticos dentro da classe que armazenem os percentuais dessas faixas.

   Crie também, um método que seja capaz de alterar os valores percentuais de cada uma das faixas.

   Observação: Note que não está sendo solicitado que as faixas sejam alteráveis. Apenas os percentuais.