Universidade Federal de São Carlos

ORIENTAÇÃO A OBJETOS: CLASSES

Disciplina: Programação Orientada a Objetos

Data de entrega: até 01 de outubro de 2019.

Professor: Delano Medeiros Beder

1 Enunciado

A atividade T1 consiste em implementar em C++ as classes conforme descritas abaixo. Os atributos das classes devem ser **privados**. Organize suas classes no namespace **poo**.

1. [1,50] Defina a classe Pessoa cujos objetos representam pessoas. Cada objeto dessa classe deve guardar os seguintes dados de pessoa: nome e CPF. Escreva os seguintes métodos/construtores para esta classe:

Nome	Descrição
Pessoa(string, int)	Construtor capaz de setar os atributos do objeto. Esse construtor deve ser único.
~Pessoa()	Destrutor da classe.
string getNome()	Método responsável por retornar o nome da pessoa.
int getCPF()	Método responsável por retornar o número CPF da pessoa.
void imprime()	Método que imprime as informações (nome, CPF) de uma pessoa.

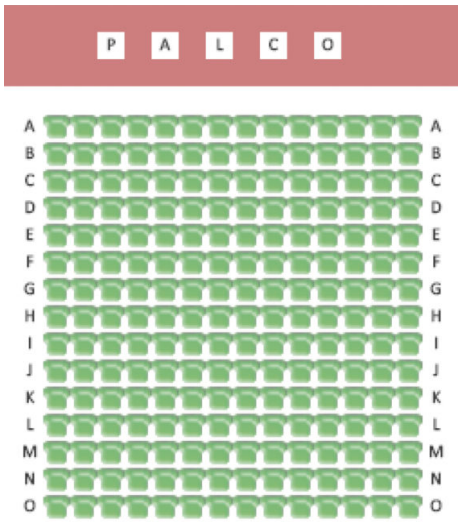
2. [2,50] Defina a classe Aluno (subclasse da classe Pessoa) cujos objetos representam alunos matriculados em uma disciplina. Cada objeto dessa classe deve guardar os seguintes dados do aluno: RA, nota da 1ª prova, nota da 2ª prova, nota do 1º trabalho, nota do 2º trabalho e nota do 3º trabalho. Escreva os seguintes métodos/construtores para esta classe:

Nome	Descrição
Aluno(string, int, int, double, double, double, double, double)	Construtor capaz de setar os atributos do objeto. Esse construtor deve ser único.
~Aluno()	Destrutor da classe.
int getRA()	Método responsável por retornar o RA do aluno.
void imprime()	Método que imprime as informações (nome, CPF, média final) de um aluno.
double media()	Método responsável por calcular a média final (MF) do aluno $MF = \frac{MP \times 8 + MT \times 2}{10}, MP = \frac{P1+P2}{2}, MT = \frac{T1+2\times T2+3\times T3}{6}$
bool aprovado()	Método responsável por retornar verdadeiro se o aluno foi aprovado ($MF \geq 6.0$) e falso, caso contrário.
bool sac()	Método responsável por retornar verdadeiro se o aluno ficou em SAC – Sistema de Avaliação Complementar ($5.0 \leq MF < 6.0$) e falso, caso contrário.
double notaSAC()	Método responsável por calcular qual a nota mínima necessária, na prova de avaliação complementar (SAC), para aprovação na disciplina. Para o aluno ser aprovado após a prova de avaliação complementar (SAC) precisa atender a seguinte regra: $\frac{SAC+MF}{2} \geq 6.0$. Caso o aluno não ficou em SAC, retornar 0.

3. [2,50] Defina a classe DataHorario com os dados: dia, mês, ano, hora, minuto e segundo. A classe deverá dispor dos seguintes métodos/construtores:

Nome	Descrição
DataHorario(int, int, int, int, int, int)	Construtor capaz de setar os atributos. Este construtor verifica se a data/horário estão corretos, caso não esteja, a data/horário é configurada como 01/01/0001 00:00:00. Esse construtor deve ser único.
~DataHorario()	Destrutor da classe.
int compara(DataHorario&)	Método que recebe como parâmetro um outro objeto da Classe DataHorario, compara com a data/horário corrente e retorna: **0** se os valores forem iguais; **1** se a data/horário corrente for maior que a do parâmetro; **-1** se a data/horário do parâmetro for maior que a corrente.
int getDia()	Método responsável por retornar o dia da data.
int getMes()	Método responsável por retornar o mês da data.
int getAno()	Método responsável por retornar o ano da data.
int getHora()	Método responsável por retornar a hora do horário.
int getMinuto()	Método responsável por retornar o minuto do horário.
int getSegundo()	Método responsável por retornar o segundo do horário.
void imprime(bool)	Método responsável pela impressão das informações de uma data/horário. Formato DD/MM/YYY hh:mm:ss [AM/PM]. O paramêtro booleano indica se deve-se utilizar o sistema horário de 12 horas (AM/PM). Exemplo: 31/08/2019 15:57:10 ou 31/08/2019 3:57:10 PM.
void imprimePorExtenso()	Método responsável pela impressão das informações de uma data/horário por extenso. Exemplo: 31 de Agosto de 2019 – 15 horas, 57 minutos e 10 segundos.

4. [3,50] Defina a classe Sessao cujos objetos representam sessões de um determinado teatro que acontecem em determinada data/horário. O teatro possui no máximo 210 poltronas (distribuídas em 15 fileiras de 14 poltronas), e a classe permite controlar a ocupação das poltronas. Cada objeto dessa classe deve guardar os seguintes dados (dentre outros) da sessão de teatro: nome da peça, data/horário. A classe deve ter os seguintes construtores/métodos:



Nome	Descrição
Sessao(string,DataHorario&)	Construtor capaz de setar os atributos: nome da peça e data/horário (instância da classe DataHorario definida na questão anterior).
~Sessao()	Destrutor da classe.
string proximoLivre()	Método responsável por retornar o número da próxima poltrona livre (formato [A-O][1-14]). Retorna "cheio" se não houver poltrona disponível na sessão de teatro.
bool verifica(string)	Método responsável por verificar se a poltrona recebida (formato [A-O][1-14]) como parâmetro está ocupada.
bool ocupa(string, Pessoa&)	Método responsável por ocupar determinada poltrona da sessão de teatro, cujo número da poltrona é recebido (formato [A-O][1-14]) como parâmetro, e retornar verdadeiro se a poltrona não estiver ocupada (operação foi bem sucedida) e falso caso contrário. O segundo parâmetro representa um expectador (instância da classe Pessoa definida anteriormente) que "comprou" um ingresso da sessão de teatro.
bool desocupa(string)	Método responsável por desocupar determinada poltrona da sessão de teatro, cujo número da poltrona é recebido como parâmetro, e retornar verdadeiro se a poltrona estiver ocupada (operação foi bem sucedida) e falso caso contrário.
int vagas()	Método responsável por retornar o número de poltronas vagas disponíveis (não ocupadas) na sessão de teatro.
void imprime()	Método responsável pela impressão das informações de uma sessão de teatro (nome da peça, data/horário e lista dos expectadores com suas respectivas poltronas).

2 Observações importantes

2.1 Sobre a elaboração e entrega:

- Este exercício-programa deve ser elaborado individualmente.
- Você deve utilizar **apenas** os conceitos apresentados em aula.
- Você deve implementar as classes em C++.
- Organize suas classes no namespace **poo**.
- Compacte o código-fonte das classes em C++ (e se possível, o projeto Netbeans ou Code-Blocks) e entregue somente este arquivo no ambiente moodle. O arquivo <RA>.zip deve conter o código-fonte das classes em C++.

  Exemplo: 1234567.zip (cuidado para não enviar arquivos errados!)
- O prazo de entrega é o dia 01 de outubro de 2019 às 23h55.
- A entrega será feita unicamente pelo ambiente moodle (`https://ava.ead.ufscar.br`). Não serão aceitos trabalhos enviados por email.
- Guarde uma cópia dos arquivos entregues.

2.2 Sobre a avaliação:

- Não serão toleradas cópias! Exercícios copiados (com ou sem eventuais disfarces) receberão nota ZERO. O exercício do aluno alvo da cópia também receberá nota ZERO.
- Exercícios com erros de sintaxe (ou seja, erros de compilação) receberão nota ZERO.
- Os exercícios serão avaliados segundo os seguintes critérios:
  - Soma simples dos valores obtidos nos itens de 1 a 2
    1. Atendimento às normas de boas práticas de programação (comentários, endentação, nomes de variáveis, estruturação do código, etc) [0..20]
    2. Corretude na implementação da atividade [0..80]