Universidade Federal de São Carlos

ORIENTAÇÃO A OBJETOS: CLASSES

Disciplina: Programação Orientada a Objetos

Data de entrega: até 27 de Janeiro de 2023.

Professor: Delano Medeiros Beder

1 Enunciado

A atividade T1 consiste em implementar em C++ as classes conforme descritas abaixo:

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

Nome	Descrição
Pessoa(string, string)	Construtor capaz de setar os atributos do objeto. Esse construtor deve ser único.
string getCPF()	Método responsável por retornar o CPF da pessoa.
string getNome()	Método responsável por retornar o nome da pessoa.
void imprime()	Método responsável pela impressão das informações (CPF e nome) 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 possuir os seguintes atributos: RA, nota da 1^a prova, nota da 2^a prova e nota do trabalho. Os atributos da classe devem ser **privados**. Escreva os seguintes métodos/construtores para esta classe:

Nome	Descrição
Aluno(string, string, int, double, double, double)	Construtor capaz de setar os atributos do objeto. Esse construtor deve ser único.
int getRA()	Método responsável por retornar o RA do aluno.
double media()	Método responsável por calcular a média final (MF) do aluno – cada prova tem peso 7 e o trabalho tem peso 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.
void imprime()	Método responsável pela impressão das informações (CPF, nome, RA e média final) de um aluno.

3. [2,00] 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: 31/12/2022 15:57:10 ou 31/12/2022 3:57:10 PM.
void imprimePorExtenso()	Método responsável pela impressão das informações de uma data/horário por extenso. Exemplo: 31 de Dezembro de 2022 – 15 horas, 57 minutos e 10 segundos.

4. [4,00] Defina a classe Voo cujos objeto representam vôos que acontecem em determinada data e horário. Cada objeto dessa classe deve guardar os seguintes dados do vôo: número e data/horário. Cada vôo possui no máximo 100 passageiros, e a classe permite controlar a ocupação das vagas. Os atributos da classe devem ser **privados**. A classe deve ter os seguintes construtores/métodos:

Nome	Descrição
Voo(int, DataHorario&)	Construtor capaz de setar os atributos: número do vôo, e data/horário (instância da classe DataHorario definida na questão anterior).
int proximoLivre()	Método responsável por retornar o número da próxima poltrona livre. Retorna zero se não houver poltrona disponível no vôo.
bool verifica(int)	Método responsável por verificar se o número da poltrona recebido como parâmetro está ocupada.
bool ocupa(int, Pessoa&)	Método responsável por ocupar determinada poltrona do vôo, cujo número é recebido como parâmetro, e retornar verdadeiro se a poltrona não estiver ocupada (operação foi bem sucedida) e falso caso contrário. O segundo parâmetro representa um passageiro (instância da classe Pessoa definida anteriormente) que "comprou" uma passagem para o vôo.
bool desocupa(int)	Método responsável por desocupar determinada poltrona do vôo, cujo número 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) no vôo.
void imprime()	Método responsável pela impressão das informações de um vôo (número, data/horário, quantidade de vagas e informações dos passageiros).
Métodos getters e setters	Métodos responsáveis por ler e alterar cada um dos atributos em separado.

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++.
- Compacte o código-fonte das classes em C++ (e se possível, o projeto CodeBlocks, VisualCode, Makefile etc) 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 27 de Janeiro de 2023 às 23h55.
- A entrega será feita unicamente pelo ambiente moodle (`https://ava2.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/atributos, estruturação do código, etc) [0..20]
    2. Corretude na implementação da atividade [0..80]