Data: 08/05/2018 – 1º Semestre de 2017.
Prof. Dr. Jean M. Simão - DAINF/UTFPR Curitiba-Central.

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ - Campus de Curitiba Central – **Dep. Acadêmico de Informática** (DAINF). Disciplina: **Técnicas de Programação – CSE20**.
Prova sobre linguagem C++ / Diagrama de Classes (*UML*) / Orientação a Objetos - 1ª Parcial

Nome do Aluno: _____ **Turma**: S71
Curso: _____ Horário de Começo: _____ **Horário de Fim**: _____

Leia toda a prova, pois os enunciados estão completados uns nos outros.
Utilize os bons princípios de projeto e programação orientada a objetos.
Em tempo, questões de mesmo peso, idem para os tópicos dentro delas (quando houver).

(Questão - 1) (a) Em um programa C++ (*console*) para futebol de robôs simulados, crie uma classe abstrata *Robo* com um método virtual puro chamado *mover*. **(b)** Ainda, esta classe terá um atributo protegido *float* chamado *id* com seu respectivo *get* e *set*, bem como outro atributo protegido, estático e inteiro chamado *quantidade* com seu respectivo *get* estático. **(c)** Essa classe também permitirá duas classes derivadas dela: *Goleiro* e *Atacante*. Ainda, essas duas classes terão cada qual um atributo privado, respectivamente booleano *titular* e inteiro *n_gois*. **(d)** Por fim, o método redefinido em classe derivada, a partir da classe base, irá apenas permitir informar a suposta funcionalidade e dados de instância via combinação de texto e valores de atributos (e.g. '*Robô id' 2.0 ', número de gois' 0, 'atacando!* ').

(Questão - 2) (a) No contexto do programa C++ em questão, crie uma classe chamada *Fila*, usando encadeamento simples, para endereços de objetos que possam ser apontados como *Robo*. **(b)** Para tal, crie e utilize classe auxiliar aninhada chamada *Elemento*. Em suma, cada endereço apontado na fila será tratado por um objeto de *Elemento*, tanto quanto cada *Elemento* tratará o encadeamento pertinente. **(c)** Essa classe *Fila* deve ter método de inclusão, além de construtora e destrutora. **(d)** Por fim, essa classe Fila deve ter um método de percorrimento que começa pelo 1º elemento e vai até último, o qual chama polimorficamente o método *mover* de cada objeto apontado enquanto *Robo*.
Obs.: *Alternativamente a abordagem de encadeamento, desde que com as mesmas funcionalidades, pode-se compor a classe Fila usando: (1) o componente queue da STL, mas valendo 3/4 da questão. (2) outro componente da STL, mas valendo 2/4 da questão; (3) alocação e deslocação dinâmica de vetor, mas valendo 2/4 da questão; ou (4) vetor usual de tamanho fixo, mas valendo 1/4 da questão.*

(Questão - 3) (a) Crie uma classe *Time* com um atributo string *nome,* um atributo inteiro *id* e um método *jogar*, bem como instancie nela (enquanto atributos) pelos menos um objeto de *Goleiro* e um de *Atacante*, chamando explicitamente alguma construtora de cada um desses objetos **(b)** Ainda, a classe *Time* permitirá apontamentos para qualquer instância de derivados de *Robo* por meio de um objeto da classe *Fila*. **(c)** Por fim, inclua o endereço de cada um dos objetos de *Goleiro* e *Atacante* nesse objeto de *Fila* de apontamentos de *Robo*, fazendo o uso de *cast* apropriado. **(d)** Por fim, use o objeto de *Fila* para implementar o método *jogar*, sendo que este apenas chama o método de percorrimento da fila.

(Questão - 4) (a) Crie uma classe *Jogo* sendo que a instância ou o objeto dela (na função *main*) se chamará *Robocup*. Em *Jogo* haverá como atributos duas instâncias da classe *Time*, chamadas de *TA* e *TB* e dois métodos chamados de *nomesTimes* e *executar*. **(b)** No método *executar*, faça um laço de repetição que varia de 1 a 10, sendo que em cada passagem ora um time deve jogar, ora outro. **(c)** Ainda, o operador de fluxo << será sobrecarregado no tocante a classe *Jogo* para permitir informar o nome de cada um dos dois times, sendo testado com o *Robocup*. **(d)** Ainda, nesta sobrecarga, também se informará a quantidade de robôs via chamada estática de *getQuantidade* da classe *Robo*.

(Questão - 5) (a) Primeiramente, antes das demais questões, elabore um diagrama em *UML* das classes solicitadas nelas e de seus relacionamentos. **(b)** Ainda, tanto em *UML* quanto em C++, para cada classe solicitada supõem-se que haverá construtora(s), destrutora e gets/sets apropriados sem necessidade de fazer todos durante esta prova. Entretanto, respeitando as questões anteriores, ao menos fazer constar os exemplos explicitamente solicitados nelas. Por fim, pode-se ter atributos e métodos complementares nas classes caso necessário.

A interpretação faz parte do conteúdo da prova! Inclua comentários (se for o caso) para deixar explicitas suas decisões.