UNIVERSIDADE FEDERAL DO CEARÁ
Campus de Quixadá
Prof. Thiago Werlley
QXD0008 e QXD0009- Programação Orientada Objeto

Trabalho
2022.2

TRABALHO PRÁTICO II

Instruções

A implementação deverá obrigatoriamente ser feita em **C++**.

1. Em **C++** todas as classes devem ser implementadas com separação da interface **.h** e da implementação **.cc**. Deverá ser criado um **Makefile** para compilar o projeto.
   a. Opcionalmente, podem ser criados os subdiretórios **include**, **obj**, **lib** e **src**
   b. Em **C++** não há um tipo Date **Date** nativo. Você poderá criar um tipo ou uma classe **Date** ou fazer uma adaptação usando os tipos disponíveis.
   c. Implemente os destrutores conforme a necessidade de cada classe.

ATENÇÃO

A detecção de cópia de parte ou de todo código-fonte, de qualquer origem, implicará reprovação direta no trabalho. Partes do código cujas ideias foram desenvolvidas em colaboração com outro(s) aluno(s) devem ser devidamente documentadas em comentários no referido trecho. O que **NÃO** autoriza a cópia de trechos de código. Portanto, compartilhem ideias, soluções, modos de resolver o problema, mas **não o código**. Qualquer dúvida entrem em contato com o professor.

Entrega

- A entrega do código--fonte será feita pelo Moodle até o dia **28 de dezembro de 2022 até as 23:30.**
- Também deverá ser entregue um relatório (impresso e em mãos) sobre o trabalho contendo:
  * Os itens especificados na parte 2 deste enunciado;
  * Relato das dificuldades encontradas durante a realização do trabalho e soluções encontradas;
  * Referências de sites e outros materiais utilizados para confecção de trabalhos, incluindo consultas a colegas (especificar quais).

Avaliação

- Funcionamento adequado do programa
  * Compilação (códigos que não compilem serão zerados, e warnings diminuirão a nota);
  * Corretude.
- Atendimento ao enunciado do trabalho;
- Comentários;
- Indentação do código;
- Adequação da estrutura do programa (algoritmos e contêineres da STL), em relação ao problema tratado;
- Relatório.

Enunciado

Neste semestre é pedido aos alunos de POO que implementem um sistema de gerenciamento de coleções de mídia, baseado na *Standard Templates Library* (STL). Para solução do problema, deve ser escolhido o contêiner da STL mais adequado ao contexto e utilizá-lo em toda sua funcionalidade, incluindo os métodos que implementam algoritmos de manipulação de dados. Não basta apenas armazenar dados em um contêiner, retirar os dados do mesmo e depois resolver o problema. Analogamente, não é suficiente implementar algoritmos que já estão prontos na STL ou substituir funcionalidades da STL.

Campeonato de futebol: jogos, times e jogadores

Tarefa

Implementar uma estrutura de classes que permita armazenar e manipular dados de jogadores, times e jogos. Essa estrutura será utilizada para montar um protótipo de jogo de simulação estilo "Football Manager"', no qual a pessoa que está jogando assume o papel de um técnico, escalando o time e um software simula os jogos entre times.

Existem vários jogos derivados do "Football Manager" como o Elifoot, Brasfoot e Cham- pionship Manager. Todos com o mesmo estilo.

Nessa parte 1 você deve se preocupar **APENAS** com a implementação orientada a objetos que envolve cada entidade do problema. Um exemplo mínimo é fornecido na Figura, que inclui as classes (que devem ser obrigatoriamente em quantidade e nome iguais ao do diagrama), bem como seus atributos e métodos. Você está livre para criar mais métodos, atributos e construtores. Mas cuidado para não exagerar.

Repare que o diagrama indica a relação entre as classes, entre as quais temos herança (é-um) e uso (dependência).

Após implementar as classes você deve criar um programa principal (main() que instancie dois objetos "Time" contendo cinco objetos "Jogador" cada (entre os quais: um goleiro, dois defensores e dois atacantes), e também um objeto "Partida" que será com- posto de dois times.

Requisitos

Classe **Jogador** e derivadas:

* **Habilidade**: o método **getHabilidade**() irá retornar um número inteiro de 0 a 100 com a habilidade daquele jogador. Um objeto "Jogador" tem sua **habilidade** dada pelo atributo protegido **habilidade**. No entanto, cada subclasse deverá sobrescrever o método **getHabilidade**() de forma a calcular a habilidade de acordo com seus próprios atributos. O cálculo da habilidade deverá ser feito da seguinte forma:
  - Goleiro: ((habilidade*5) + (((int)(altura*100))*2) + (reflexos*3))/10
  - Defensor: ((habilidade*5) + (cobertura*3) + (desarme*2))/10
  - Atacante: ((habilidade*5) + (velocidade*2) + (tecnica*3))/10
* **Gols**: o método **somaGol**() deverá somar um gol ao registro de gols do jogador, que é inicializado com 0 e vai aumentando conforme o jogador faz um gol em uma partida.

Classe **Time**:

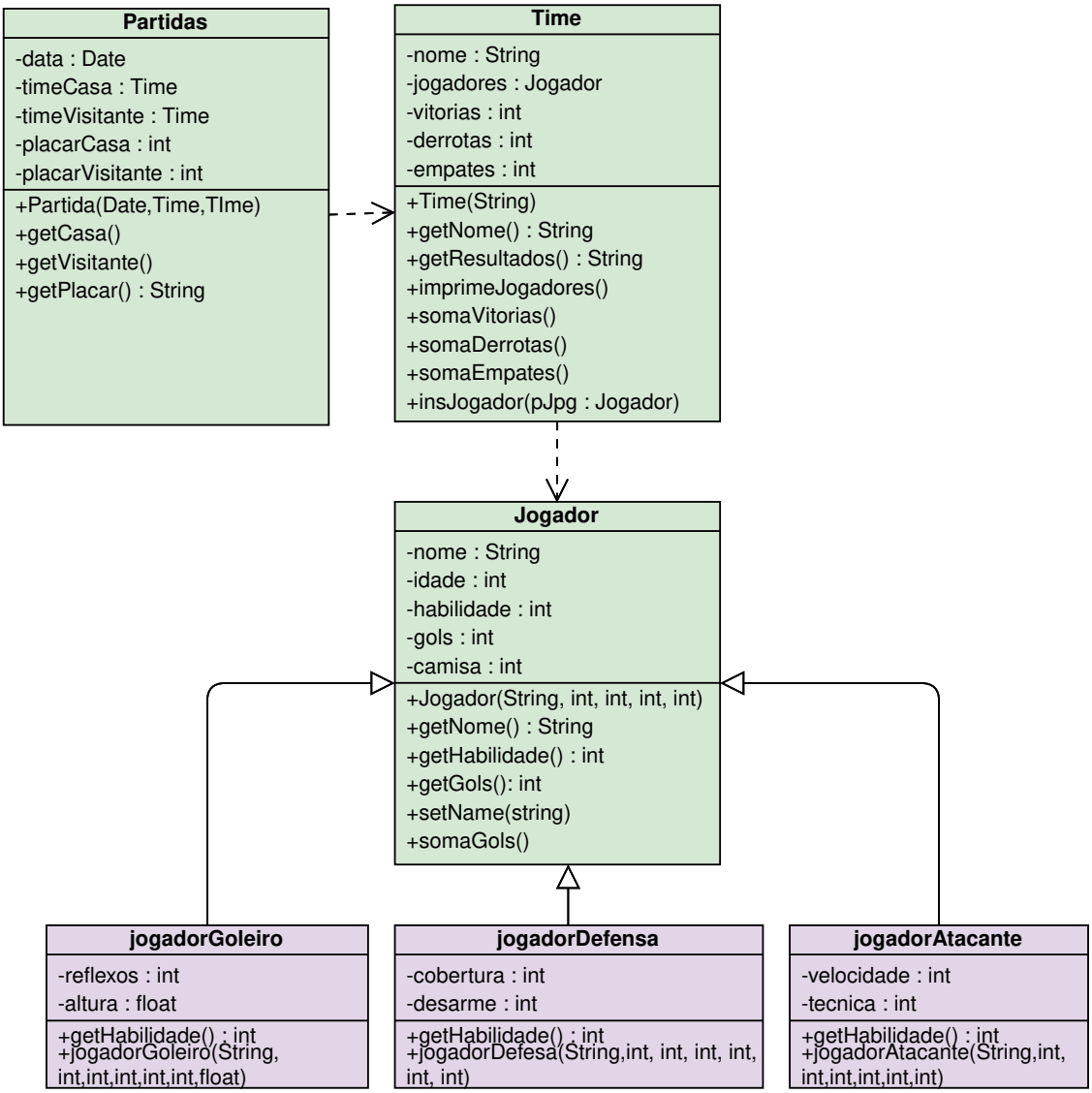
* O construtor de Time recebe apenas o nome do time. Os jogadores são inseridos posteriormente. Nessa primeira implementação não há remoção de jogadores.
* O número de vitórias, derrotas e empates é inicializado com 0 e aumenta conforme o resultado dos jogos.
* O método getResultados retorna uma string com os resultados de um time, no formato: Vitorias: 0, Empates: 0, Derrotas: 0.

Classe **Partida**:

* O construtor recebe uma data e dois times.
* Os métodos golCasa() e golVisitante() adicionam gols aos atributos placarCasa e placarvisitante, respectivamente.
* O método getPlacar() retorna uma string com o resultado do jogo, no formato: Nome do Time da Casa 0×0 Nome do Time Visitante.

Parte 1 – Implementação

Realize a implementação do sistema de campeonato de futebol conforme o diagrama UML abaixo, em que os jogadores são «*actor*» e isso indica que algum contêiner da STL deverá ser utilizado.



OBRIGATÓRIO : para esta prática crie um **CRUD**

a) Dado o nome do jogador, exibir o nome de todos os jogadores do time, ordenados pelas classes derivadas;

b) Dado o nome do jogador, exibir o número de vitorias, derrotas e empates;

c) Dado o nome do jogador, exibir o numero de gols realizados por partidas

d) Dado a data da partida, exibir todas os times que jogaram naquele dia;

e) Dados o nome do Time, exibir se o time é da casa ou visitante;

f) Dados o nome do time, exibir o placar;

g) Dados o nome do jogador, exibir todos os seus atributos, inclusive o nome do time e as partidas jogadas;