SCC0204 (Programação Orientada a Objetos) — Prof. Moacir P. Ponti Junior

Trabalho Prático — Parte 1

Implemente sua atividade sozinho sem compartilhar, olhar código de seus colegas, ou buscar na Internet. Procure usar apenas os conceitos já vistos nas aulas.

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 Championship 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 composto 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`

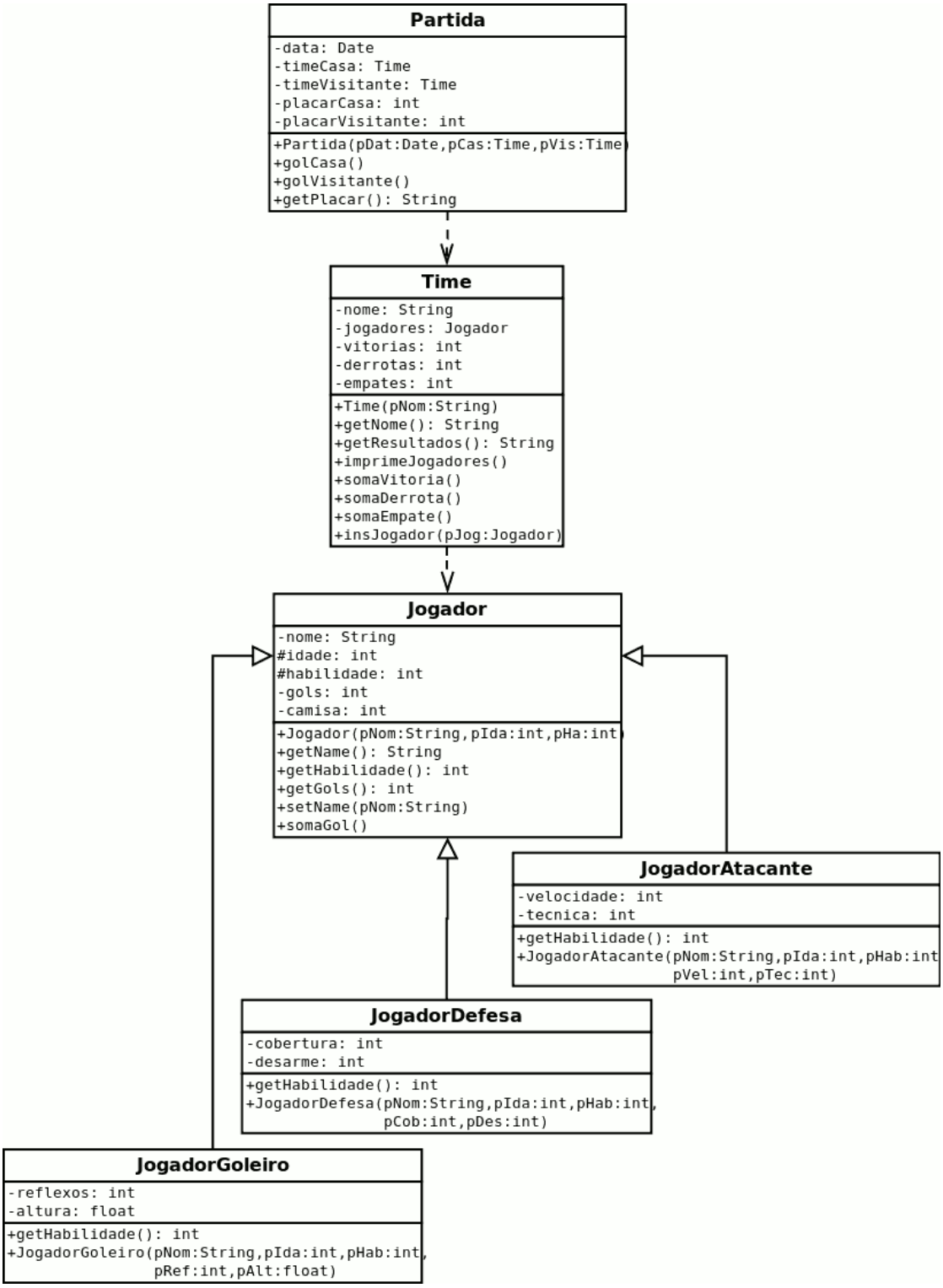

Figura 1: Diagrama de classes UML com a estrutura mínima que deverá ser implementada

- **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 x 0 Nome do Time Visitante`.

Instruções

A implementação deverá obrigatoriamente ser feita em **Java** e **C++**. Colocar seus projeto em pastas separadas: uma pasta `Java` e uma pasta `CC`, e compactar as duas.

1. Em **Java** deverá ser criado um Pacote `Jogadores` que deverá conter a classe `Jogador` e suas derivadas, e que será importado pelas classes `Time` e `Partida`.
2. 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.
   - Opcionalmente, podem ser criados os subdiretórios `include`, `obj`, `lib` e `src`.
   - Em **C++** não há um tipo `Date` nativo. Você poderá criar um tipo ou uma classe `Date` ou fazer uma adaptação usando os tipos disponíveis.
   - Implemente os destrutores conforme a necessidade de cada classe.

Dúvidas conceituais deverão ser levadas preferencialmente ao Monitor Especial (Eduardo) ou Estagiário PAE (Ricardo), cujos contatos estão na página da disciplina. Envie e-mail com o assunto `[poo-trab1] Duvida`.

ATENÇÃO: A detecção de cópia de parte ou de todo código-fonte, de qualquer origem, implicará reprovação direta no trabalho. Partes do código cujas **idéias** 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.