

## Atividade JUnit (M3)

Considere que você precisa desenvolver um programa em Java para manipular dados de jogadores, times e jogos de futebol. Cada jogador tem uma nota de 0 a 100 que corresponde a sua habilidade. Entretanto, o cálculo da habilidade varia de acordo com a posição do jogador e com o valor de seus atributos, os quais também variam de 0 a 100. Um goleiro possui os atributos altura e reflexos. Considere que o atributo altura é a altura real do jogador em centímetros normalizada para um valor entre 0 e 100. Para isso, a altura máxima a ser considerada é 210 centímetros. Desta forma, qualquer jogador com 210 centímetros ou mais de altura terá a nota 100. Um jogador de 181 centímetros terá a nota 86 (considerando o arredondamento HALF\_EVEN). Um defensor possui os atributos cobertura e desarme. Um atacante possui os atributos velocidade e técnica. O cálculo da habilidade é dado pelas seguintes fórmulas:

- Goleiro:  $\text{altura} * 4 + \text{reflexos} * 6$
- Defensor:  $\text{cobertura} * 6 + \text{desarme} * 4$
- Atacante:  $\text{velocidade} * 4 + \text{técnica} * 6$

O programa deve controlar a quantidade de gols feita por cada jogador. Este valor é acumulativo, pois um jogador pode fazer gols em diferentes partidas. Além do nome do jogador, o programa deve calcular sua idade atual. Cada jogador tem um número de camisa único dentro de seu time (um time não pode ter dois jogadores com um mesmo número de camisa).

Um time possui um nome e jogadores que podem ser adicionados ou removidos. Um time completo é formado por 5 jogadores, sendo 1 goleiro, 2 defensores e 2 atacantes. Além disso, o programa deve controlar os resultados obtidos pelo time (vitórias, empates e derrotas). Ao final de cada partida, os resultados de cada time devem ser atualizados. Uma vitória vale 3 pontos, empate vale 1 e derrota não pontua.

Uma partida é composta de dois times, onde um é o time da casa e o outro é o visitante. Além da data da partida, o programa deve guardar o placar da partida (quantos gols o time da casa fez e quantos gols o time visitante fez). Desta forma, será possível saber quem venceu a partida ou se houve empate.

O programa deve permitir cadastrar várias partidas e depois simular a execução destas partidas, atualizando estatísticas de times e jogadores. Na simulação, considere a pontuação total das habilidades de cada time (quem tem a maior pontuação, tem mais chance de sair vitorioso). Além disso, se o time que estiver jogando em casa tem uma chance adicional. Você pode definir sua própria fórmula, mas ela deve estar explicada no código (javadoc).

### Orientações

1. A pasta do projeto deve ser compactada (formato zip). O arquivo compactado deve ser renomeado antes de ser colocado na Intranet. O formato para o nome do arquivo deve ser "T<n>\_<Nome1>\_<Nome2>.zip", onde:
  - <n> é o número do trabalho solicitado
  - <Nome1> é o nome de um dos alunos (não precisa ser completo, mas o suficiente para identificá-lo na turma)
  - <Nome2> é o nome do outro aluno participante

Os projetos devem ter rigorosamente esta nomenclatura, pois a ferramenta do Material Didático não renomeia o arquivo baixado com o nome dos alunos que fizeram o upload. Além disso, facilitará a identificação de qual projeto pertence a qual equipe, quando carregados no IntelliJ. Exemplos para nomes válidos:

- T2\_TonyStark\_SteveRogers.zip
- T2\_JessicaJones\_NatashaRomanova.zip

**Atividade JUnit (M3)**

2. Para cada classe do seu programa, crie uma classe de teste (JUnit). As classes de teste (teste de unidade) devem estar em uma pasta diferente da pasta do código do programa.
3. Em cada classe de teste, implemente os diferentes casos de teste. Para cada caso de teste, indique a condição de teste e explique as técnicas utilizadas para a definição dos dados (partição de equivalência, análise do valor limite, árvore de decisão).
4. Considere casos de teste positivos e negativos.
5. Como critérios da avaliação, serão considerados: grau de cobertura dos testes em relação à especificação; qualidade dos testes; adoção adequada das técnicas; qualidade interna do programa; qualidade interna dos casos de teste.

## Atividade JUnit (M3)

Para a validação, crie casos de teste para avaliar o programa desenvolvido. Não há a necessidade de criar uma interface com o usuário. Tudo pode ser demonstrado pelos casos de teste. Para facilitar, considere a redefinição do método `toString()`.

Testes esperados:

- Cadastrar vários times.
- Cadastrar vários jogadores.
- Classificar os times de acordo com a habilidade total de seus jogadores.
- Classificar os times de acordo com suas estatísticas. Para ordenar, considere total de pontos, número de vitórias, número de empates e saldo de gols.
- Adicionar jogadores em um time.
- Remover jogadores de um time.
- Cadastrar várias partidas.
- Simular a execução das partidas.
- Avaliar se a simulação foi executada corretamente.