

## 2022\_1 - PROGRAMAÇÃO E DESENVOLVIMENTO DE SOFTWARE II - TA\_TN - METATURMA

PAINEL > MINHAS TURMAS > 2022\_1 - PROGRAMAÇÃO E DESENVOLVIMENTO DE SOFTWARE II - TA\_TN - METATURMA > GERAL  
> L02E01 - TREINADOR POKÉMON (4,0 PTS)


 Descrição



 Enviar

 Editar

 Visualizar envios

### L02E01 - Treinador Pokémon (4,0 pts)

 Data de entrega: sexta, 1 Jul 2022, 23:59

 Arquivos requeridos: evolutionBall.hpp, evolutionBall.cpp, healthBall.hpp, healthBall.cpp, pokebola.hpp, pokebola.cpp, pokemon.hpp, pokemon.cpp, pokemonCapturado.hpp, pokemonCapturado.cpp, treinador.hpp, treinador.cpp ( [Baixar](#))

Tipo de trabalho:  Trabalho individual

Herança e Composição	
VPL: 1	Nome: Treinador Pokemon

#### Objetivo:

Seu objetivo neste exercício é usar os conceitos de *herança e composição* para simular uma parte do universo pokémon. No final, sua aplicação deverá permitir **gerenciar as pokébolas de um treinador e capturar novos pokémons**. Não será preciso implementar a main.cpp, mas você pode baixá-la [aqui](#) se desejar reproduzir o exercício em sua máquina.

#### TADs:

Classe Treinador	
Atributos:	<b>private std::string nome</b> → Nome do treinador <b>private std::vector&lt;EvolutionBall*&gt; evolution_balls</b> → EvolutionBalls do treinador <b>private std::vector&lt;HealthBall*&gt; health_balls</b> → HealthBalls do treinador
Métodos: (Todos os métodos descritos abaixo devem ser públicos)	
<b>Treinador(std::string _nome)</b> → Construtor que inicializa as variáveis <b>~Treinador()</b> → Libera a memória alocada para as pokébolas (evolution_balls e health_balls) <b>HealthBall* selecionarHealthBall(int _id)</b> → Retorna uma <i>HealthBall</i> pelo seu <i>id</i> ou <i>nullptr</i> se não a encontrar <b>EvolutionBall* selecionarEvolBall(int _id)</b> → Retorna uma <i>EvolutionBall</i> pelo seu <i>id</i> ou <i>nullptr</i> se não a encontrar <b>void adicionarPokebola(HealthBall* pokebola)</b> → Adicionar uma nova <i>HealthBall</i> a lista de health_balls <b>void adicionarPokebola(EvolutionBall* pokebola)</b> → Adicionar uma nova <i>EvolutionBall</i> a lista de evolution_balls <b>void listarPokemons()</b> → Lista as informações dos pokémons que o treinador possui. Quando o treinador <b>não</b> tem pokébolas a saída deve ser: Treinador: [nome_treinador] O treinador não possui pokebolas A saída para um treinador que possui 2 pokébolas (1 <i>HealthBall</i> e 1 <i>EvolutionBall</i> ) e uma contém um pokémon e outra não: Treinador: [nome_treinador] HealthBall ID: [id_pokebola] Pokemon: [nome_pokemon], [tipo_pokemon], [forcaAtaque], [forcaDefesa], [saude] EvolutionBall ID: [id_pokebola] A pokebola não possui um pokemon Quando a pokébola possui um pokémon capturado são exibidas as informações dele e quando ela não possui nenhum pokémon é exibida a mensagem "A pokebola não possui um pokemon."	
<b>Observações:</b> Note que no exemplo da saída do método void listarPokemons() você <b>NÃO</b> deve printar os colchetes também, eles são apenas uma forma de indicar que o que está entre os colchetes é um campo do objeto.	

Classe Pokebola	
Atributos:	<b>private static int count</b> → Contador de pokébolas criadas <b>protected int id</b> → Identificador da pokébola, atualizado de acordo com o valor do contador <b>protected PokemonCapturado* pokemon</b> → Pokémon capturado da pokébola

Métodos: (Todos os métodos descritos abaixo devem ser públicos)	
<p><b>Pokebola()</b> → Inicializa as variáveis. O construtor deve incrementar o contador <b>depois</b> de atribuir o id da pokebola como o valor do contador.</p> <p><b>~Pokebola()</b> → Libera a memória alocada para o pokémon capturado.</p> <p><b>int getId()</b> → Retorna o Id da pokébola</p> <p><b>void guardarPokemon()</b> → Marca o pokémon com o status "dormindo", caso a pokébola possua um pokémon</p> <p><b>Pokemon* liberarPokemon()</b> → Se a pokébola não tem pokémon retorna <i>nullptr</i>, mas se ela tem remove o status de dormindo do pokémon e retorna um ponteiro para ele</p> <p><b>bool capturar(Pokemon&amp; _pokemon)</b> → Deve se obter um número aleatório entre 0 e 1 utilizando a <code>rand()</code> da biblioteca <code>&lt;cstdlib&gt;</code>:</p> <p>→ Se o valor obtido for maior que 0.5 o pokémon é capturado e colocado na pokebola e você deve retornar <i>true</i>. Para colocá-lo na pokébola você irá precisar de criar uma instância <code>PokemonCapturado</code> que tenha as mesmas informações que o pokémon recebido pela função</p> <p>→ Se o valor obtido for menor ou igual que 0.5 você deve retornar <i>false</i> indicando que o pokémon não foi capturado</p> <p>→ ATENÇÃO: Você não precisa informar a seed, pois ela já é informada no <code>main.cpp</code>.</p>	

Classe HealthBall : Pokebola	
Atributos:	<p><b>private time_t ultimoUso</b> → Indica quando foi o último uso da habilidade de cura desta pokébola. O tipo <code>time_t</code> deve ser utilizado com a biblioteca <code>&lt;ctime&gt;</code>. Ele representa o tempo em segundos. Se ele for igual a 0 é o equivalente às 00:00 de 1 de Janeiro de 1970 (UTC)</p> <p><b>private double intervalo</b> → Indica qual o intervalo em segundos que pode se utilizar a habilidade de cura novamente</p>
Métodos: (Todos os métodos descritos abaixo devem ser públicos)	
<p><b>HealthBall(double _intervalo)</b> → <i>HealthBall</i> é uma pokébola especial que permite que o treinador cure seu pokémon restaurando sua saúde ao máximo. Essa habilidade só pode ser utilizado de tempos em tempos que é estabelecida pelo atributo "intervalo" dessa pokébola</p> <p><b>bool recuperarPokemon()</b> → Estabelece a seguinte regra:</p> <p>→ Se a pokébola <b>não</b> possuir um pokémon então a habilidade não pode ser utilizada, retornando <i>false</i>.</p> <p>→ Se a pokébola possui um pokémon, então a habilidade só pode ser utilizada se a diferença entre a data de último uso e a data atual for maior do que o "intervalo", se isso for verdade deve se retornar <i>true</i> indicando que a habilidade foi usada, caso o contrário, deve se retornar <i>false</i>. Para saber a data atual o método <code>time</code> da biblioteca <code>&lt;ctime&gt;</code> pode ajudar e para calcular a diferença entre as datas o método <code>difftime</code> da biblioteca <code>&lt;ctime&gt;</code> também pode ajudar. Lembre-se de restaurar a saúde nesse momento (<code>maxSaude</code>). Dica: não se esqueça também de atualizar a <code>ultimoUso</code> também!</p>	

Classe EvolutionBall : Pokebola	
Atributos:	<p><b>private double taxaPoder</b> → Indica qual a taxa que o poder do pokémon será aumentado após a evolução</p> <p><b>private bool habilidadeUsada</b> → Indica se a habilidade de evoluir já foi usada ou não</p>
Métodos: (Todos os métodos descritos abaixo devem ser públicos)	
<p><b>EvolutionBall(double _taxaPoder)</b> → <i>EvolutionBall</i> é uma pokébola especial que tem a habilidade de evoluir uma única vez o pokémon capturado. O método construtor é responsável por inicializar o <code>taxaPoder</code> que é uma variável que indica o quanto o poder do pokémon vai aumentar após a evolução.</p> <p><b>bool evoluirPokemon()</b> → Estabelece a seguinte regra:</p> <p>→ Se a pokébola <b>não</b> possuir um pokémon então a habilidade não pode ser utilizada, retornando <i>false</i>.</p> <p>→ Se a pokébola possui um pokémon, então a habilidade só pode ser utilizada se já não tiver sido usada antes. Caso utilizada deve retornar <i>true</i> e se não for utilizada deve retornar <i>false</i>. Lembre-se que após a habilidade ser utilizada a flag deve ser atualizada para <i>true</i> e você deve chamar o método <code>evoluir()</code> de <code>Pokemon</code>.</p>	

Pokemon	
Atributos:	<p><b>protected std::string nome</b> → Nome do pokémon</p> <p><b>protected std::string tipo</b> → Tipo do pokémon</p> <p><b>protected double forcaAtaque</b> → Indicador da força de ataque do pokémon</p> <p><b>protected double forcaDefesa</b> → Indicador da força de defesa do pokémon</p> <p><b>protected std::string proxEvolucao</b> → Nome da próxima evolução do pokémon</p> <p><b>protected double saude</b> → Indicador da saúde do pokémon</p>
Métodos: (Todos os métodos descritos abaixo devem ser públicos)	
<p><b>Pokemon(std::string _nome, std::string _tipo, std::string _pEvol, double _fA, double _fD, double _saude)</b> → <code>_pEvol</code> é referente a <i>proximaEvolucao</i>, <code>_fA</code> é referente a <i>forcaAtaque</i> e <code>_fD</code> é referente a <i>forcaDefesa</i>. O construtor deve inicializar os atributos do objeto</p> <p><b>std::string getName()</b> → Retorna o nome do pokémon</p> <p><b>void maxSaude()</b> → O valor máximo da saúde é 100. Este método deve preencher a saúde do pokemon com 100.</p> <p><b>void info()</b> → Printa as informações a respeito do pokémon no seguinte formato:</p> <p>Pokemon: [nome_pokemon], [tipo_pokemon], [forcaAtaque], [forcaDefesa], [saude]</p>	
<p><b>Observações:</b></p> <p>Note que no exemplo da saída do método <code>void info()</code> você <b>NÃO</b> deve printar os colchetes também, eles são apenas uma forma de indicar que o que está entre os colchetes é um campo do objeto.</p>	

PokemonCapturado : Pokemon
----------------------------

<b>Atributos:</b>	<b>private bool evoluído</b> → Indicador se pokémon já foi evoluído <b>private bool dormindo</b> → Indicador se o pokémon está dormindo
<b>Métodos:</b>	<b>PokemonCapturado(Pokemon&amp; _pok)</b> → Inicializa as variáveis: <i>evoluído</i> será <i>false</i> e <i>dormindo</i> será <i>true</i> . A referência <i>_pok</i> deve ser passada para o construtor da classe pai, o que estamos fazendo é utilizando o construtor de cópia da classe pai (você pode ler mais sobre construtores de cópia neste <a href="#">link</a> ) <b>void evoluir(double taxaPoder)</b> → Este método vai evoluir o pokémon setando o atributo “ <i>evoluído</i> ” como <i>true</i> . O processo de evoluir precisa: → Atualizar o nome do pokémon para o nome da sua evolução → Colocar o nome da próxima evolução como vazia → Aumentar a força de defesa e ataque. A “ <i>taxaPoder</i> ” é um número entre 0 e 1 que representa qual fator de aumento que a força de defesa e ataque irão sofrer

Você tem liberdade para implementar quaisquer outros métodos na TAD que julgar necessário. Lembre-se que getters e setters podem ser importantes quando atributos são privados ou protegidos e precisamos acessá-los de fora da classe.

#### Exemplos de entrada e saída:

<b>Exemplo 1</b>	
<b>Entrada:</b> Anne 3 1 h 0 kakuna inseto 9 30.9 beedrill 63 e 3 jigglypuff normal 18 15.1 wigglytuff 13 h 2 odish planta 22.5 16.9 gloom 80 h 1 clefairy fada 12.1 22 clefable 10 q h 1 i 3000 h 1 e 3 q	<b>Saída:</b> Treinador: Anne HealthBall ID: 0 A pokebola não possui um pokemon HealthBall ID: 1 A pokebola não possui um pokemon HealthBall ID: 2 A pokebola não possui um pokemon EvolutionBall ID: 3 A pokebola não possui um pokemon ----- Pokémon kakuna capturado. Pokémon jigglypuff NÃO capturado. Pokémon odish NÃO capturado. Pokémon clefairy capturado. ----- Treinador: Anne HealthBall ID: 0 Pokemon: kakuna, inseto, 9, 30.9, 63 HealthBall ID: 1 Pokemon: clefairy, fada, 12.1, 22, 10 HealthBall ID: 2 A pokebola não possui um pokemon EvolutionBall ID: 3 A pokebola não possui um pokemon ----- HealthBall 1 Pokemon: clefairy, fada, 12.1, 22, 100 Intervalo 3000 milisegundos HealthBall 1 Pokemon: clefairy, fada, 12.1, 22, 100 EvolutionBall 3 Pokémon NÃO evoluído.
<b>Exemplo 2</b>	
<b>Entrada:</b> Rui 0 1 e 0 jigglypuff normal 18 15.1 wigglytuff 13 q e 0 q	<b>Saída:</b> Treinador: Rui EvolutionBall ID: 0 A pokebola não possui um pokemon ----- Pokémon jigglypuff capturado. ----- Treinador: Rui EvolutionBall ID: 0 Pokemon: jigglypuff, normal, 18, 15.1, 13 ----- EvolutionBall 0 Pokemon: wigglytuff, normal, 36, 30.2, 13
<b>Exemplo 3</b>	
<b>Entrada:</b> Louis 0 0 q q	<b>Saída:</b> Treinador: Louis O treinador não possui pokebolas ----- Treinador: Louis O treinador não possui pokebolas -----

#### Dicas:

→ O tipo de dados `time_t` representa o tempo em segundos. Se ele for igual a 0 é o equivalente às 00:00 de 1 de Janeiro de 1970 (UTC).

~> O tempo atual pode ser obtido por meio da função `time(time_t* t)` que recebe um ponteiro do tipo `time_t`.

~> `int rand()` gera um valor aleatório entre 0 e `RAND_MAX`, `RAND_MAX` depende da biblioteca que está implementando mas é pelo menos maior que 32767 em qualquer implementação de standart library.

~> `double difftime (time_t end, time_t beginning)` recebe duas variáveis do tipo `time_t` e retorna um `double` representando o tempo entre elas.

---

#### Links Úteis:

- Sobre o tipo [time\\_t](#) da biblioteca `<ctime>`
- Sobre o método [rand](#) da biblioteca `<ctime>`
- Sobre o [difftime](#) da biblioteca `<ctime>`
- Sobre o método [time](#) da biblioteca `<ctime>`

[VPL](#)

◀ L01E08 - Imagem (4,0 pts)

Seguir para...

L02E02 - Makefile (1,0 pt) ▶