

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:	<pre>private std::string nome → Nome do treinador private std::vector<EvolutionBall*> evolution_balls → EvolutionBalls do treinador private std::vector<HealthBall*> health_balls → HealthBalls do treinador</pre>
Métodos: (Todos os métodos descritos abaixo devem ser públicos)	
<pre>Treinador(std::string _nome) → Construtor que inicializa as variáveis ~Treinador() → Libera a memória alocada para as pokébolas (evolution_balls e health_balls) HealthBall* selecionarHealthBall(int _id) → Retorna uma HealthBall pelo seu id ou nullptr se não a encontrar EvolutionBall* selecionarEvolBall(int _id) → Retorna uma EvolutionBall pelo seu id ou nullptr se não a encontrar void adicionarPokebola(HealthBall* pokebola) → Adicionar uma nova HealthBall a lista de health_balls void adicionarPokebola(EvolutionBall* pokebola) → Adicionar uma nova EvolutionBall a lista de evolution_balls void listarPokemons() → Lista as informações dos pokémons que o treinador possui. Quando o treinador não tem pokébolas a saída deve ser: Treinador: [nome_treinador] O treinador não possui pokebolas A saída para um treinador que possui 2 pokébolas (1 HealthBall e 1 EvolutionBall) 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."</pre>	
Observações: Note que no exemplo da saída do método void listarPokemons() você NÃO deve imprimir 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:	<pre>private static int count → Contador de pokébolas criadas protected int id → Identificador da pokébola, atualizado de acordo com o valor do contador protected PokemonCapturado* pokemon → Pokémon capturado da pokébola</pre>

Métodos: (Todos os métodos descritos abaixo devem ser públicos)	
	<p>Pokebola() → Inicializa as variáveis. O construtor deve incrementar o contador depois de atribuir o id da pokebola como o valor do contador.</p> <p>~Pokebola() → Libera a memória alocada para o pokémon capturado.</p> <p>int getId() → Retorna o Id da pokébola</p> <p>void guardarPokemon() → Marca o pokémon com o status "dormindo", caso a pokébola possua um pokémon</p> <p>Pokemon* liberarPokemon() → 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>bool capturar(Pokemon& _pokemon) → Deve se obter um número aleatório entre 0 e 1 utilizando a <code>rand()</code> da biblioteca <code><cstdlib></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>private time_t ultimoUso → 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><ctime></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>private double intervalo → 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>HealthBall(double _intervalo) → <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>bool recuperarPokemon() → Estabelece a seguinte regra:</p> <p>~> Se a pokébola não 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><ctime></code> pode ajudar e para calcular a diferença entre as datas o método <code>difftime</code> da biblioteca <code><ctime></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>private double taxaPoder → Indica qual a taxa que o poder do pokémon será aumentado após a evolução</p> <p>private bool habilidadeUsada → Indica se a habilidade de evoluir já foi usada ou não</p>
Métodos: (Todos os métodos descritos abaixo devem ser públicos)	
	<p>EvolutionBall(double _taxaPoder) → <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>bool evoluirPokemon() → Estabelece a seguinte regra:</p> <p>~> Se a pokébola não 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>protected std::string nome → Nome do pokémon</p> <p>protected std::string tipo → Tipo do pokémon</p> <p>protected double forcaAtaque → Indicador da força de ataque do pokémon</p> <p>protected double forcaDefesa → Indicador da força de defesa do pokémon</p> <p>protected std::string proxEvolucao → Nome da próxima evolução do pokémon</p> <p>protected double saude → Indicador da saúde do pokémon</p>
Métodos: (Todos os métodos descritos abaixo devem ser públicos)	
	<p>Pokemon(std::string _nome, std::string _tipo, std::string _pEvol, double _fA, double _fD, double _saude) → <code>_pEvol</code> é referente a <code>proximaEvolucao</code>, <code>_fA</code> é referente a <code>forcaAtaque</code> e <code>_fD</code> é referente a <code>forcaDefesa</code>. O construtor deve inicializar os atributos do objeto</p> <p>std::string getName() → Retorna o nome do pokémon</p> <p>void maxSaude() → O valor máximo da saúde é 100. Este método deve preencher a saúde do pokemon com 100.</p> <p>void info() → Printa as informações a respeito do pokémon no seguinte formato:</p> <p>Pokemon: [nome_pokemon], [tipo_pokemon], [forcaAtaque], [forcaDefesa], [saude]</p>
Observações:	<p>Note que no exemplo da saída do método <code>void info()</code> você NÃO 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

Atributos:	<div><div>private bool evoluído → Indicador se pokémon já foi evoluído</div><div>private bool dormindo → Indicador se o pokémon está dormindo</div></div>
Métodos:	<div><div>PokemonCapturado (Pokemon& _pok) → Inicializa as variáveis: evoluído será false e dormindo será true. A referência _pok deve ser passada para o contrutor 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 link)</div><div>void evoluir (double taxaPoder) → Este método vai evoluir o pokémon setando o atributo "evoluído" como true. O processo de evoluir precisa:<div><div>~> Atualizar o nome do pokémon para o nome da sua evolução</div><div>~> Colocar o nome da próxima evolução como vazia</div><div>~> Aumentar a força de defesa e ataque. A "taxaPoder" é um número entre 0 e 1 que representa qual fator de aumento que a força de defesa e ataque irão sofrer</div></div></div></div>

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:

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

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) ▶