Projeto - Batalha Pokemon

Team - 10

Celso Junio Simões de Oliveira santos

Felipe Matheus Guimarães Santos

Leandro Marques Venceslau de Souza

Leonardo Dutra Pereira

Matheus Ferreira Coelho

Introdução:

O desenvolvimento de códigos e softwares é um dos aspectos fundamentais em Sistemas de Informação. Nesse contexto, o uso de boas práticas de programação aliado a conceitos de Orientação a Objetos e Test Driven Development (TDD) é recomendado na criação de um código estável, funcional e de fácil leitura.

O objetivo deste trabalho é implementar um jogo baseado nos da franquia Pokémon, utilizando os conceitos passados em sala de aula nesse semestre, no curso de Programação e Desenvolvimento de Software II. Para tal procuramos aplicar: Abstração, Encapsulamento, Herança, Polimorfismo, modularidade e componentes reutilizáveis, tudo isso no intuito de aproveitarmos funções comuns entre as aplicações, assim como testes de exceção em métodos que julgamos necessário sua aplicação.

Do Projeto:

O projeto sofreu diversas modificações e refatorações baseandas no TDD para a remoção de problemas como: métodos longos, números mágicos, classes invejosas, etc. Todos os integrantes participaram ativamente, e demonstraram interesse na elaboração do projeto.

Step-by-step do Jogo:

O jogo é iniciado solicitando a entrada de um nome para o jogador. Constrói-se um objeto do tipo Treinador que recebe como parâmetros o nome do jogador e o nível de liderança dele (no caso 1 pois ele é um novato).

Logo após a função introdução é executada, recebendo como parâmetro o objeto Treinador criado. Ela confere ao usuário três pokémons iniciais para que ele possa batalhar. O usuário também tem a oportunidade de dar apelidos aos seus Pokemon.

Em seguida chama-se a funcão treinador_info que também recebe como parâmetro o objeto Treinador. Essa função é responsável por informar o jogador sobre os atributos e características de seus Pokemon.

Depois a função escolher_pokemon é iniciada, a qual permite selecionar e levar à batalha um dos Pokemon previamente adicionados. Chama-se então a função batalha_x1 parametrizada com o objeto jogador, o pokemon escolhido e o nível de dificuldade.

Durante a batalha um jogador com nível de liderança 4 (mestre) terá acesso a 4 habilidades específicas do Pokemon selecionado por escolher_pokemon. No começo, no entanto, o jogador é considerado novato e tem acesso apenas à uma habilidade. Ao ganhar uma batalha adiciona-se um ao nível de liderança do jogador até um valor máximo 4.

Ao final de cada batalha o usuário tem três opções:

1) Capturar o pokemon inimigo : Caso a captura seja feita com sucesso o pokemon inimigo é adicionado ao vetor que armazena pokemons, presente no objeto Treinador.

2) Batalhar novamente, selecionando-se tanto o pokemon a ser levado à batalha quanto o nível de dificuldade da próxima luta.

3) Ver seu Pokedéx, chamando novamente a função treinador_info.

É importante denotar que a função batalha_x1 está estruturada recursivamente. Nas opções 1 e 3 acima é perguntado ao jogador se ele deseja continuar batalhando, caso afirmativo então um comportamento idêntico à escolha 2) ocorre, ou seja, seleciona-se a dificuldade do jogo, o pokemon pertencente ao jogador e, logo em seguida, outro método batalha_x1 é chamado.

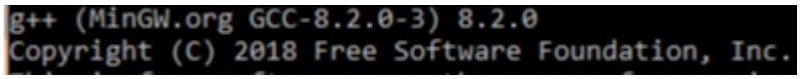
Executando o programa:

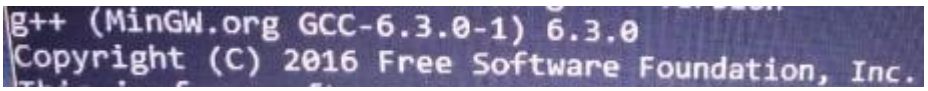
Nosso projeto foi preparado com um arquivo Makefile para facilitar sua execução; para iniciá-lo, siga os passos:
Execute o terminal de sua máquina, e vá ao diretório/pasta “20191-team-10” após, utilize um dos seguintes comandos:
make: compila todo o código
make tests: executa os testes de unidade
make coverage: mostra a cobertura dos testes em relação ao código
make valgrind: executa o valgrind em busca de memory leaks
make clean: limpa tudo
make run: roda o main

Compilação e execução:

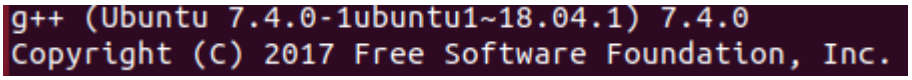
O projeto foi compilado e executado normalmente nos seguintes sistemas operacionais e seus respectivos compiladores nas versões:

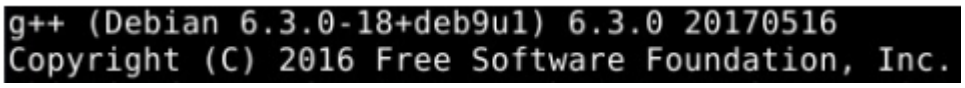
MinGW (Windows 7/10):





G++ (Ubuntu/Debian):





Especificações do Código:

A seguir estão apresentadas diversas informações a respeito do código desenvolvido, além de um breve detalhamento das classes e funções.

Hierarquia de Classes

Esta lista de hierarquias está parcialmente ordenada (ordem alfabética):

- Exception
  - Excpt_Entrada_Inicial
  - Excpt_Habilidade_Invalida
  - Excpt_Lideranca_Invalida
  - Excpt_Nocaute

- Pokemon
  - Pokemon_Agua
    - Blastoise
    - Squirtle
    - Wartotle

  - Pokemon_Fogo
    - Charizard
    - Charmander
    - Charmeleon

  - Pokemon_Grama
    - Bulbasauro
    - Ivysaur
    - Venosauro

- Treinador

Lista de Classes:

Blastoise
Bulbasauro
Charizard
Charmander
Charmeleon
Excpt_Num_Pokemon_Invalido
Excpt_Habilidade_Invalida
Excpt_Lideranca_Invalida
Excpt_Nocaute

Ivysaur
Pokemon
Pokemon_Agua
Pokemon_Fogo
Pokemon_Grama
Squirtle
Treinador
Venosauro
Wartotle

Classe “Pokemon” :

Pokemon: É uma Superclasse que contém diversas variáveis protegidas referentes aos atributos do Pokemon, as quais serão herdadas pelas classes filhas listadas abaixo. Existe também uma variável pública current_hp que será alterada nas sub-classes pelos métodos polimórficos referentes à batalha. Também separamos os Métodos de Polimorfismo e os referentes à manipulação de TADs.

Pokemon_Agua: Atribui valor às variáveis de resistência e fraqueza; subclasses:

- **Charmander** - Objeto pokemon
- **Charmeleon** - Objeto pokemon
- **Charizard** - Objeto pokemon

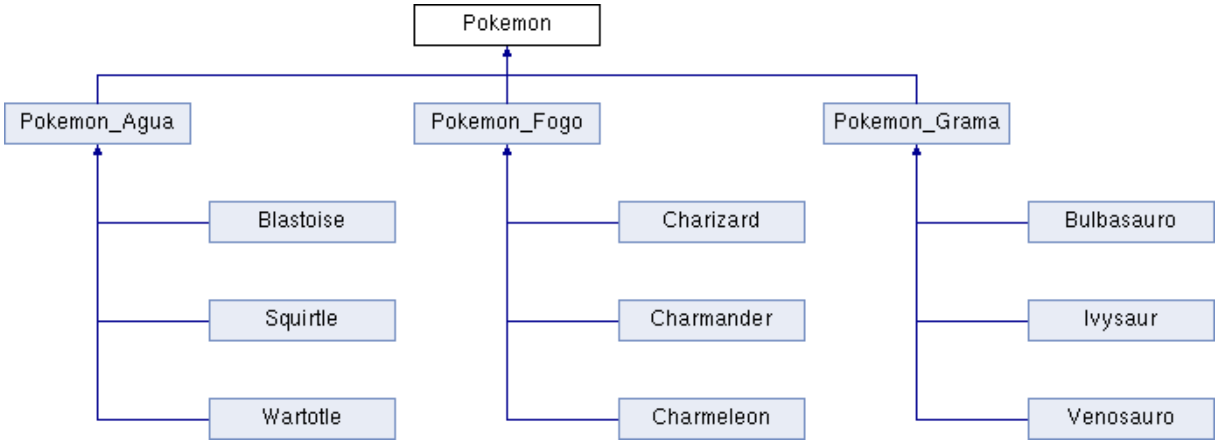
Pokemon_Grama: Atribui valor às variáveis de resistência e fraqueza; subclasses:

- **Charmander** - Objeto pokemon
- **Charmeleon** - Objeto pokemon
- **Charizard** - Objeto pokemon

Pokemon_Fogo: Atribui valor às variáveis de resistência e fraqueza; subclasses:

- **Charmander** - Objeto pokemon
- **Charmeleon** - Objeto pokemon
- **Charizard** - Objeto pokemon

Diagrama da Superclasse “Pokemon” e suas respectivas subclasses:



Classe “Treinador”:

É a classe mais próxima da interação com o usuário. Contém variáveis privadas com base nos princípios do encapsulamento, para que somente a própria classe tenha acesso às mesmas. Além da variável pública Liderança que influenciará diretamente nas habilidades dos pokemons e na batalha. Seus métodos de Get e set são necessários para definir a quantidade de itens e pokemons atribuídos ao treinador.

“Exceções.h”

É o arquivo que contém uma lista de classes voltado unicamente a verificação de casos de exceção, e posteriormente, jogar o tipo de exceção específica.

Class Excpt_Nocaute - Verifica um pokemon nocauteado

Class Excpt_Lideranca_Invalida - Verifica se o valor atribuído a Liderança está entre 1 e 4

Class Excpt_Habilidade_Invalida - Verifica se o treinador tem liderança suficiente para usar determinada habilidade.

Class Excpt_Num_Pokemon_Invalido - Verifica se o treinador possui um número de pokemons válido.

Arquivos e funções

Alguns outros arquivos essenciais foram utilizados para melhor organização do projeto, e criação de funções específicas para interagir com as classes. São eles:

batalha.h
introdução.h
pokedex_ascii.h
interface.h
pokebola.h
pokedex.h

Batalha.h: É o arquivo que contém, como seu nome diz, as funções destinadas às batalhas em nosso projeto. Tais são elas: gera oponentes, resete de hp, rola dados, escolher pós batalha, encerrar batalha e, a mais importante, batalha_x1; esta última em específico recebe, como parâmetros, um treinador, seu pokémon escolhido e um inteiro que determinará a dificuldade do oponente; ela é utilizada para o combate entre os pokemons. Exceções e seus respectivos tratamentos foram feitos dentro desse arquivo para que todas as funções pudessem rodar corretamente sem nenhum erro.

Introdução.h: É o arquivo que realiza o primeiro contato do programa com a tela de exibição que executa o programa; faz a introdução ao programa. Possui uma função com todo o texto introdutório do game, e uma uma classe exceção (Excpt_Entrada_Inicial) que faz a verificação e tratamento da primeira entrada recebida pelo programa.

Interface.h: É um *header* que tem como propósito armazenar funções relacionadas a interface, ou seja, aquelas funções que possuem print e inputs mais extensos. Tais são elas, funções de escolhas de: pokebola, pokemon, habilidade, número, capturar ou nao; função de dor_captura e bool de checar pokebolas. É o arquivo que auxilia as interações.

Pokebola.h: É um arquivo que contém as funções complementares da captura de um pokemon, utilizados após a batalha; funcionando, como indica seu nome, como uma pokebola em Pokemon. As funções são: capturar_pokemon, pokemon_capturado e utilizar_pokebola. Também são atribuídos nesse arquivo, por meio de “defines”, valores a palavras intuitivas que irão evitar o uso de números mágicos nas funções (boas práticas sempre).

Pokedex.h: É um arquivo destinado unicamente a uma função, a função treinador_info, que apresenta status gerais do treinador (como nome, liderança, número de pokebolas, etc…) e de todos os pokemons que o mesmo possui.

Pokedex_ASCII.h: É um arquivo que possui a função print_ascii_art, destinada a imprimir na tela os ascii arts correspondentes ao identificadores dos pokemons.

Polimorfismo Aplicado

No nosso trabalho, aplicamos os conceitos de polimorfismo principalmente na Superclasse Pokemon e suas classes filhas, seguindo a seguinte lógica:

Um Pokemon X possui determinado tipo (fogo,agua,grama) e pertence à raça Pokemon. Ex: um Squirtle é um Pokemon, mais especificamente um Pokemon de Agua.

Cada Pokemon_Tipo tem uma série de fraquezas e resistências. Essas características são utilizadas no método polimórfico atacar que estabelece diferenças em sua funcionalidade baseado no objeto que utiliza essa função.

Testes

Foram realizados vários testes de Unidade para assegurar o bom funcionamento das classes e funções acessíveis do programa; sendo utilizado a ferramenta para testes “Doctest”, disponibilizada gratuitamente no Github, alocando-a em uma pasta “third_party”; é a mesma ferramenta utilizada pelos professores nos VPLs da matéria.
Os testes foram divididos em 3 arquivos:
testes_treinador.cpp : Aqui foram testados: o Contrutor da classe treinador, seu funcionamento correto, as definições dos parâmetros da classe setados automaticamente, os getters e setters da classe e sua alocação correta, os métodos das funções de pokebolas e capturas, os métodos relacionados a adição de pokemons a lista do jogador, e sua função pokedex. Todos os testes passaram conforme esperado.

testes_pokemon.cpp : Neste arquivo foram testadas: os construtores dos objetos pokemon in game, as características especiais de fraqueza e resistência de cada tipo diferente de pokemon, além dos getters de status (hp, defesa, ataque, etc..) de um pokemon listado no treinador. Todos os testes passaram conforme esperado.

testes_batalha.cpp : Este arquivo foi destinado unicamente ao teste da função batalha_x1, que une as classes treinador e pokemon. O teste passou.

Considerações Finais

O trabalho foi uma ótima maneira de testar os ensinamentos passados ao longo do semestre. A principal dificuldade encontrada foi decidir quais dentre tantas abstrações e padrões para escrita de código seriam aplicados ao projeto, considerando restrições como tempo, sintaxe da linguagem e organização das tarefas referentes à cada membro do grupo.

No mais, foi uma boa experiência =].

Referências

Slides da matéria (Google drive)
https://pt.stackoverflow.com/ - Dúvidas em geral quanto a implementações
https://github.com/ongtam/doctest - Doctest

Observação: O Doxygen gerou um arquivo em formato RTF grande (149 páginas no nosso caso), por isso a Documentação em texto foi feita manualmente para uma melhor compreensão. Mesmo assim disponibilizamos o arquivo em HTML gerado pelo Doxygen na mesma pasta que este arquivo de Documentação, no Github. O arquivo em HTML contém todas as linhas de código feitas/utilizadas com breves descrições e devidamente separadas.