UNICAMP

Universidade Estadual de Campinas
Instituto de Computação

Laboratório 4

Interfaces e Arquivos

MC322 - Programação Orientada a Objetos

Professor: Marcos Raimundo
PEDs: Augusto Cesar / Heigon Soldera
PAD: Matheus Seiji Luna Noda

INSTITUTO DE COMPUTAÇÃO

1. Descrição Geral

No laboratório quatro, iremos utilizar arquivos XML e interfaces para inicializar o Jogo com jogadores, montros e tesouros. Além de criar uma classe para conter a lógica do jogo, como pode ser visto no Diagrama a seguir.[1] .

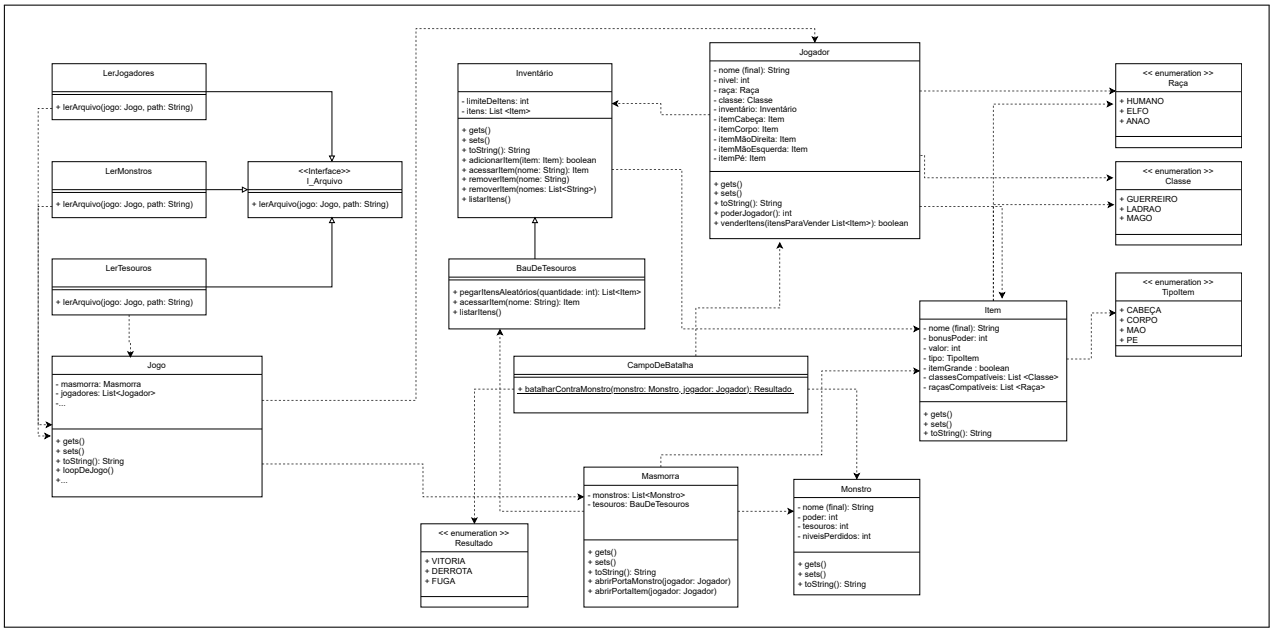

Figura 1: Diagrama de Classe

2. Objetivos

Os principais objetivos deste laboratório consistem em:

- Consolidação dos conteúdos vistos nos labs anteriores;
- Interfaces e Polimorfismo;
- Uso de arquivos para carregamento dos dados do jogo;

3. Atividades

A principal atividade desse laboratório é passar o loop de jogo para a classe Jogo e inicializar os jogadores, monstros e tesouros a partir de arquivos. Além de implementar uma nova regra na lógica do jogo para iniciar os jogadores com seus itens.

[1] Note que algumas das funcionalidades requeridas não estão explicitamente demonstradas no diagrama de classes. Tal abordagem visa um dos objetivos desse laboratório, o qual é a ampliação da capacidade de abstração por parte dos alunos em como aplicar os conceitos visto em aula para a resolução de problemas.

4. Descrição das Classes

Algumas classes foram criadas, a seguir segue alguns destaques importantes:

Classe - Jogo

No laboratório anterior o *loop* de funcionamento principal do jogo deveria ser executado dentro da *main* do programa. O objetivo agora é enviar a lógica criada na *main* para uma classe específica encapsulando o funcionamento. Isso trará benefícios como maior organização, reutilização facilitada, manutenção simplificada e melhoria da legibilidade do código. Não há necessidade de alterar a lógica geral desenvolvida no último laboratório.

Interface - I_Arquivo

Responsável por ler os arquivos, o leitor de cada classe específica deve implementar ela.

Exemplo de Arquivo de Jogadores com Leitor

Segue um exemplo de um arquivo XML contendo dois jogadores, no caso de jogador, podemos assumir que os outros atributos não iniciem com valores, pois se referem a itens que os jogadores irão ganhar. Além disso, segue também um leitor desse arquivo.

```
<Jogadores>
    <Jogador>
        <nome>Joao</nome>
        <nivel>1</nivel>
        <raca>HUMANO</raca>
        <classe>GUERREIRO</classe>
    </Jogador>
    <Jogador>
        <nome>Maria</nome>
        <nivel>1</nivel>
        <raca>ELFO</raca>
        <classe>LADRAO</classe>
    </Jogador>
</Jogadores>
```

```
public class LerJogadores implements I_Arquivo {
    @Override
    public void lerArquivo(Jogo jogo, String path) {
        List<Jogador> jogadores = new ArrayList<>();

        try {
            File file = new File(path);
            DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(file);
            doc.getDocumentElement().normalize();

            NodeList nodeList = doc.getElementsByTagName("Jogador");

            for (int i = 0; i < nodeList.getLength(); i++) {
                Element jogadorElement = (Element) nodeList.item(i);

                String nome = jogadorElement.getElementsByTagName("nome").item(0).
    getTextContent();
                int nivel = Integer.parseInt(jogadorElement.getElementsByTagName("nivel").
    item(0).getTextContent());
                Raca raca = Raca.valueOf(jogadorElement.getElementsByTagName("raca").item(0)
    .getTextContent());
                Classe classe = Classe.valueOf(jogadorElement.getElementsByTagName("classe")
    .item(0).getTextContent());

                Jogador jogador = new Jogador(nome, nivel, raca, classe, new Inventario());
                jogadores.add(jogador);
            }
        } catch (Exception e) {
            System.err.println("Erro ao ler o arquivo: " + e.getMessage());
            e.printStackTrace();
```

```
        }

        jogo.setJogadores(jogadores);
    }
}
```

Exemplo de Arquivo de Monstros e Itens

Aqui são apresentados exemplos de arquivos de Monstro e Itens, basta alterar o código fornecido para se adaptar a cada caso, todos sendo herdeiros da mesma Interface e implementando o mesmo método.

```
<Monstros>
    <Monstro>
        <nome>Ogro</nome>
        <poder>15</poder>
        <tesouros>2</tesouros>
        <niveisPerdidos>1</niveisPerdidos>
    </Monstro>
    <Monstro>
        <nome>Goblin</nome>
        <poder>8</poder>
        <tesouros>1</tesouros>
        <niveisPerdidos>0</niveisPerdidos>
    </Monstro>
</Monstros>
```

```
<Itens>
    <Item>
        <nome>Espada Longa</nome>
        <bonusPoder>5</bonusPoder>
        <valor>500</valor>
        <tipo>MAO</tipo>
        <itemGrande>true</itemGrande>
        <classesCompativeis>
            <classe>GUERREIRO</classe>
            <classe>ARQUEIRO</classe>
        </classesCompativeis>
        <racasCompativeis>
            <raca>HUMANO</raca>
            <raca>ELFO</raca>
        </racasCompativeis>
    </Item>
    <Item>
        <nome>Elmo Viking</nome>
        <bonusPoder>10</bonusPoder>
        <valor>200</valor>
        <tipo>CABECA</tipo>
        <itemGrande>false</itemGrande>
        <classesCompativeis>
            <classe>GUERREIRO</classe>
        </classesCompativeis>
        <racasCompativeis>
            <raca>HUMANO</raca>
            <raca>ELFO</raca>
            <raca>ANAO</raca>
        </racasCompativeis>
    </Item>
</Itens>
```

5. Execução

A execução desse laboratório consiste em, na função Main, inicializar um objeto da classe Jogo e passar ela pelos 3 leitores de arquivo. Dessa forma, a classe Jogo terá o jogo pronto para iniciar, com todos os jogadores, monstros e itens. Porém dessa vez terá um novo passo antes de iniciar o loop de jogo padrão desenvolvido anteriormente, ao iniciar, cada jogador deve receber 5 itens da lista de itens presentes no Baú

de Tesouros e ter a chance de equipá-los. Esse irá compor seu poder, já que, por padrão, devem iniciar sem itens e no nível 1.

6. Avaliação

- Entrega realizada dentro do prazo estipulado;
- Qualidade do código desenvolvido (tabulação, comentários);
- Implementação da lógica da leitura de arquivos usando Interface.
- Desenvolvimento correto dos métodos de acesso.

7. Entrega

- **A entrega do Laboratório é realizada exclusivamente via Github**.
- Utilize os horários de laboratório e atendimentos para tirar eventuais dúvidas de submissão e também relacionadas ao desenvolvimento do laboratório.
- **Prazo de Entrega:** 07/05 - 19h