Universidade Presbiteriana Mackenzie

ESTRUTURA DE DADOS II

Revisão: Programação Orientada a Objetos com Java
Atividade (máx. três alunos)

Objetivo

Revisar tópicos de Estrutura de Dados I, incluindo Programação Orientada a Objetos (POO) com Java.

Instruções

- A atividade deve ser resolvida usando a linguagem Java.
- Inclua a identificação do grupo (nome completo e RA de cada integrante) no início de cada arquivo de código, como comentário.
- Inclua todas as referências (livros, artigos, sites, vídeos, entre outros) consultadas para solucionar a atividade como comentário no arquivo `.java` que contém a `main()`.

Enunciado

1. Crie uma classe Java (ex. `Movie`) para armazenar os dados de um filme.

a) A classe deve conter três atributos privados:
- Título (texto)
- Ano de lançamento (número com quatro dígitos)
- Pontuação (intervalo 0.0 – 10.0)

b) Inclua os métodos públicos *getters*/*setters* de cada atributo do filme.

c) A classe deve conter dois métodos construtores:
- Um construtor sem parâmetros, definindo os seguintes valores-padrão para cada atributo:
  - Título: `"N/A"`
  - Ano de lançamento: `0`
  - Pontuação: `0.0`
- Um construtor com parâmetros, permitindo iniciar todos os atributos da classe.

d) Sobrescreva o método `toString()`, que deve retornar todos os dados de um filme no seguinte formato:
`"Título (Ano) Pontuação"`

Por exemplo:
`"The Intouchables (2011) 8.5"`

2. Crie outra classe Java (ex. `Program`) que contenha a `main()`.

a) Ao iniciar a execução, o programa deve exibir um menu com as seguintes opções:
```
1. Carregar dados
2. Exibir dados
3. Sair
```

b) A opção 1 (Carregar dados) lê o arquivo `imdb.txt` fornecido com esse enunciado, que contém a seguinte estrutura:

- Linha 1: Título do primeiro filme.
- Linha 2: Ano de lançamento do primeiro filme.
- Linha 3: Pontuação do primeiro filme, de acordo com o ranking IMDb.
- Linhas 4, 5 e 6: Título, ano de lançamento e pontuação do segundo filme, respectivamente.
- E assim por diante...

Para cada trio de informações (título, ano, pontuação) lido do arquivo txt, seu código deve armazenar as informações em um objeto do tipo `Movie` (ou o nome que você deu para a sua classe Java do item **1**).

Observações:

- Você deve escolher uma estrutura de dados adequada para mapear o conteúdo do arquivo `imdb.txt` na memória principal.
- Lembre-se de fechar o arquivo após todo o conteúdo ser transferido para a memória principal!

c) A opção 2 (Exibir dados) exibe as informações de cada filme na tela, seguindo o formato:

```
#N: Título (Ano) Pontuação
```

Sendo:

- `#N`: A posição do filme no arquivo `imdb.txt`;
- `Título`: O título do filme;
- `Ano`: O ano de lançamento do filme;
- `Pontuação`: A pontuação IMDb do filme.

Por exemplo:

```
#2: The Godfather (1972) 9.2
```

As informações devem vir dos objetos que foram criados durante a execução do programa.

d) A opção 3 (Sair) encerra a execução do programa. Esta deve ser a única maneira de encerrar o programa.

Entrega

Compacte o código-fonte (somente arquivos `*.java`) no **formato `zip`**.

Atenção: O arquivo `zip` não deve conter arquivos intermediários e/ou pastas geradas pelo compilador/IDE (ex. arquivos `*.class`, etc.).

Prazo de entrega: via link do Moodle até 05/03/2024 23:59.

Universidade Presbiteriana Mackenzie

ESTRUTURA DE DADOS II

Critérios de avaliação

A nota da atividade é calculada de acordo com os critérios da tabela a seguir.

ITEM AVALIADO	PONTUAÇÃO MÁXIMA
1a. Atributos privados da classe que representa um filme.	0,5
1b. *Getters*/*setters* de cada atributo da classe que representa um filme.	0,5
1c. Métodos construtores da classe que representa um filme.	0,5
1d. Sobrescrita do método `toString()` da classe que representa um filme.	0,5
2a. Menu de opções.	1,0
2b. Carregar dados.	3,0
2c. Exibir dados.	2,0
2d. Encerrar programa.	1,0
3. Funcionamento geral do programa, de acordo com o enunciado.	1,0

Tabela 1 - Critérios de avaliação.

A tabela a seguir contém critérios de avaliação que podem **reduzir** a nota final da atividade.

ITEM INDESEJÁVEL	REDUÇÃO DE NOTA
O projeto é cópia de outro projeto.	Projeto é zerado
Há erros de compilação e/ou o programa trava durante a execução[1].	-1,0
Não há identificação do grupo. Não há indicação de referências. Arquivos enviados em formatos incorretos. Arquivos e/ou pastas intermediárias que são criadas no processo de compilação ou pela IDE foram enviadas junto com o código-fonte.	-1,0

Tabela 2 - Critérios de avaliação (redução de nota).

O código-fonte será compilado com o compilador `javac` (21.0.2) na plataforma Windows da seguinte forma:

```
> javac *.java -encoding utf8
```

O código compilado será executado com `java` (21.0.2) na plataforma Windows da seguinte forma:

```
> java <Classe>
```

Sendo que `<Classe>` deve ser substituído pelo nome da classe que contém o método `public static void main(String[] args)`.

[1] Sobre erros de compilação: considere apenas erros. Não há problema se o projeto tiver *warnings* (embora *warnings* podem avisar sobre possíveis travamentos em tempo de execução, como loop infinito, divisão por zero, etc.).

ESTRUTURA DE DADOS II

Apêndice: Algumas dicas...

Exemplo de método que transforma todo o conteúdo de um arquivo texto em uma única string:

```java
/**
 * Cria uma string com o conteúdo do arquivo texto passado no parâmetro filename.
 *
 * @param filename Nome do arquivo texto a ser lido pelo método.
 * @return String contendo todo o conteúdo do arquivo texto.
 * @throws FileNotFoundException Arquivo passado no parâmetro filename não existe (exceção do FileInputStream).
 * @throws IOException Problema ao ler conteúdo do arquivo texto (exceção do BufferedReader).
 */
public static String loadTextFileToString(String filename) throws FileNotFoundException, IOException {
    InputStream is = new FileInputStream(filename);
    InputStreamReader isr = new InputStreamReader(is, "UTF-8");
    BufferedReader br = new BufferedReader(isr);

    StringBuilder sb = new StringBuilder();
    while (true) {
        String line = br.readLine();
        if (line == null) {
            break;
        }
        sb.append(line).append('\n');
    }

    is.close();

    return sb.toString();
}
```

Exemplo de código que divide uma string (`str`) em várias strings (`lines`) a partir da quebra de linhas.

```java
String[] lines = str.split("#|\\r?\\n|\\r");

for (String line : lines) {
    System.out.println(line);
}
```