Prova III – Tipo C - Técnicas de programação I – DSM - Prof. Arley - 28/11/2023

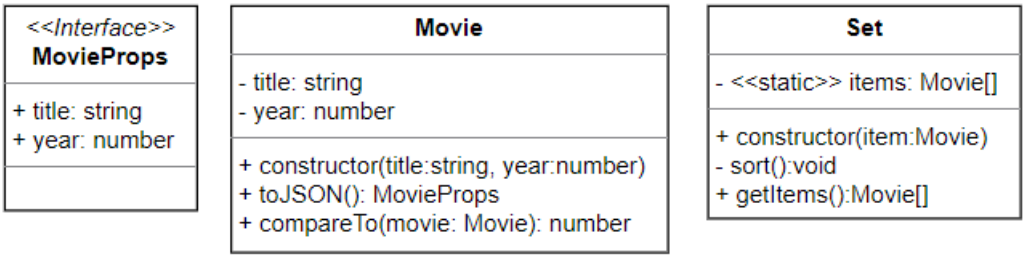
Instruções:

- A prova é individual e com consulta ao próprio material e internet. Não é permitido consultar material de outra pessoa da sala;
- Codificar todos os exercícios em uma única aplicação back-end;
- A prova encerra-se às 20h50. Ao terminar, você deverá chamar o professor para apresentar o código.

Exercício 1: (1 pt.) Ao final o projeto deverá ter a estrutura mostrada ao lado.

- As interfaces deverão estar no arquivo Types.ts.

Exercício 2: (3 pts.) Codificar as classes Movie e Set, e a interface MovieProps considerando os seguintes requisitos:



- Classe Movie:
  - constructor: inicializa as propriedades;
  - toJSON: retorna as propriedades no formato {title,year};

- compareTo: recebe um objeto do tipo Movie como parâmetro e retorna -1, 1, ou 0, se o title do objeto atual vem antes, depois ou igual ao title do objeto recebido como parâmetro. Dica: use o método localeCompare da classe String (https://www.w3schools.com/jsref/jsref_localecompare.asp).

- Classe Set:
  - constructor:
    - Adiciona o objeto recebido como parâmetro no array items;
    - Chama o método sort para ordenar os elementos do array items;
  - sort: ordena os elementos do array sort. Dica: use o método sort da classe Array (https://www.w3schools.com/jsref/jsref_sort.asp).
- Na função objetivo da rota /um deverá ser criado um objeto do tipo Set (usando como parâmetro no construtor um objeto Movie). A função objetivo retornará o conteúdo do array itens da classe Set.

Para cada vez que a rota é chamada será adicionado um elemento no array items da classe Set. Como resultado a rota /um retornará os elementos do array itens:

1ª vez que a rota é solicitada:

localhost:3013/um/Matrix/1999

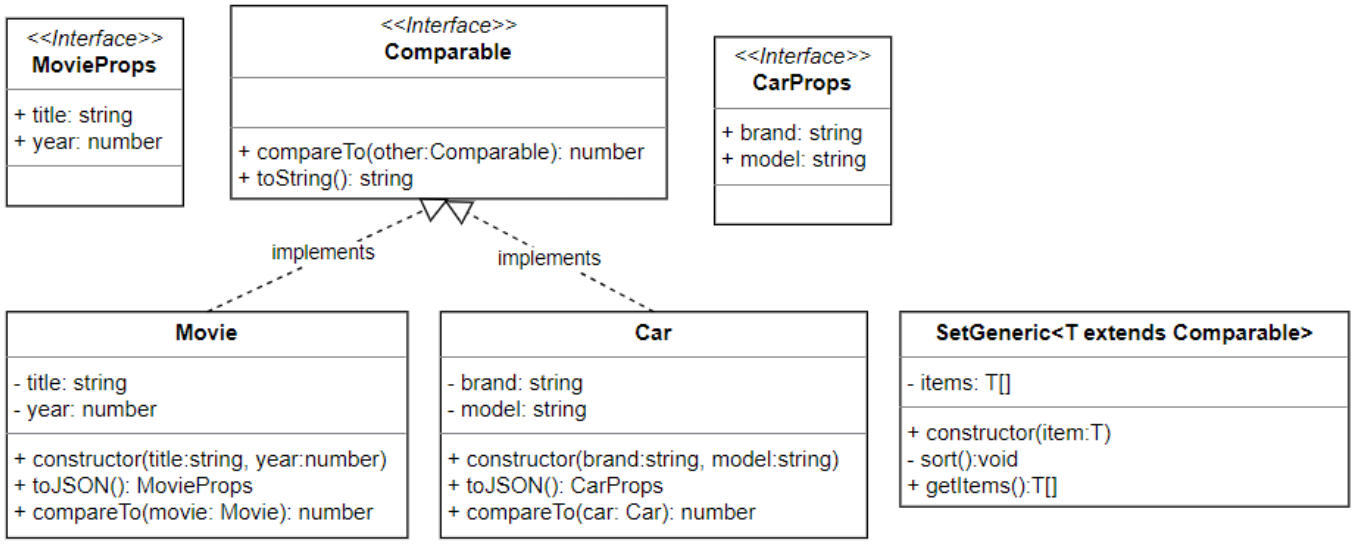
```
{"filmes":[{"title":"Matrix","year":1999}]}
```

2ª vez que a rota é solicitada:

localhost:3013/um/forrest gump/1994

```
{"filmes":[{"title":"forrest gump","year":1994},
           {"title":"Matrix","year":1999}]}
```

Exercício 3: (3 pts.) Codificar as classes Car e SetGeneric, e as interfaces CarProps e Comparable considerando os seguintes requisitos:

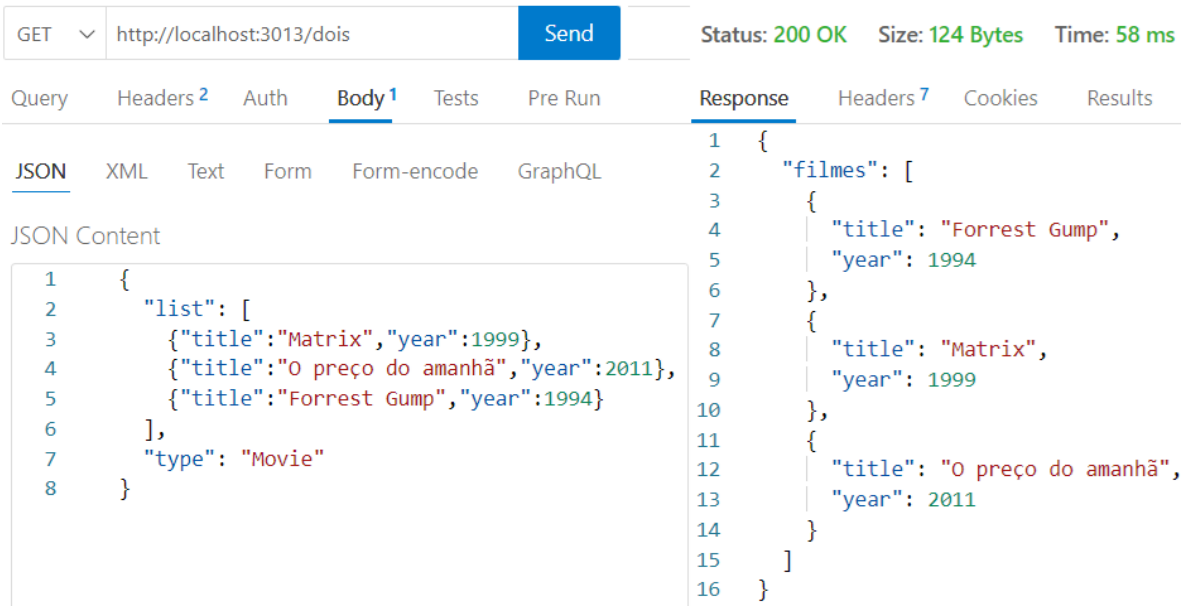


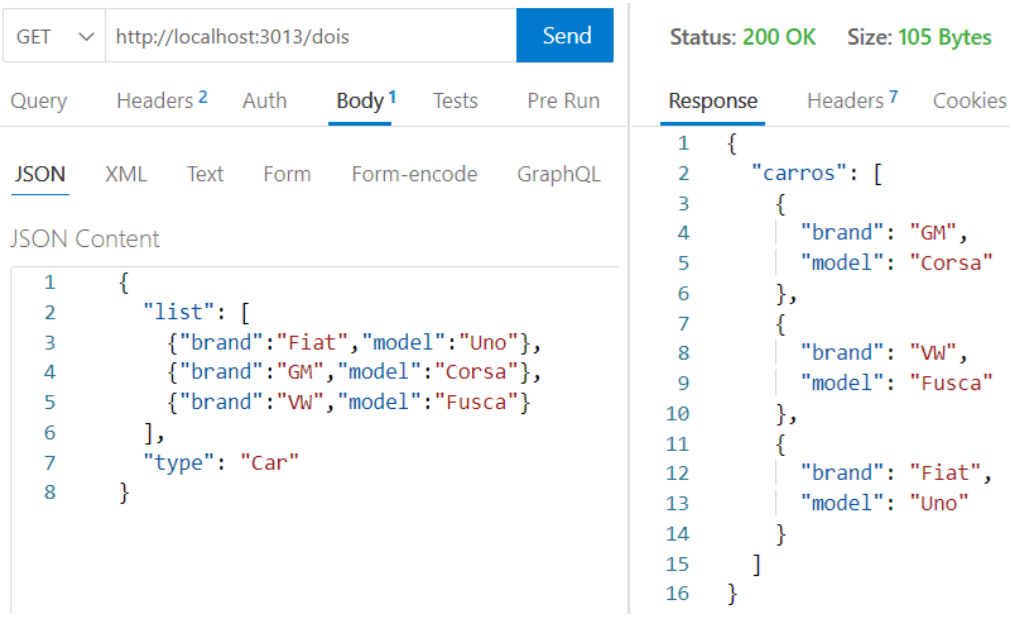
- Classe Movie deverá implementar a interface Comparable;

Prova III – Tipo C - Técnicas de programação I – DSM - Prof. Arley - 28/11/2023

- Classe Car deverá ter a mesma estrutura da classe Movie. O método compareTo deverá comparar usando a propriedade model;
- Classe SetGeneric é semelhante a classe Set com exceção do constructor:
  - constructor:
    - Coloca o array recebido como parâmetro na propriedade items;
    - Chama o método sort para ordenar os elementos do array items;
- Na função objetivo da rota /dois deverá ser criado um objeto do tipo SetGeneric passando um array de Movie ou Car.

Para cada vez que a rota é chamada deverá passado pelo body a lista de filmes ou carros e o tipo de dado dos elementos da lista (Movie ou Car). A rota retornará os filmes ou carros ordenados em ordem alfabética:





Exercício 4: (3 pts.) Alterar o construtor da classe SetGeneric para chamar o método append da classe File para salvar os dados recebidos no arquivo dados.txt.

A seguir tem-se o conteúdo do arquivo dados.txt após chamar a rota /dois nos dois exemplos anteriores:

```
Forrest Gump,1994
Matrix,1999
O preço do amanhã,2011
GM,Corsa
VW,Fusca
Fiat,Uno
```

A seguir tem-se o código do arquivo src/classes/File.ts. Será necessário instalar os pacotes fs-extra, @types/fs-extra e async-mutex.

```
import fs from "fs-extra";
import { Mutex } from "async-mutex";
import { Comparable } from "./Types";

export default class File {
  private static filename = "./src/classes/dados.txt";
  private static writeMutex = new Mutex();

  static async append(items: Comparable[]): Promise<void> | never {
    const release = await this.writeMutex.acquire();
    try {
      const texto = await fs.readFile(this.filename, "utf8");
      let lines: string[] = items.map( item => item.toString()+"\r\n" );
      await fs.writeFile(this.filename, texto + lines.join(""), "utf8");
    } catch (e:any) {
      throw new Error(e.message);
    } finally {
      release();
    }
  }
}
```