Instituto Superior de Engenharia de Lisboa
Licenciatura em Engenharia Informática e de Computadores
Modelação e Padrões de Desenho– Semestre de Verão – 2017/2018

Trabalho 1

Objectivos: Prática com funções de primeira classe, expressões lambda, funções de ordem superior, `Iterable<T>`

Data limite de entrega: **16 Abril de 2018**

NOTA:

1. A solução entregue deve incluir **todos os testes unitários necessários** para validar o correcto funcionamento das funcionalidades pedidas.
2. Este trabalho deve ser desenvolvido usando como base o projecto Gradle `movapi` disponibilizado em https://github.com/isel-leic-mpd/movapi.
   Copie toda a solução **incluindo o ficheiro .gitignore** para o repositório Github do grupo de MPD.
3. Cada trabalho será desenvolvido num novo módulo dentro do projecto `movapi`.

Implemente a biblioteca `movlazy` que disponibiliza informação detalhada sobre filmes. Os dados são obtidos a partir de uma API RESTful: https://api.themoviedb.org. O modelo de domínio é formado pelas entidades: `SearchItem`, `Movie`, `CastItem` e `Actor` e obedece à especificação apresentada no diagrama de classes da Figura 1.

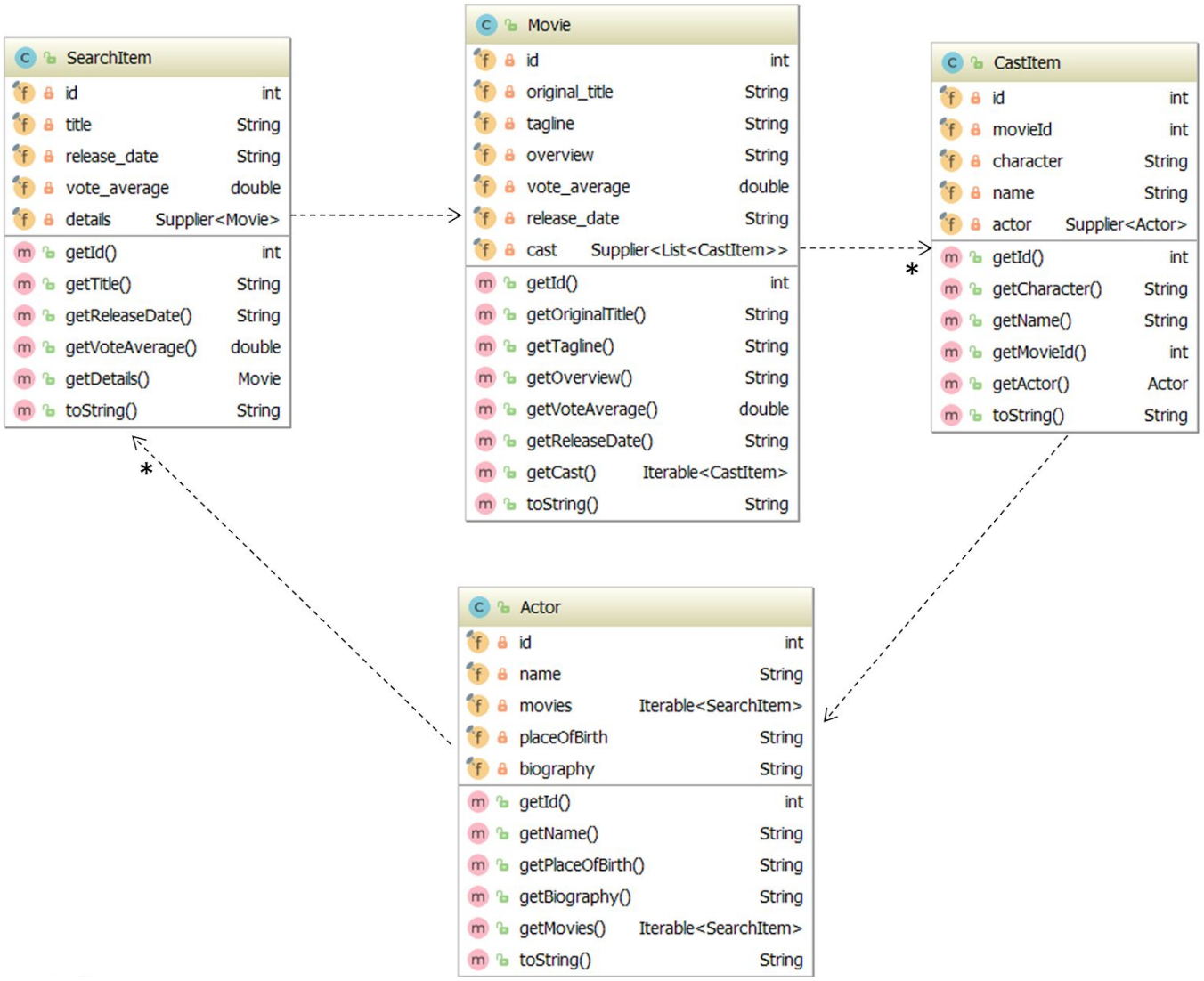
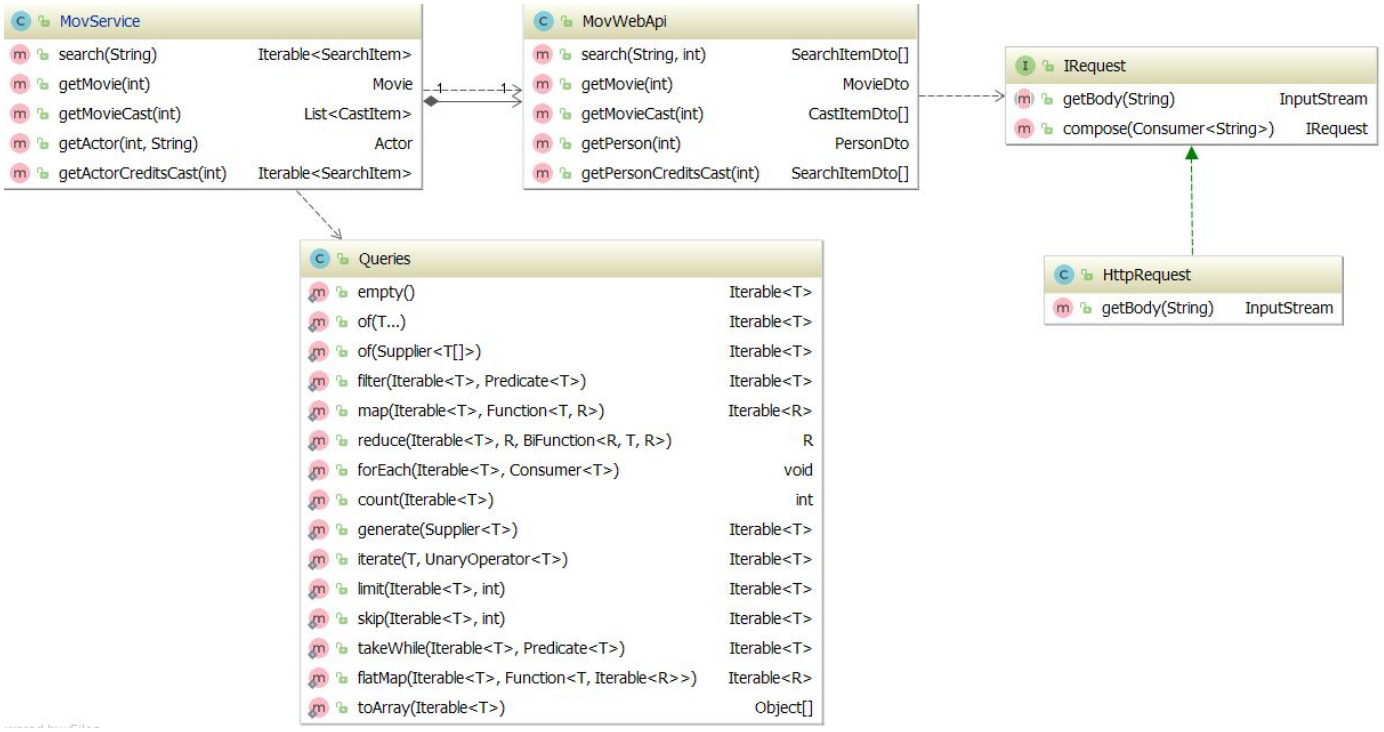


Figura 1

As classes do modelo de domínio estão implementadas no módulo `movlazy`. As relações entre as entidades de domínio são mantidas da seguinte forma:

- Todas as relações são *lazy*, ou seja o pedido à fonte de dados só é feito quando consultada a propriedade que fornece os objectos de domínio, e.g. `getDetails()`, `getCast()`, `getActor()` e `getMovies()`. Cada uma destas propriedades é suportada num campo de um tipo com semântica *lazy*, e.g. `Supplier` ou `Iterable`.
- As instâncias de `Movie`, `CastItem` e `Actor` devem ser mantidas num cache da instância de `MovService` cuja estrutura já está definida nos campos `movies`, `cast` e `actors`.
- Não faça cache das instâncias de `SearchItem`.

Neste trabalho deve completar a implementação de `MovService`, `MovWebApi`, `Queries::flatMap`, `Queries::takeWhile` e do package `movlazy.dto`, de modo a passar com sucesso todos os testes unitários.
O método `search()` de `MovService` é o único método que já se encontra implementado retornando um `Iterable` que faz pedidos de forma lazy de todas as páginas com os resultados da pesquisa de um filme, faltando a implementação dos métodos `flatMap()` e `takeWhile()` para o seu correcto funcionamento.



Além das classes e métodos fornecidos poderá ter que criar outros métodos e classes. Por exemplo, no caso do package `movlazy.dto` terá que adicionar classes em falta que seguem o modelo de dados fornecido pela API RESTful https://api.themoviedb.org.

Implemente uma nova classe de testes unitários semelhante a `MovServiceTestForWarGames`, mas para uma pesquisa de outro tema. Na implementação dessa classe de testes, **reutilize o máximo o código já existente** .

As informações da fonte de dados são obtidas a partir dos seguintes *endpoints* da API RESTful https://api.themoviedb.org:

- *https://developers.themoviedb.org/3/search/search-movies*
- https://developers.themoviedb.org/3/movies
- https://developers.themoviedb.org/3/movies/get-movie-credits
- *https://developers.themoviedb.org/3/people*

A API RESTful: https://api.themoviedb.org tem limite de pedidos. Para que não ultrapasse esse limite poderá controlar o número de pedidos através de bibliotecas auxiliares como **`RateLimiter`** do Guava.
Nos testes unitários MovServiceTestForWarGames é apresentado um exemplo de utilização de RateLimiter.

Os resultados da API RESTful podem ser convertidos através da biblioteca Gson para instâncias de classes pré-definidas (DTOs).