Gmail **Guilherme Bedin <guilherme0bedin@gmail.com>**

# #7DaysOfCode - Java 6/7: 👩‍💻 Criando abstrações

1 mensagem

**Paulo Silveira** <paulo@7daysofcode.io> 12 de junho de 2022 08:03
Responder a: reply-caelum57945.activehosted.1328.8171.46734@d32a.emsend3.com
Para: Guilherme Bedin <guilherme0bedin@gmail.com>


Olá Guilherme Bedin, tudo bem? Você está quase chegando no final do projeto, hein? Espero muito que esteja gostando de desenvolvê-lo, assim como eu!

Bora lá pra mais um desafio?

O seu código evoluiu bastante e, olhando para o método `main`, já dá para identificar 3 responsabilidades **principais**:

1. chamar a API
2. parsear o JSON
3. gerar o HTML

Provavelmente, você desenvolveu algo como:

```
public static void main(String[] args) throws
Exception {

    String apiKey = "<sua chave>";
    String json = new
ImdbApiClient(apiKey).getBody();
```

```
    List<Movie> movies = new
ImdbMovieJsonParser(json).parse();

    PrintWriter writer = new
PrintWriter("content.html");
    new HtmlGenerator(writer).generate(movies);
    writer.close();
}
```

Você usou a API do IMDB como exemplo para implementar o seu projeto, mas existem várias outras APIs por aí que também fornecem informações sobre filmes e séries, como, por exemplo, a API da Marvel.

Com a API da Marvel, você pode buscar séries e histórias em quadrinhos. Fique à vontade para analisar e testar a API, ela funciona de maneira similar à API do IMDB.

Voltando para o seu código e com a API da Marvel em mente, o seu desafio será que o **HTML seja gerado independentemente do conteúdo em questão** (seja ele um filme, uma série, uma história em quadrinhos ou outro).

Você deverá deixar o seu código mais genérico, ou seja, preparado para receber dados de outras APIs. Para isso, entram em cena as **interfaces**, que permitem implementações diferentes.

Então, vamos lá: o seu modelo deverá **implementar uma nova interface que irá definir o comportamento comum de um conteúdo**. Você pode chamá-la de `Content`, e ela poderá ter os seguintes métodos:

```
public interface Content {
    String title();
    String urlImage();
    String rating();
    String year();
```

```
}
```

E a sua classe (ou record) `Movie` se tornará um `Content`, dessa forma:

```
public class Movie implements Content {...}
```

Sendo assim, você também poderá pensar em uma abstração para o parser de JSON. Você pode criar uma interface chamada '`JsonParser`':

```
public interface JsonParser{
    public List<? extends Content> parse();
}
```

Repare que o método devolve uma lista que possui elementos do tipo `<? extends Content>`. Como o `Movie` implementa a interface `Content`, esse código vai funcionar!

A partir daí, você poderá usar a nova interface `JsonParser` na classe `ImdbMovieJsonParser`:

```
public class ImdbMovieJsonParser implements
JsonParser{
    //…
}
```

Resumindo, você criará duas abstrações: uma para o seu modelo chamado de `Content` e outra para o `JsonParser`. Basta que futuras implementações sigam essas interfaces e o seu gerador de HTML continuará funcionando! Ou seja, você desacoplou o parseamento do JSON da geração de HTML.

## EXERCÍCIO EXTRAS

Acha que acabou por aí? Que nada! Vamos brincar um pouco mais com Java:

1. Voltando para a motivação inicial, a API da Marvel, crie uma interface chamada '`APIClient`' com um método `getBody()`.

2. Tente consumir a API da Marvel criando um cliente da API e `JsonParser`. Aqui também, você precisará de uma conta para gerar a API Key (chave pública e privada). *Obs: Esse exercício é bastante trabalhoso, reserve um tempo para ele :)*

3. Adicione um novo método na interface chamado `type()`. Esse método irá devolver o tipo do conteúdo em questão, por exemplo: `Movie`, `Series`, `ComicBook`, etc.

## DICA

Sobre a abstração criada, `<? extends Content>`, você viu que, como o `Movie` implementa a interface `Content`, esse código funcionará. Mas o legal é que ele também funcionaria para uma possível classe `Series` (inglês para “série” ou “seriado”), desde que ela também implementasse a interface `Content`.

## EXTRA

Você pode ler mais sobre Orientação a Objetos e interfaces neste artigo fantástico do blog da Alura.

Como sempre, não deixe de compartilhar o seu código no seu GitHub e nas suas redes sociais com a hashtag **#7DaysOfCode**, e também com **#feedback7DoC** caso você queira alguma ajuda.

Bons estudos e até amanhã!

**Paulo Silveira**
CEO e fundador da Alura