Desafio 2:

Prólogo: Seu objetivo durante o desafio 2 deve ser entender o conceito de POO, entender cada um de seus pilares e também as possibilidades que eles nos proporcionam no desenvolvimento de API's. Além do entendimento desses conceitos, o foco deve estar em saber criar as suas próprias classes de entidade;

a) Com suas palavras defina o que é programação orientada a objetos (POO) e cite seus pilares? (máx 10 linhas)
b) Exemplifique e explique um cenário de abstração;
c) Exemplifique e explique um cenário de encapsulamento;
d) Exemplifique e explique um cenário de herança;
e) Exemplifique e explique um cenário de polimorfismo;
f) Cite 5 vantagens da POO;
g) Agora vamos imaginar um sistema de controle financeiro, onde teremos dois tipos de entidades, categoria e lançamento. Categoria, será nossa forma de agrupar os lançamentos (Exemplo de categorias: Lazer, Saúde, Alimentação, Salário, Freela, etc). O lançamento por sua vez poderá ter os seguintes estados, se é despesa ou receita, se está pago ou pendente, data do acontecimento e valor;

Entidade	Características	Ações
Categoria (Category)	id (long), name (string), description (string)	getId, setId, getName, setName, getDescription, setDescription, toString
Lançamento (Entry)	id (long), name (string), description (string), type (string), amount (string), date (string), paid (boolean), categoryId (long)	getId, setId, getName, setName, getDescription, setDescription, getType, setType, getAmount, setAmount, getDate, setDate, getPaid, isPaid, getCategoryId, setCategoryId, toString

```
// Exemplo de categoria
{
      id: 4,
      name: 'Salário',
      description: 'Recebimento de Salário'
}
```

```
// Exemplo de lançamento
{
      id: 3,
      name: 'Salário na Empresa X',
      description: 'Adiantamento quinzenal',
      type: "revenue",
      amount: "4405,49",
      date: "15/09/2021",
      paid: true,
```

```
      categoryId: 4
}
```

Hands-on:

i. Implemente uma classe para cada entidade com suas respectivas características e ações;
ii. Implemente 2 construtores para cada entidade, um vazio e outro completo;
iii. Implemente um método toString() para cada entidade;

Agora em tempo de execução:

iv. Crie um objeto do tipo categoria utilizando o construtor para popular;
v. Crie um objeto do tipo categoria, porém agora vazio e utilize os métodos set para popular o objeto;
vi. Realize um println com o método toString() para cada um dos objetos criados;
vii. Repita os passos 4, 5, 6 para lançamentos;

Dica: Entre no site https://start.spring.io/ e gere um projeto spring.

Group: trilha.back

Artifact/Name: financys

```
package trilha.back.finacys;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class FinacysApplication {

    public static void main(String[] args) {
        SpringApplication.run(FinacysApplication.class, args);

        // Insira seu código aqui
    }

}
```