

# Modelação e Padrões de Desenho

LEIRT 2024

## Enunciado do Trabalho 1

Data limite de entrega: 8 de abril

**Objectivos:** Revisões. Utilização elementar de padrões de desenho. Introspeção e Reflexão. Operações genéricas de pesquisa

### Notas:

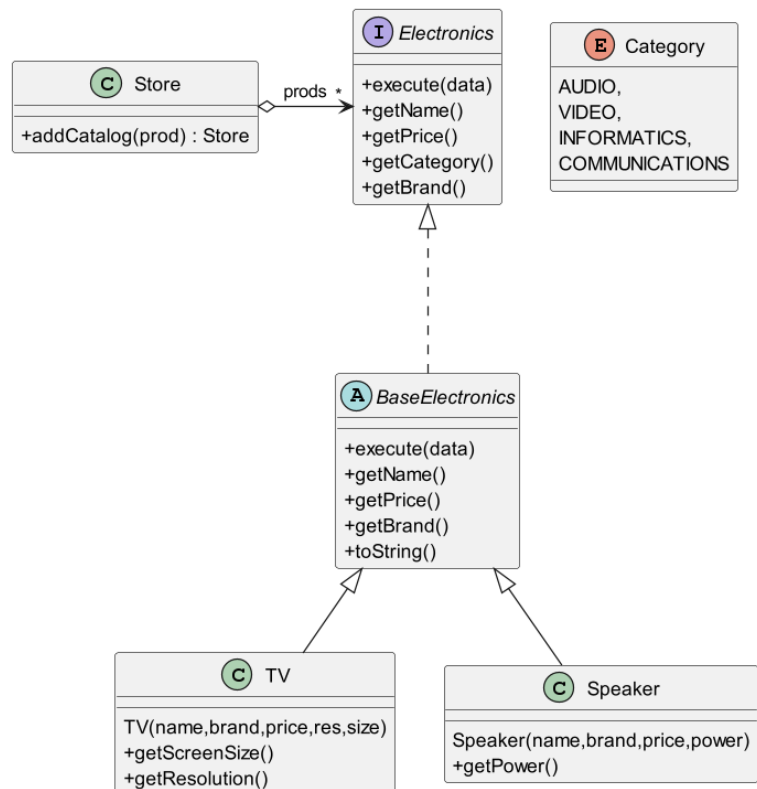
- A solução entregue deve incluir todos os testes unitários necessários para validar o correto funcionamento das funcionalidades pedidas, para além dos que já são fornecidos. As funcionalidades que não possam ser avaliadas com testes unitários serão demonstradas na discussão.
- Este, como todos os outros trabalhos, será desenvolvido num novo módulo dentro do vosso repositório. O repositório modelo associado a cada grupo já contém o módulo deste primeiro trabalho com algum código inicial, apresentado na hierarquia abaixo

## I Parte – Revisões e Padrões de Desenho

Pretende-se modelar uma loja de venda de produtos eletrónicos. Considerando a hierarquia de produtos definida no código inicial do trabalho:

### 1. Complete a hierarquia de produtos com as seguintes classes e funcionalidades:

- Redefina os métodos **toString** das classes **TV** e **Speaker** por forma a apresentar as características específicas de cada produto.
- Execute o teste **productsFromSamsungTests**. Justifique a razão da sua falha e corrija o necessário nas classes da hierarquia de produtos para que o teste tenha sucesso
- Acrescente a classe **SmartPhone**, da categoria **COMMUNICATIONS**, com os atributos adicionais **resolution** e **screenSize**, idênticos ao da classe **TV**, e ainda o atributo **batteryCapacity** com a capacidade da bateria em mAh.
- Acrescente a classe **Notebook**, da categoria **INFORMATICS**, também com os atributos adicionais **resolution** e **screenSize**, e ainda os atributos **batteryCapacity** com a capacidade da bateria em mAh e **usbPorts** com a quantidade de portas usb.
- Note que **Smartphone**, **TV** e **Notebook** possuem ecrã. Encontre uma forma de identificar produtos com ecrã, com os atributos resolução e dimensão, sem acrescentar métodos para esse efeito.
- Acrescente a classe **Promo** que implementa a interface **Electronics** e cujas instâncias colocam um produto existente em promoção, com desconto definido em percentagem.
- Acrescente a classe **Pack** que representa um artigo composto por vários produtos. As instâncias da classe **Pack** devem implementar a interface **Iterable<Electronics>** de modo a permitirem iterar sobre os seus produtos. Notem que pode haver instâncias de **Pack** que contenham *sub packs*.



2. Substitua na classe **Store** a coleção de produtos, que atualmente é um **ArrayList**, por um **SortedSet**, de modo a manter o catálogo ordenado por marca e nome de produto.
3. Implemente os métodos de pesquisa definidos na classe **Store**.
4. Acrescente o necessário à classe **Store** e à hierarquia de produtos eletrônicos de modo a permitir salvar (serializar) o catálogo numa **Stream** de texto em formato JSON e repor (desserializar) o catálogo a partir do conteúdo de uma **Stream** de texto nesse formato. Utilize a biblioteca `org.json` para auxiliar nessa conversão. A dependência para a biblioteca já está definida no projeto disponibilizado no vosso repositório de grupo.

## II Parte – introspeção e reflexão

5. Usando reflexão, generalize a serialização/desserialização em JSON para qualquer objeto arbitrário. Sugere-se criar a classe **ReflexUtils** com os seguintes métodos públicos:

```
void saveToFile(Object o, String fileName)
void saveToWriter(Object o, Writer w)
```

6. Crie a anotação `@Internal` para identificar campos que não devem ser usados na serialização/desserialização
7. Crie a anotação `@JsonName(String name)` para identificar campos que tenham nome diferente no formato JSON fonte/destino
8. Altere a implementação da questão 4 da Parte I para tirar partido da infraestrutura criada na 2ª parte

## III Parte – Métodos genéricos de pesquisa, transformação e redução de Iterables

9. Implemente na classe **QueryUtils** o método estático  
`<T,U> U reduce(Iterable<T> src, U initial, BiFunction<U,T,U> combiner)`  
Que reduz a um valor **U** o **Iterable<T>** passado por parâmetro usando a função **combiner**
10. Reimplemente os métodos de pesquisa definidas na classe **Store** utilizando unicamente os métodos genéricos **filter**, **map**, **flatMap** definidos nas aulas e o método **reduce** definido na questão anterior.