# Parte 1

Desenvolva uma pequena aplicação que permita armazenar informação relativa a um conjunto de bicicletas disponíveis para venda numa loja especializada. A loja comercializa essencialmente bicicletas de dois tipos: bicicletas de montanha e de carga/distribuição. As bicicletas possuem: identificador (`id`), número de velocidades (`numberOfGears`), cor principal (`mainColor`), diâmetro das rodas (`wellSize`) e comprimento (`bikeLenght`). Adicionalmente, podem possuir um assento regulável (`adjustableSeatpost`) e o preço (`price`). Para além dos dados já mencionados, cada tipo bicicleta possui características próprias:

- A **Bicicleta de montanha** possui um número de luzes (`numberofLights`), tipo de suspensão (`suspension`) (p.ex. suspensão simples, dupla, ou sem suspensão) e um conjunto de utensílios/acessórios (`bikeTools`) aplicável apenas a este tipo de bicicleta (p.ex. garrafa de água, kit para substituição do pneu, etc). A bicicleta de montanha não possui um limite de utensílios.
- A **Bicicleta de carga/distribuição** possui a localização (`isFrontBasket`), a capacidade do cesto (`basketDelivery`) e um conjunto de máximo de 10 patrocínios (`Sponsors`).

**Resolução parcial:**

```java
/**
 * Método construtor para a criação de uma instância de
 * {@link Biclycle bicicleta}.
 *
 * @param id Identificador da bicicleta
 * @param numberOfGears Número de velocidades
 * @param mainColor Cor da bicicleta
 * @param weelSize Diâmetro das rodas
 * @param bikeLenght Comprimento da bicicleta
 * @param adjustableSeatpost Saber se assento é ajustável
 * @param price Preço da bicicleta
 */
public Bicycle(int id, int numberOfGears, String mainColor, float weelSize,
        float bikeLenght, boolean adjustableSeatpost, float price) {
    this.id = id;
    this.numberOfGears = numberOfGears;
    this.mainColor = mainColor;
    this.weelSize = weelSize;
    this.bikeLenght = bikeLenght;
    this.adjustableSeatpost = adjustableSeatpost;
    this.price = price;
}

/**
 * Retorna o {@link Bicycle#id id} de uma bicicleta
 * @return
 */
public int getId() {
    return id;
}
```

Método construtor e excerto dos métodos de acesso da classe `Bicycle`

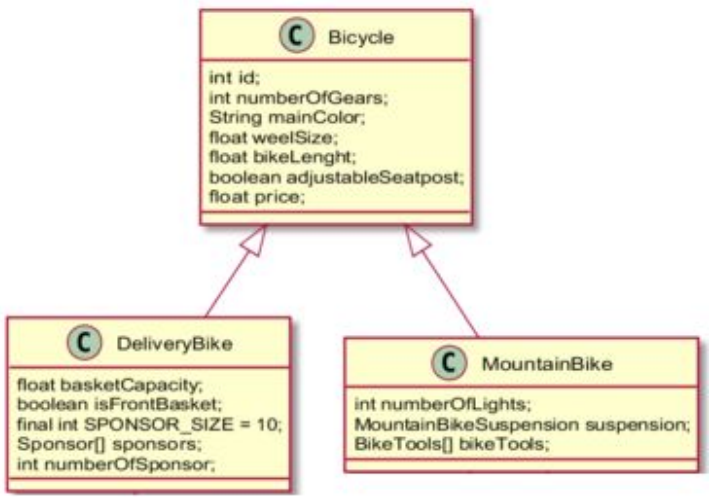
```java
public class MountainBike extends Bicycle{
    /**
     * Número de luzes
     */
    private int numberOfLights;
     /**
     * Tipo de suspensão
     */
    private MountainBikeSuspension suspension;
     /**
     * Utensílios existentes neste tipo de bicicleta
     */
    private BikeTools[] bikeTools;

    /**
     * Método construtor para a criação de uma instância de
     * {@link MountainBike bicicleta de montanha} sem utensílios.
     *
     * @param numberOfLights Número de luzes
     * @param suspension Tipo de suspensão
     * @param id Identificador da bicicleta
     * @param numberOfGears Número de velocidades
     * @param mainColor Cor da bicicleta
     * @param weelSize Diâmetro das rodas
     * @param bikeLenght Comprimento da bicicleta
     * @param adjustableSeatpost Saber se assento é ajustável
     * @param price Preço da bicicleta
     */
    public MountainBike(int numberOfLights, MountainBikeSuspension suspension,
            int id, int numberOfGears, String mainColor, float weelSize,
            float bikeLenght, boolean adjustableSeatpost, float price) {
        super(id, numberOfGears, mainColor, weelSize, bikeLenght, adjustableSeatpost,
                price);
        this.numberOfLights = numberOfLights;
        this.suspension = suspension;
    }
```

Excerto da classe `MountainBike`

## Exercício 1

1.1. Num projeto `pp_fp07`, com um package: `pp_fp07.bikeStore`, implemente o código Java necessário para representar a estrutura descrita. Tenha por referência o seguinte excerto da estrutura de classes:




Excerto da estrutura para representação da informação (classe - C e enumeração - E).

Considere que:

- Deve garantir o encapsulamento de todas as classes criadas;
- Deve criar métodos de acesso necessários para todas as classes criadas;
- Num *package* específico, deve criar as enumerações necessárias para suportar o problema apresentado.
- Para as coleções de ferramentas (BikeTool) e patrocinadores (Sponsors), deverá criar métodos que permitam o acesso controlado a cada uma das coleções. Por exemplo, deverá criar métodos de **inserção, edição, remoção e listagem** de elementos.

1.2. Crie a classe `BikeDemo` e teste as classes implementadas, inicializando alguns elementos relativos a bicicletas de montanha e bicicletas e de carga/distribuição.

1.3. Crie uma classe `BicycleManagment` que armazene um vetor de bicicletas (`Bicycle[]`). Crie um método para adicionar bicicletas ao vetor até a um máximo de 20. De seguida e utilizando a classe `BikeDemo`, teste a classe criada, adicionando as bicicletas que criou na alínea 1.1. Crie ainda um método na classe `BicycleManagment`, que imprima todos os dados das bicicletas contidas no vetor que criou.

**Gere o JavaDoc para o projecto utilizado na resolução desta ficha de trabalho.**

# Parte 2

Considere como base para a resolução da parte 2, o exemplo da Pizzaria apresentado na ficha prática 6.

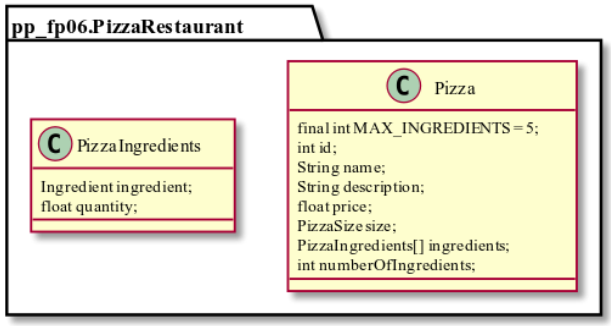
## Exercício 1

1.1. Considere que cada ingrediente de uma pizza é identificado pelo seu código, nome, unidade de medida (por exemplo: Gramas, Litros ou Unidades) e o número de calorias associadas. No entanto, é necessário que uma pizza possua (pelo menos) dois tipos de ingredientes:

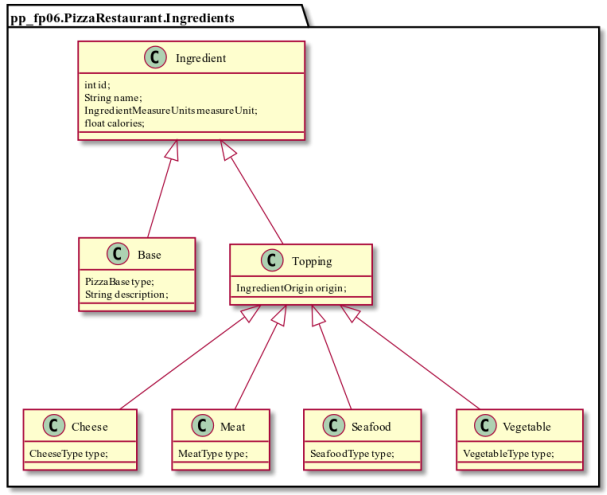
- Base, que descreve o tipo de massa utilizada (Massa alta ou massa fina);
- Cobertura da pizza que engloba alguns tipos de ingredientes e os seus tipos:
    - Queijo (Mozzarela, Serra, etc);
    - Carne (Porco, Vaca, Salame, etc);
    - Vegetal (Tomate, Cebola, Cogumelos, etc);
    - Frutos do mar(Camarão, Lagosta, etc).

Como os clientes da pizzaria são muito exigentes em relações aos produtos que são utilizados para confecionar a pizza, qualquer tipo de ingrediente que faça parte da cobertura da `Pizza` possui um atributo que permite identificar a sua origem (nacional ou importado).

Implemente as alterações necessárias de modo a que possa refletir no exercício da ficha prática 6 todos os requisitos apresentados. Tenha por base o seguinte diagrama para o auxiliar na resolução da exercício:




Excerto da estrutura para representação das classes `Pizza` e `PizzaIngredients`




Excerto da estrutura para representação das classes e respetivas subclasses

**Resolução parcial:**

```java
public class Meat extends Topping {
    private MeatType type;

    /**
     * Método construtor para a criação de uma instância de {@link Meat meat}
     *
     * @param id Identificação de um {@link Ingredient ingrediente}
     * @param name Nome do ingrediente {@link Ingredient ingrediente}
     * @param measureUnit {@link IngredientMeasureUnits Unidade}
     * de medida do {@link Ingredient ingrediente}
     * @param calories Número de calorias associado a um
     * {@link Ingredient ingrediente}
     * @param type Tipo de {@link MeatType carne}
     * @param origin {@link IngredientOrigin Origem} da carne
     */
    public Meat(int id, String name, IngredientMeasureUnits measureUnit,
            float calories, MeatType type, IngredientOrigin origin) {
        super(origin, id, name, measureUnit, calories);
        this.type = type;
    }

    /**...3 lines */
    public MeatType getType() {...3 lines }

    /**...3 lines */
    public void setType(MeatType type) {...3 lines }

}
```

**Figura 2:** Excerto da classe `Meat`

1.2. Realize as alterações necessárias para que a unidade de medida por defeito dos ingredientes do tipo

`PizzaBase` seja obrigatoriamente gramas.

1.3. Altere a classe `PizzaDemo` de forma a testar as alterações realizadas. Crie pelo menos um ingrediente de cada tipo.

1.4. Na classe `Pizza`, adicione/altere métodos que permitam:

- Que apenas seja possível adicionar ingredientes do tipo `Topping` quando a pizza já possuir um ingrediente do tipo `Base`;
- Não deverá ser possível ter mais do que um ingrediente do tipo `Base`;
- No máximo, a pizza deverá possuir 5 ingredientes do tipo `Topping`;
- Defina um tipo da pizza tendo por base dos ingredientes que esta possui, considerando a seguinte classificação:
    - Pizza de carne: Contém apenas ingredientes do tipo `Meat`.
    - Pizza do mar: Apenas ingredientes do tipo `Seadfood`;
    - Pizza vegetariana: Apenas ingredientes do tipo `Vegetable`;

Teste as alterações na classe `PizzaDemo`. Crie um método para imprimir o conteúdo (todos os atributos) de todos os ingredientes de uma `pizza`.

**Gere o JavaDoc para o projecto utilizado na resolução desta ficha de trabalho.**