P.PORTO ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

IPP – Instituto Politécnico do Porto

LEI/LSIRC

PP – Paradigmas de Programação

2º Semestre

Ficha Prática 7

Sumário
- Herança

Documentação complementar Java:
- Inheritance

Nota: A resolução desta ficha prática pressupõe a utilização da ferramenta git para ajudar na gestão de versões do software desenvolvido.
Documentação complementar Git:
- learngitbranching
- gitexplorer

Parte 1

Desenvolva uma API que permita gerir informação relativa a um conjunto de bicicletas produzidas por uma empresa especializada. A empresa produz e comercializa essencialmente bicicletas de dois tipos: bicicletas de montanha e de estrada. De forma forma geral as bicicletas são caracterizadas por: identificador (`id`), número de velocidades (`numberOfGears`), cor principal (`mainColor`), diâmetro das rodas (`wellSize`), o tipo de travões (`brakes`) (pinças identificados internamente como "P" ou hidraúlicos identificados internamente como "H") e o `material` em que foram construídas (Carbono ou Alumínio). Além disso, é importante indicar o preço (`price`) e os anos de garantia (`guarantee`) definidos pela empresa para a bicicleta. 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 possui um limite de 10 utensílios.
- A **Bicicleta de estrada** possui a identificação das fitas utilizadas no guiador (`handlebelt`), o tamanho do quadro (`frameSize`) e um conjunto de observações (texto livre) que o cliente poderá indicar para a construção da bicicleta. Além disso, a bicicleta de estrada é construída por defeito em carbono e com travões hidráulicos.

O excerto de código para suporte à resolução da ficha prática encontra-se alojado num repositório do GitHub. No **repositório** pode encontrar um excerto da resolução do problema apresentado. Para o utilizar devidamente, deverá seguir as instruções apresentadas na ficha prática anterior.

Exercício 1

1.1. Num projeto `pp_fp07`, com um package: `pp_fp07.bikeStore`, implemente o código Java necessário para representar a estrutura descrita anteriormente.

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) das bicicletas de montanha e conjunto de observações das bicicletas de estada, 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 retorne um representação textual de todos os dados das bicicletas contidas no vetor que criou.

1.4. Documente o código desenvolvido (analise os exemplos javadoc fornecidos nas classes de exemplo).

1.5. Submeta uma nova versão do seu projeto no repositório de acordo com as instruções indicadas anteriormente.

1.6. Crie um repositório no GitHub (ou qualquer outro host git à sua escolha) e armazene o código produzido no repositório criado. Para isso:

- `git remote add origin <link do repositório>` (para associar o repositório do GitHub criado com o seu repositório local)
- `git push -u origin master` (para submeter os commits do repositório local no repositório remoto)

Parte 2

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

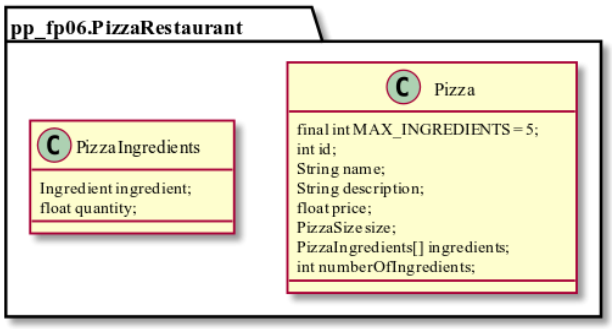
Exercício 1

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:

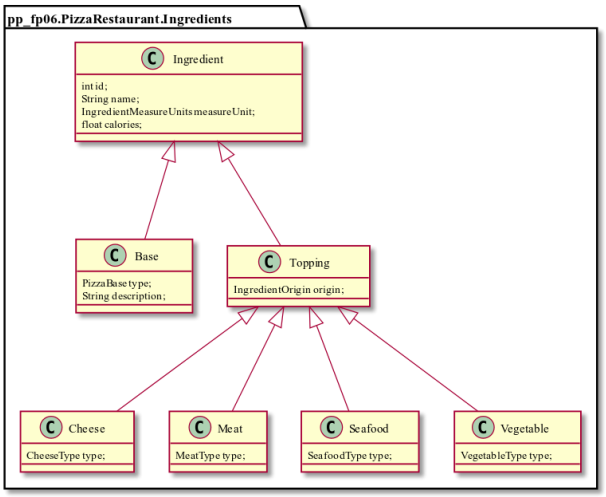
- Base, que descreve o tipo de massa utilizada (Massa alta ou massa fina);
- Cobertura da pizza que engloba alguns tipos de ingredientes e os seus tipos:
    - Queijo (Mozzarela, Serra, etc);
    - Carne (Porco, Vaca, Salame, etc);
    - Vegetal (Tomate, Cebola, Cogumelos, etc);
    - Frutos do mar(Camarão,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

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`.

1.5. Documente o código desenvolvido e gere o javadoc para o projeto.

1.6. Submeta uma nova versão do seu projeto no repositório remoto de acordo com as instruções indicadas anteriormente.