Exercícios POO

6 Desenvolvimento da aplicação *SuperMercado*

6.1 Descrição do problema

O objectivo deste exercício é o desenvolvimento de uma aplicação que permita gerir um conjunto de *carrinhos* e *artigos* num sistema de gestão de compras de um supermercado. Pretende-se que a aplicação suporte as compras que usam o serviço de entrega a casa. Cada carrinho está equipado com um computador de bordo que permite ao cliente saber em qualquer momento que artigos estão no carrinho (e como estão organizados), estando todos os carrinhos do supermercado ligados a uma central de pagamentos.

Apresentam-se de seguida as características dos vários elementos envolvidos na aplicação. Cada carrinho tem associado um *identificador* (único) e uma *capacidade* máxima. Cada artigo tem associado um *nome* único, um volume e um *preço*. Um carrinho pode conter vários artigos, mas a sua capacidade nunca deve ser excedida. Os carrinhos podem ser reutilizados à medida que vão sendo pagos.

A aplicação deve então permitir:

1. Criar um novo carrinho no sistema (comando `CRIA CARRINHO`). São fornecidos o identificador do carrinho e a sua capacidade (volume). Em caso de sucesso o carrinho é adicionado (Carrinho criado com sucesso.). A operação falha se já existe um carrinho com o identificador dado (Carrinho existente!).

2. Criar um novo artigo no sistema (comando `CRIA ARTIGO`). São fornecidos o nome, o preço e o volume do artigo. Em caso de sucesso o artigo é adicionado (Artigo criado com sucesso.). A operação falha se já existe um artigo com o identificador dado (Artigo existente!).

3. Adicionar um artigo a um carrinho (comando `DEPOSITA`). São fornecidos o nome do artigo e o identificador do carrinho. Considere que é possível adicionar vários artigos semelhantes a um carrinho. Em caso de sucesso o artigo é adicionado ao carrinho (Artigo adicionado com sucesso.). A operação falha se: (1) não existe um carrinho com o identificador dado (Carrinho inexistente!); ou (2) não existe um artigo com o identificador dado (Artigo inexistente!); ou (3) o carrinho está demasiado cheio para o artigo caber (Capacidade excedida!).

4. Remover artigo do carrinho (comando `REMOVE`). São fornecidos o nome do artigo e identificador do carrinho. Considere que é possível remover o mesmo tipo de artigo várias vezes de um carrinho (ou seja, pode remover vários artigos semelhantes do carrinho - por exemplo, pode remover sucessivamente 3 pacotes de leite, um de cada vez). Em caso de sucesso o artigo é removido do carrinho (Artigo removido com sucesso.). A operação falha se: (1) não existe um carrinho com o identificador dado (Carrinho inexistente!); ou (2) não existe um artigo com o identificador dado no carrinho (Artigo inexistente no carrinho!).

5. Listar o conteúdo de um carrinho (comando `LISTA`). É fornecido o identificador do carrinho. Em caso de sucesso a listagem deve mostrar o conteúdo do carrinho, nomeadamente os nomes e preços de todos os artigos contidos no saco. Os artigos devem ser listados por ordem de inserção. Se por acaso o carrinho estiver vazio, deve mostrar a mensagem correspondente (Carrinho vazio!). A operação falha se não existir um carrinho com o identificador dado (Carrinho inexistente!).

6. Pagar um carrinho (comando `PAGA`). É fornecido o identificador do carrinho. Em caso de sucesso a operação deve indicar o preço total do conteúdo do carrinho, ou seja, a soma dos preços de todos os artigos do carrinho e deve também esvaziar o carrinho, que pode então ser reutilizado. A operação falha se: (1) não existe um carrinho com o identificador dado (Carrinho inexistente!); ou (2) se o carrinho está vazio (Carrinho vazio!).

7. Sair da aplicação (comando `SAIR`). A operação tem sempre sucesso.

6.2 Exemplo de interacção com a aplicação

Desenvolva a sua aplicação para que esta garanta o modelo de interacção ilustrado no exemplo seguinte:

```
CRIA CARRINHO carro1 20↵
Carrinho criado com sucesso.↵
↵
CRIA CARRINHO carro1 15↵
Carrinho existente!↵
↵
CRIA CARRINHO carro2 15↵
Carrinho criado com sucesso.↵
↵
CRIA ARTIGO arroz 108 2↵
Artigo criado com sucesso.↵
↵
CRIA ARTIGO arroz 70 2↵
Artigo existente!↵
↵
CRIA ARTIGO papelhigienico 1299 10↵
Artigo criado com sucesso.↵
↵
CRIA ARTIGO ervilhas 69 1↵
Artigo criado com sucesso.↵
↵
DEPOSITA arroz carro1↵
Artigo adicionado com sucesso.↵
↵
DEPOSITA arroz carro1↵
Artigo adicionado com sucesso.↵
↵
DEPOSITA ervilhas carro1↵
Artigo adicionado com sucesso.↵
↵
DEPOSITA arroz carro1↵
Artigo adicionado com sucesso.↵
↵
```

```
DEPOSITA meias carro1↵
Artigo inexistente!↵
↵
DEPOSITA papelhigienico carro1↵
Artigo adicionado com sucesso.↵
↵
DEPOSITA papelhigienico carro1↵
Capacidade excedida!↵
↵
DEPOSITA ervilhas carro1↵
Artigo adicionado com sucesso.↵
↵
REMOVE ervilhas carro1↵
Artigo removido com sucesso.↵
↵
REMOVE playstation carro1↵
Artigo inexistente no carrinho!↵
↵
DEPOSITA meias carro2↵
Artigo inexistente!↵
↵
DEPOSITA ervilhas carro2↵
Artigo adicionado com sucesso.↵
↵
DEPOSITA arroz carro2↵
Artigo adicionado com sucesso.↵
↵
DEPOSITA ervilhas carro3↵
Carrinho inexistente!↵
↵
LISTA carro1↵
arroz 108↵
arroz 108↵
arroz 108↵
papelhigienico 1299↵
ervilhas 69↵
↵
LISTA carro2↵
ervilhas 69↵
arroz 108↵
↵
LISTA carro3↵
Carrinho inexistente!↵
↵
PAGA carro1↵
1692↵
↵
LISTA carro1↵
Carrinho vazio!↵
↵
LISTA carro2↵
ervilhas 69↵
arroz 108↵
↵
SAIR↵
Volte sempre.↵
↵
```

6.3 Desenvolvimento

Desenvolva a sua aplicação de acordo com as seguintes fases:

1. Numa primeira fase, desenvolva o(s) interface(s) de suporte à aplicação *SuperMercado*. Recorde que o uso de tipos genéricos pode ajudar a optimizar o código da sua aplicação. Desenhe o diagrama de classes e interfaces da aplicação.

2. Numa segunda fase implemente e teste a aplicação *SuperMercado*.

3. Submeta o código fonte da aplicação ao *Mooshak*.

Ficheiros de testes

Os testes do Mooshak verificam de forma incremental a implementação dos vários comandos:

- Ficheiro de teste: `teste1_CRIA_CARRINHO` − `in.txt` (5 pontos)
  Comandos testados: `CRIA CARRINHO, SAIR`
  Contexto: são testadas as condições onde os comandos têm sucesso e onde os comandos falham.

- Ficheiro de teste: `teste2_CRIA_ARTIGO` − `in.txt` (5 pontos)
  Comandos testados: `CRIA ARTIGO, SAIR`
  Contexto: são testadas as condições onde os comandos têm sucesso e onde os comandos falham.

- Ficheiro de teste: `teste3_DEPOSITA` − `in.txt` (10 pontos)
  Comandos testados: todos os comandos dos testes 1 e 2 e o comando `DEPOSITA`
  Contexto: são testadas as condições onde o comando `DEPOSITA` tem sucesso.

- Ficheiro de teste: `teste4_DEPOSITA` − `pre` − `in.txt` (10 pontos)
  Comandos testados: todos os comandos dos testes 1 a 3.
  Contexto: são testadas as condições onde o comando `DEPOSITA` tem sucesso e também as onde falha.

- Ficheiro de teste: `teste5_REMOVE` − `in.txt` (10 pontos)
  Comandos testados: todos os comandos dos testes 1 a 4.
  Contexto: são testadas as condições onde o comando `REMOVE` tem sucesso.

- Ficheiro de teste: `teste6_REMOVE` − `pre` − `in.txt` (10 pontos)
  Comandos testados: todos os comandos dos testes 1 a 5.
  Contexto: são testadas as condições onde o comando `REMOVE` tem sucesso e também as onde falha.

- Ficheiro de teste: `teste7_LISTA` − `in.txt` (10 pontos)
  Comandos testados: todos os comandos dos testes 1 a 6.
  Contexto: são testadas as condições onde o comando `LISTA` tem sucesso.

- Ficheiro de teste: `teste8_REMOVE` − `pre` − `in.txt` (10 pontos)
  Comandos testados: todos os comandos dos testes 1 a 7.
  Contexto: são testadas as condições onde o comando `LISTA` tem sucesso e também as onde falha.

- Ficheiro de teste: `teste9_PAGA` − `in.txt` (10 pontos)
  Comandos testados: todos os comandos dos testes 1 a 8.
  Contexto: são testadas as condições onde o comando `PAGA` tem sucesso.

- Ficheiro de teste: `teste10_PAGA` − `pre` − `in.txt` (10 pontos)
  Comandos testados: todos os comandos dos testes 1 a 9.
  Contexto: são testadas as condições onde o comando `PAGA` tem sucesso e também as onde falha.

- Ficheiro de teste: `enunciado` − `in.txt` (10 pontos)
  Comandos testados: todos.
  Contexto: testa tudo como no enunciado.