PROJETO PRÁTICO DE AEDA

Tema 2 - Transações Online (Parte 1)

Turma 4 - Grupo 3:
João Marinho (up201905952@fe.up.pt)
Miguel Rodrigues (up201906042@fe.up.pt)
Tiago Silva (up201906045@fe.up.pt)

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

DESCRIÇÃO DO PROBLEMA

Aspetos centrais sobre o funcionamento de uma loja *Online*.

IMPLEMENTAÇÃO DE UMA LOJA ONLINE

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

- A empresa *BuyNow de*cidiu implementar um sistema para a venda dos seus produtos online:
  - Necessária informação dos clientes e o registo das transações efetuadas através deste sistema.

- O cliente é caraterizado pelo seu nome e pelo seu NIF
  - O cliente tem a opção de se tornar um cliente registrado, onde além de ser caraterizado pelos atríbutos acima, este é identificado com o seu e-mail pessoal:
  - Ao cliente registrado é atríbuido um ID, estes devem também ficar armazenados no sistema para que possam realizar mais compras futuramente.

- As transações são caraterizadas por uma data, um carrinho de produtos e o seu valor.
- Para o pagamento das transações existem 3 métodos:
  - *MBWay*, onde o deve ser fornecido o número de telemóvel do cliente;
  - Multibanco, onde deve ser gerada uma referência para o pagamento;
  - Cartão de Crédito, onde deve ser guardado o número de cartão usado durante o pagamento;
- Este sistema deve manter o acompanhamento dos Stocks dos produtos:
  - Permitir a reposição de stock da loja *online* a partir das lojas físicas (se possível);
  - Caso, o ponto acima não seja exequível, então reabastecer os stocks através do fornecedor.

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

DESCRIÇÃO DA SOLUÇÃO

Contornos do problema e algoritmos relevantes.

CONTORNOS DO PROBLEMA

- Como funciona uma loja online?
  - Quais os processos e os mecanismos que dão forma às lojas online nos dias de hoje?
  - Como é que as empresas usam as lojas online para aumentar as suas vendas e reduzir custos?

- Implementação de Classes
  - Classes devem representar cada mecanismo ou ferramenta encontrados a partir dos pontos acima;
  - Leitura cuidada da descrição do problema permite que se comece o desenho da solução.

- Testar a Solução
  - Garantir que a solução encontrada se encontra dentro da especificação proposta;
  - Trabalhar com a escalabilidade do sistema em mente para uma implementação mais simples de possíveis futuras novas funcionalidades.

- No entanto este processo é iterativo, sendo crucial que se repita várias vezes, no caso de haver aspetos menos bem concebidos.

SOLUÇÃO SIMPLIFICADA

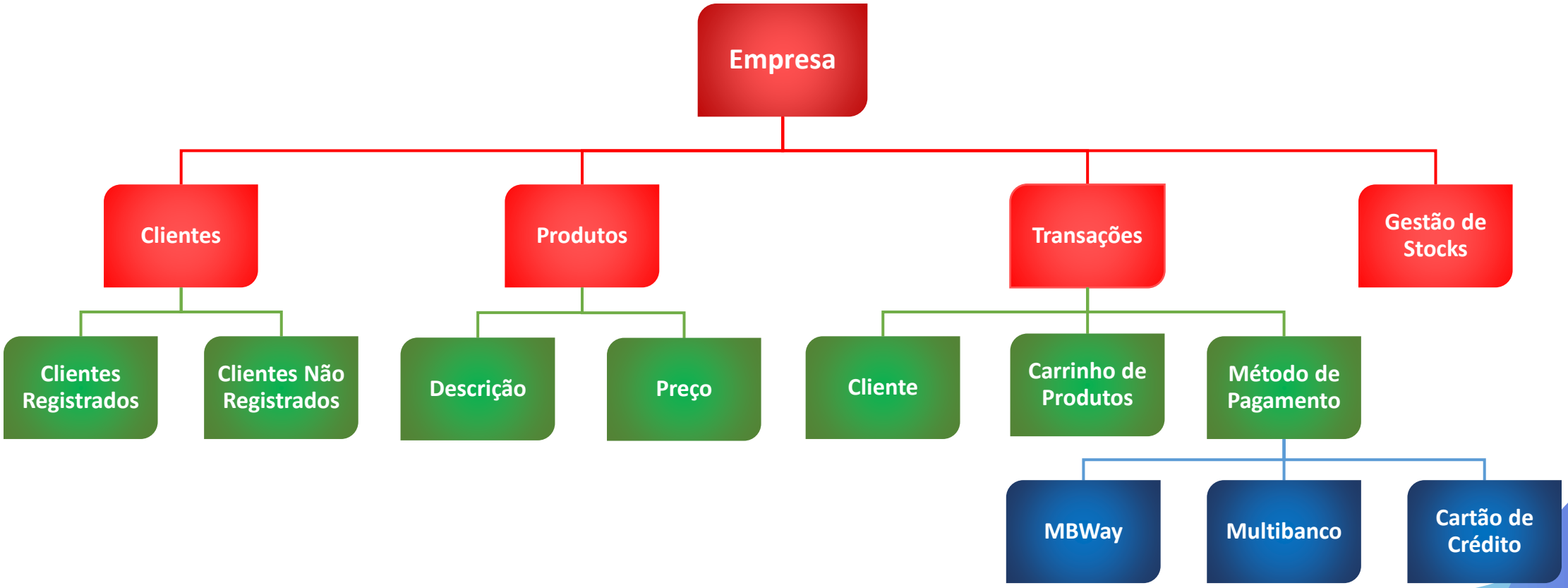

Figura 1 - Rascunho de uma possível solução.

IMPORTÂNCIA DOS ALGORITMOS

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

- Algoritmos de Ordenação:
  - **std::sort()** com o uso do *Introsort*, este algoritmo é um mistura do *Quicksort* com o *Heapsort* de forma a evitar o pior cenário de complexidade do *Quicksort* - $O(n^2)$;
  - A complexidade temporal média do **std::sort()** é $O(n \log(n))$;
  - A complexidade espacial é constante para o **std::sort()** é $O(1)$, visto que esta função recebe como parâmetros do tipo *RandomAcessIterators*, permintindo uma passagem de valores por referência.

- Algoritmos de Pesquisa:
  - Ambos os algoritmos tem complexidade espacial constante, ou seja, $O(1)$;
  - *Sequential Search* com complexidade temporal $O(n)$:
    - Usada para procura em estruturas de dados lineares de pequena dimensão, desordenadas. Este algoritmo pode ser encontrado no código sob a forma de **std::find_if()** ou implementado à base de *for's* e/ou *for range based loops*.
  - *Binary Search* com complexidade temporal $O(\log n)$:
    - Usada para procura em estruturas de dados ordenadas. Este algoritmo encontra-se no código implementada, por exemplo, em **OnlineStore::findClient()** bem como a implementação do respetivo *operator* < para o tipo de variável em causa.

- Fonte: (cppreference.com, 2020)

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

DIAGRAMAS DE CLASSES

Atributos e métodos das diferentes classes e subclasses.

Classe *Company*

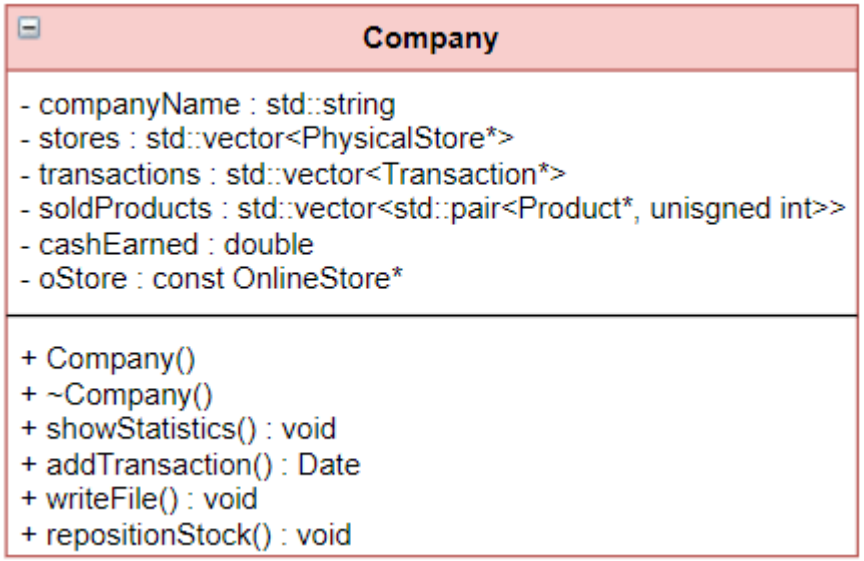

Figura 2 - Diagrama da classe *Company*

Classe *Store*

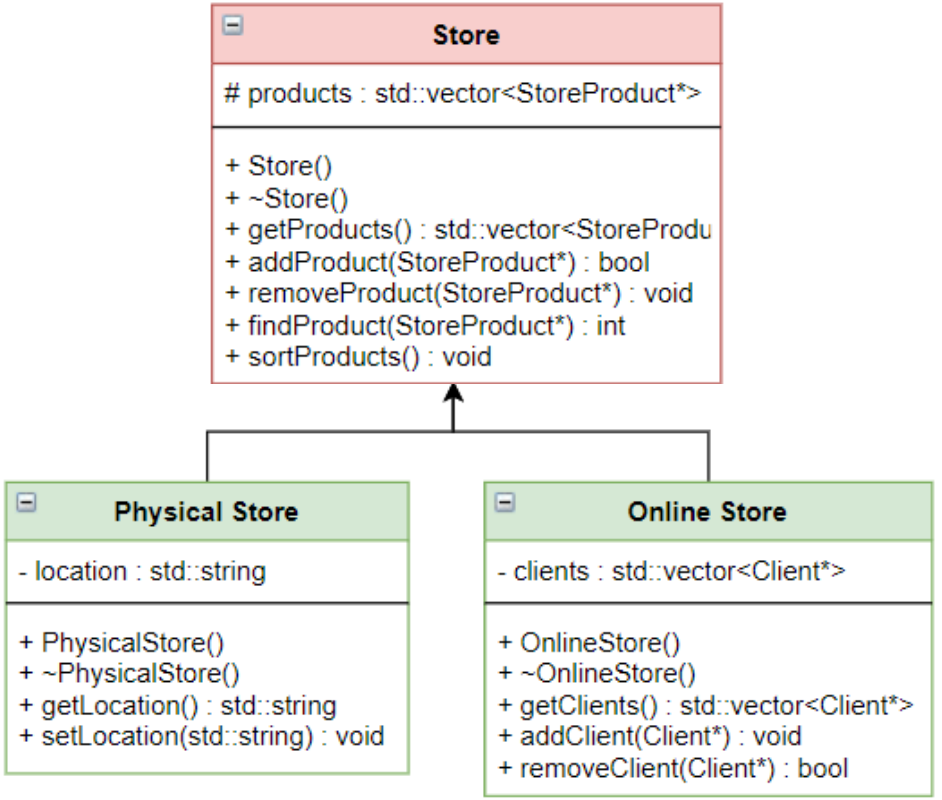

Figura 3 - Diagrama da classe *Store* e respetivas subclasses.

Classes *Product* e *StoreProduct*

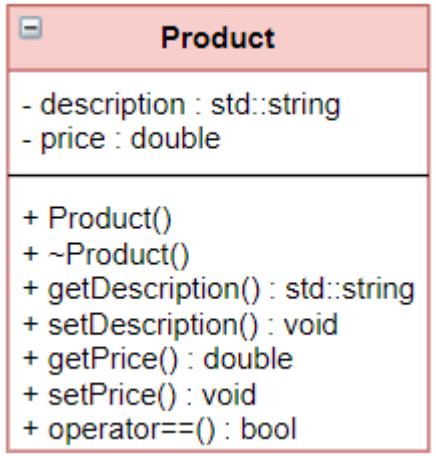

Figura 4 - Diagrama da classe *Product*

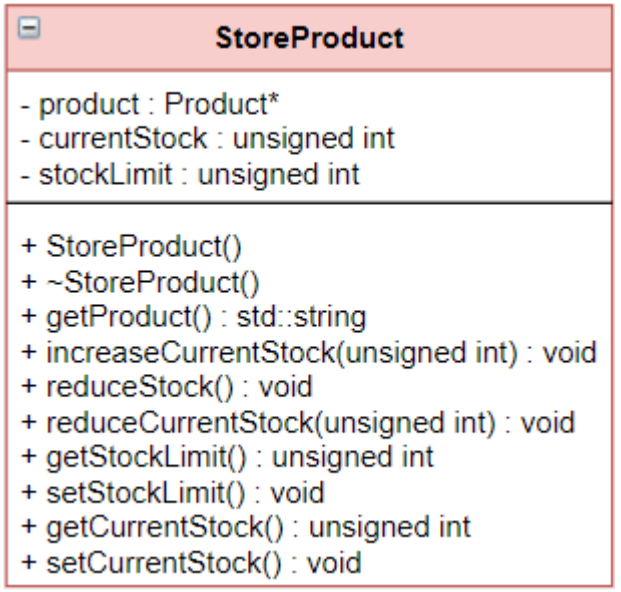

Figura 5 - Diagrama da classe Store*Product*

Classe *Transaction*

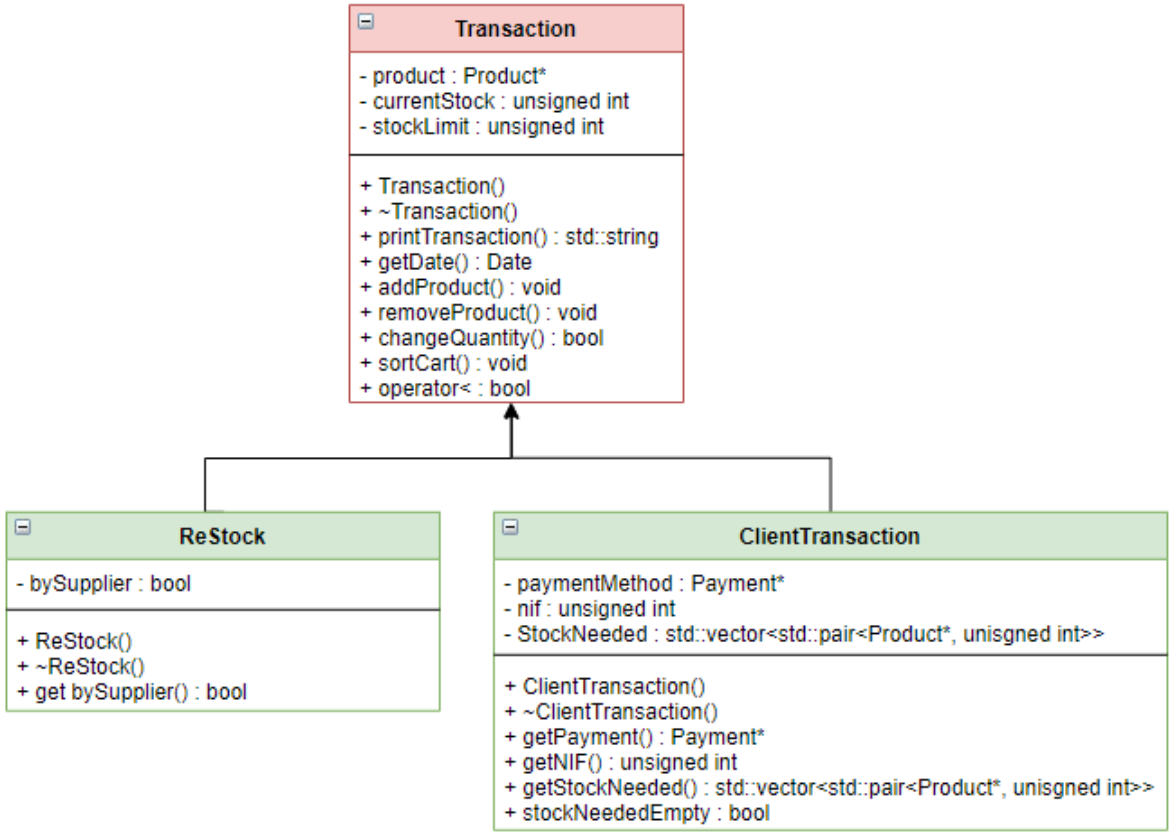

Figura 6 - Diagrama da classe *Transaction* e respetivas subclasses.

Classe *Client*

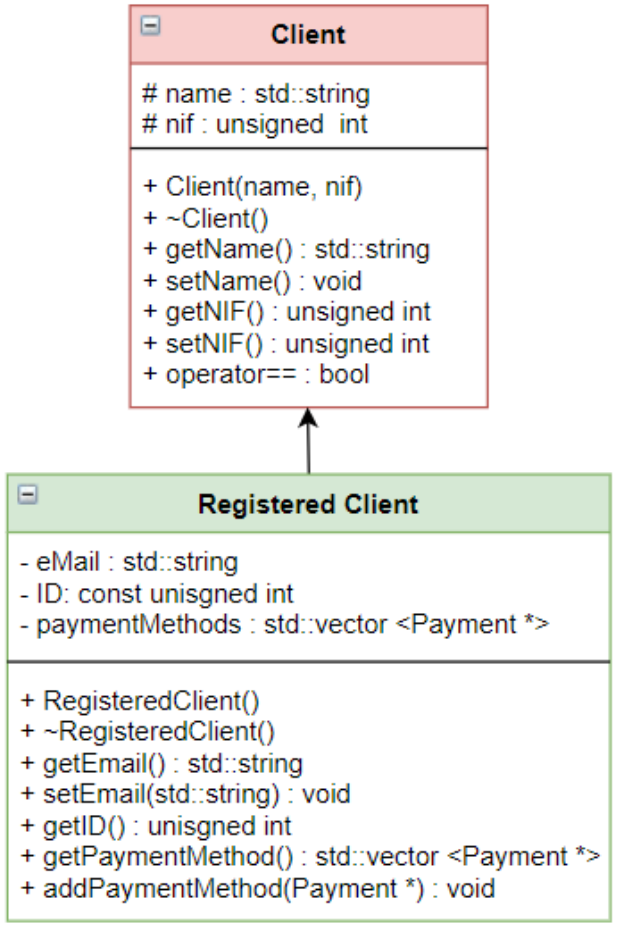

Figura 7 - Diagrama da classe *Client* e *RegisteredClient*.

Classe *Payment*

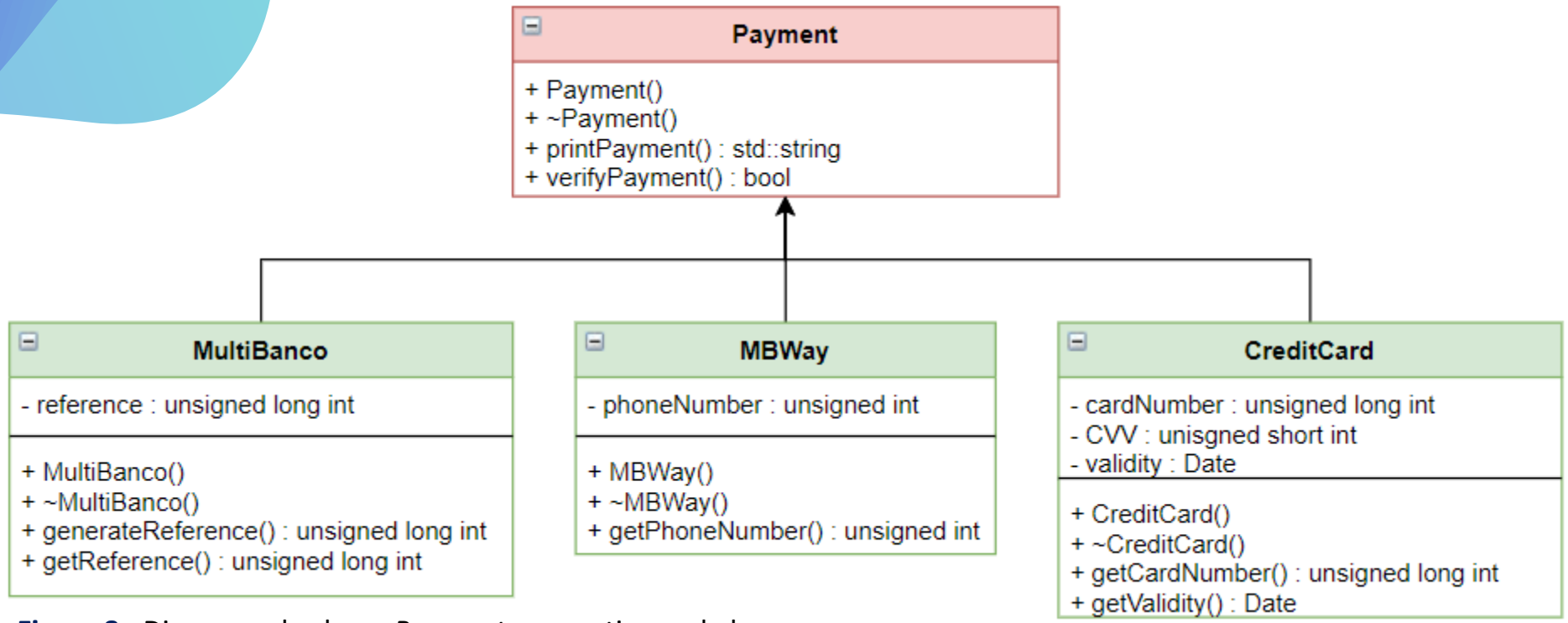

Figura 8 - Diagrama da classe *Payment* e respetivas subclasses.

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

TRATAMENTO DE EXCEÇÕES

De que forma o programa lida com os erros?

LANÇAMENTO DE EXCEÇÕES

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

- As exceções são o principal mecanismo para se lidar com possíveis erros que possam surgir durante a utilização do programa desenvolvido.
- Existem várias exceções lançadas pelos métodos das classes implementadas, que ao serem posteriormente tratadas alertam o usuário sobre o erro(s) que ocorreram.

- **Exemplos**:
  - *NotEnoughStock*
    - Lançada quando um cliente adiciona ao carrinho um produto for a de stock.
  - *ProductDoesNotExist*
    - Exceção lançada quando um cliente adiciona ou procura por um produto que não se encontra listado em loja.
  - *InvalidPayment*
    - Arremessada quando um método de pagamento não respeita os requesitos pré-determinados.
  - *InvalidDate*
    - A seu arremesso dá-se sempre que uma data é inserida com valores inválidos.

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

ESTRUTURA DOS FICHEIROS

Processamento dos dados dos clientes e dos produtos.

FICHEIROS E A SUA UTILIDADE

- Para o armazenamento dos dados relativos a transações, listas de clientes e produtos foram usados ficheiros de texto.

- O programa, quando é iniciado, efetua a leitura dos conteúdos do ficheiro armazenando-os na estrutura de dados apropriada. Por outro lado, a escrita dos dados para o ficheiro de texto ocorre no quando o utilizador encerra o programa.

COMPANY.TXT

- **Nome da Empresa**
  - Localização das lojas físicas
- **Lista de Clientes**
  - Clientes Não Registados
  - Clientes Registados
    - E-Mail e métodos de pagamento
- **Lista de Produtos**
  - Descrição e Preço
  - Referências de Stock para cada loja
    - Stock mínimo
    - Stock disponível
- **Lista de Transações**
  - Para efeitos estatísticos

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

FUNCIONALIDADES

Facilidade de uso na ótica do utilizador.

FUNCIONALIDADES

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

- O programa permite, ao seu utilizador, de uma forma simples e intuitiva a execução de várias operações de forma a tornar mais fácil a sua utilização.

- **CRUD** (*Create*, Read, *Update*, *Delete*) - (OK):
  - Todos os objetos das classes *Client*, *StoreProduct*, *Product* e *Store* possuem métodos que cumprem este tipo de operações.

- **Listagem** - (OK):
  - É possível ao utilizador fazer a listagem dos produtos que se encontram à venda na loja.
  - Também é dada ao utilizador a possibilidade de ver o seu carrinho de compras de uma forma ordenada.

- **Pesquisa** - (OK):
  - O utilizador deste sistema pode procurar por um determinado produto que esteja à venda em alguma das lojas.
  - O administrador do sistema pode, por exemplo, ver estatísticas de vendas num determinado período de tempo à sua escolha, para tal, foram implementados algoritmos de pesquisa sobre as estruturas de dados que armazenam as Transações.

FUNCIONALIDADES EXTRA

- Como foi dito anteriormente o programa procura trazer ao utilizador uma plataforma simples e de utilização amigável para compras online.

- Como tal, uma das ferramentas mais úteis ao administrador é as reposições de stock automáticas. Otimizando a eficiência e os custos para a companhia.
  - Primeiro, quando se atinge um esgotamento de stock na loja *online*, procura-se nas lojas físicas se há stock suficiente para repor na loja *online*.
  - Caso não seja possível essa reposição através das lojas físicas, então adquire-se produto ao fornecedor.

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

DIFICULDADES

Quais as maiores desafios enfrentados pelo grupo?

DISTRIBUIÇÃO DAS TAREFAS

- Neste projeto, coube a cada elemento do grupo implementar um conjunto específico de classes e respetivos métodos, cada um destes com as suas particularidades.
- Além disso a coordenação e distribuição equitativa das tarefas entre todos os elementos do grupo mostrou-se, desde o ínicio, crucial para que fossem atingidos os objetivos esperados.

- **João Marinho** (1/3)
  - Conceção inicial dos Diagramas;
  - Classes *Product*, *StoreProduct* e reposições de stock a partir de class *Company*;
  - Desenvolvimento da interface (vísivel para o cliente).
- **Miguel Rodrigues** (1/3)
  - Conceção inicial dos Diagramas;
  - Classes *Date*, *Transaction*, *Payment* e as respetivas subclasses;
  - Desenvolvimento do relatório.
- **Tiago Silva** (1/3)
  - Conceção inicial dos Diagramas;
  - Classes *Company*, *Client* e *Store* e respetivas subclasses;
  - Desenvolvimento da interface (vísivel para administrador).