Licenciatura em Engenharia de Sistemas Informáticos

U.C. Programação Orientada a Objetos (POO)

2024/2025

FASE 1

Docente: Luís Ferreira

Aluno: José António da Cunha Alves

Nº 27967

08/11/2024

Conteúdo

Enunciado

1 Motivação

Pretende-se que sejam desenvolvidas soluções em C# para problemas reais de complexidade moderada. Serão identificadas classes, definidas estruturas de dados e implementados os principais processos que permitam suportar essas soluções. Pretende-se ainda contribuir para a boa redação de relatórios.

2 Objetivos

• Consolidar conceitos basilares do Paradigma Orientado a Objetos;
• Analisar problemas reais;
• Desenvolver capacidades de programação em C#;
• Potenciar a experiência no desenvolvimento de software;
• Assimilar o conteúdo da Unidade Curricular.

3 Problema a Explorar

(vii) Comércio eletrónico: sistema que permita a gestão de uma loja online. *keywords*: produtos, categorias, garantias, stocks, clientes, campanhas, vendas, marcas.

Introdução

Este documento é uma descrição do trabalho realizado na primeira fase do trabalho prático da unidade curricular de Programação Orientada a Objetos.
Neste trabalho, é proposto desenvolver um programa que torne possível a gestão de uma loja online, tendo como termos indispensáveis *produtos, garantias, vendas, clientes, categorias, stocks, campanhas* e *marcas*. Assim sendo, é necessário que o programa contemple funções que permitam todo o tipo de operações de gestão, desde criação de ficha de produto/cliente, bem como a gestão dos stocks dos respetivos produtos, datas de fim de garantias, datas de venda, entre outros. Esta primeira fase contempla apenas a definição de classes indispensáveis ao projeto, bem como as funcionalidades básicas de gestão das mesmas.

Este trabalho tem como objetivos: a consolidação de conceitos basilares do Paradigma Orientado a Objetos; a análise de problemas reais, neste caso, de gestão de uma loja; o desenvolvimento de capacidades de programação em *C#*; o potenciamento da experiência no desenvolvimento de *software* e a assimilação do conteúdo lecionado na Unidade Curricular em questão.

Revisão de Literatura

- Programação Orientada a Objetos – Material das aulas;
- Documentação – Doxygen Quick Reference;
- Qualidade do Código – *Clean Code – A Handbook of Agile Software Craftsmanship*; de Robert C. Martin.
- .NET Coding Conventions – Microsoft.

Convenções de Nomenclatura Utilizadas (*Microsoft .NET Coding Conventions*)

Classes e Interfaces:

- Classes e tipos públicos devem ter nomes em **PascalCase**.
- Interfaces devem começar com a letra "I", seguida de um nome em PascalCase (ex: *ISale*, *IClient*).

Métodos:

- Deve utilizar-se **PascalCase** para métodos públicos e internos (ex: *AddClient*, *RemoveProductFromSale*).
- Métodos devem ter nomes que descrevam claramente a ação ou propósito (ex: *AddClientToSale*).

Propriedades e Campos:

- Propriedades públicas e internas em **PascalCase** (ex: *MakeList*, *ClientList*).
- Atributos privados e variáveis de instância usam **camelCase** e um prefixo '_' (ex: *_id*, *_durantionInYears*).
- Constantes e campos *readonly* podem ser nomeados em **PascalCase** (*MaxProducts*).

Variáveis Locais e Parâmetros:

- Deve utilizar-se **camelCase** para variáveis locais e parâmetros (ex: *campList*).
- Devem escolher-se nomes descritivos para melhorar a clareza, evitando abreviações excessivas.

Padrões e Convenções de Estilo (*Microsoft .NET Coding Conventions*)

Espaçamento e Identação:

- Identação com 4 espaços (não usar tabulações), para manter o padrão da maioria dos editores de C#.

Colocação de Chaves:

- Devem usar-se chavetas de abertura '{' na mesma linha da declaração (*if*, *for*, *while*), e chavetas de fecho numa linha nova.

Comentários e *XML Documentation*:

- Devem comentar-se métodos e classes utilizando **comentários XML** (///). Descreva parâmetros e valor de retorno, incluindo exceções lançadas.
- Deve utilizar-se *tags* XML como *<summary>, <param>, <returns>, <exception>*, para fornecer documentação completa.

Nomes de Ficheiros:

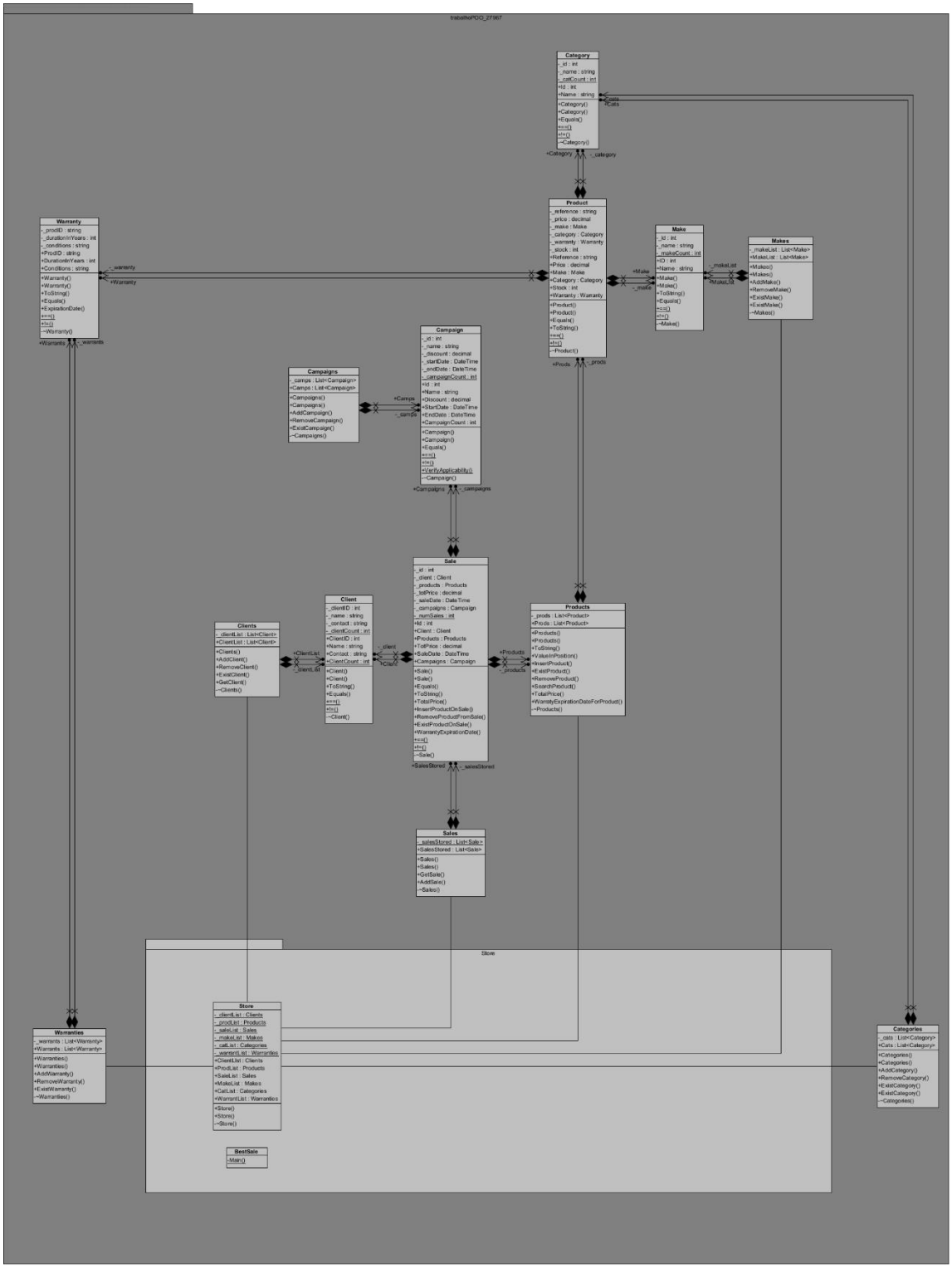
- Ficheiros de código devem ser nomeados de acordo com a classe pública principal que ele contém. Por exemplo, a classe *Client* deve estar no arquivo *Client.cs*.

Tratamento de Exceções:

- Deve utilizar exceções claras e específicas.
- Deve evitar capturar exceções genéricas sem tratamento adequado.

Trabalho Desenvolvido

Diagrama de Classes:



Estrutura de Classes:

- ***BestSale*** – Classe Principal.
- ***Campaign*** – Definição de atributos de campanha, propriedades e métodos para gestão das mesmas.
- ***Campaigns*** – Classe de agregação de ***Campaign***, que contém os métodos para gestão da pluralidade.
- ***Category*** – Definição de atributos de categoria, propriedades e métodos para gestão das mesmas.
- ***Categories*** – Classe de agregação de ***Category*** que contém os métodos para gestão da pluralidade.
- ***Client*** – Definição de atributos de cliente, propriedades e métodos para gestão dos mesmos.
- ***Clients*** – Classe de agregação de ***Client***, que contém os métodos para a gestão da pluralidade.
- ***Make*** – Definição de atributos de marca, propriedades e métodos para gestão das mesmas.
- ***Makes*** – Classe de agregação de ***Make***, que contém os métodos para gestão da pluralidade.
- ***Product*** – Definição de atributos de produto, propriedades e métodos para gestão dos mesmos.
- ***Products*** – Classe de agregação de ***Product***, que contém os métodos para gestão da pluralidade.
- ***Sale*** – Definição de atributos de venda e propriedades e métodos para gestão das mesmas.
- ***Sales*** – Classe de agregação de ***Sales***, que contém métodos para a gestão da pluralidade.
- ***Store*** – Definição de atributos de loja, propriedades e métodos para gestão da mesma.
- ***Warranty*** – Definição de atributos de garantia, propriedades e métodos para gestão das mesmas.
- ***Warranties*** - Classe de agregação de ***Warranty***, que contém métodos para a gestão da pluralidade.

Ponto de Situação e Trabalho Futuro

O programa resultante é ainda bastante embrionário, visto que há ainda alterações que devem ser feitas. Estas alterações visam a otimização do programa, bem como a tentativa de melhoria da própria implementação, no que à qualidade do código diz respeito.

Neste momento, foram apenas definidas as classes que deverão ser utilizadas ao longo do processo, não considerando necessariamente estruturas de dados adequadas ao melhor desempenho, sendo essa uma das possíveis alterações futuras. Mais ainda, deve também ser implementado todo o procedimento de tratamento de exceções, visto ser também um tema a ser ainda abordado em aula.

A constante revisão e limpeza do código é também um fator a ter em conta.