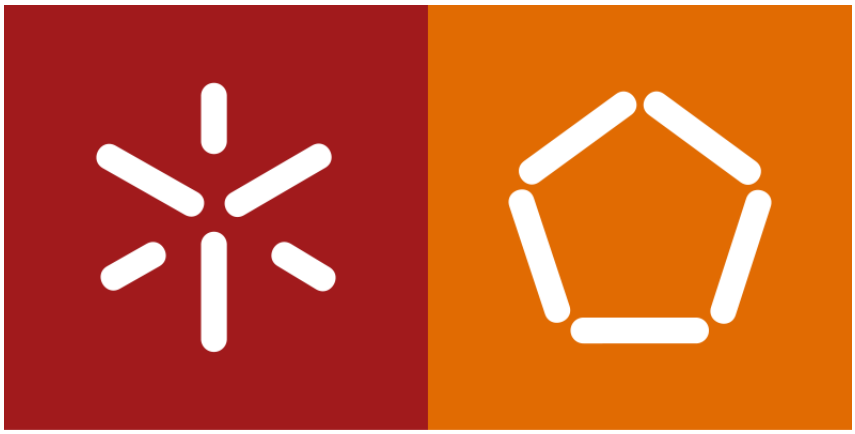

Universidade do Minho

Escola de Engenharia
Relatório

Trabalho Prático de Programação Orientada a Objetos 22/23

Grupo 58:
91672 Luís Ferreira
93258 Bernardo Lima
94595 João Pedro Cardoso

Professor:
António Nestor Ribeiro

Contents

1 Motivação

Este trabalho foi realizado no âmbito da unidade curricular Programação Orientada a Objetos, no ano letivo 2022-2023.

O propósito principal do mesmo é forçar os alunos a aglomerar e consolidar os conhecimentos aprendidos nas aulas.

Desde a noção básica de classes e objetos, aos conceitos mais práticos como encapsulamento, modularidade e abstração. Passando também por conceitos mais exclusivos do paradigma orientado a obejetos, como as noções de herança, polimorfismo, e overriding de métodos.

Dada ao requisito de ser realizado um diagrama de classes para o mesmo trabalho, passamos também pelo campo da modelação, mais especificamente sobre a metodologia UML. É nessa mesma faceta do trabalho que também compreendemos melhor os conceitos como, composição, agregação, associação e dependência.

2 Arquitetura e Decisões tomadas

2.1 Arquitetura da aplicação

O projeto foi dividido em 4 packages, cada uma refernente a uma componente da nossa solução.

- **Time** - Responsável por guardar e alterar a data relativa ao sistema.
- **Items** - Onde estão declarados todos os métodos e variáveis de instância relativos às classes onde é guardada a informação sobre artigos.
- **Transporation** - Onde estão declarados todos os métodos e variáveis de instância relativos às classes onde é guardada a informação sobre transporadoras e também a classe Encomenda que agrega os artigos para expedição.
- **Users** - Onde estão guardadas todos as variáveis de instância da classe *Users.java*, assim como compra, venda e devolução de artigos.

2.1.1 Package Time

```
import java.util.Collection;
import java.time.LocalDate;
import java.util.Scanner;
import java.time.DateTimeException;

public class Data {
    public static LocalDate tempo;
```

Figura 1: Código da classe Data.java

Contém a classe *Date.java*, constitúida por uma variável estática pública e 4 métodos.

- ***LocalDate* tempo** - Contém a data que a loja vai utilizar, funciona como uma variável global.
- **public static void addDays()** - Recebe o número de dias a avançar e efettua soma, atualizando a variável tempo.
- **public static void checkEncomendas(Collection<Encomenda> encomendas, LocalDate date)** - imprime as encomendas que ainda se encontram em transito
- **void setData(LocalDate x)** - Setter de tempo.
- **public static void startTempo() throws DateTimeException** - Recebe a data inicial, verificando se a mesma é válida e atualiza o valor da variável tempo.

2.1.2 Package Items

É nesta package que começamos a aplicar alguns conceitos de programação orientada a objetos. Nomeadamente: Herança e Polimorfismo.

É constituída por 6 classes.

- **Artigo.java**
- **Mala.java**
- **MalaPremium.java**
- **Sapatilha.java**
- **SapatilhaPremium.java**
- **Tshirt.java**

Artigo.java

No programa, por exemplo, a classe SapatilhaPremium extende a classe Sapatilha que por sua vez extende a classe Artigo.

Decidimos esta implementação pois a mesma permite que as subclasses herdem as propriedades e comportamentos da sua superclasse, reduzindo a quantidade de código duplicado e permitindo maior reutilização do mesmo.

Diferenciação entre artigos usados e novos

Outra decisão importante foi como fazer a separação entre artigos novos e usados, decidimos que a maneira mais sucinta seria acrescentar uma variável de número inteiro, chamada *num-donos*.

Da seguinte maneira, um artigo é usado, se *num-donos* for maior que 0, e novo caso o contrário seja verdade. Esta decisão acrecenteu uma simplicidade ao fator de desvalorização de artigos, pois assim conseguimos diretamente decidir se, e o quanto descontar, utilizando apenas uma variável.

```java
package Items;
import java.util.Objects;


public class Artigo {
    private String descricao;
    private String marca;
    private String item_id;
    private String transportadora;
    private double preco;
    private double desconto;
    private int num_donos;
    private int stock;
```

Figura 2: Código da classe Artigo.java

A classe artigo é constituída por:

- 8 variáveis de instância
- Métodos standard (toString,Clone,Equals,gets e sets, construtores..)

Sapatilha.java

```java
package Items;

import Time.Data;

import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

public class Sapatilha extends Artigo{
    private int tamanho;
    private boolean atacadores;
    private String cor;
    private int ano_colecao;
```

Figura 3: Código da classe Sapatilha.java

Na classe sapatilha, tivemos que acrescentar váriáveis que não se encontravam presentes num objeto da classe artigo.

Assim, Sapatilha.java é constituído por:

- 4 variáveis de instância
- Métodos standard
- Métodos de cálculo de preço
- Métodos de cálculo e validação do desconto
- Métodos de escrita para database

Mala.java

```java
package Items;

import Time.Data;

import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

public class Mala extends Artigo {
    private String dimensao; //pode ser pequena média ou grande
    private int ano_colecao;
    private String material;
```

Figura 4: Código da classe Mala.java

De forma análoga à classe anterior, Mala.java é constituída por:

- 3 variáveis de instância
- Métodos standard
- Métodos de cálculo de preço
- Métodos de cálculo e validação do desconto
- Métodos de escrita para database

Tshirt.java

```java
package Items;


import Transportation.Transportadora;

import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.List;
import java.util.Scanner;

import static Transportation.Transportadora.readDatabaseTransportadora;

public class Tshirt extends Artigo{
    private String tamanho;
    private String padrao; //pode ser lisa, riscas ou palmeiras

    //Lisas nunca têm desconto, as restantes têm 50% de desconto se forem usadas
```

Figura 5: Código da classe Mala.java

Novamente, Tshirt.java é constituída por:

- 2 variáveis de instância
- Métodos standard
- Métodos de cálculo e validação do desconto
- Métodos de escrita para database

SapatilhaPremium.java

```java
public class SapatilhaPremium extends Sapatilha{
    private String autor;
    //sapatilhas premium têm autores reconhecidos que aumentam o preço com o passar do anos

    public SapatilhaPremium(){
        super();
        this.autor = "";
    }

    public SapatilhaPremium(SapatilhaPremium sapatilhaPremium){
        super(sapatilhaPremium);
        this.autor = sapatilhaPremium.getAutor();
    }

    public SapatilhaPremium(String descricao, String marca, String item_id,String transportadora, double preco, double desconto, int num_donos,int stock, int tamanho, boolean atacadores, String cor, int ano_colecao, String autor) {
        super(descricao, marca, item_id,transportadora, preco, desconto, num_donos, stock, tamanho, atacadores, cor, ano_colecao);
        this.autor = autor;
    }

    //getter and setter

    public String getAutor() {
        return autor;
    }

    public void setAutor(String autor) {
        this.autor = autor;
    }


    public SapatilhaPremium clone() {
        return new SapatilhaPremium(this);
    }

    public void calculaPreco(){
        this.setDesconto(2*(Data.tempo.getYear()-this.getAno_colecao()));
        this.setPreco(this.getPreco()*(1+this.getDesconto()/100)+this.taxaSatisfacao());
    }
```

Figura 6: Código de Sapatilha Premium

é constituído por:

- 3 variáveis de instância
- Métodos standard
- Métodos de cálculo de preço
- Métodos de cálculo e validação do desconto
- Métodos de escrita para database

MalaPremium.java

```java
public class MalaPremium extends Mala{
    //malas premium, o preço sobre x% por ano
    private String autor;

    public MalaPremium(){
        super();
        this.autor = "";
    }

    public MalaPremium(MalaPremium malaPremium){
        super(malaPremium);
        this.autor = malaPremium.getAutor();
    }

    public MalaPremium(String descricao, String marca, String item_id, String transportadora, double preco, double desconto, int num_donos, int stock, String dimensao, int ano_colecao, String material, String autor) {
        super(descricao, marca, item_id,transportadora, preco, desconto, num_donos, stock, dimensao, ano_colecao, material);
        this.autor = autor;
    }

    //getter and setter

    public String getAutor() {
        return autor;
    }

    public void setAutor(String autor) {
        this.autor = autor;
    }


    public MalaPremium clone() {
        return new MalaPremium(this);
    }

    public void calculaPreco(){
        this.setDesconto(2*(Data.tempo.getYear()-this.getAno_colecao()));
        this.setPreco(this.getPreco()*(1+this.getDesconto())/100+this.taxaSatisfacao());
    }
```

Figura 7: Código de Mala Premium

é constituído por:

- 3 variáveis de instância
- Métodos standard
- Métodos de cálculo de preço
- Métodos de cálculo e validação do desconto
- Métodos de escrita para database

2.1.3 Package Transportation

Inclui duas classes de java, **Encomenda.java** e **Transportadora.java**, assim como uma base de dados denominada *transportadoras.txt*

Encomenda.java

```java
package Transportation;

import Items.*;
import Time.Data;
import Users.Purchases;


import java.io.*;
import java.time.format.DateTimeFormatter;
import java.util.Scanner;
import java.time.LocalDate;
import java.util.ArrayList;
import java.util.Collection;
import java.util.List;
import java.util.Objects;

import static Time.Data.tempo;

public class Encomenda {
    private String encomendaId;
    private Collection<Artigo> colecao;
    private int dimensao;
    private double preco_final;
    private String estado;
    private LocalDate data_criacao;
    private String buyerEmail;
```

Figura 8: Código de Encomenda.java

Esta classe é constituída por:

- 7 variáveis de instânica
- Métodos Standard
- Métodos de leitura/escrita em base de dados
- 1 método que calcula o preço final

Importância desta classe

O uso mais relevante desta classe é permite-nos tratar do estado das encomendas, atualizar ou devolver caso ainda se encontre em trânsito.

Tem ainda um método crucial para o funcionamento da loja, que é o *public double calcularPrecoFinal(ArrayList<Artigo> colecao)*

Que ao receber uma coleção de artigos, tendo em conta que artigos diferentes possuem fórmulas de preço diferentes, faz os cálculos apropriados, e ainda acresce o preço de envio da transportadora de cada artigo.

Com esta função conseguimos apresentar o valor correto da encomenda.

Transportadora.java

```
package Transportation;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class Transportadora {
    private String nome;
    private double preço_pequena;//1 artigo
    private double preço_media;//2 a 5 artigos
    private double preço_grande;// 6 ou mais
    private int tamanho;
```

Figura 9: Código de Transportadora.java

Esta classe é constituída por:

- 5 variáveis de instânica
- Métodos Standard
- Métodos de leitura da base de dados
- Métodos de cálculo de imposto

Importância desta classe

Nesta classe, os métodos mais importantes são o cálculo de imposto, que é relevante para o cálculo do preço final da encomenda, e a leitura da base de dados, para obter as transportadoras e seus preços para cada tipo de pacote: pequeno, médio ou grande.

2.1.4 Package Users

Por fim chegamos à package mais complexa, que pega em tudo que fizemos até agora, e une as funcionalidades, de forma a pôr a nossa loja em funcionamento.

É composta por 6 classes:

- **User.java**
- **BuyOrSell.java**
- **Buy.java**
- **Sell.java**
- **Purchases.java**
- **Queries.java**
- **Pack.java**

User.java

```java
package Users;

import java.io.*;
import java.util.ArrayList;
import java.util.List;
import java.util.Objects;
import java.util.Scanner;

public class User {
    private String name;
    private String userId;
    private String email;
    private String morada;
    private String nif;
```

Figura 10: Código de User.java

Esta classe é constituída por:

- 5 variáveis de instânica
- Métodos Standard
- Métodos de leitura/escrita em base de dados
- Métodos de registo e login

Esta classe é relativamente simples, os métodos mais relevantes são o método de criação de ID's e de login, sendo que o primeiro adiciona entradas ao ficheiro "users.txt" e o segundo averigua se está lá presente o utilizador inserido no login.

BuyOrSell.java

Esta classe é constituída por:

```java
public static void buyOrSellArticle(String userEmail) {
    boolean running = true;
    Scanner scanner = new Scanner(System.in);
    //clear the screen
    System.out.print("\033[H\033[2J");
    System.out.flush();

    while(running) {
        System.out.println("Welcome to the shop, today is "+ tempo + ",\nWhat are you here for?: \n 1) Buy \n
        int choice = scanner.nextInt();
        switch (choice) {
            case 0:
                System.out.println("Exiting program...");
                System.exit(0);
            case 1:
                Buy.buyArticle(userEmail);
                break;
            case 2:
                Sell.sellArticle(userEmail);
                break;
```

Figura 11: Código de BuyOrSell.java

- Um método responsável pelo menu da loja

Importância desta classe

O nosso programa têm essencialmente 3 menus principais, menu de login, menu de loja e menu de compra.
Esta classe apresenta o menu de loja, lendo e processando o input do utilizador, e redirecionando para a função correspondente à escolha do utilizador.

Por exemplo, se escolhermos comprar será chamado o método responsável pela compra, o *public static void buyArticle(String userEmail)*, o que nos leva à classe **Buy.java**

Buy.java

```
import static Users.BuyOrSell.buyOrSellArticle;
import static Users.Purchases.getAllSales;
import static Users.Purchases.removeItemFromUserStock;

public class Buy {
    public static void buyArticle(String userEmail) {
        System.out.print("\033[H\033[2J");
        System.out.flush();

        boolean running = true;
        Scanner scanner = new Scanner(System.in);
        Encomenda encomenda = new Encomenda(new ArrayList<>(),0,0.0,"Em transito", Data.tempo,userEmail, "");
        ArrayList<Artigo> cart = new ArrayList<>();
        ArrayList<Artigo> stock = getAllSales(userEmail);
        String itemId;

        while(running) {
            System.out.println("Select if you want to:\n 1) See the stock\n 2) Add an item to your shopping ca

            int choice;
            while (true) {
                if (scanner.hasNextInt()) {
                    choice = scanner.nextInt();
                    break;
                } else {
```

Figura 12: Código de Buy.java

Caso pretenda vender, é chamado o método da classe Sell.java

Sell.java

```
import static Users.BuyOrSell.buyOrSellArticle;
import static Users.Purchases.getAllSales;
import static Users.Purchases.removeItemFromUserStock;

public class Buy {
    public static void buyArticle(String userEmail) {
        System.out.print("\033[H\033[2J");
        System.out.flush();

        boolean running = true;
        Scanner scanner = new Scanner(System.in);
        Encomenda encomenda = new Encomenda(new ArrayList<>(),0,0.0,"Em transito", Data.tempo,userEmail, "");
        ArrayList<Artigo> cart = new ArrayList<>();
        ArrayList<Artigo> stock = getAllSales(userEmail);
        String itemId;

        while(running) {
            System.out.println("Select if you want to:\n 1) See the stock\n 2) Add an item to your shopping ca

            int choice;
            while (true) {
                if (scanner.hasNextInt()) {
                    choice = scanner.nextInt();
                    break;
                } else {
```

Figura 13: Código de Sell.java

Este método, apenas receberá input do utilizador e criará um artigo de acordo com a informação mencionada. Acrescentado o mesmo à base de dados *stock.txt*

Queries.java

```
package Users;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;
import static Users.BuyOrSell.buyOrSellArticle;

public class Queries {
    public static void statisticsMenu(String userEmail) {
        System.out.print("\033[H\033[2J");
        System.out.flush();

        boolean running = true;
        Scanner scanner = new Scanner(System.in);
        String BUY_FILE = "Files/buyhistory.txt";
        String SELL_FILE = "Files/sellhistory.txt";
        String ORDERS_FILE = "Files/orders.txt";

        while (running) {
            System.out.println(
                    "Select one of the following: \n 1) Seller who billed the most \n 2) Carrier which has the highest billing volume \n 3) Ranking of the biggest buyers \n
            int choice = scanner.nextInt();
            switch (choice) {
```

Figura 14: Código de Queries.java

Queries é a classe onde estao contidas as funções de cálculos de estatística.

Purchases.java

```java
package Users;

import Items.*;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class Purchases {
    public static void printUserSales(String email, String FILE) {
        try {
            Scanner scanner = new Scanner(new File(FILE));
            String itemType = "";

            while (scanner.hasNextLine()) {
                String line = scanner.nextLine();
                String[] values = line.split(":");

                if (values[0].equals(email)) {
                    itemType = values[3].substring(0, 2);
```

Figura 15: Código de Purchases.java

A classe purchases é constituída por 4 métodos.

- Um de impressão de as vendas de um user
- Um de leitura das vendas de um user a partir de um ficheiro
- Um de remoção de artigos do stock de um user
- Um de extração de informação sobre produtos de um user a partir de uma string.

Pack.java

```java
import java.util.Scanner;

import static Transportation.Encomenda.removePackageFromFile;
import static Users.BuyOrSell.buyOrSellArticle;

public class Pack {
    public static void packagesMenu(String userEmail){
        System.out.print("\033[H\033[2J");
        System.out.flush();

        boolean running = true;
        Scanner scanner = new Scanner(System.in);
        ArrayList<Encomenda> encomendas;
        encomendas = Encomenda.readPackagesFromFile();
        for (Encomenda encomenda : encomendas) {
            if(encomenda.getBuyerEmail().equals(userEmail)) System.out.println(encomenda.toString2());
        }

        while(running) {
            System.out.println("\nSelect: \n 1)Return Article\n\n  0) Exit:");
            int choice = scanner.nextInt();
            switch (choice) {
                case 1:
                    System.out.println("Type the package ID (The article needs to still be in transit):");
                    String packageID = scanner.next();
                    removePackageFromFile(packageID);
                    break;
                case 0:
                    buyOrSellArticle(userEmail);
                    return;
                default:
```

Figura 16: Código de Pack.java

Importância desta classe

Nesta classe está englobada num menu, a componente de devoluções. O utilizador encontrar-se-á no menu de devolução até não pretende devolver mais nenhum artigo, sendo redirecionado de novo para o menu da loja

2.2 A nossa Main

UserSellerManager.java

```java
import Time.Data;
import Users.User;

import java.io.IOException;
import java.util.ArrayList;
import java.util.InputMismatchException;
import java.util.Scanner;

import static Users.BuyOrSell.buyOrSellArticle;
import static Users.User.*;

public class UserSellerManager {

    private static final String USERS_FILE = "Files/users.txt";

    public static void main(String[] args) throws IOException {
        Scanner scanner = new Scanner(System.in);
        boolean running = true;
        while (running) {
            boolean loggedIn = false;
            String userId = "";
            String userEmail = "";
            // Para inicializar em que data começa
            Data.startTempo();
            while (!loggedIn) {
                System.out.println("\\Welcome to Vintagio/");
                System.out.println("Choose an option:");
                System.out.println(" 1. Log in");
                System.out.println(" 2. Create new user");
                System.out.println(" 0. Quit");
                try {
                    int choice = scanner.nextInt();
                    switch (choice) {
                        case 0:
                            System.out.println("Exiting program...");
```

Figura 17: Código de UserSellerManager.java

Por fim temos a nossa "Main", a função central do nosso projeto, onde o programa é executado, e onde estão interligadas todas as suas fucionalidades previamente descritas.

Passaremos a descrever melhor os seu funcionamento na próxima secção.

3 Descrição da Aplicação Desenvolvida

3.1 Modelação da estrutra do Programa

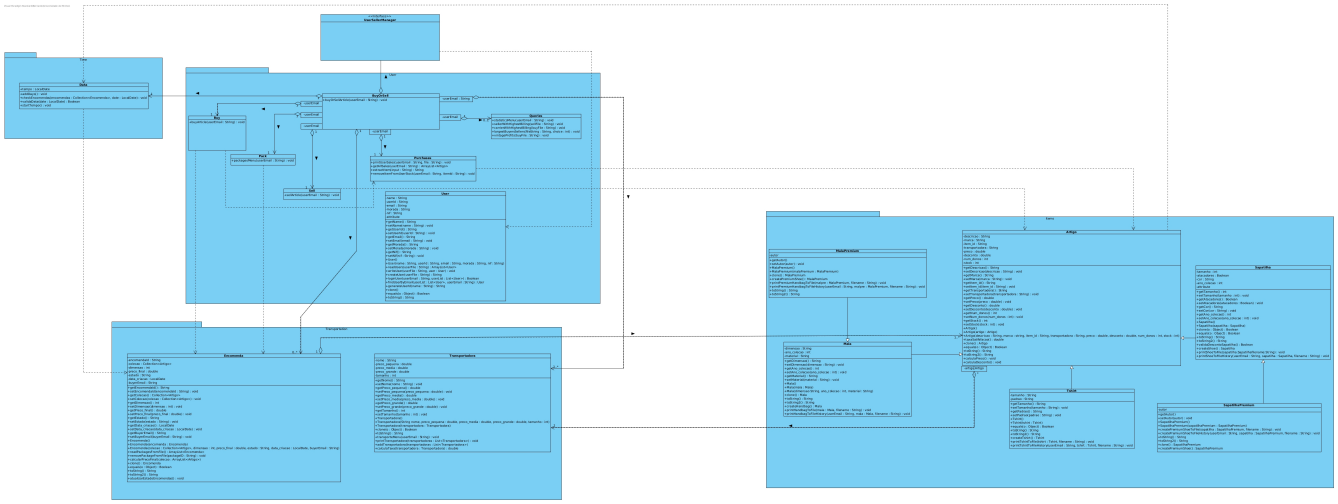

Figura 18: Diagrama de classes

3.2 Modelação do Funcionamento do programa

Durante o nosso estudo da componente de modelação do diagrama de classes, econtramos um diagrama que nos é muito útil para descrever o funcionamento da aplicação, o diagrama de atividades.

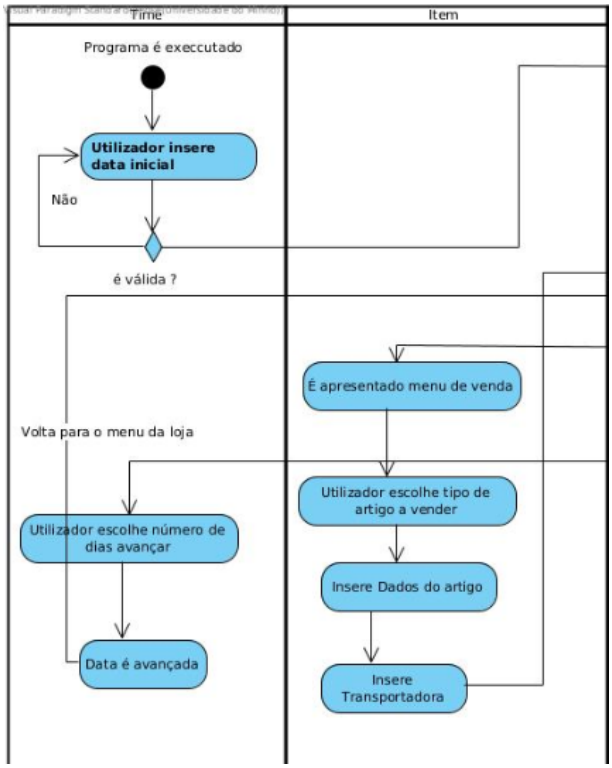

Figura 19: Sequência de execução pré-login + Escolha Vender

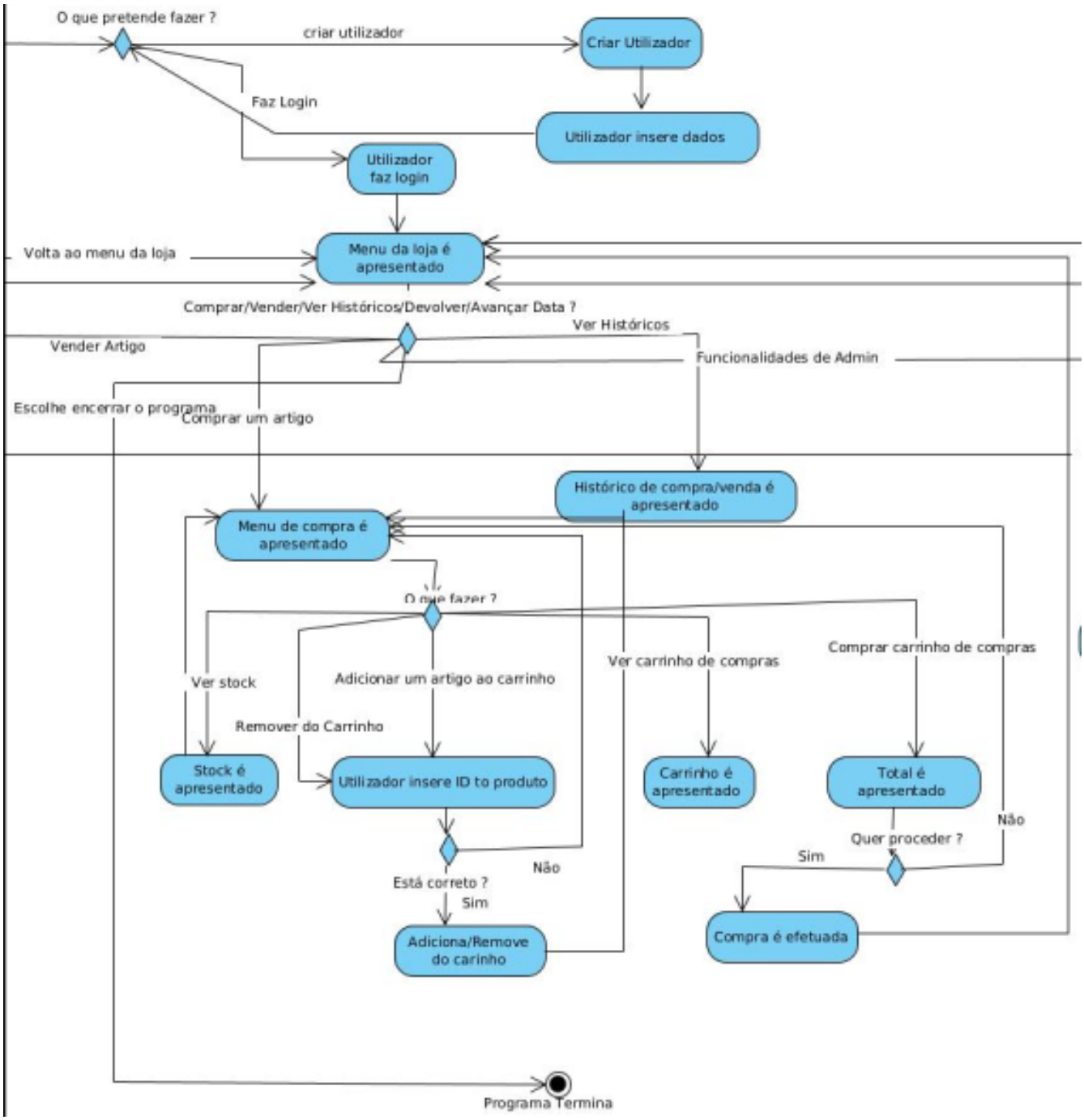

Figura 20: Sequência de execução após login, com exceção de funcionalidadades de Admin

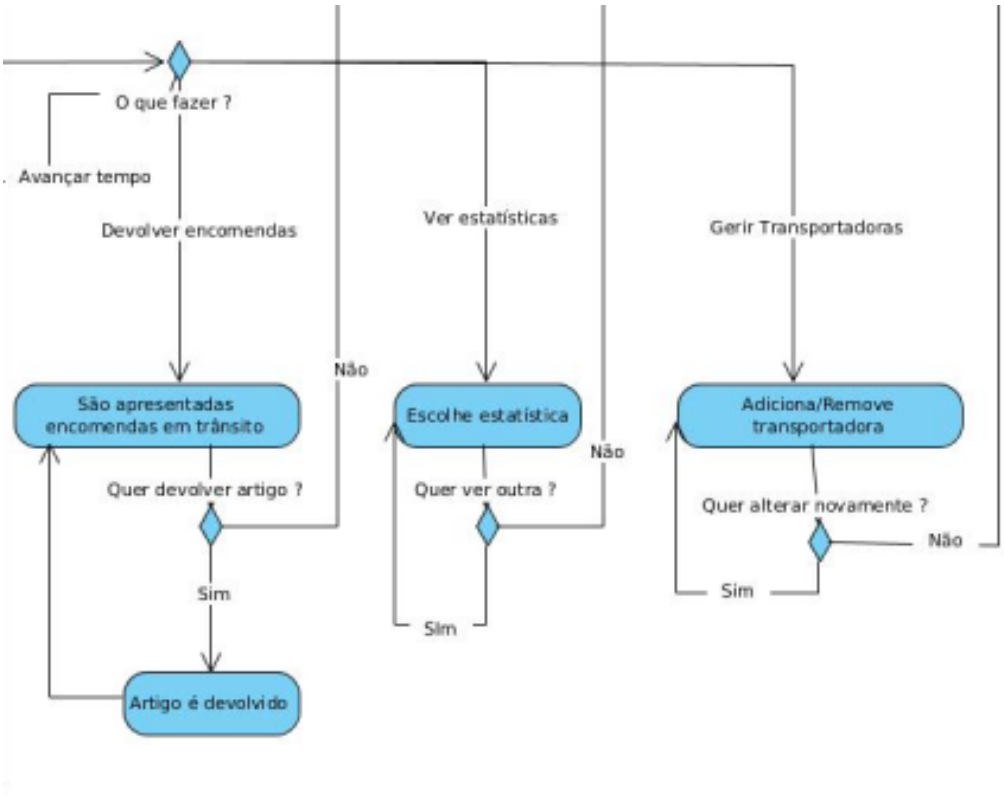

Figura 21: Sequência de execução após escolha de funcionalidades de Admin

Descrição textual do funcionamento da Vintage

Como foi necessário dividir a imagem do diagrama de atividades passaremos a explicá-lo melhor via textual.
Vamos percorrer a execução do programa. Após clicar em executar, vamos ter ao:

Menu de Data

1. Pede uma data
2. Verifica se a mesma é válida
3. É válida ? Define como data do sistema
4. Não ? lança exceção e pede que insira um válida.

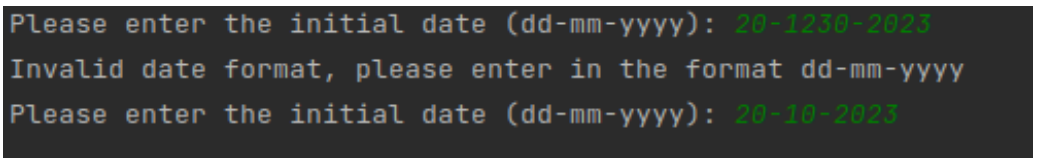

Figura 22: Menu data

Após ter selecionada uma data, avançamos para o:

Menu de Login

O menu login tem duas funcionalidades

- Fazer Login
- Criar novo User

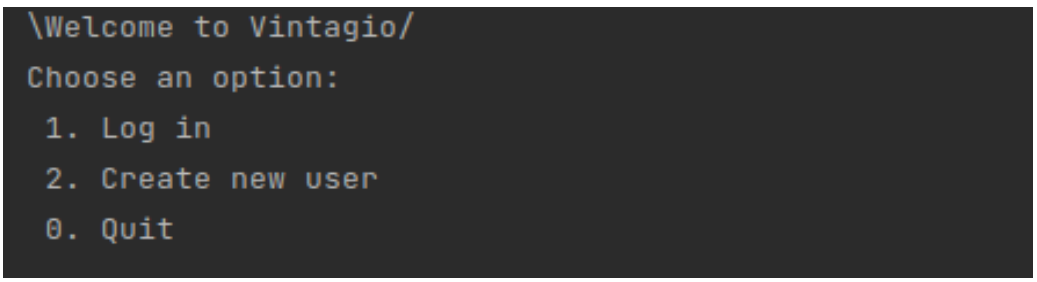

Figura 23: Menu Login

Após ser realizado o login somos levados para o menu mais importante, o menu de loja

Menu de Loja

Funciona como uma éspecie de "homepage". É partir deste que todas as funcionalidades se tornam acessíveis ao user.

```
Logged in as user with email: a

Welcome to the shop, today is 2023-10-20,
What are you here for?:
 1) Buy
 2) Sell
 3) Buy history
 4) Sell history
 5) Check and Return packages
 6) Check Admin Functionality

  0) exit
```

Figura 24: Menu Compra

Se pretender comprar, é levado para o *Menu de Compra*.

Menu de Compra

```
Select if you want to:
 1) See the stock
 2) Add an item to your shopping cart
 3) Remove an item from your shopping cart
 4) Display shopping cart
 5) Buy shopping cart
```

Figura 25: Menu Compra

Assumindo que o utilizador escolheu a primeira:

```
Sapatilha-> Descricao: a, Marca: a, ID do Item: SN54038, Transportadora: UPS, Preco: 25.0, Desconto: 0.0, Num_donos: 2, Stock: 27, Tamanho: 2, Atacadore

Tshirt-> Descricao: a, Marca: a, ID do Item: TN71063, Transportadora: CTT, Preco: 25.0, Desconto: 0.0, Num_donos: 1, Stock: 21, Tamanho: L, Padrao: plai

Select if you want to:
 1) See the stock
 2) Add an item to your shopping cart
 3) Remove an item from your shopping cart
 4) Display shopping cart
 5) Buy shopping cart

  0) exit:
```

Figura 26: Opção ver stock

Assumindo que o utilizador viu um artigo que gostou e pretende comprar, escolhe a opção 2 que lhe vai pedir um ID.

```
Select the item you want to add to your shopping cart typing the id:
invalido
Inserted ID doesn't exist.

Select if you want to:
 1) See the stock
 2) Add an item to your shopping cart
 3) Remove an item from your shopping cart
 4) Display shopping cart
 5) Buy shopping cart

  0) exit:
```

Figura 27: Inserção de ID errado

Depois de inserir um artigo ao carrinho, pode sempre remover, no entanto assumindo que o user sabe o que quer decididiu confirmar se estava mesmo a comprar o que deseja.

```
Your shopping cart:
Tshirt-> Descricao: Tshirt, Marca: Adibos, ID do Item: TN300000, Transportadora: UPS, Preco: 100.0, Desconto: 1.0, Num_donos: 0, Stock: 23, Tamanho:
Premium Mala-> Descricao: premium hand, Marca: asd, ID do Item: HP42870, Transportadora: Fedex, Preco: 100.0, Desconto: 0.0, Num_donos: 0, Stock: 3
Tshirt-> Descricao: isabela, Marca: valnisios, ID do Item: TN75472, Transportadora: CTT, Preco: 34790.0, Desconto: 0.0, Num_donos: 1, Stock: 9997,


Select if you want to:
 1) See the stock
 2) Add an item to your shopping cart
 3) Remove an item from your shopping cart
 4) Display shopping cart
 5) Buy shopping cart

  0) exit:
```

Figura 28: Listar items do carrinho

Se tudo estiver correto o user decide confirmar a compra.

```
Select if you want to:
 1) See the stock
 2) Add an item to your shopping cart
 3) Remove an item from your shopping cart
 4) Display shopping cart
 5) Buy shopping cart

  0) exit:
5
Your total comes to: 17526.175€, do you want to proceed?
[1) Yes, 2) No]
```

Figura 29: Confirmação compra

Após a confirmação de compra a encomenda é criada e encontrar-se-á disponível para devolução durante 2 dias.

Ora, vamos assumir que o user na verdade arrependeu-se e pretende devolver.

Voltou a menu de loja e de lá, escolheu as funcionalidades de administrator.

```
Logged in as user with email: a

Welcome to the shop, today is 2023-10-20,
What are you here for?:
 1) Buy
 2) Sell
 3) Buy history
 4) Sell history
 5) Check and Return packages
 6) Check Admin Functionality

  0) exit
```

Figura 30: Menu Compra

Lá, escolheu a opção de devolver encomendas.

```
5
Encomenda{ID: 60974, colecao: [], dimensao: 3, preco_final: 74827.75, estado: Entregue, data_criacao: 2022-11-25}
Encomenda{ID: 6569, colecao: [], dimensao: 1, preco_final: 16.75, estado: Em transito, data_criacao: 2022-11-25}

Select:
 1)Return Article

  0) Exit:
```

Figura 31: Menu Devolução

Por fim o nosso user decidiu saber quem era o melhor vendedor da vintage.

```
Select one of the following:
 1) Seller who billed the most
 2) Carrier which has the highest billing volume
 3) Ranking of the biggest buyers
 4) Ranking of the biggest sellers
 5) Vintage Profits

  0) Exit:
1

Seller with highest billing: isabela@boda.com (€34790.00)
```

Figura 32: Estatística de melhor vendedor

4 Conclusão e Análise Crítica

Este relatório foi escrito com a intenção de explicar e expôr o que foi o nosso trabalho prático à unidade curricular Programação Orientada a Objetos.

Enfrentamos desafios durante o projeto, principalmente na integração de certos conhecimentos, como o parsing e criação de arquivos "txt" para armazenar o conteúdo da loja. Além disso, a gestão temporal atrasou-nos um pouco, já que foi a última implementação antes das "queries", e tivemos dificuldade em integrar a data de forma que a loja fosse atualizada com o passar do tempo.

Embora haja outras melhorias que poderíamos ter feito, como a automatização, estamos satisfeitos com o resultado final e nossa solução atual.

Agradecemos a oportunidade de aprender e crescer academicamente. Sabemos que o conhecimento de métodos de programação em um dos paradigmas mais usados atualmente é uma valiosa contribuição para nosso currículo e futuro e esperamos conseguir pôr em prática estes conhecimentos em breve.