**INSTITUTO SUPERIOR POLITÉCNICO DE TECNOLOGIAS E CIÊNCIAS**

**DEPARTAMENTO DE ENGENHARIAS E TECNOLOGIAS**

**PROGRAMAÇÃO III**

RELATÓRIO DO TRABALHO PRÁTICO

EXAME FINAL

**TEMA:**

# APLICAÇÃO PARA GESTÃO DE FILAS DE SUPERMERCADOS


| **INTEGRANTES DO GRUPO** | |
|---|---|
| José Domingos Cassua N'donge - 20200689<br>Kuenda João António Mayeye - 20201219<br>Rui Yuri Joaquim Malemba - 20201580 | |
| **CURSO:** ENGENHARIA INF5<br>**TURMA:** M1 | **DOCENTE:**<br>Sediangani Daniel N. Sofrimento |
| Data de Realização do Projecto: Junho de 2023 | |

## ✓ CLASSES, HERANÇA E POLIMORFISMO

Durante o desenvolvimento de nossa aplicação optamos pela utilização de conceitos de programação orientado a objecto como: classes, encapsulamento e herança.

As classes implementadas são:

- Simulação: Uma classe abstrata que possui atributos e alguns métodos que serão extendidos pelas classes ModoManual e ModoAutomatico;

- ModoManual: Possui atributos e métodos utilizados durante a execução manual;

- ModoAutomatico: Possui atributos e métodos utilizados durante a execução automática;

- Caixa: Possui os atributos e métodos utilizado pelos caixas.

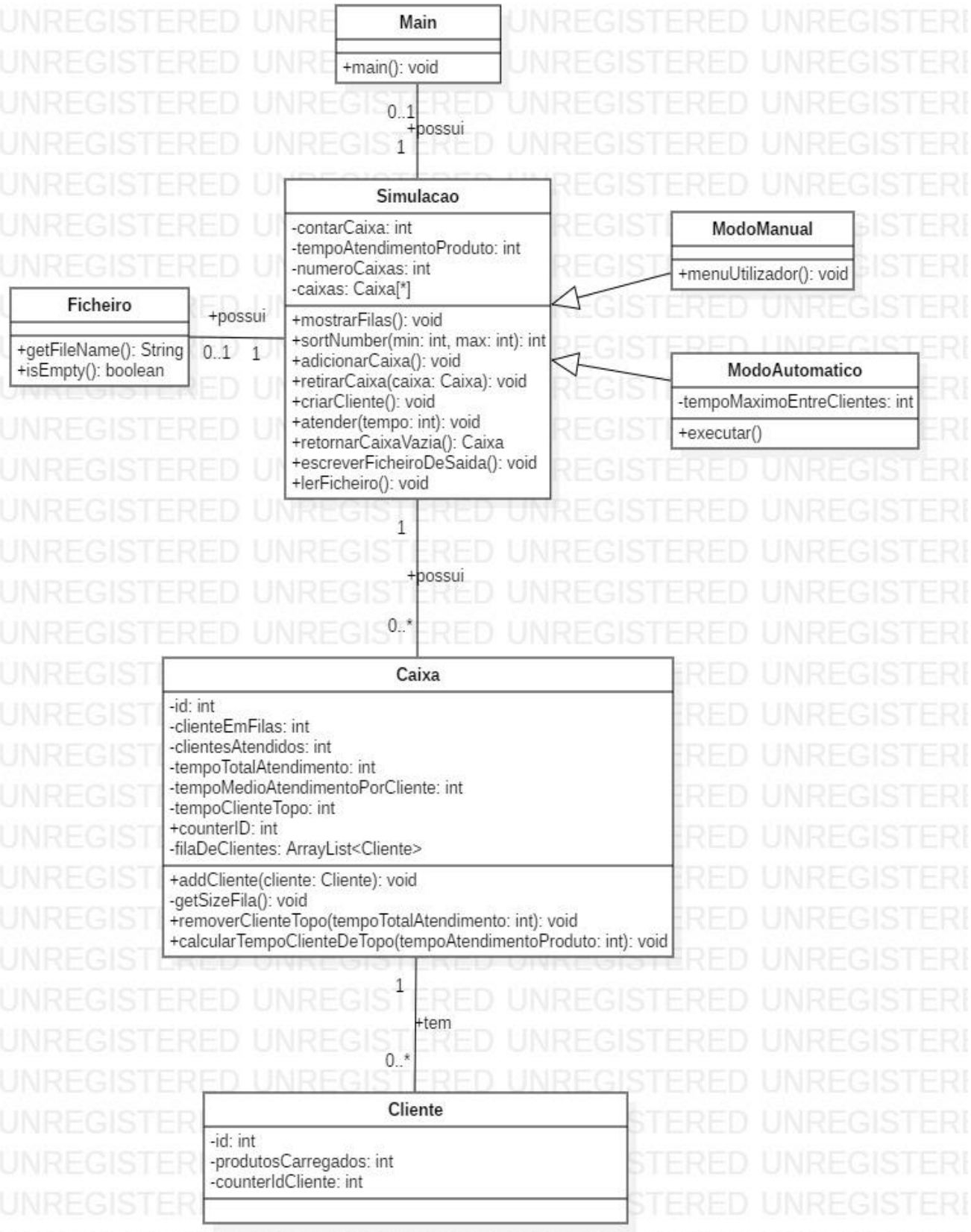
- Cliente: Possui os atributos e métodos utilizado pelos clientes em fila.

- Ficheiro: Classe que possuem apenas dois métodos para manipular ficheiro, getFileName e isEmpty.

- Main: Classe principal e responsável pela apresentação do menu para escolha do modo de execução.

Na aplicação as classes ModoManual e ModoAutomatico herdam as propriedades e métodos da classe Simulação. Esta decisão foi tomada após constatar que tanto o ModoManual como o ModoAutomatico são apresentados no enunciado como simulação, e que possuem características da simulação, aqui vai uma nota, na simulação existem apenas dois métodos abastractos que são implmentados nos dois modos, escreverFicheiroDeSaida() e lerFicheiro().

## ✓ Diagrama de classes



## ✓ Algoritmos mais complexos

Os algoritmos implementados mais complexos são:

- Atender T tempo

  Implementamos esse algoritmo com a seguinte lógica, primeiro criamos um loop para percorrer todas caixas adicionados na simulação e atendemos os clientes de topo dessas caixas, validando as três opções do enunciado, no caso da última opcão, fizemos t = t – tempoDeTopo e zeramos o contador i para atender em todas caixas novamente mas agora com o novo t.

- Retirar caixa

  Implementar este algoritmo foi díficil, uma vez que o grupo optou por utilizar vetor na simulação para armazenar as caixas, sempre que eliminamos a caixa com fila de clientes vazia, atribuimos null a esta caixa e reduzimos o contador de caixas criadas ou existentes, dessa maneira o loop para percorrer as caixas nunca chega numa posição em que a caixa é null.

- Ler Ficheiro

  Esta foi também uma das funções ou métodos mais complexos a ser implementada, a sua lógica é, começamos por ler linha a linha do ficheiro .txt e pegamos cada informação da linha a ser lida, no caso os números inteiros, para os números que possuem mais de um dígito, criamos um loop interno para ler caracter a caracter da string(linha a ser lida do ficheiro .txt), os caracteres que são dígitos concatenamos na string str e depois do loop convertemos essa string para um tipo inteiro e só atribuimos o valor inteiro aos atributos respectivos através dos métodos sets e gets.

## ✓ Organização do Código

O código foi distribuído em 3 packages nomeadamente: entidades, auxiliares e views.

Entidades: Nesta package, encontram-se os ficheiros das classes Caixa, Cliente, ModoAutomatico, ModoManual e classe abstrata Simulacao.

Auxiliar: Nesta package encontra-se o ficheiro da classe Ficheiro.

View: Nesta package encontra-se o ficheiro da classe Main.

## ✓ Opções Técnicas Tomadas

Para as caixas optamos por usar vetores pois o número de caixa será inalterável durante a execução da aplicação.

Ao eliminar caixa, decidimos atribuir null a mesma e reduzimos o contador de caixas válidas em uma unidade.

Optamos por criar o método estático isEmpty() para verificar se o ficheiro possui ou não alguma informação, no caso de não ter, pedimos ao utilizador que insira os parâmetros da simulação, no caso de ter alguma informação o utilizador já não precisa digitar esses parâmetros, salvo no caso de ler o ficheiro que não possui parâmetro tempo máximo entre clientes, estando o utilizador a executar o programa no modo automático.

Para a representação de clientes na fila optamos pelo uso de arraylist, pois o número de clientes em fila não é definido e portanto podem ser adicionados incontáveis clientes na fila.