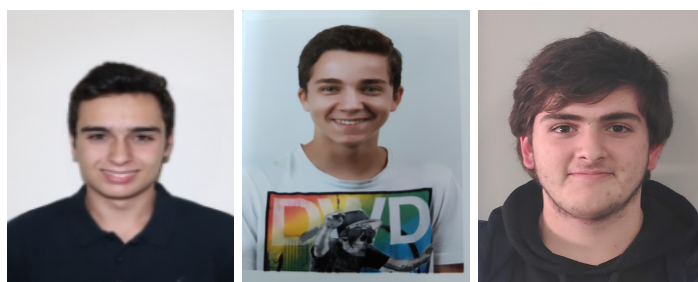




UNIVERSIDADE DO MINHO  
Departamento de Informática

PROGRAMAÇÃO ORIENTADA AOS OBJETOS

## Projeto prático de programação orientada aos objetos



**Feito por:**

João Gonçalves - A95019

João Martins - A96215

Rodrigo Pereira - A96561

21 de maio de 2022

## Conteúdo

<b>1</b>	<b>Introdução</b>	<b>1</b>
<b>2</b>	<b>Diagrama de classes - Compacto</b>	<b>2</b>
<b>3</b>	<b>Classes</b>	<b>2</b>
3.1	SmartDevice . . . . .	2
3.2	SmartBulb, SmartSpeaker e SmartCamera . . . . .	2
3.3	CasaInteligente . . . . .	3
3.4	FornecedoresEnergia . . . . .	3
3.5	Faturas . . . . .	3
3.6	CommunityModel . . . . .	3
3.7	Interface CommunityChanges . . . . .	4
3.8	Model-View-Controller(MVC) . . . . .	4
<b>4</b>	<b>Carregar dispositivos, fornecedores e casas</b>	<b>4</b>
<b>5</b>	<b>Avançar o tempo e Guardar o estado do programa</b>	<b>5</b>
5.1	Avançar o tempo . . . . .	5
5.2	Guardar o estado do programa . . . . .	5
<b>6</b>	<b>Estatísticas acerca do estado do programa</b>	<b>6</b>
<b>7</b>	<b>Alterações no estado do programa</b>	<b>7</b>
<b>8</b>	<b>Auto-simulação</b>	<b>8</b>
<b>9</b>	<b>Conclusão</b>	<b>8</b>

## 1 Introdução

Neste trabalho, foi nos proposto um sistema que monitorize e registre a informação sobre o consumo energético das habitações de uma comunidade. No nosso projeto, como pedido, existe em cada casa, um conjunto muito alargado de dispositivos (SmartDevices) como as SmartBulb, as SmartCamera e os SmartSpeaker. Cada um destes dispositivos possui características particulares, mas todos podem ser ligados e desligados e têm associado o seu consumo energético. O consumo energético de cada um pode variar dependendo do seu estado de funcionamento. Para além disto, as casas agrupam estes dispositivos em divisões e controlam os mesmos, e estas são agrupadas em uma comunidade com os respetivos fornecedores disponíveis.

## 2 Diagrama de classes - Compacto

Nesta secção apresentamos uma versão compactada da nossa estrutura de dados para contextualização e apresentação da arquitetura do nosso projeto.

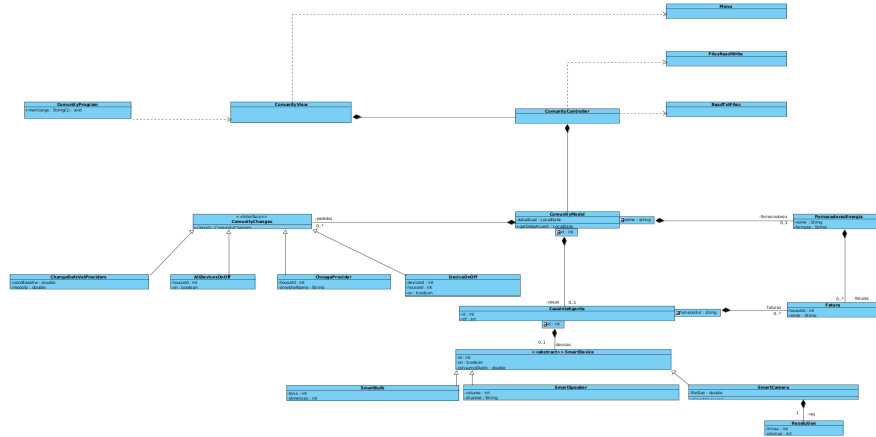


Figura 1: Class Diagram

## 3 Classes

### 3.1 SmartDevice

Esta classe representa os Smart Devices presentes na casa, e estes por sua vez têm como atributos o identificador único do device, o seu consumo diário, o custo instalação e o atributo que indica se está ligado ou desligado.

### 3.2 SmartBulb, SmartSpeaker e SmartCamera

**SmartCamera:** As Smart Cameras, para além dos atributos normais dos Smart Devices, têm como atributos adicionais a sua resolução e o tamanho do ficheiro onde é gravada a informação. O consumo energético é calculado em função do tamanho do ficheiro que geram multiplicado pela resolução da imagem. A classe Resolution apresenta como variáveis de instância as linhas e colunas.

**SmartBulb:** Os Smart Bulbs, têm como atributos o tom da lampada e a sua dimensão em cm. O consumo de energia diário é definido pela seguinte fórmula: (consumo diário da lâmpada em watts) \* (fator em função do tipo de luz emitida). O consumo de energia durante um período de tempo em dias é definido pelo seguinte fórmula: (consumo diário dependente das condições de utilização da lâmpada) \* (período do consumo em dias).

**SmartSpeaker:** Os Smart Speakers, apresentam como variáveis de instância o volume, o canal e a marca. O consumo de energia diário é definido pela seguinte fórmula:  $(\text{consumo diário da coluna em watts}) + 0.02 * \text{volume}$ . O consumo de energia durante um período de tempo em dias é definido pelo seguinte fórmula:  $(\text{consumo diário dependente das condições de utilização da coluna}) * (\text{período do consumo em dias})$ .

**Nota:** O consumo existente em cada dispositivo é inserido pelo utilizador e é considerado como o consumo diário em estado de utilização normal.

### 3.3 CasaInteligente

A classe CasaInteligente apresenta um dono, um nif associado à casa, um id, um Map de devices, cuja key é um inteiro representando o identificador único do device, associada a um Smart Device, um Map de localizações, cuja chave é o nome da divisão da casa que está associada a uma lista com os códigos dos devices. Também apresenta como atributo um fornecedor, e um Map de faturas, que associa faturas a um fornecedor. Escolhemos identificar a casa com um identificador único, gerado automaticamente, por facilidade de execução de queries e para a sua referenciação.

### 3.4 FornecedoresEnergia

A classe FornecedoresEnergia apresenta como variáveis de instância o nome do Fornecedor, a formula para determinar o calculo do preço da energia, e uma lista de faturas associadas a esse fornecedor. A fórmula é definida no momento de criação do fornecedor. Caso não seja passada a fórmula, é atribuída uma fórmula geral por nós definida  $((\text{consumo}/1000)*\text{valorBaseKw}*(1+\text{imposto}))$ .

### 3.5 Faturas

A classe Fatura tem como atributos principais o número da casa a que pertence a fatura, o nome do dono da casa, o nif da casa, o fornecedor associado à fatura, a data desde onde foi aplicada a fatura tal como a data até onde foi aplicada, o consumo especificado na fatura e o custo associado.

### 3.6 ComunityModel

Esta é a classe que faz a gestão das casas e fornecedores existentes no programa. No programa, esta classe faz o controlo a nível de mudança do estado interno dos objetos existentes. Possui um Map dos fornecedores, onde a key é o nome do fornecedor, e um Map das casas onde a key é o id da casa respetiva. Para além disto, possui a data atual em que o programa se encontra e uma lista de pedidos a executar.

### 3.7 Interface CommunityChanges

Esta interface está associada com os pedidos, as alterações que podem ser requeridas e para isso agrupamos as seguintes classe AllDevicesOnff, DeviceOnoff, ChangeProvider e ChangeDefsValsProvider para podermos listar os 4 tipos de pedidos diferentes, ligar todos os devices de uma casa, ligar um dispositivo de uma casa, alterar um fornecedor de uma casa, e alterar os valores base de um fornecedor.

### 3.8 Model-View-Controller(MVC)

No nosso projeto o MVC está representado pelas seguintes classes: Community-Model, CommunityView e CommunityController. Implementamos a classe CommunityView de forma a não conhecer qualquer tipo dos objetos existentes no nosso modelo, assim sendo apenas gere os menus principais (criados a partir da classe Menu). Já o controller é chamado a partir da view para executar as opções existentes nos menus e enviar ao model os métodos que devem ser aplicados.

## 4 Carregar dispositivos, fornecedores e casas

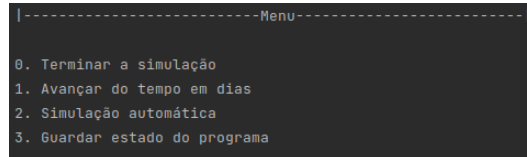
```
|-----|
|  Programa de Gestão Energética de uma Comunidade  |
|-----|
|-----Menu-----|
0. Carregar através do menu
1. Carregar através de um ficheiro de texto
2. Carregar através de um ficheiro com o estado do programa
|-----|
```

Figura 2: Modos de carregamento de dispositivos, fornecedores e casas

No nosso projeto, possuímos algumas formas de fazer o carregamento de dispositivos, fornecedores e casas. Como se pode ver na figura 1, a primeira forma de carregamento dos dados é através do próprio menu do programa, que depois de selecionada a opção fica à espera de input do utilizador. A segunda forma de carregamento é através de um ficheiro de texto, sendo essa a forma mais usual de se carregar os dados, sendo que o utilizador só necessita de escrever os dados no ficheiro e utilizá-lo quando quiser. A terceira forma de carregamento dos dados é através de um ficheiro binário que tenha guardado um estado do programa. Nestes dois ultimos casos, o menu irá pedir o nome do ficheiro logo de seguida da escolha da opção.

**Nota:** O input esperado na primeira opção é uma linha de cada vez e com a mesma estrutura do ficheiro usado na opção 2.

## 5 Avançar o tempo e Guardar o estado do programa



```
|-----Menu-----|
0. Terminar a simulação
1. Avançar do tempo em dias
2. Simulação automática
3. Guardar estado do programa
```

Figura 3: Opções de avançar o tempo e guardar estado do programa

### 5.1 Avançar o tempo

O avançar no tempo é a fase mais importante do nosso programa. Para que ocorra o avançar do tempo é necessário passar o número de dias a avançar. Em seguida, é calculado o consumo para todas as casas nesse período e passada uma fatura para cada casa que é guardada tanto na mesma casa quanto no fornecedor que a passou. Se a fatura passada for a primeira dessa casa é adicionado ao total o custo de instalação dos dispositivos da casa em questão.

De cada vez que avançamos o tempo, são executados os pedidos (alterções) que foram passados no avançar do tempo anterior, ou seja, o programa segue a seguinte estrutura:

Avançar no tempo –> Queries –> Alterações –> Avançar no tempo –> Execução das Alterações –> Queries –> Novas Alterações –> ... .

### 5.2 Guardar o estado do programa

É possível a certo momento da simulação, guardar o estado do programa, ou seja o estado das Casas, Fornecedores e Smart Devices vai ser conservado num ficheiro em binário.

## 6 Estatísticas acerca do estado do programa

```
|-----Menu-----|
0. Avançar
1. Qual a casa que mais gastou no período decorrido
2. Qual o comercializador com maior volume de faturação
3. Lista das faturas emitidas por um comercializador
4. Os n maiores consumidores durante um período
```

Figura 4: Queries disponíveis para fazer estatísticas acerca do estado do programa

Depois de ser feito o avanço no tempo, o utilizador pode pedir ao programa que faça estatísticas sobre o programa, podendo escolher uma de 4 opções:

- **Querie 1 (Qual a casa que mais gastou no período decorrido)**  
- Para a realização desta querie, criamos um método `casaQueMaisGastouUltimoPeriodo` na classe `CommunityModel`, que é chamado no método `queriesFase` da classe `CommunityController`, quando é escolhida a primeira opção. De modo a conseguir a casa que mais gastou no período decorrido, temos de percorrer a lista de fornecedores, e para cada fornecedor retirar as faturas cuja data limite corresponda à data atual e colocá-las num `TreeSet` de faturas, ordenando as faturas por ordem decrescente de consumo. De seguida é só retirar o número da casa associado à primeira fatura do `TreeSet`, e se a fatura não existir deve ser dada uma `FaturaException`, com a mensagem de que a fatura não existe.
- **Querie 2 (Qual o comercializador com maior volume de faturação)**  
- Para a realização desta querie foi feito um método `fornecedorMaiorFaturacao` na classe `CommunityModel`, que é chamado no método `queriesFase` da classe `CommunityController`, quando é escolhida a segunda opção. Para conseguir o fornecedor com a maior faturação, adicionamos todos os fornecedores a um `TreeSet` que os ordena decrescentemente pela soma dos ganhos de cada fornecedor nas faturas. De modo a conseguir esse fornecedor, basta retirar o primeiro elemento do Set de fornecedores e retornar o seu nome.
- **Querie 3 (Lista das faturas emitidas por um comercializador)** -  
Esta querie é realizada no método `faturasComercializador` da classe `CommunityModel`, que é chamado no método `queriesFase` da classe `CommunityController`, quando é escolhida a terceira opção. O método `faturasComercializador` acede ao map de fornecedores e retorna as faturas dos fornecedores associado ao nome do fornecedor associado.
- **Querie 4 (Os n maiores consumidores durante um período)** -  
Esta querie é realizada no método `ordenaMaioresConsumidores`, que é

chamado no método `queriesFase` da classe `CommunityController`, quando é escolhida a quarta opção. Para conseguir ordenar o top n de casas mais consumidoras, utilizamos um `TreeSet` ordenado por ordem decrescente de consumo onde adicionamos todas as casas pertencentes ao map das casas, e de seguida adicionamos a um `ArrayList` o top n de casas mais consumidoras e retornamos essa lista.

## 7 Alterações no estado do programa

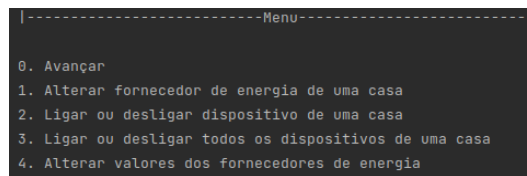


Figura 5: Alterações disponíveis ao estado do programa

No nosso projeto como pedido, é possível fazer as seguintes alterações:

- **Alterar fornecedor de energia de uma casa** - Quando o utilizador escolhe esta opção será pedido que indique o identificador da casa onde quer fazer a alteração e de seguida será pedido que indique o nome do novo fornecedor da casa. Se o id da Casa não existir, devolve uma exceção de que a Casa não existe, e se o nome do fornecedor não existir, então o programa devolve uma exceção de que o Fornecedor não existe.
- **Ligar ou desligar dispositivo de uma casa** - Quando o utilizador escolhe esta opção, então será pedido que indique o id da casa onde se vai fazer a alteração, e se essa casa existir então será pedido o nome da divisão onde se encontra o dispositivo. Dado o nome da divisão, e se essa existir então será pedido o id do device a ligar/desligar, fazendo assim as devidas alterações.
- **Ligar ou desligar todos os dispositivos de uma casa** - Quando é escolhida esta opção, será pedido o id da casa a fazer as alterações, e se a casa existir então é pedido que ligue ou desligue todos os dispositivos da casa.
- **Alterar valores dos fornecedores de energia** - Quando o utilizador escolhe esta opção será pedido que diga o novo valor base por kw e o novo valor do imposto, para aplicar a todos os fornecedores.



## 8 Auto-simulação

```
-- As datas são representadas sempre no seguinte formato: (yyyy.MM.dd)
~~~

-- Opções de alterações nas casas e/ou fornecedores
data, idCasa, idDispositivo, ligar
data, idCasa, idDispositivo, desligar
data, idCasa, ligar
data, idCasa, desligar
data, idCasa, novoFornecedor
data, nomeFornecedor, valorBaseKw, imposto

-- Como avançar no tempo
~~~
Avançar, dataInicio, diasAvançar
```

Figura 6: Modo de auto-simulação

A forma usada para automatizar a simulação permite realizar todas as operações possíveis no nosso programa tanto em casas e fornecedores. Para avançar no tempo, é necessário usar a forma indicada na Figura 3 com a data de início e o número de dias a avançar.

**Nota:** As datas apresentadas no ficheiro para automatizar a simulação seguem uma ordem crescente, ou seja, a data da próxima linha no ficheiro será sempre igual ou superior à anterior.

## 9 Conclusão

Neste trabalho, implementamos um programa de gestão de dispositivos de uma comunidade, permitindo obter um maior conhecimento acerca da linguagem Java, da forma como utilizar classes abstratas e suas subclasses, interfaces, uma visão simples do modelo MVC, e também da elaboração de diagramas de classe. Penso que cumprimos com os requisitos propostos com especial atenção ao encapsulamento seguindo uma estratégia de composição.