



**Universidade do Minho**  
Escola de Engenharia

# Programação Orientada aos Objetos

Trabalho Prático

grupo 17

2ºano - 2ºsemestre



Joana Branco (A96584)



Joana Pereira (A97588)



Marta Sá (A97158)

Braga, 21 de maio de 2022

## Introdução

O problema consiste na construção de um programa que seja capaz de controlar e registrar todo o tipo de informação referente aos diversos dispositivos presentes numa variedade de casas inteligentes de forma a respeitar o encapsulamento, a abstração de implementação e a capacidade de evoluir a aplicação de forma controlada.

Na constituição de cada casa existem divisões compostas por diferentes tipos de dispositivos únicos, sendo que cada um deles é caracterizado de maneira diferente e deve ser possível fazer o controle de ligar e desligar o mesmo. Para além disso, o consumo energético também é algo calculado de acordo com o desempenho do dispositivo e há um custo de instalação associado a cada um deles.

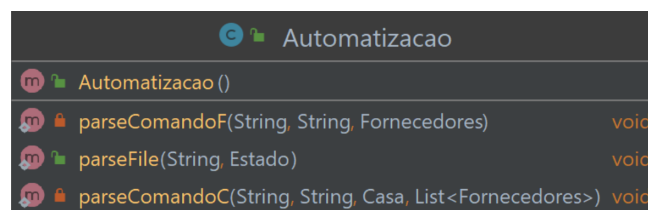
Desta forma, o nosso projeto baseia-se numa estratégia de composição.

## Classes

A divisão dos diferentes aspetos a considerar neste problema foi concretizada aquando do desenvolvimento do projeto, ou seja, de acordo com a necessidade de implementar uma nova funcionalidade criou-se uma nova classe.

Através de métodos que permitem o parsing de um ficheiro, fomos capazes de automatizar o programa de forma a executar algumas funcionalidades tais como: ligar/desligar um dispositivo de uma específica casa, ligar/desligar todos os dispositivos numa divisão de uma casa, mudar o comercializador de energia de uma casa e ainda alterar o desconto e a fórmula de consumo de energia de cada um deles. Para cumprir este mesmo objetivo foi criada a classe **Automatizacao**. É importante realçar que estas funcionalidades só são executadas se a data associada às mesmas forem posteriores à data atual do programa, o que contribui para a consistência do mesmo. A data inicial escolhida foi 01/01/2018.

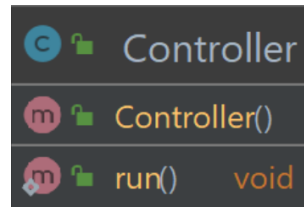
A classe referida foi implementada da seguinte forma:



Quanto ao nível interativo e estético do programa, foram desenvolvidas as classes **Controller** e suas filiações. São nestas referidas anteriormente onde, enquanto corre o Estado que vai ser explicado mais à frente, é possível comunicar entre as classes que fazem parte de um todo. Assim sendo, Casa, Estado, Estatística, Fornecedores e Simulação são menus onde é executável todos os componentes respetivos às mesmas e onde se pode

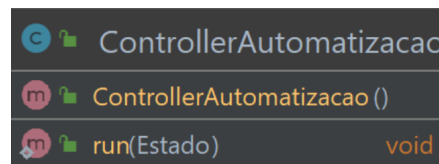
interagir com o programa. Conforme o Controller desejado, vão ser lidas as linhas do ficheiro inicial de input, as opções escolhidas pelo utilizador e o conteúdo inserido pelo mesmo. As classes deste tipo são: **ControllerAutomatizacao**, **ControllerCasa**, **ControllerEstado**, **ControllerEstatistica**, **ControllerFornecedores** e **ControllerSimulacao**.

Como a classe foi implementada:



		Controller
		Controller()
		run() void

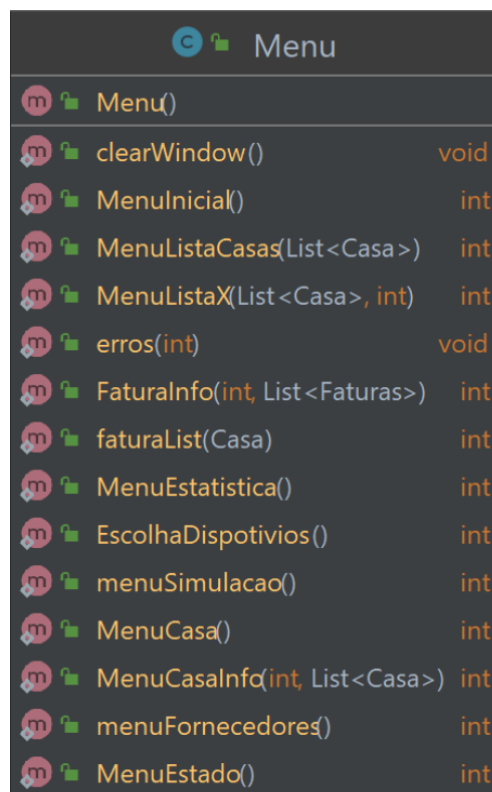
Um exemplo do ControllerAutomatizacao:



		ControllerAutomatizacao
		ControllerAutomatizacao ()
		run(Estado) void

A classe **Menu** trata da parte visual do programa, onde é permitido ao utilizador escolher entre as diversas opções, retroceder nas mesmas e tomar decisões. Aqui foram implementados os menus relativos a cada parâmetro para avaliação e mencionado no parágrafo anterior. Para além disto, foi implementado o método que dá a mensagem especificada de cada erro, de acordo com o problema.

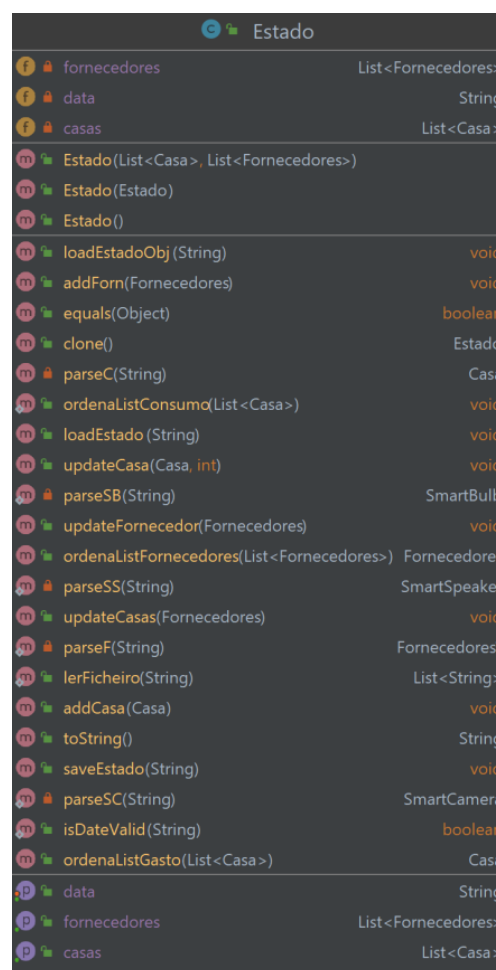
Estrutura da classe referida:



		Menu
		Menu()
		clearWindow() void
		MenuInicial() int
		MenuListaCasas(List<Casa>) int
		MenuListaX(List<Casa>, int) int
		erros(int) void
		FaturaInfo(int, List<Faturas>) int
		faturaList(Casa) int
		MenuEstatistica() int
		EscolhaDispositivos() int
		menuSimulacao() int
		MenuCasa() int
		MenuCasaInfo(int, List<Casa>) int
		menuFornecedores() int
		MenuEstado() int

De maneira a facilitar o processo do input com a informação sobre diversas casas implementou-se a classe **Estado**. O ficheiro de input é lido e o seu conteúdo vai ser organizado e direcionado para as restantes classes por forma a que o acesso à informação seja disponibilizado, basicamente inicializando ou adicionando a um conjunto de dados. Também é permitida a memorização de acordo com as atualizações feitas (por exemplo, adicionar uma casa, ligar um dispositivo, etc). A interface Serializable é a que se implementou pois é pretendido guardar a informação num ficheiro. Neste caso, esta informação seria atualizada no ficheiro original de input.

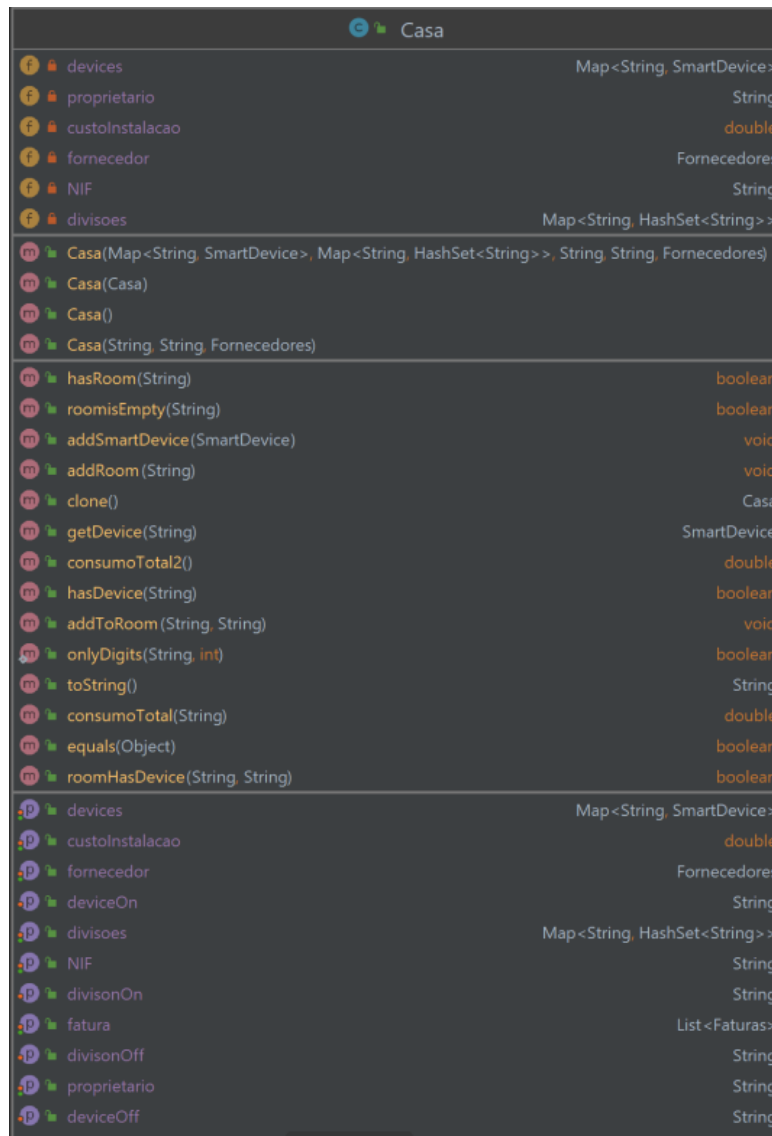
Como a classe foi implementada:



A classe **Casa** procura fazer a gestão com todas as informações relativas a cada uma das casas inteligentes tais como, as suas divisões, dispositivos, número de contribuinte, nome do proprietário, fornecedor mas também, o seu consumo energético. Também aqui era útil e necessário implementar a interface Serializable. Quanto às estruturas de dados desenvolvidas nesta casa, pode-se afirmar que se baseiam num HashMap. Os dados

relativos aos dispositivos estão organizados num HashMap onde a chave é o id do dispositivo e o valor associado é o próprio dispositivo. Já as divisões também seguem a mesma estrutura em que a chave é o nome da divisão e o valor associado é um HashSet de strings com os id's dos dispositivos daquela divisão.

A classe referida foi implementada da seguinte forma:



Da referida anteriormente “derivam” assim as classes **SmartDevice** e, por consequente, as **SmartBulb**, **SmartSpeaker** e **SmartCamera**. A **SmartDevice** reúne os dispositivos, identificando-os com um id único e qual o estado deles, se está ligado ou não, só depois os diferenciando pelos diferentes tipos. Em cada uma das classes relativas às lâmpadas, altifalantes e câmaras são apresentadas as diferentes características de cada um deles tais como, o tom e dimensão, volume, canal e marca e, a resolução e tamanho. Na classe generalizada também foram criados métodos que alteram o estado de um dispositivo assim como, permitem saber o consumo de energia do mesmo.

### Estrutura da classe SmartDevice:

SmartDevice		
f	on	boolean
m	SmartDevice(SmartDevice)	
m	SmartDevice(String, boolean)	
m	SmartDevice()	
m	toString()	String
m	clone()	SmartDevice
m	consumoEnergia()	double
m	equals(Object)	boolean
m	hashCode()	int
p	ID	String
p	on	boolean

### Classes “derivadas” de SmartDevice:

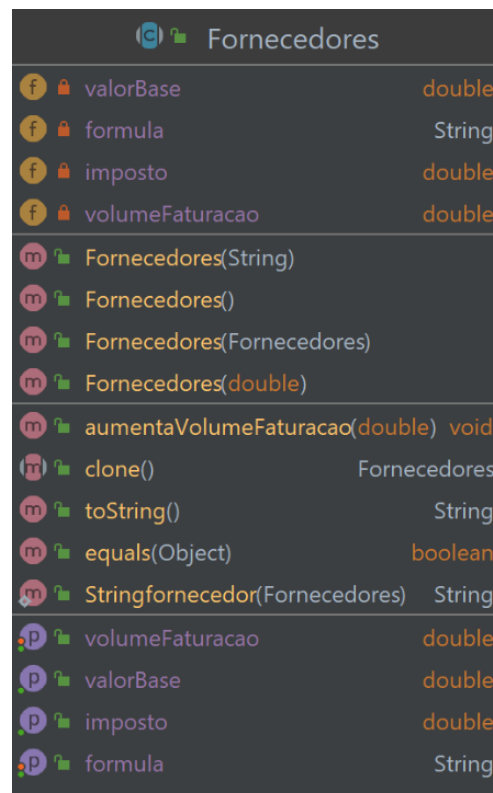
SmartBulb		
f	tone	int
f	dimensao	double
m	SmartBulb()	
m	SmartBulb(String, boolean, int, double)	
m	SmartBulb(SmartBulb)	
m	toString()	String
m	clone()	SmartBulb
m	hashCode()	int
m	equals(Object)	boolean
m	consumoEnergia()	double
p	dimensao	double
p	tone	int

SmartSpeaker		
f	volume	int
f	marca	String
f	channel	String
m	SmartSpeaker()	
m	SmartSpeaker(SmartSpeaker)	
m	SmartSpeaker(String, boolean, int, String, String)	
m	SmartSpeaker(int, String, String)	
m	toString()	String
m	equals(Object)	boolean
m	volumeDown()	void
m	volumeUp()	void
m	clone()	SmartSpeaker
m	hashCode()	int
m	consumoEnergia()	double
p	volume	int
p	channel	String
p	marca	String

SmartCamera		
f	resolution	double
f	size	double
m	SmartCamera(SmartCamera)	
m	SmartCamera()	
m	SmartCamera(double, double)	
m	SmartCamera(String, boolean, double, double)	
m	consumoEnergia()	double
m	hashCode()	int
m	equals(Object)	boolean
m	clone()	SmartCamera
m	toString()	String
p	size	double
p	resolution	double

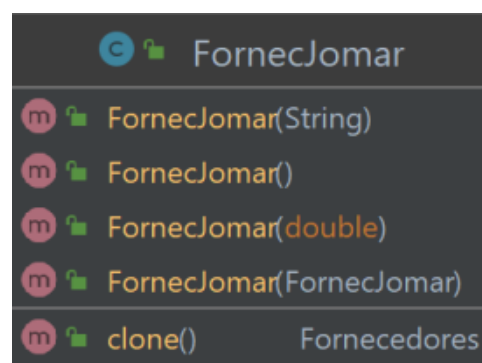
De modo a definir os comercializadores de energia criou-se a classe **Fornecedores**. Aqui foram fixados o valor base e o imposto e desenvolvida a fórmula de consumo de energia a qual é diferente para cada um dos fornecedores. Como os fornecedores têm demasiados métodos em comum, representamos a classe Fornecedor como abstrata. Isto permite criar uma hierarquia. Desenvolveram-se sub-classes de Fornecedor: **FornecEDP**, **FornecEndesa** e **FornecJomar**, tendo em conta que existem três diferentes empresas em que o consumo de energia para cada uma delas vai ser diferente.

Como a classe foi implementada:



Fornecedores		
f	valorBase	double
f	formula	String
f	imposto	double
f	volumeFaturacao	double
<hr/>		
m	Fornecedores(String)	
m	Fornecedores()	
m	Fornecedores(Fornecedores)	
m	Fornecedores(double)	
<hr/>		
m	umentaVolumeFaturacao(double)	void
m	clone()	Fornecedores
m	toString()	String
m	equals(Object)	boolean
m	Stringfornecedor(Fornecedores)	String
<hr/>		
p	volumeFaturacao	double
p	valorBase	double
p	imposto	double
p	formula	String

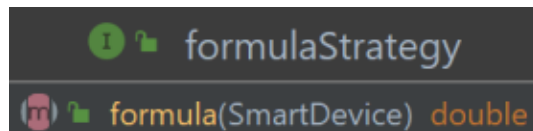
Um exemplo do FornecJomar, um dos tipos de Fornecedores:



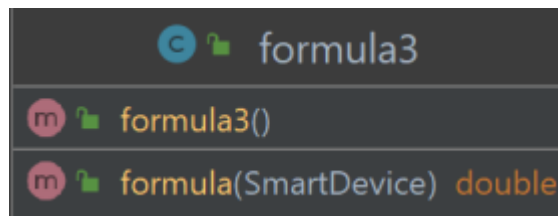
FornecJomar		
m	FornecJomar(String)	
m	FornecJomar()	
m	FornecJomar(double)	
m	FornecJomar(FornecJomar)	
<hr/>		
m	clone()	Fornecedores

As classes do tipo Formula tem como objetivo criar diferentes fórmulas para os diferentes fornecedores. Cada um deles associa o valor base e o imposto com o consumo de energia de cada uma das casas. Com a implementação da interface **formulaStrategy**, é possível admitir que o utilizador possa alterar os valores da fórmula e, assim, alterar a fórmula de cada um dos fornecedores. As classes deste tipo são: **formula1**, **formula2**, **formula3**.

Estrutura da interface:

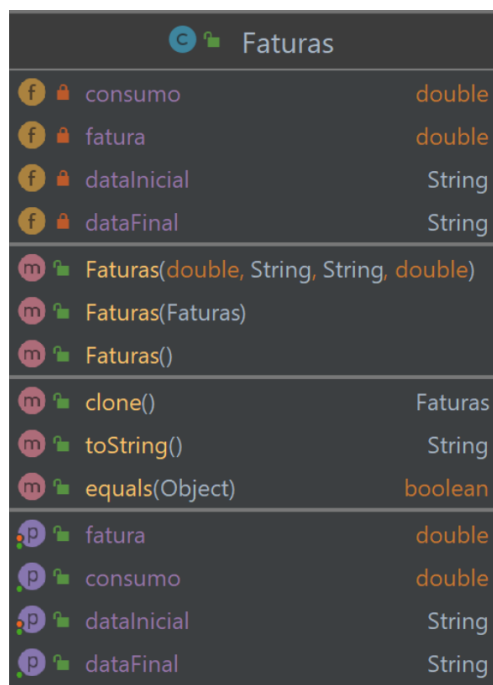


Um exemplo de uma das fórmulas possível:



Para fazer a simulação de todo o processo, envolvendo os fornecedores e a casa, criou-se a classe **Faturas**. Neste caso, para se poder guardar o output num ficheiro implementou-se a interface Serializable. Dadas uma data inicial e outra final, é possível fazer a emissão de uma fatura inserindo-a numa lista de faturas pertencente a cada casa. A data final de uma certa fatura será a data inicial da fatura seguinte caso se efetue o avançar do tempo.

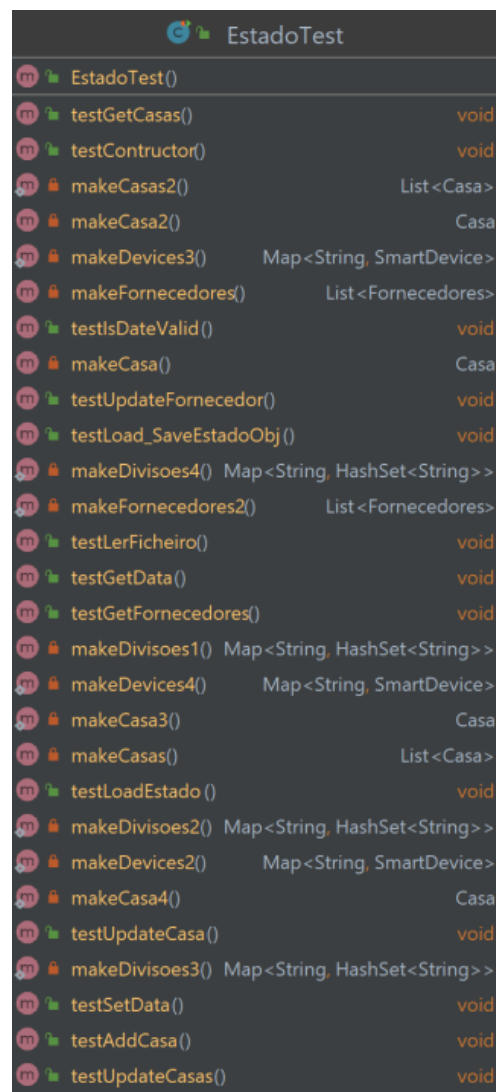
A classe referida foi implementada da seguinte forma:





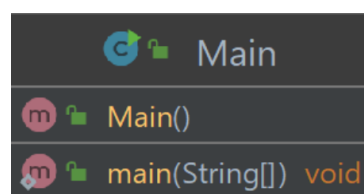
Já as classes do tipo **Test** foram criadas de maneira a ser verificado cada um dos métodos elaborados nas suas respectivas classes. Desta forma, é possível provar que métodos foram corretamente inicializados e definidos e de que cálculos foram bem efetuados pela máquina, entre outros. Também se confere se os métodos implementados estão de acordo com os tipos das variáveis e dos seus parâmetros, o que permite que o trabalho e o programa faça sentido e funcione como esperado. As classes deste tipo são: **FaturaTest**, **EstadoTest**, **FornecedoresTest**, **SmartDeviceTest**, **SmartSpeakerTest**, **SmartBulbTest** e **SmartCameraTest**.

Exemplo de um Test, neste caso, relativa ao Estado:



Juntando todas as classes anteriormente referidas, executamos o programa com a classe **Main**.

Estrutura da classe:



## **Diagrama**

A estrutura de cada uma das classes foi mostrada e explicada no tópico Classes de forma a ser mais legível. Algumas classes foram omitidas porque se tornaria informação redundante. Na figura apresentada no Anexo (última página) é possível verificar as ligações entre as classes e como estas se relacionam. A imagem do diagrama também pode ser consultada na pasta “images”.

## **Desenvolvimento**

A execução do programa baseia-se em diferentes passos sequenciais em que, em algumas situações, uns dependem de outros.

Inicialmente é necessário carregar o ficheiro original de input com a informação de todas as casas, no **Menu Estado**. Sem este passo, não é possível escolher as outras opções como carregar um novo ficheiro ou salvar estado. De seguida é permitido o acesso ao Menu Fornecedores, Menu Casa e Simulação.

O **Menu Fornecedores** admite a informação dos três fornecedores, no nosso caso, permite a modificação da fórmula de cálculo do consumo de energia e a visualização das fórmulas existentes.

No **Menu Casa** está disponível a lista com a informação das casas, é possível escolher a opção de ligar/desligar dispositivos, assim como, toda uma divisão. Também é permitido criar uma casa, adicionar divisões e dispositivos e, também, alterar o fornecedor de energia de uma certa casa. De ter em conta que, os inputs inseridos pelo utilizador devem respeitar a estrutura e o tipo pedido. Em específico, se pretender alterar o fornecedor, esta operação só será realizada se já tiverem passado 30 dias desde a data em que aderiu a esse dado fornecedor. É de evidenciar que estas alterações são momentâneas mas só são contabilizadas e efetivamente úteis depois de ser emitida a fatura.

Já no **Menu Simulacao** somos capazes de visualizar faturas. As faturas só podem ser analisadas depois de haver a emissão das mesmas. E, assim, se segue para o Menu Estatísticas. É importante realçar, se forem emitidas mais que uma fatura, as estatísticas vão ser referentes à última. De relembrar que o input deve respeitar o formato indicado.

O **Menu Estatísticas** contém a casa com o maior consumo de energia num dado período de tempo, o fornecedor com maior valor de faturação, as faturas emitidas por um fornecedor no estado atual, os consumidores com maior gasto durante um determinado período de tempo e fornecedor com maior valor de faturação, também durante um período de tempo.

É de destacar que em cada um dos Menus há a liberdade de escolher voltar para trás e, ainda, no Menu Inicial pode-se sair com “0”.

### Exemplo de Execução:

Assumindo que carregou-se o ficheiro original e, seleccionando 2) Menu Casa.

```
-----  
                        MENU CASA  
-----  
1) Lista de Casas.  
2) Ligar/Desligar dispositivo.  
3) Ligar/Desligar todos os dispositivos de uma divisão.  
4) Criar casa.  
5) Adicionar divisões.  
6) Adicionar dispositivos.  
7) Alterar fornecedor.  
0) Voltar atrás.  
-----  
Selecione a opção pretendida:  
|
```

Escolhendo a opção 1) Lista de Casas e de seguida o índice da casa, neste caso, escolheu-se a casa 134.

```
-----  
                        CASA INFO  
-----  
Casa {  
  
Sala de Estar -> Dispositivos:  
SmarSpeaker (2811611,OF): Marca-> Philips Channel-> RTP Antena 1 98.3 FM Volume-> 70 Consumo Diário-> 20.0  
SmarSpeaker (888010,OF): Marca-> Sony Channel-> TSF Radio Noticias Volume-> 26 Consumo Diário-> 20.0  
SmarCamera (3993490,OF): Resolução-> 3840.0 Tamanho-> 2160.0 Consumo Diário-> 100.0  
SmartBulb (2382854,OF): Tone-> 1 Dimensao-> 14.67 Consumo Diário-> 1.0  
SmartBulb (1327351,OF): Tone-> 1 Dimensao-> 0.21 Consumo Diário-> 1.0  
SmarSpeaker (6971941,OF): Marca-> Goodis Channel-> RFM Oceano Pacifico Volume-> 4 Consumo Diário-> 20.0  
SmarSpeaker (322356,OF): Marca-> Sennheiser Channel-> M80 Radio Volume-> 50 Consumo Diário-> 20.0
```

Relembrando que 0) faz voltar para o menu anterior. Escolheu-se ligar a SmartBulb com o id 2382854 que está na divisão Sala de Estar. Para isto, no Menu Casa selecciona-se 2).

```
-----  
                        MENU CASA  
-----  
1) Lista de Casas.  
2) Ligar/Desligar dispositivo.  
3) Ligar/Desligar todos os dispositivos de uma divisão.  
4) Criar casa.  
5) Adicionar divisões.  
6) Adicionar dispositivos.  
7) Alterar fornecedor.  
0) Voltar atrás.  
-----  
Selecione a opção pretendida:  
2  
-----  
Insira o índice da casa:  
134  
-----  
Insira o id:  
2382854  
-----  
Ligar ou Desligar?  
ligar
```

Verificando se esta última operação foi bem sucedida, visualiza-se novamente a 1) Lista de Casas onde tem a informação relativa à 134.

```
-----  
CASA INFO  
  
Casa {  
  
Sala de Estar -> Dispositivos:  
SmarSpeaker (2811611,OF): Marca-> Philips Channel-> RTP Antena 1 98.3 FM Volume-> 70 Consumo Diário-> 20.0  
SmarSpeaker (888818,OF): Marca-> Sony Channel-> TSF Radio Noticias Volume-> 26 Consumo Diário-> 20.0  
SmarCamera (3993490,OF): Resolução-> 3840.0 Tamanho-> 2160.0 Consumo Diário-> 100.0  
SmartBulb (2382854,ON): Tone-> 1 Dimensao-> 14.67 Consumo Diário-> 1.0  
SmartBulb (1327351,OF): Tone-> 1 Dimensao-> 0.21 Consumo Diário-> 1.0  
SmarSpeaker (6971941,OF): Marca-> Goodis Channel-> RFM Oceano Pacifico Volume-> 4 Consumo Diário-> 20.0  
SmarSpeaker (322356,OF): Marca-> Sennheiser Channel-> M80 Radio Volume-> 50 Consumo Diário-> 20.0
```

Se agora quiser adicionar um dispositivo devo seleccionar 6) no Menu Casa.

```
Selecione a opção pretendida:  
6  
Insira o índice da casa:  
134  
Insira a divisão:  
Sala de Estar  
Off ou On:  
ON  
Insira o ID:  
1234  
-----  
DISPOSITIVO  
  
1) SmartBulb.  
2) SmartCamera.  
3) SmartSpeaker.  
  
-----  
Selecione a opção pretendida:  
3  
Insira a marca:  
Sony  
Insira a rádio :  
123  
Insira o volume :  
30
```

E, verificando novamente se a última operação foi bem sucedida.

```
-----  
CASA INFO  
  
Casa {  
  
Sala de Estar -> Dispositivos:  
SmarSpeaker (2811611,OF): Marca-> Philips Channel-> RTP Antena 1 98.3 FM Volume-> 70 Consumo Diário-> 20.0  
SmarSpeaker (888818,OF): Marca-> Sony Channel-> TSF Radio Noticias Volume-> 26 Consumo Diário-> 20.0  
SmarSpeaker (1234,ON): Marca-> Sony Channel-> 123 Volume-> 30 Consumo Diário-> 20.0  
SmarCamera (3993490,OF): Resolução-> 3840.0 Tamanho-> 2160.0 Consumo Diário-> 100.0  
SmartBulb (2382854,ON): Tone-> 1 Dimensao-> 14.67 Consumo Diário-> 1.0  
SmartBulb (1327351,OF): Tone-> 1 Dimensao-> 0.21 Consumo Diário-> 1.0  
SmarSpeaker (6971941,OF): Marca-> Goodis Channel-> RFM Oceano Pacifico Volume-> 4 Consumo Diário-> 20.0  
SmarSpeaker (322356,OF): Marca-> Sennheiser Channel-> M80 Radio Volume-> 50 Consumo Diário-> 20.0
```

Experimentando agora o Menu Simulacao, escolheu-se 4) no Menu Inicial.

```
-----  
MENU INICIAL  
  
1) Estado.  
2) Menu Casa.  
3) Menu Fornecedores.  
4) Simulação.  
5) Automatização.  
0) Sair.  
  
-----  
  
Selecione a opção pretendida:  
4
```

De notar que é necessário fazer 1) Emissão de Faturas antes do restante e, que a data escolhida no input deve ser superior a 01/01/2018, que é considerada como data atual.

```
-----  
SIMULACAO  
  
1) Emissão faturas.  
2) Faturas.  
3) Estatísticas.  
0) Voltar atrás.  
  
-----  
  
Selecione a opção pretendida:  
1  
  
Insira a data no formato dd/MM/yyyy:  
01/01/2018
```

Pedindo a lista de faturas de uma casa em específico, escolhe-se 2) no Menu Simulacao e posteriormente 134, que é o índice da casa para este exemplo. É possível ver os detalhes de cada uma das faturas optando por uma delas.

```
-----  
LISTAS DE FATURAS  
  
1) Data Inicial: 01/01/2018 Data Final: 21/02/2019  
2) Data Inicial: 21/02/2019 Data Final: 23/07/2020  
0) Voltar atrás  
  
-----
```

Se seleccionar 3) Estatísticas, estas vão ser referentes às faturas que imprimem a partir da data 23/07/2020.

```
-----  
MENU ESTATÍSTICAS  
  
1) Casa com maior gasto no período máximo.  
2) Fornecedor com maior volume de facturação.  
3) Faturas emitidas por um fornecedor no estado atual.  
4) Top x consumidores durante um determinado período.  
5) Fornecedor com maior volume de facturação num determinado intervalo.  
0) Voltar atrás.  
  
-----  
  
Selecione a opção pretendida:  
.  
  
Insira o x:  
.
```

É possível saber os consumidores que mais gastaram num determinado período de tempo, que é o que se obtém na figura seguinte. Também se tem acesso à informação de cada uma das casas se se escolher um dos números indicado na lista.

```
-----  
LISTAS DE CASAS  
  
1) Casa1 -> NIF: 707666276 Proprietário: Joao Pedro Malheiro da Costa  
2) Casa2 -> NIF: 134655929 Proprietário: Miguel Velho Raposo  
3) Casa3 -> NIF: 637220731 Proprietário: Diogo Filipe Duraes Ribeiro  
0) Voltar atrás.  
  
-----
```

## **Conclusão**

Este projeto permitiu desenvolver diferentes capacidades relacionadas com toda a linguagem de Java e o paradigma de programação orientada por objetos.

Tendo como retrospectiva os resultados obtidos pela execução do programa, podemos afirmar que correu tudo, na maior parte das vezes, de forma positiva. Tivemos contacto com diferentes contextos o que permitiu ir melhorando o trabalho, tornando-o mais eficiente.

Achamos que conseguimos cumprir com todos os objetivos propostos no enunciado, nomeadamente no que toca à manutenção do encapsulamento e à capacidade de evoluir o projeto de forma controlada. Apesar de ter sido investido demasiado tempo a adicionar e corrigir detalhes, sentimos que todo o trabalho foi bem executado e estamos satisfeitas com o mesmo.

## Anexo

