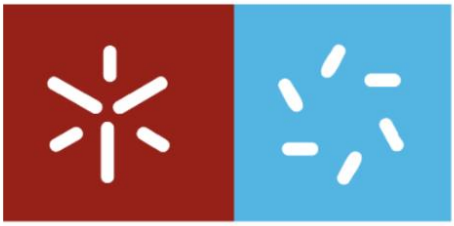

Universidade do Minho

Projeto prático de Programação Orientada aos Objetos

Licenciatura em Ciências da Computação

Grupo n.º 9

2021/2022



Hugo Costa (a96059) Nuno Costa (a97610) Sara Fontes (a92999)

Conteúdo

Introdução e principais desafios

O projeto consiste no desenvolvimento, em Java, de um sistema que monitorize e registe toda a informação relativa ao consumo energético de habitações de uma comunidade.

Consideramos que o maior desafio que enfrentamos durante o desenvolvimento do projeto foi a resolução de alguns erros que foram surgindo e que nos impediam de avançar.

Classes

a. SmartDevice

Começamos por criar uma classe com a qual conseguíssemos ter acesso a métodos específicos das classes de cada um dos três dispositivos em causa neste projeto, como por exemplo:

```
public abstract void turnOn();
public abstract void turnOff();
public String getId();
```

Contém apenas um identificador que nos indica uma **`string`** que identifica um determinado aparelho.

```
private String id;
```

Decidimos neste trabalho, ao criar, definir todos os aparelhos a ON para, se for necessário, poder alterar o seu estado.

b. SmartBulb

SmartBulb é uma classe mais específica da classe SmartDevice, isto é, é uma subclasse onde atribuímos a uma lâmpada um estado (ON ou OFF), uma tonalidade, uma dimensão, um valor de consumo e um valor de preço por instalação. O consumo final calculado para uma SmartBulb varia consoante a tonalidade e o consumo base.

```
private Estado estado; //ON ou OFF
private int n_estado; // usado para converter num Estado
private Tonalidade tone; //NEUTRAL, COLD ou WARM
private double dimensões;
private double consumoB;
private double consumoF;
private double precoInstalacao;
```

c. SmartSpeaker

Tal como a classe anterior, SmartSpeaker é uma subclasse de SmartDevice, onde atribuímos valores a uma coluna inteligente, tais como estado, n_estado, marca, volume, a radio a transmitir e valores de consumo base e preço por instalação. Numa SmartSpeaker o consume varia consoante a marca da coluna e o volume.

```
private Estado estado; //ON ou OFF
private int n_estado; // usado para converter num Estado
private String marca;
private int volume;
private String radioOnline;
private double consumoBase;
private double precoInstalacao;
```

d. SmartCamera

Igualmente, a SmartCamera é uma subclasse de SmartDevice onde atribuímos um estado, n_estado, valores da resolução, tempo ligado, tamanho do pacote de vídeo resultante, valores de consumo, uma data de início e preço por instalação. O seu consumo depende da resolução da imagem e do tamanho do ficheiro que gera.

```
private Estado estado;
private int n_estado; // usado para converter num Estado
private int x;
private int y;
private long tempoLigado;
private double tamanhoPacote;
private double consumo;
private LocalDateTime dataInicial;
private double precoInstalacao;
```

e. Fatura

A classe Fatura contém informação mais detalhada sobre os custos dos aparelhos inteligentes na casa: o identificador da casa, o valor do consumo, o custo e as datas de início e fim de consumo.

```
private String idCasa;
private double consume;
private double custo;
private LocalDateTime dataInical;
private LocalDateTime dataFinal;
```

f. FornecedorEnergia

A classe FornecedorEnergia é responsável pela identificação de cada fornecedor de energia associado às casas, tendo como identificadores: o nome da empresa que fornece energia à casa, o imposto, o valor base que cobra, o desconto, o conjunto de casas que lhe estão associadas e as faturas que geram.

```
public class FornecedorEnergia {
public String nomeEmpresa;
public double imposto;
public double valorBase;
public double desconto;
private Map<String, Casa> conj_Casas; // Id da casa -> Casa
private Map<String, List<String>> faturas; //Id da casa -> faturas dessa casa
```

g. Casa

Nesta classe criamos métodos que “constroem” uma casa, adicionando divisões, dispositivos, entre outros. Eis alguns exemplos:

```
public void turn_On_Casa();
public void add_Divisao(String div);
public void remove_Divisao(String div);
```

Para tal existem os identificadores: identificador da casa, a morada, o nome e o NIF do proprietário, um conjunto de dispositivos associados, um conjunto de divisões e por fim um fornecedor de energia associado a essa casa.

```
private String idCasa;
private String morada;
private String nome; //Nome do proprietário
private String NIF; //NIF do proprietário
public Map<String, SmartDevice> dispositivos; //conjunto de dispositivos
public Map<String, List<String>> divisoes; // Nome da divisão -> Dispositivos
private FornecedorEnergia fornecedor;
```

h. Cidade

A classe Cidade contém um conjunto de casas com vários métodos que permitem obter informações sobre casas e, também a criação desses mesmos conjuntos, como por exemplo:

```
public FornecedorEnergia getFornecedorDeCadaCasa(String idCasa);
public double getConsumoCasaPeriodo(Casa casa, long periodo);
public double volumeFatFornecedor (FornecedorEnergia fornecedor, long
periodo);
public void add_Casa(String idCasa, Casa casa);
```

Os seus identificadores são um conjunto de casas, fornecedores e faturas.

```
public Map<String, Casa> casas = new HashMap<>();
public Map<String, List<String>> fornecedores = new HashMap<>();
public Map<String, List<String>> faturas = new HashMap<>();
```

i. Parser

"Parser" foi adiantada pela equipa docente da disciplina, mas posteriormente alterada adaptada ao nosso código. Esta permite a leitura de um ficheiro com uma listagem de casas com respetivas informações, bem como todos os nomes de fornecedores de energia. Um dos métodos que o constitui é, por exemplo, o método que lê o ficheiro:

```
public List<String> lerFicheiro(String nomeFich);
```

j. Main Menu

O programa tem início com este menu.

1. Menu Criar Cidade

Abre um segundo menu onde temos as seguintes opções:

"1 – Houses" onde podemos criar alterar ou remover uma casa;

"2 – Energy Suppliers" onde podemos acrescentar, alterar ou remover um fornecedor de energia;

"3 – Advance in time" onde podemos criar vários ciclos;

"4 – Simulation" com as respetivas opções:

"1 – House that spent the most in a certain period of time" que indica a casa que gasta mais num dado período de tempo dado como parâmetro;

"2 – Supplier with the highest invoicing volume" indica o fornecedor com maior volume de faturação;

"3 - Invoices list's issued by a supplier" lista todas as faturas emitidas por um fornecedor de energia;

"4 - Ordering of the largest energy consumers during this period" indica a ordem decrescente de maiores consumidores num dado período;

"5 - Save state in object file" guarda o ficheiro;

"6 – Load state in object file" carrega o ficheiro;

"7 – Check state" onde podemos ver as informações de casas, dispositivos, etc.

2. Menu Import a log File

Nesta opção podemos importar um ficheiro log para o programa executar a análise do mesmo.

k. Main

A classe Main é a que dá início a todo o programa, invocando a classe Menu.

Estrutura do Projeto

O nosso projeto segue a estrutura Model View Controller (MVC), estando por isso organizado em três camadas:

- A camada de dados (o modelo) é composta pela Classe SmartDevice, SmartBulb, SmartSpeaker, SmartCamera, Casa, Cidade, Fornecedor de Energia, Faturas.
- A camada de interação com o utilizador (vista, ou apresentação) é composta unicamente pela Classe Menu.
- A camada de controlo do fluxo do programa (o controlador) é composta pelo Main e pela Classe Parser.

Todo o nosso projeto baseia-se na ideia de encapsulamento, e portanto a relação entre classes é de composição.

Diagrama de Classes

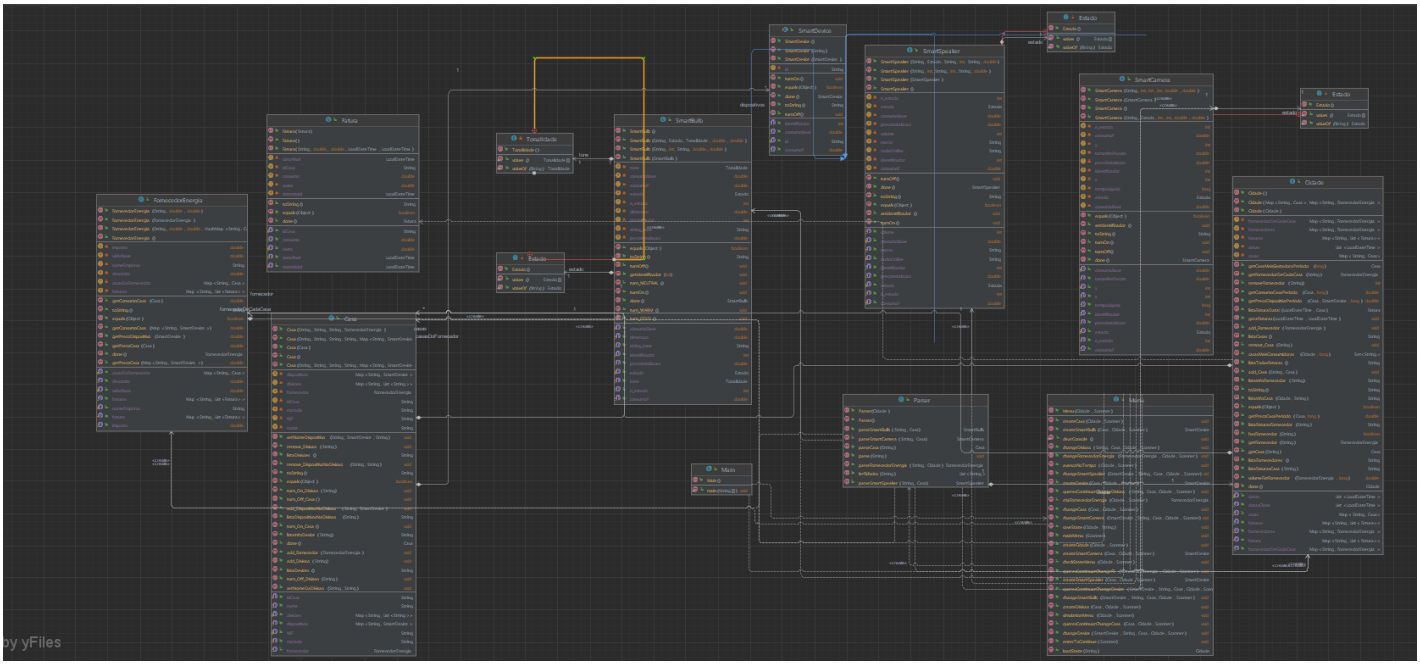

Figura 1: Diagrama de classes do programa, gerado pelo IntelliJ

Conclusão

A nível geral, e tendo em conta o que foi explicado nos capítulos anteriores, como grupo achamos que todos os objetivos foram cumpridos e apesar das dificuldades que fomos encontrando o grupo conseguiu superar de uma forma muito boa, sempre com um olhar crítico e a pensar no próximo passo. Acreditamos que respondemos de forma correta ao problema apresentado pela equipa docente da disciplina.