

Universidade do Minho

UNIVERSIDADE DO MINHO

LICENCIATURA EM CIÊNCIAS DA COMPUTAÇÃO

Relatório do projeto: SmartDevices, SmartHouses e Controlo/Eficiência Energética

POO- Trabalho Prático  
Grupo 10

Maio de 2022

Uma imagem com pessoa, parede, homem, interior

Descrição gerada automaticamenteUma imagem com pessoa, parede, interior, propriedade

Descrição gerada automaticamenteUma imagem com pessoa, parede, vestuário

Descrição gerada automaticamente

Catarina Quintas João Guedes Pedro Martins

A91650 A94013 A91681

Índice

1. **Introdução . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4**

1. **Classes . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5**
   1. SmartDevice . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .5
   2. SmartBulb . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .5
   3. SmartCamera . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
   4. SmartSpeaker . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
   5. Casa. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
   6. FornecedorEnergia . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
   7. Ambient . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
   8. Invoice. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .7
   9. Parser. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .8

2.10.View e Controller. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .8

2.11 Java Docs . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

2.12 JUnit . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

2.13 Menu . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .9

1. **Exceptions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12**
   1. DeviceExistsInDivisionExceptions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
   2. DivisionExistsExceptions. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
   3. HouseNotFoundExceptions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
   4. ParserExceptions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .12
2. **Estrutura do projeto . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13**
3. **Diagrama de classes . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14**
4. **Conclusão. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15**

**Capítulo 1**

**Introdução**

Este projeto consistiu no desenvolvimento de uma aplicação se pretende construir um sistema que monitorize e registe a informação sobre o consumo energético das habitações de uma comunidade na linguagem de programação Java, de forma a colocar em prática os conhecimentos adquiridos ao longo do semestre.

Cada casa é constituída pelo proprietário, Nif, os SmartDevices agrupados por divisão e o fornecedor. Dependendo do seu tipo, o SmartDevice possuí um conjunto de caraterísticas diferentes e estes poderão ser ligados e desligados, registando assim o seu consumo energético. Cada fornecedor tem um preço diferente para a venda da energia de acordo com os seus critérios comerciais e não podem existir casas sem fornecedor.

**Capítulo 2**

**Classes**

**2.1 SmartDevice**

public enum Status{

OFF,

ON

}

private String factoryID; //Código único definido pelo fabricante   
private double mCost; // Custo de instalação   
private Status status; // Status: OFF/ ON do smartDevice   
protected long start; // Ligar o smartDevice  
protected long finish; // Desligar o smartDevice  
protected long timeElapsed; // Tempo percorrido com o SmartDevice ligado

Começamos por fazer uma classe mãe dos aparelhos, onde temos os identificadores gerais representados acima. Com isto podemos proceder à criação de um dispositivo mais genérico onde identificamos o seu FactoryId, mCost, Status entre outros.

**2.2 SmartBulb**

public enum LightMode{

NEUTRAL, ( 10 w )

WARM, ( 15 w )

COLD ( 20 w )   
}

private LightMode mode; // Tonalidade da luz

private int dimension; // Dimensão

private double dailyConsumption; // Consumo diário

”SmartBulb” é uma subclasse de ”SmartDevice”, que cria os SmartDevices do tipo SmartBulb. Nesta subclasse, dependendo da tonalidade (Neutral/Warm/Cold ) da lâmpada, iremos obter diferentes consumos.   
Por exemplo: Considerando que a lâmpada está no modo “Neutral” (10w), o consumo da energia irá ser dado pela fórmula:

* aux = (double)( 15 \* (dimension\* dimension))/1000;
  1. **SmartCamera**

private int resolution; //resolução da camera

private double fileSize; // tamanho do ficheiro

protected double dailyConsumption; // consumo diário da camera

“SmartCamera” é a subclasse de SmartDevice, onde são criados os SmartDevices do tipo SmartCamera, adicionando às variáveis de instância da sua superclasse: a variável resolução da camera, tamanho do ficheiro e o consumo diário da camera

* 1. **SmartSpeaker**

private int volume; // volume da coluna

private String radio; // radio da coluna

private String brand; // marca da coluna

private double dailyConsumption; // consumo diário da coluna

“SmartSpeaker” é a subclasse de SmartDevice, onde são criados os dispositivos do tipo SmartSpeaker. As variáveis de instância desta classe são: volume, radio, marca e o consumo diário.

* 1. **Casa**

private String owner;  
private int NIF;  
private HashMap<String,ArrayList<SmartDevice>>divisions;  
private String provider;

Nesta classe são guardadas todas as informações de uma casa, nomeadamente o nome e o nif único de cada proprietário, os Smartdevices de cada divisão e por último o fornecedor. Para cada divisão, é listada uma sequência dos dispositivos que estão registados nessa mesma divisão.

* 1. **FornecedorEnergia**

private String company;  
private double dailyEnergyCost;  
private double tax;

“FornecedorEnergia” é uma classe que providencia energia para as casas, sendo que cada casa pode escolher o seu próprio fornecedor. Nesta classe existe 2 variaveis de instância do tipo double que correspondem ao iva e o custo de KW/h por dia. Para enriquecer a nossa solução acrescentamos campos para o IVA energético e para o custo de KW/h por dia no ficheiro de logs.txt

A screenshot of a computer

Description automatically generated with medium confidence

* 1. **Ambient**

private Calendar calendar;

“Ambient” é a classe responsável pela gestão do tempo, ou seja, a partir desta classe iremos poder avançar no tempo para deste modo ocorrer a simulação de custos.

**2.8 Invoice**

private String code;  
private Date date;  
private double consumoTotal;  
private FornecedorEnergia Provider;  
private Casa house;  
private ArrayList <String> codeIDs;

“Invoice” é a classe onde irão ser criadas as faturas. Esta classe irá ter toda a informação de uma fatura de energia relativa a uma casa num determinado período de tempo, tendo como informação o código, data, consumo total de energia, fornecedor, casa e o armazenamento de todos os códigos de faturas.

**2.9 Parser**

”Parser” é inspirado no código dado pela equipa de docente da disciplina, mas tendo algumas alterações e adições importantes para o funcionamento da nossa parte para adaptar ao nosso código. Esta classe vai permitir importar os dados de um ficheiro para o programa estando preparada para qualquer eventualidade de erro no ficheiro a ler, tornando assim o ambiente do programa mais seguro e dinâmico.

**2.10 View e Controller**

Onde estão localizados os componentes essenciais para um ambiente MVC (Model-View-Controller), evidenciado nas aulas de POO. O utilizador somente necessita de colocar em funcionamento a class View e o programa tratará do resto. Diminuindo a complexidade para novos users.

**2.11 Java Docs**

Para mais informações foi gerada uma documentação do projeto em inglês onde pode ser consultada a especificação de cada método presente em cada class, facilitando assim o trabalho dos docentes de POO e de futuros utilizadores desta aplicação.

**2.12 JUnit**

Para um código mais seguro contra erros, durante o desenvolvimento da aplicação foi utilizado o JUnit para verificar a existência de qualquer tipo de erros. Mais uma vez, seguindo as recomendações dos docentes de POO.

**2.13 Menu**

Esta classe cria o menu inicial da aplicação. Intuitiva para qualquer utilizador.

Segue abaixo a árvore de comandos do projeto, bem como alguns exemplos de uso.

Text

Description automatically generated

**Simular 15 dias para um cliente, carregando configurações por ficheiro:**

Text

Description automatically generated

Text

Description automatically generated**Exemplo de fatura para esse cliente:**

As faturas são guardadas para ficheiros ‘.txt’ na pasta /**invoices** com o código próprio.

**Text

Description automatically generated with medium confidenceCriação de um ambiente sem necessitar de ficheiro:**

**Capítulo 3**

**Exceptions**

**3.1 DeviceExistsInDivisionExceptions**

Esta exception foi criada no âmbito de quando o dispositivo já está definido e já existe na divisão, alertando o utilizador para este problema.

**3.2 DivisionExistsExceptions**

Esta exception foi criada no âmbito de quando a divisão já está definida e impede a sobreposição de comandos sobre a mesma, alertando o utilizador para este problema.

**3.3 HouseNotFoundExceptions**

Esta exception foi criada no âmbito de quando a casa não está definida ou quando é definida incorretamente, alertando o utilizador para este problema.

**3.4 ParserExceptions**

Esta exception foi criada no âmbito de quando o parser é utilizado incorretamente e/ou a informação dada no ficheiro é definida incorretamente, alertando o utilizador para este problema.

**Capítulo 4**

**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 pelas Classes SmartDevice, SmartSpeaker, SmartBulb, SmartCamera, Casa e ForcenedorEnergia.
* A camada de interação com o utilizador (a vista, ou apresentação) é composta unicamente pela classe View.
* A camada de controlo do fluxo do programa (o controlador) é composta pela classe Controller.

Todo o projeto baseia-se na ideia de facilidade de utilização para o utilizador.

**Capítulo 5**

**Diagrama de Classes**

Graphical user interface

Description automatically generated

Diagrama de classes do programa, gerado pelo IntelliJ

Informações extra podem ser consultadas no JavaDocs presente na pasta **docs/**

**Capítulo 6**

**Conclusão**

Em suma, consideramos que a realização deste projeto foi bastante vantajosa, uma vez que não só nos possibilitou uma melhor consolidação dos tópicos abordados nas aulas, como também permitiu-nos aplicar esses conhecimentos numa aplicação em java de maior escala.  Consideramos, ainda, que conseguimos cumprir os objetivos propostos, no entanto, existem melhorias possíveis a realizar, como por exemplo colocar os dispositivos a ligar/desligar durante a simulação, pois neste momento estes só podem ser ligados/desligados no início ou no final da simulação, bem como a troca de fornecedor por parte das casas.