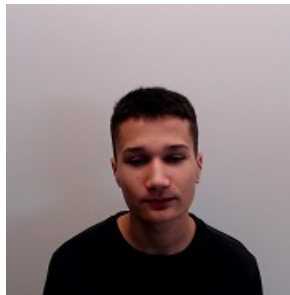

Monitorização e Gestão do consumo energético a habitações

TRABALHO REALIZADO POR:

BEATRIZ RIBEIRO MONTEIRO
BENJAMIM MELEIRO RODRIGUES
BIANCA ARAÚJO DO VALE



A95437
Beatriz Monteiro



A93323
Benjamim Rodrigues



A95835
Bianca Vale

Índice

1	Introdução	1
2	Arquitetura de Classes	1
2.1	GestorConsumoAPP	1
2.2	Model	2
2.2.1	SmartHouse	2
2.2.2	SmartDevices	2
2.2.3	Comercializadores	3
2.2.4	Fatura	3
2.2.5	GestorComunidade	4
2.3	Files	4
2.3.1	Parser	4
2.4	Controller	4
2.4.1	Interpretador	4
2.4.2	Input	4
2.5	View	4
2.5.1	Apresentacao	5
2.5.2	Output	5
3	Descrição da Aplicação	5
3.1	Mensagem de Entrada e Menu Inicial	5
3.2	Menu Principal	6
3.3	Menu Consultas	6
3.4	Menu Alterações	7
3.5	Menu Visualizar Dados	8
4	Conclusão e Reflexão Crítica	8

1 Introdução

No âmbito da Unidade Curricular de Programação Orientada aos Objetos, foi nos proposto a implementação de um programa que monitorize e registre a informação sobre o consumo energético das habitações de uma comunidade.

Os focos prioritários deste trabalho foram a modularidade, o encapsulamento de dados e o código reutilizável que são essenciais para uma boa prática da programação orientada aos objetos.

Ao longo deste relatório vamos explicar com mais detalhe o desenvolvimento das várias componentes da solução implementada para responder ao que o enunciado propunha.

2 Arquitetura de Classes

Para que seja possível desenvolver esta plataforma é necessário organizar os dados numa estrutura que, apesar de ser compacta, seja de rápido e fácil acesso.

As classes elementares da nossa estrutura representam Casas, Comercializadores, Fatura e SmartDevice.

Com o decorrer do projeto, chegamos à conclusão que para que as classes pudessem interagir entre si, seria necessária uma classe que as agregasse e, também uma classe que fosse capaz de interagir com o utilizador. Deste modo, implementamos o modelo MVC, para alcançar a independência das camadas, bem como para conseguir interagir com o utilizador de forma segura.

Desta forma, e com o intuito de controlar o modelo do programa, todos os dados das classes referidas em cima são agregados numa Base de Dados, que tem como nome GestorConsumoAPP.



2.1 GestorConsumoAPP

Esta é a classe do trabalho que incorpora a main que permite a execução do programa. Contém os vários módulos do MVC (*Model View Controller*) e onde se executa o interpretador.

```
1  GestorComunidade gc = new GestorComunidade(); // Model
2  Interpretador i; // Controlador
3  Apresentacao a = new Apresentacao(); // View
4  Parser parser = new Parser(); // Parser
```

2.2 Model

2.2.1 SmartHouse

Esta classe contém a informação de cada Casa Inteligente(*SmartHouse*).

Para guardar as Faturas associadas a uma dada Casa utilizamos uma Lista de Faturas, onde estão guardadas todas as faturas daquela casa.

Cada casa tem um Map onde estão guardados todos os seus dispositivos, sendo a key do Map o ID do dispositivo. Temos ainda outro Map que guarda a lista dos IDs de todos os dispositivos de uma dada divisão da casa, sendo a divisão a key do Map.

Para além disso, cada Casa tem ainda um Fornecedor(Comercializador) associado.

```
1 // Nome do proprietario
2 private String name;
3 // NIF do proprietario
4 private int NIF;
5 // Fornecedor de energia
6 private Comercializador comercializador;
7 // Lista de todas as Faturas da Casa
8 private List<Fatura> faturas;
9
10 // Custo de instalacao
11 private double custos_instalacao;
12
13 // Map com os todos dispositivos da casa
14 private Map<UUID, SmartDevice> devices;
15 // Map todos os dispositivos de uma certa divisao
16 private Map<String, List<UUID>> divisions;
```

2.2.2 SmartDevices

Esta classe contém o ID do dispositivo e o seu estado, isto é, se este se encontra ligado ou desligado. Como esta é uma classe abstrata uma vez que servirá de super-classe para as classes que definem os diferentes dispositivos.

```
1 // ID do dispositivo
2 private UUID id;
3 // Estado(Ligado == true/ Desligado == false)
4 private boolean state;
```

Existem tres tipos de SmartDevices:

SmartBulb Esta classe estende da classe SmartDevice para descrever um dispositivo do tipo lâmpada(SmartBulb) com os atributos que são apresentados abaixo.

```
1 // Definicao dos diferentes modos da lampada
2 private static final int COLD = 0;
3 private static final int NEUTRAL = 1;
4 private static final int WARM = 2;
5 // Custo de instalacao da lampada
6 private static final int custo_instalacao = 1;
7 // Atributos
8 private int tonalidade;
9 private double dimensao;
10 private double consumoDiario;
```

SmartCamera Esta classe estende da classe SmartDevice para descrever um dispositivo do tipo câmera(SmartCamera) com os atributos que são apresentados abaixo.

```
1 // Custo de instalacao da camera
2 private static final int custo_instalacao = 5;
3 // Atributos
4 private int resolX;
5 private int resolY;
6 private double tamanhoFicheiros;
```

SmartSpeaker Esta classe estende da classe SmartDevice para descrever um dispositivo do tipo coluna(SmartSpeaker) com os atributos que são apresentados abaixo.

```
1 // Custo de instalacao da coluna
2 private static final int custo_instalacao = 3;
3 // Atributos
4 private int volume;
5 private String estacaoRadio;
6 private String marca;
7 private double consumoDiario;
```

2.2.3 Comercializadores

Neste package encontram-se 3 classes que servem de template para criar fornecedores de energia.

```
1 private String nome;
2 private double desconto = 0.9;
3
```

```
1 private String nome;
2 private double descontoMenor = 1, descontoMaior = 0.75;
3
```

```
1 private String nome;
2 private double descontoMenor = 1.1, descontoMaior = 0.7;
3
```

2.2.4 Fatura

Esta classe contém a informação de cada Fatura.

```
1 // Intervalo de tempo da fatura
2 private LocalDate inicio, fim;
3 // Consumo de energia
4 private double consumo;
5 // Custo do consumo de energia, Custo de Instalacao
6 private double custos_consumo, custos_instalacao;
7 // Fornecedor que emite a fatura
8 private String comercializador;
```

2.2.5 GestorComunidade

Esta classe é responsável por gerir

```
1 // Map das Casas
2 Map<Integer, SmartHouse> casas;
3 Map<String, Comercializador> comercializadores;
```

2.3 Files

2.3.1 Parser

Responsável pela leitura, tratamento e armazenamento de dados que se encontram no ficheiro de logs.

```
1 public class Parser {
2     public static List<String> readFile(String fileName){...}
3
4     public void parse(GestorComunidade gc) throws
LinhaIncorretaException {...}
5 }
```

2.4 Controller

2.4.1 Interpretador

A classe Interpretador é responsável por interagir com o utilizador e implementa a interface InterfaceInterpretador.

```
1 // Leitura de Input do utilizador
2 private final Input in;
3 // View do Programa
4 Apresentacao ap;
5 // Modelo da Programa
6 GestorComunidade gc;
```

2.4.2 Input

Esta classe tem implementados métodos que permitem a leitura diferentes tipos de variáveis introduzidos pelo utilizador.

```
1 public class Input {
2     public int readInt () {...}
3     public String readline() {...}
4     public double readDouble() {...}
5     public LocalDate readLocalDate() {...}
6     public void close() {...}
7 }
```

2.5 View

2.5.1 Apresentacao

Esta classe implementa a interface InterfaceApp e é responsável por devolver os resultados visuais ao utilizador.

```
1 //Imprime os outputs
2 private final Output out;
3 // Apresenta o responsavel pelos outputs relacionados os
  resultados
4 private ApresentacaoMain ap;
```

2.5.2 Output

A classe Output implementa métodos que permitem passar a informação ao utilizador.

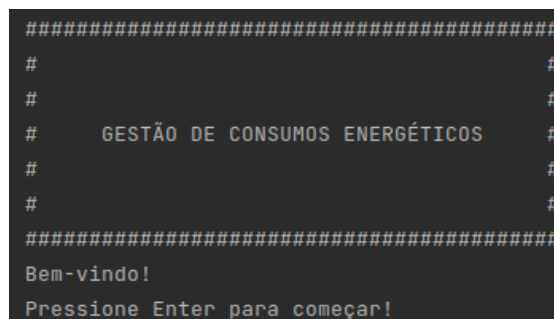
```
1 public class Output {
2     // Apresenta da mensagem com \n no final
3     public void printMessage(String message){...}
4     // Apresenta a mensagem numa linha
5     public void printline(String m){...}
6     // Apresenta a mensagem "Op o N o Disponivel!\n"
7     public void printOpInvalida(){...}
8     // Apresenta a mensagem "A altera o ir ser executada a
  proxima vez que correr a simulacao!\n"
9     public void printNextSimulation(){...}
10    // Template para fazer Menus
11    public void printMenus(String []menu, String message, int type
  ){...}
12 }
```

3 Descrição da Aplicação

O modo de interação com o utilizador escolhido foi uma interface de linha de comandos.

3.1 Mensagem de Entrada e Menu Inicial

Quando o utilizador entra no programa aparece-lhe uma mensagem de boas vindas na linha de comandos e é-lhe pedido para pressionar a tecla Enter para continuar.



```
#####
#
#
#   GESTÃO DE CONSUMOS ENERGÉTICOS   #
#
#
#####
Bem-vindo!
Pressione Enter para começar!
```

Após clicar na tecla pedida o programa avança e aparece o Menu Inicial que permite ao utilizador escolher se pretende

- Carregar os dados de um ficheiro.

-
- Começar uma simulação nova.

```
MENU INICIAL
1 | Carregar Dados
2 | Nova Simulação
9 | Sair
```

Carregar os dados de um ficheiro Carrega os dados guardados num ficheiro indicado pelo utilizador.

Começar uma simulação nova É apresentada uma mensagem que situa temporalmente a simulação na data atual e pede para inserir uma data futura para determinar o avanço do tempo desejado.

```
A simulação encontra-se no dia de hoje (21/05/2022).
Determine o avanço do tempo com o padrão dia/mês/ano!
```

3.2 Menu Principal

Após inserir o avanço do tempo, é apresentado o Menu Principal do programa. Este menu permite ao utilizador escolher se quer carregar ou guardar dados, consultar estatísticas da comunidade, fazer alterações nos dados atuais do programa, visualizar os dados da comunidade, seguir com a simulação, fazendo com que os dados atualizem ou sair, terminando assim o programa.

```
MENU PRINCIPAL
1 | Dar save/load do estado do programa
2 | Consultar estatísticas da vizinhança
3 | Fazer alterações nos dispositivos, fornecedores, casas
4 | Visualizar dados do programa
0 | Continuar Simulação
9 | Sair
Escolha a sua opção:
```

3.3 Menu Consultas

O Menu de Consultas permite ao utilizador

- Consultar a casa que gastou mais dinheiro.
- Consultar qual o fornecedor que teve maior volume de faturação.
- Listar as faturas emitidas por um dado fornecedor.
- Listar os n maiores consumidores de energia durante um dado período.

```
MENU CONSULTAS
1 | Casa que mais gastou
2 | Fornecedor com maior volume de faturação
3 | Listar as faturas emitidas por um fornecedor
4 | Listar os maiores consumidor de energia durante um periodo
Escolha a sua opção:
```

Consultar a casa que gastou mais dinheiro É apresentada a casa que mais dinheiro gastou juntamente com a quantidade de energia que consumiu.

Consultar qual o fornecedor que teve maior volume de faturação É apresentado o fornecedor que teve o maior volume de faturação

Listar as faturas emitidas por um dado fornecedor É apresentada uma lista de todas as faturas emitida pelo fornecedor inserido pelo utilizador

Listar os n maiores consumidores de energia durante um dado período É apresentada a lista dos n, número inserido pelo utilizador, maiores consumidores de energia durante um intervalo de tempo definido pelo utilizador.

3.4 Menu Alterações

O Menu de Alterações permite o utilizador escolher uma das seguintes opções

- Mudar o fornecedor de uma casa.
- Ligar/Desligar os dispositivos de uma divisão de uma casa.
- Alterar os valores de algum fornecedor.

```
MENU ALTERAÇÕES
1 | Mudar o fornecedor de alguma casa
2 | Ligar/Desligar os dispositivos de alguma divisão de alguma casa
3 | Alterar os valores de algum fornecedor
Escolha a sua opção:
```

Mudar o fornecedor de uma casa São apresentadas as casas da vizinhança e pedido ao utilizador para inserir o NIF do proprietário da casa para alterar o fornecedor de energia.

Ligar/Desligar os dispositivos de uma divisão de uma casa É apresentado o Menu Ligar e Desligar Dispositivo, o utilizador tem de escolher se pretende ligar ou desligar um dado dispositivo.

```
MENU LIGAR E DESLIGAR DISPOSITIVO
1 | Ligar
2 | Desligar
```

Após escolher a opção aparece o Menu Ligar/Desligar Divisão ou Dispositivo que permite o utilizador escolher se quer alterar o estado de um dispositivo específico ou de todos os dispositivos de uma divisão.

```
LIGAR/DESLIGAR DIVISAO OU DISPOSITIVO
1 | Divisão
2 | Dispositivo
```

Sendo posteriormente apresentadas as casas da vizinhança para escolher qual será o alvo da alteração.

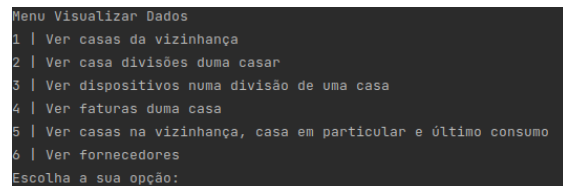
Alterar os valores de algum fornecedor É apresentada a lista dos fornecedores da vizinhança e permite ao utilizador escolher a qual é que pretende alterar os valores e definir os novos descontos.

3.5 Menu Visualizar Dados

Neste Menu o utilizador tens as seguintes opções:

- Ver casas da vizinhança
- Ver casa divisões duma casa
- Ver dispositivos numa divisão de uma casa
- Ver faturas duma casa
- Ver casas na vizinhança, casa em particular e último consumo
- Ver fornecedores

Dependendo da opção escolhida serão apresentados os dados do projeto.



```
Menu Visualizar Dados
1 | Ver casas da vizinhança
2 | Ver casa divisões duma casar
3 | Ver dispositivos numa divisão de uma casa
4 | Ver faturas duma casa
5 | Ver casas na vizinhança, casa em particular e último consumo
6 | Ver fornecedores
Escolha a sua opção:
```

4 Conclusão e Reflexão Crítica

Atingidos os objetivos deste projeto, desenvolvido ao longo do semestre, sentimos que pudemos aprofundar todos os conhecimentos e aspetos adquiridos obedecendo ao paradigma de programação orientada aos objetos e utilizando linguagens como o Java, principalmente em termos de abstração, reutilização de código e modularidade.

Em suma, consideramos que os objetivos do trabalho foram atingidos. Sentimos algumas dificuldades a juntar as diferentes partes do trabalho no interpretador, porém, tentamos procurar as diversas soluções que nos foram lecionadas ao longo do semestre.