

# Projeto POO

## Simulação de Casas Inteligentes

Bruno Gião a96544

Miguel Vaz a72161

João Cruz a95375

Grupo 1

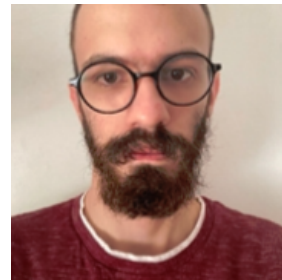
UM, LCC, 2021/2022



Bruno Gião a96544



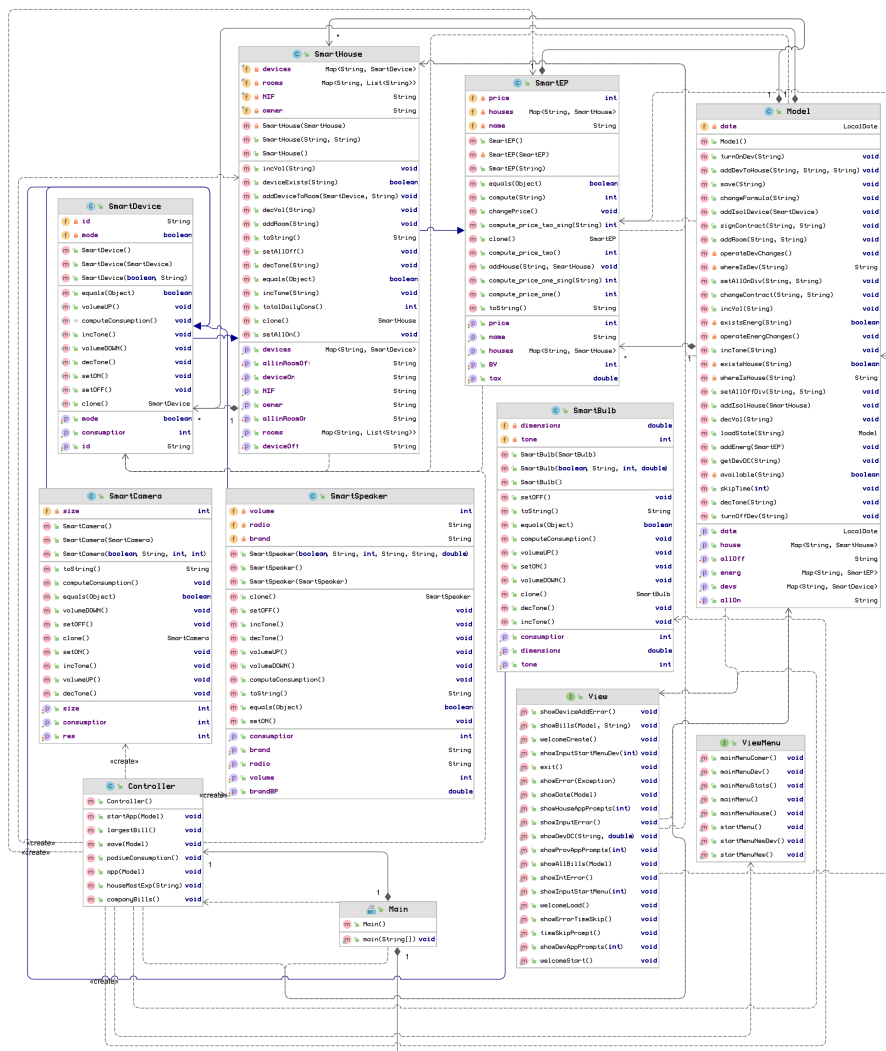
João Cruz a95375



Miguel Vaz a72161

# Contents

<b>1</b>	<b>Introdução</b>	<b>4</b>
<b>2</b>	<b>Model</b>	<b>4</b>
2.1	Classes Base . . . . .	4
2.1.1	SmartDevices . . . . .	4
2.1.2	SmartHouse . . . . .	4
2.1.3	Fornecedor de Energia . . . . .	4
<b>3</b>	<b>View</b>	<b>4</b>
<b>4</b>	<b>Controller</b>	<b>4</b>
<b>5</b>	<b>Simulador e Decisões de Grupo</b>	<b>5</b>
5.1	Criação de uma Sessão . . . . .	5
5.2	Carregar/Guardar uma Sessão . . . . .	5
5.3	. . . . .	5
<b>6</b>	<b>Conclusão</b>	<b>6</b>



## 1 Introdução

Neste projeto, temos como objetivo criar uma simulação de ‘Casas Inteligentes’. Ora, estas usam dispositivos inteligentes que têm certas regras que os definem. Regras essas que tornam este projeto ideal para um paradigma como o de Programação Orientada a Objetos.

Através de encapsulamento, abstração, herança e composição, conseguimos realizar este projeto, assegurando a implementação devida de POO.

Neste Projeto, foi utilizada a arquitetura típica de MVC (Model-View-Controller), que consiste de dividir as tarefas do programa em classes diferentes. i.e. O Model trata de gerir dos dados em memória, o View trata da User Interface, o Controller gere a logica do programa.

## 2 Model

Conforme foi explicado na introdução, o Model é a classe que trata da memória e da gestao de dados em memória, isto é, qualquer estrutura de dados, ou manipulação de uma estrutura de dados é feita nesta secção da arquitetura.

### 2.1 Classes Base

De modo a implementar a tal ‘simulação’ é necessário ter classes ‘base’, nomeadamente as dos SmartDevices, SmartHouses e SmartEP.

#### 2.1.1 SmartDevices

SmartDevices são dispositivos que podem ser ligados ou desligados e têm um identificador único, no entanto, podem ser divididos em três tipos: SmartBulbs; SmartSpeakers e SmartCameras. Assim, o melhor modo de realizar esta tarefa foi claramente através do uso de uma classe Abstrata extendida pelas previamente mencionadas.

#### 2.1.2 SmartHouse

Uma SmartHouse é uma casa com dono e com uma coleção de SmartDevices, no caso deste projeto, essa coleção foi implementada através de um Map. De modo a garantir encapsulamento, estes são implementados através de composição.

#### 2.1.3 Fornecedor de Energia

Um fornecedor de energia é uma coleção de Casas, equipada com valores de impostos e de formulas para calcular preço por KWh. Fornecedores de energia também têm nome de empresa de modo a conseguir empresas armazenar num HashMap. De modo a colecionar estas casas, novamente, é utilizada composição.

## 3 View

O View é a secção da arquitetura que assegura o design e a estética do Programa, sendo assim, neste projeto decidimos implementar duas interfaces, que, em conjunto definem o View.

Estas interfaces tratam de imprimir no ecran toda a informação necessária para o utilizador conseguir correr o programa, desde menus a erros ou prompts.

## 4 Controller

O Controller é onde o programa em si corre, tratando de invocar os métodos de View e de Model, servindo, assim, como intermediario entre estas duas secções.

## **5 Simulador e Decisões de Grupo**

### **5.1 Criação de uma Sessão**

Embora uma implementação ‘estranha’, isto é, no sentido de não ser exatamente o que se imaginaria, o modo de criar uma sessão é através de criar objetos até finalmente todas as casas terem assinado um contrato e, de seguida, escolher a opção de proceder no tempo.

Ora, primeiramente, é necessário criar os objetos necessários, por qualquer ordem, no entanto para adicionar um objeto a uma casa é necessário: um, a casa existir, dois, a casa ainda não ter um contrato assinado, e três, a casa ter divisões, nomeadamente aquela qual o utilizador pretende adicionar a. Isto resulta num modo de criação sequencial e fácil de acompanhar, apesar da necessidade da casa não ter contratos assinados. Ainda assim continua a ser mais acessível do que qualquer outro modo que tentamos implementar.

### **5.2 Carregar/Guardar uma Sessão**

Através de `FileOutputStreams` e `FileInputStreams` e do uso de MVC, é possível guardar e carregar seguramente o estado da sessão em curso. Isto pois, toda a informação relevante está armazenada inteiramente no Model, sendo assim é apenas necessário escrever para um ficheiro binário o conteúdo da classe Model. Tornando o uso de sessões anteriores não só viável mas incrivelmente eficiente e acessível para o utilizador.

### **5.3**

## 6 Conclusão

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.