PROGRAMAÇÃO ORIENTADA A OBJETOS

Licenciatura em Engenharia Informática

Sistemas Operativos 2023/2024

META 2

Dinis Meireles de Sousa Falcão / David José Nobre Pires

2020130403 / 2019129618

a2020130403@isec.pt / a2019129618@isec.pt

TEMA E CONCEITOS GERAIS

Pretende-se construir em **C++** um **simulador de uma habitação** controlada por **componentes de domótica** interligados entre si. O simulador inclui **zonas da habitação**, que têm **propriedades** tais como **temperatura**, **luz**, entre outras. Estas são inspecionadas por **sensores**, os quais fornecem os valores lidos a **regras** que são geridas por **processadores de regras** que determinam o que fazer em função das leituras dos **sensores**. Nas **zonas** existem também **aparelhos**, os quais afetam as **propriedades da zona** onde se encontram e reagem a **comandos** emitidos pelos **processadores de regras**. O simulador é **controlado por comandos escritos pelo utilizador**. Existem uns quantos comandos, dando a falsa ideia de dimensão/complexidade, mas na verdade o simulador pouco mais é do que um conjunto de objetos interligados entre si e uma interface de utilizador. O simulador inclui a noção de **passagem de tempo**. Esta característica é importante para o funcionamento dos **componentes**. A passagem do tempo no simulador é totalmente independente do tempo real medido pelo computador, sendo descrita em unidades abstratas de **"instantes"** e **avança-se para o instante seguinte por comando do utilizador**.

DESCRIÇÃO DAS OPÇÕES TOMADAS

De acordo com o enunciado do Trabalho Prático de Programação Orientada a Objetos, e após efetuar uma breve análise do mesmo, decidimos que precisamos de criar as diferentes e seguintes classes genéricas: a classe **Componente**; a classe **Habitação**, que é, resumidamente, um conjunto de classes do tipo **Zona**; a classe **Propriedade**; a classe **Regras**; e a classe **Simulador**, na qual se faz a gestão de toda a interface do utilizador (incluindo a validação dos comandos), a ligação entre a mesma e a **Habitação** e a respetiva chamada e utilização dos **ficheiros fornecidos** pelo professor (**Terminal.h** e **Terminal.cpp**).

Após reunir com um dos professores, chegámos à conclusão de que a seguinte figura seria um exemplo da melhor estratégia a seguir para a Interface do Utilizador para este **Simulador**, com auxílio da biblioteca **curses.h**, ainda sem qualquer **Habitação** criada.

Figura 1 – Interface do Utilizador

DESCRIÇÃO DAS ESTRUTURAS USADAS

Como já foi dito anteriormente, foram criadas algumas classes que serão utilizadas ao longo de toda realização do Trabalho Prático.

Em relação aos tipos genéricos de componentes (**classe Componente**), estes são os Aparelhos (**classe Aparelho**), o Processador (**classe Processador**) e os Sensores (**classe Sensor**). Estas últimas três classes são **herdeiras da classe Componente**, mas têm algumas diferenças entre si. A classe **Aparelho** tem mais 4 classes de herança, sendo estas as classes **Aquecedor**, **Aspersor**, **Lâmpada** e **Refrigerador**. Cada uma destas classes é responsável pelas suas ações, ou seja, estas contêm métodos que vão afetar as **Propriedades** (cada uma à sua maneira), quando recebem comandos que são reconhecidos pelas mesmas ("*liga*", "*desliga*"). A classe **Processador** contém um conjunto de **Regras** (inicialmente vazio) e um conjunto de **Aparelhos** (inicialmente vazio). Este faz uma verificação da veracidade das **Regras** que contém, e caso estas sejam todas verdadeiras, comunica aos **Aparelhos** associados, para que estes possam fazer o que têm a fazer. A classe **Sensor** é uma classe simples, pois apenas contém uma **Propriedade** e apenas consegue fazer a leitura do valor atual da mesma.

Em relação à classe genérica **Propriedade**, esta divide-se em 7 outras classes, que são: **Fumo**, **Humidade**, **Luz**, **Radiação**, **Som**, **Temperatura** e **Vibração**. Cada uma destas possui um valor atual, e algumas têm condições de valor máximo e/ou valor mínimo.

A classe genérica **Regra** divide-se em 5 outras classes, que são: **Entre**, **Fora**, **IgualA**, **MaiorQue**, **MenorQue**. Cada uma, e dependendo de qual, têm um ou dois valores, que serão condições para os valores das **Propriedades**.

A classe **Habitação** contém um conjunto de **Zonas** e um conjunto de **Processadores** (armazena as cópias dos mesmos), e contém os métodos que a classe **Zona** contém (com a adição do parâmetro "*int IDzona*"), que servirão para executar os comandos do Utilizador na classe **Simulador**.

INTERFACE ESPECÍFICA

Como já foi dito anteriormente, inicialmente o **Simulador** é criado sem qualquer habitação, apenas com 2 janelas (**Comandos** e **Resultado de Comandos**). Após a introdução do comando "**hnova x y**" pelo utilizador, aparecerá uma nova janela:

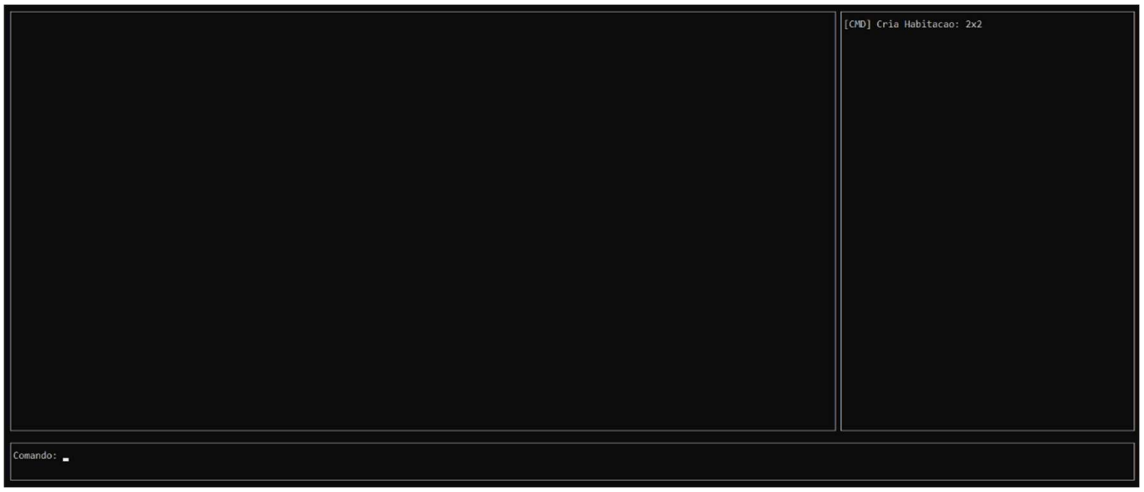

Figura 2 – Interface do Utilizador (**hnova 2 2**)

Após este comando, o utilizador poderá introduzir **Zonas** através do comando "**znova x y**"**,** mas apenas serão criadas dentro dos limites da Habitação.

Figura 3 – Interface do Utilizador (**znova 1 1** e **znova 1 2**)

Com as zonas criadas é permitido consultar as propriedades existentes em cada **Zona**, que são geradas com valores aleatórios dentro dos limites de cada uma. Além disto, também é permitido fazer a criação de novos **Componentes**, que aparecerão dentro da zona na qual foram criados.

Figura 4 – Interface do Utilizador (**cnovo 1 p liga**, **cnovo 2 a aquecedor** e **cnovo 2 s temperatura**)

É permitido fazer todos e quaisquer comandos que estão presentes no enunciado, que poderão ter ou não a sua representação visual, como já pudemos ver. Também é importante referir que qualquer comando de remoção provocará a remoção do elemento visual que foi selecionado.

DESAFIOS E DIFICULDADES

Após a defesa da Meta 1, reparámos que a estratégia de organização de classes que teríamos pensado inicialmente não seria a melhor para o desenvolvimento deste projeto, o que fez com que tivéssemos de refazer quase que por completo as classes que já tínhamos criado. Podemos dizer que foi uma **decisão muito bem tomada**, pois permitiu que realizássemos "todas" as tarefas pedidas no enunciado com muito mais facilidade do que poderia ter sido de outra maneira.

Cremos que o maior desafio deste projeto terá sido a **organização inicial de todas as classes** (o que cada uma precisava, quais as relações que teria com outras classes, que métodos deveríamos fazer, organização dos construtores, entre outros), mas assim que encontrámos a melhor solução, foi bastante **prazeroso** e **empolgante** fazer todo o restante desenvolvimento deste mesmo projeto.

Esperamos que com este possamos obter uma boa nota no mesmo e, por sua vez, obter aprovação à cadeira de **Programação Orientada a Objetos**.

FIM