Trabalho de Projeto Avançado de Sistemas
Padrão – Little Language
Atividade

Objetivo da atividade: Criar duas novas funcionalidades para o sistema em questão.
Detalhes: Cada funcionalidade é representada por uma interpretação da gramática da linguagem que resulta em um objeto de uma classe do tipo “ConcreteNonTerminal”

As funcionalidades a serem desenvolvidas no exercício devem ser comandos válidos para a linguagem, os quais devem permitir ao usuário duas coisas:
(A) Interagir com um dispositivo que esteja em um determinado cômodo da casa; e
(B) Interagir com todos os dispositivos de um determinado cômodo.
Para executar esses comandos o usuário deverá entrar com as seguintes informações, na respectiva ordem:
Comando (A): *Ação a ser realizada sobre o dispositivo, nome do dispositivo, nome do cômodo, e o andar no qual se encontra esse cômodo.* EX: ("ligar Ar Quarto 1").
Comando (B): *Ação a ser realizada sobre o dispositivo, termo da linguagem que indica todos os dispositivos, nome do cômodo, e o andar no qual se encontra esse cômodo.*
EX: ("ligar tudo Quarto 0").

Gramática da Linguagem:
<comando> := <ação><device><room>
<ação> := “ligar” | “desligar” | “abrir” | “fechar” | “trancar” | “destrancar” | <mudar>
<mudar> := “mudar” número
<device> := identificador | “tudo”
<room> := [identificador][número] | “tudo”

Passo 1.
Criar duas classes que exercem o papel de ConcreteNonTerminal com os nomes de SingleRoomSingleDevice e SingleRoomAllDevices que estendem a classe Command. Para ambas as classes:

```
public class NomeClasse extends Command
{
        public void execute() throws Exception
        {
                // Desenvolver a execução do comando de acordo com a funcionalidade
                // requerida
                …
        }

        // Colocar o atributos necessários para o funcionamento do comando
        // Observar que o classe-mãe Command já possui alguns desses atributos
        // como a casa em questão (IF_House house)
        ...
}
```

Passo 2.
Modificar o CommandFactory para criar objetos dos tipos criados no Passo 1;

```
public class CommandFactory
{
	public IF_Command createSRSD(CommandArgs args)
	{
		// Criação e retorno do objeto do tipo SingleRoomSingleDevice
		// Deve-se usar os atributos em args para setar os atributos necessários
		// em SingleRoomSingleDevice
		...
	}

	// Modificar o método 'private IF_Command createSDCommand' para retornar o
	retorno
	// da chamada à função acima caso o roomScope seja Single

	public IF_Command createSRAD(CommandArgs args)
	{
		// Criação e retorno do objeto do tipo SingleRoomAllDevices
		// Deve-se usar os atributos em args para setar os atributos necessários
		// em SingleRoomAllDevices
		…
	}

	// Modificar o método 'private IF_Command createADCommand' para retornar o
	retorno
	// da chamada à função acima caso o roomScope seja Single
}
```

Passo 3.
Inserir código na linha 122 do Parser.java para detectar números após o identificador do cômodo
3.1: Pegar próximo token do Lexer

```
token = lexer.nextToken();
```

3.2: Verificar o atributo Type do token obtido, se for *NUMBER* significa que o roomScope é *Single*
3.3: Colocar o valor do número no token como floorNumber no commandArgs
3.4: retornar Scope.Single

Passo 4.
Executar as classes de teste unitárias, SingleRoomSingleDeviceTest e SingleRoomAllDevicesTest.