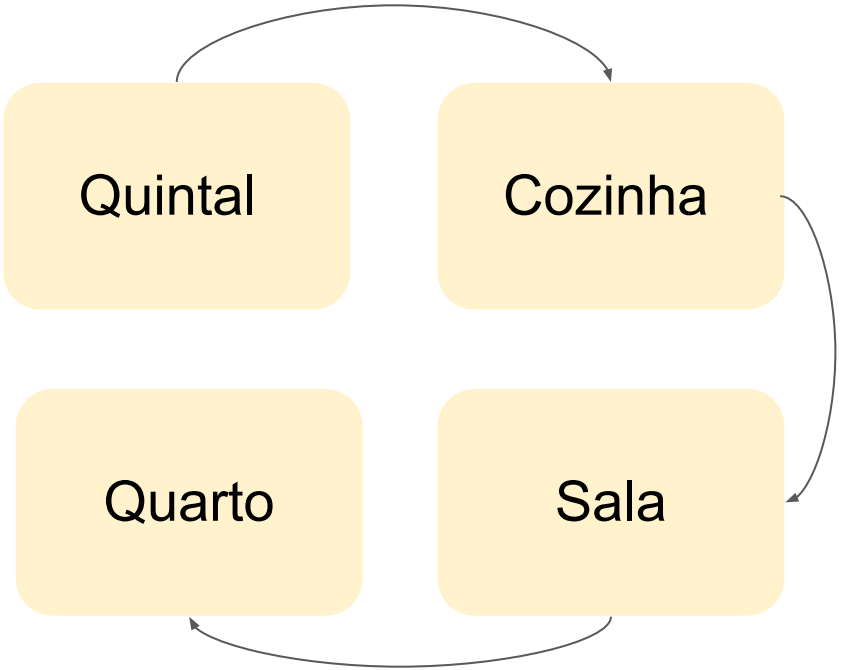
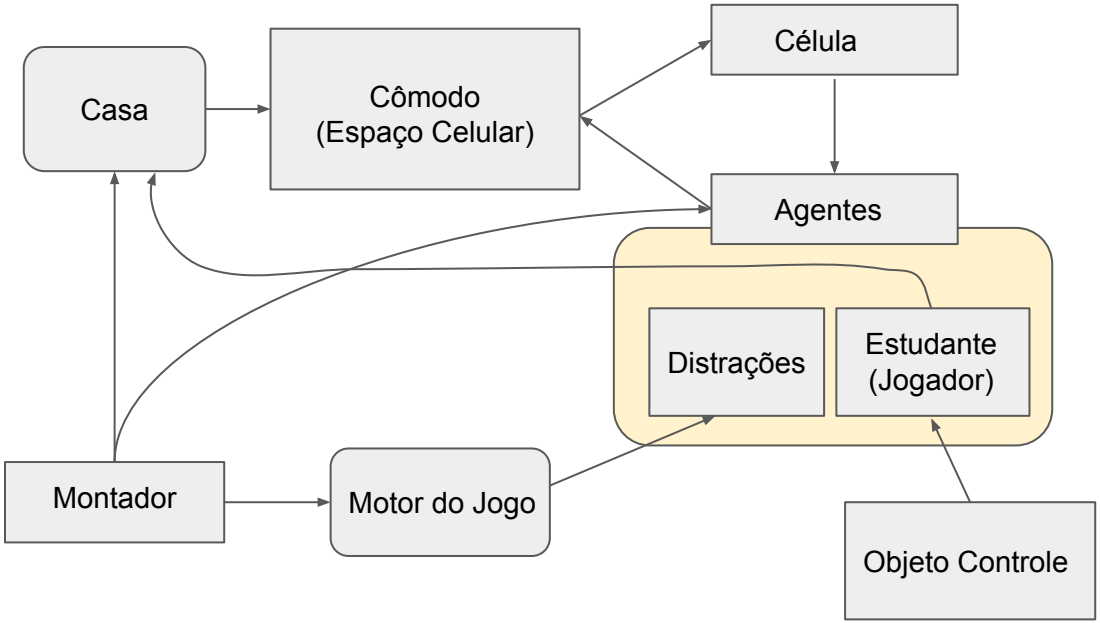
Quarentenado

Proposta do Jogo



Arquitetura

Arquitetura Inicial Proposta ...

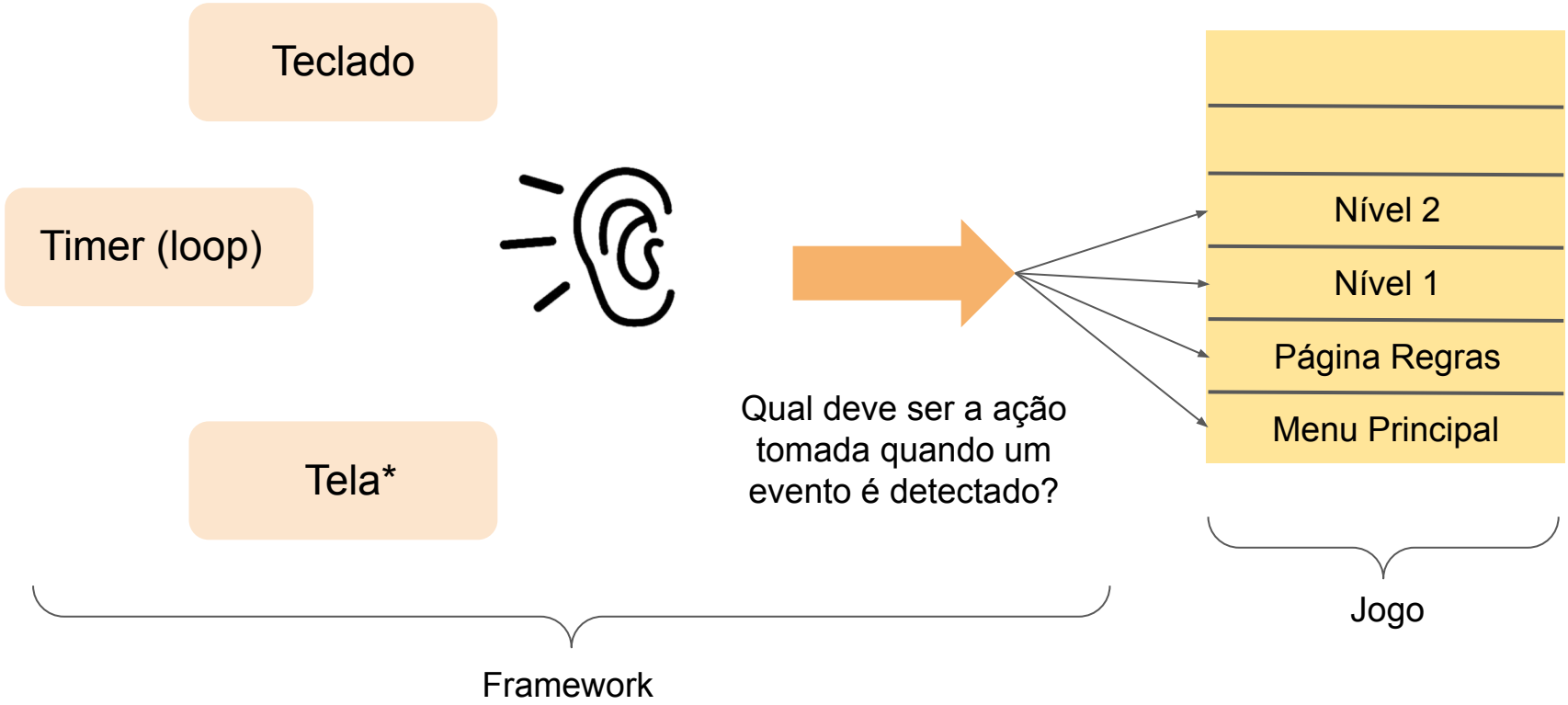


Questões:

- desconhecimento das ferramentas gráficas;
- pouca clareza do que cada parte faria;
- redundância de algumas partes: casa e cômodo.

Arquitetura

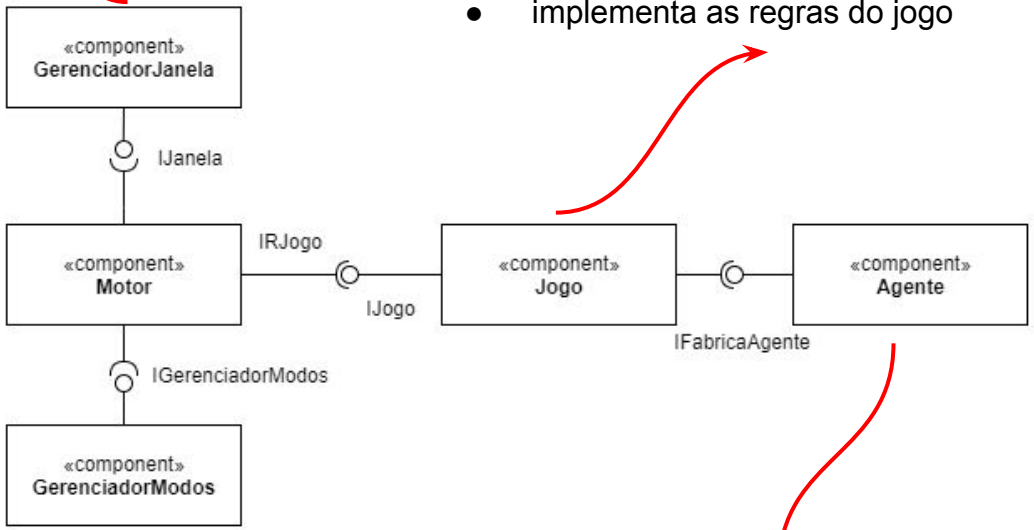
Ideia: criar uma estrutura (*Framework)* que pudesse ser usada para a criação de outros jogos.



*paintComponent: chamado pelo JavaSwing

Arquitetura

- criar a janela do jogo
- atualizar as imagens do painel
- captar eventos do teclado



- constrói o espaço celular
- implementa as regras do jogo

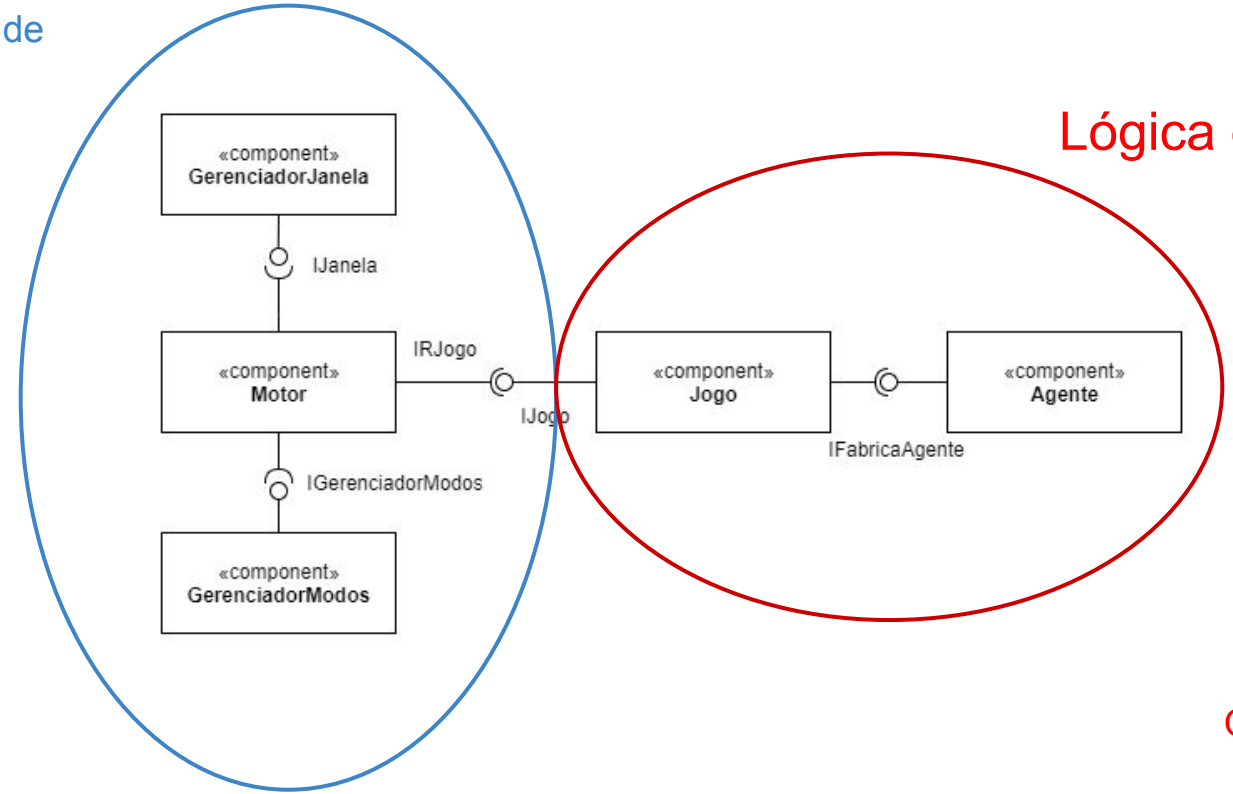
- controle do loop de jogo

- implementa a pilha de modos do jogo
- gerencia os modos

- fornece objetos do tipo Agente
- implementa o Pattern Factory

Arquitetura

pode ser usado no desenvolvimento de outros jogos



Framework

Lógica do Jogo

lógica específica do jogo Quarentenado

Destaques

Gerenciador de Modos

```
public void loop() throws ErroPilhaVazia{
    if(modos.empty()) {
        throw new ErroPilhaVazia("A pilha de modos está vazia!");
    }else {
        modos.peek().loop();
    }

}

public void pintarTela (Graphics g) throws ErroPilhaVazia{
    if(modos.empty()) {
        throw new ErroPilhaVazia("A pilha de modos está vazia!");
    }else {
        modos.peek().pintarTela(g);
    }

}

public void keyTyped(KeyEvent e) {
    if(modos.empty()) {
        System.out.println("Atenção: a pilha está vazia.");
    }else {
        modos.peek().keyTyped(e);
    }

}
```

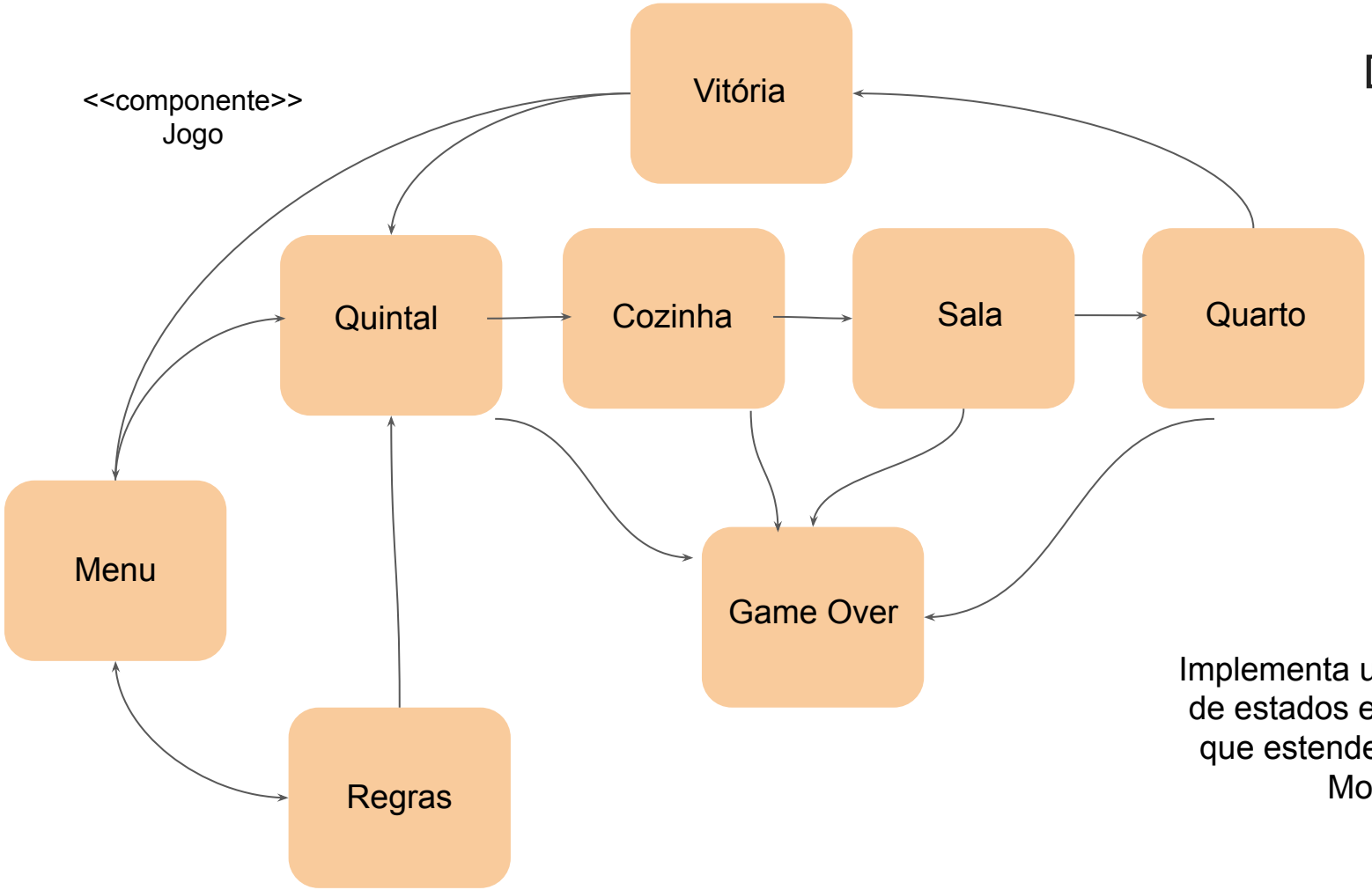
<<componente>>
GerenciadorModos

Para usar o Framework criado para o desenvolvimento de diferentes jogos ou diferentes contextos de um mesmo jogo, basta estender a classe Modo.

Modo

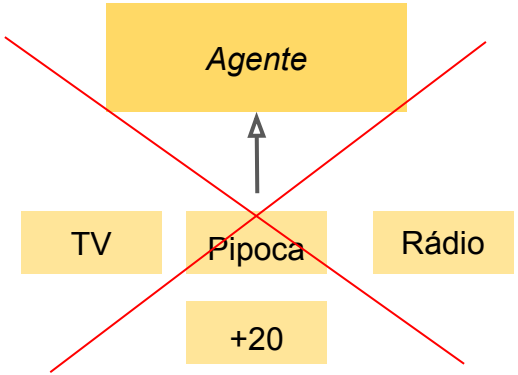
```
public abstract class Modo {
    public static GerenciadorModos meuGerenciador;

    public Modo() {};

    public void setGerenciador(IGerenciadorModos  meuGerenciador) {
        this.meuGerenciador = meuGerenciador.retornaGerenciadorModo();
    }
    public abstract void pintarTela(Graphics g);
    public abstract void loop();
    public abstract void keyTyped(KeyEvent e);
    public abstract void keyPressed(KeyEvent e);
    public abstract void keyReleased(KeyEvent e);

}
```

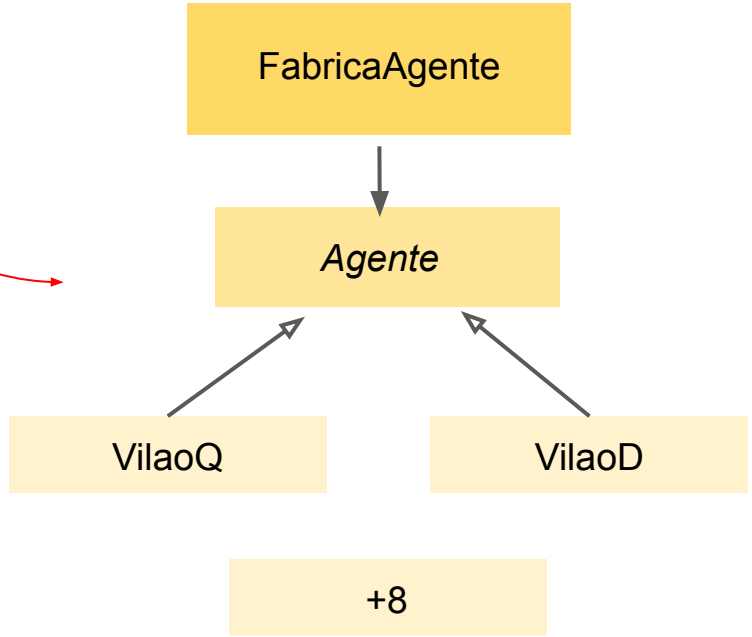
Destaques



Implementa uma máquina de estados entre objetos que estendem a classe Modo

Destaques

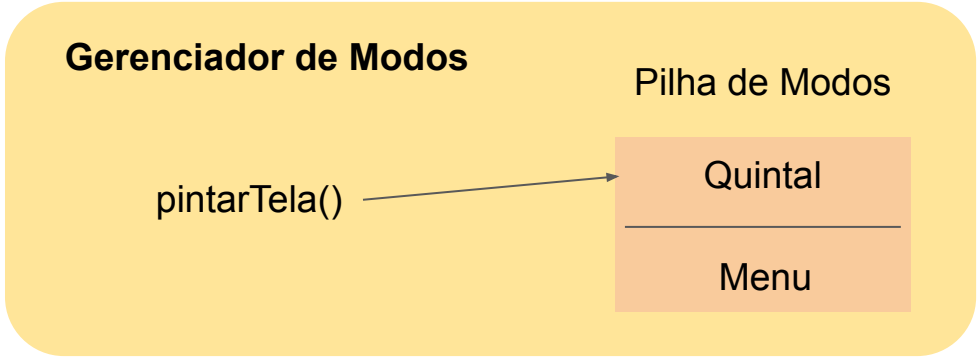




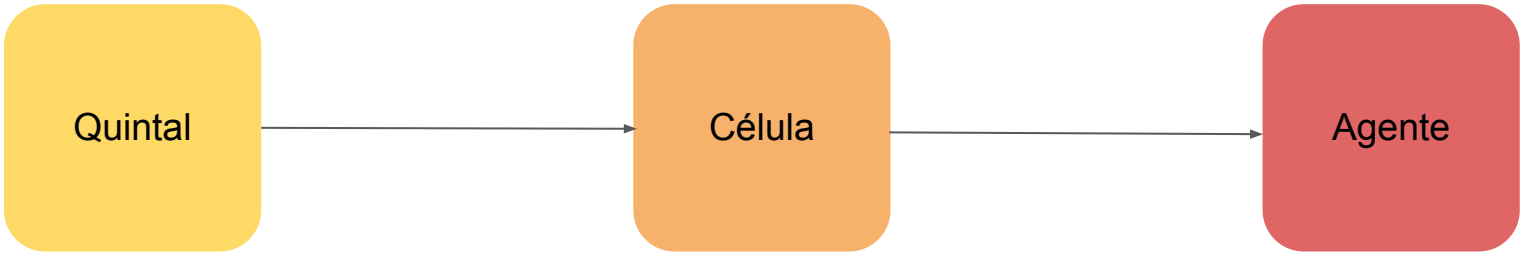
- diminui o número de classes;
- facilita criação de novos padrões de movimento;
- facilita instanciação (*pattern factory*).

<<componente>>
Agente

Retomando a ideia inicial ...



tomando como exemplo o caso do método abstrato pintarTela() da classe abstrata Modo



pinta sua tela de fundo e chama o método pintarTela() da célula

chama o método pintarTela() do agente que ele contém

cada agente se pinta

Histórico, Aprendizados e Expectativas Futuras

- Mudança na apresentação gráfica do jogo;
- Reformulação total da arquitetura do jogo;
- Mudança na arquitetura do componente Agente.

- Projetar uma boa arquitetura é um trabalho difícil e, no desenvolvimento desse jogo, foi preciso rever o projeto várias vezes;
- A divisão do código em componentes facilita mudanças e expansões no código.

- Elaboração de mais cômodos (níveis);
- Implementação de um sistema de pontuação;
- A geração de cômodos poderia ser feita através de um *pattern factory.*

FIM

Instituto de Computação - Unicamp

MC322 - A

Gabriel & Hannah

Agradecimentos à Gabriela Ferreira pela arte do jogo.