Instituto Superior de Engenharia

Politécnico de Coimbra

ENGENHARIA INFORMÁTICA 2022/2023

Relatório Programação Avançada

- **Tiago Garcia Quintas 2019128044**

Reconhecimentos

A extensa preparação para a realização deste trabalho não teria sido possível tanto sem os vários conceitos lecionados nas aulas práticas e teóricas de PA como toda a matéria disponibilizada nas plataformas virtuais de apoio aos estudantes (Inforestudante).
Quero agradecer ao Professor Álvaro Santos e ao professor João Durães pela disponibilidade e flexibilidade de horários para ensino dos conceitos necessários para a realização deste trabalho e acompanhamento regular de esclarecimento de dúvidas mesmo quando este coincidia com o tempo livre e ou com tarefas pessoais destes professores.

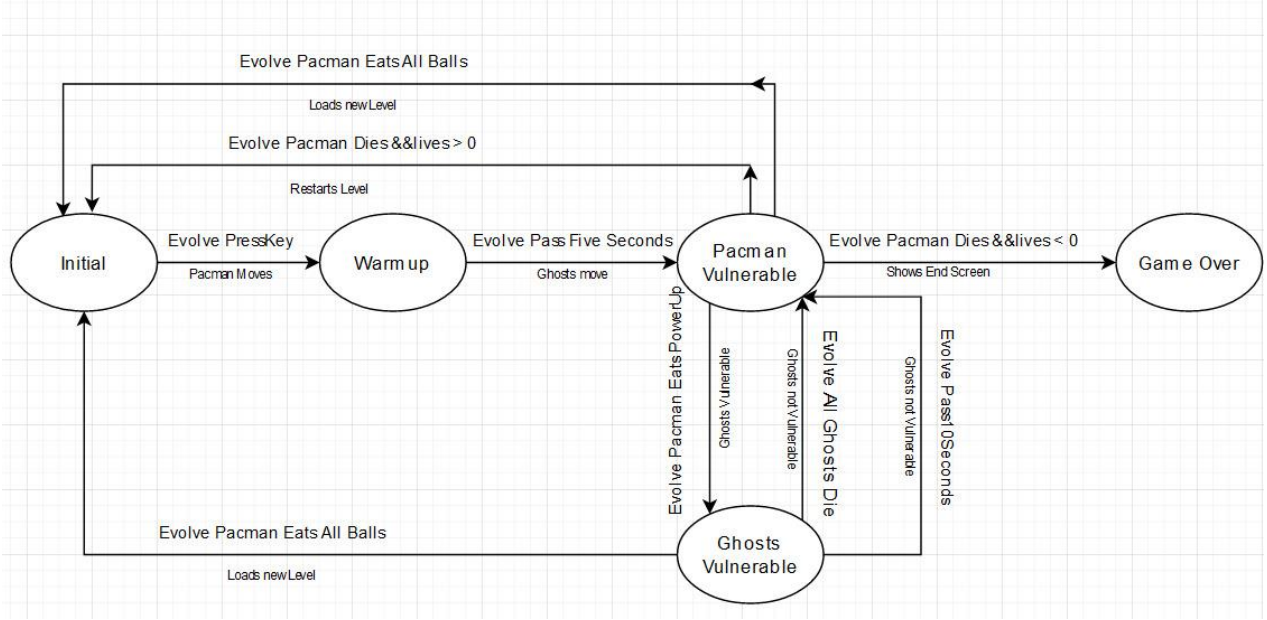
Agradeço também a todos os docentes de PA pela oportunidade de preparar este trabalho.

Introdução:

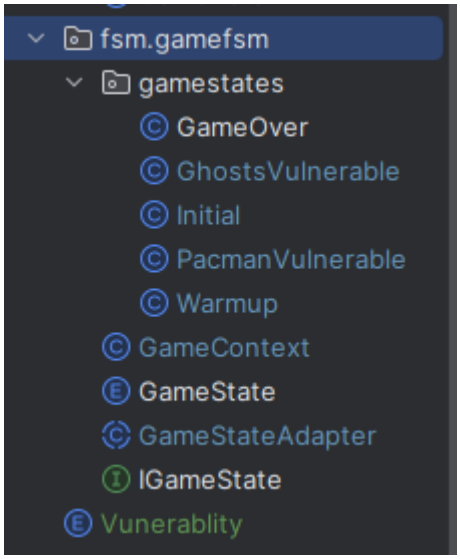
Neste trabalho foi-nos proposto implementarmos uma versão do jogo PacMan em Java utilizando os conceito dados nas aulas teóricas e práticas.

Neste breve relatório começarei por introduzir a informação relativa as classes dos vários elementos utilizados bem como a maquina de estados utilizada para gerir o decorrer do jogo.

Máquina de Estados:



A máquina de estados encontra-se na package “package pt.isec.pa.a2019128044.tinypac.model.fsm.gamefsm.gamestates;” sendo que tem como funcionalidade gerir o desenvolver do jogo, contendo uma referencia para os dados do jogo.



Context:

```java
package pt.isec.pa.a2019128044.tinypac.model.fsm.gamefsm;

import ...

Asus *
public class GameContext implements IGameEngineEvolve {
    5 usages
    private GameData data;
    4 usages
    private IGameState state;

    Asus
    public GameContext() {
        data = new GameData();
        state = GameState.INITIAL.createState( context: this, data);
    }

    Asus
    public char[][] getLevel(){
        return data.getLevel();
    }
    no usages  Asus
    public GameState getState() { return state.getState(); }

    1 usage  Asus
    void changeState(IGameState newState) {
        this.state = newState;
    }

    Asus
    public DIRECTION getDirection() { return data.getDirection(); }

    Asus
    public void setDirection(DIRECTION direction) { data.setDirection(direction)

    Asus
    @Override
    public void evolve(IGameEngine gameEngine, long currentTime) {

        if(!state.evolve(currentTime))
            gameEngine.stop();

    }
}
```

O context contem o estado atual e os dados de jogo. Esta classe gere os estados da maquina de estados.

Classes

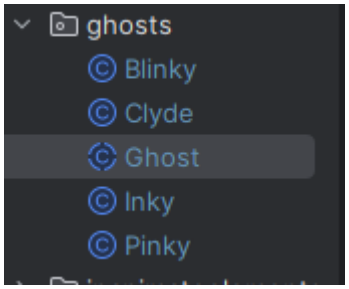
Todos os Elementos do jogo como o pacman, os fantasmas e até mesmo os elementos fixos do labirinto são derivados da classe Element que implementa a interface IMazeElement.

```java
public abstract class Element implements IMazeElement {

    protected final char symbol;
    protected Level level;

    Asus
    public Element(char symbol, Level level) {
        this.symbol = symbol;
        this.level = level;
    }

    5 overrides  new *
    public void evolve(long currentTime){};

    8 overrides  Asus
    @Override
    public char getSymbol() {
        return symbol;
    }

}
```

A classe Ghosts é derivada da Classe Element de modo a puder exercer as suas diversas funções. Todos os fantasmas são derivados desta classe. Esta classe contem um arrayList de modo a puder guardar todos os movimentos dos fantasmas sendo que quando se econtram vulneráveis deverão reverter os seus movimentos, esta funcionalidade ainda não se encontra implementada.

```java
public abstract class Ghost extends Element {

    9 usages
    protected boolean inSpawn;
    6 usages
    protected long lastMovedTime = 0;
    1 usage
    protected boolean isAlive;
    no usages
    protected boolean vulnerable;
    protected DIRECTION direction;
    1 usage
    protected List<Integer[][]> movements;

    4 usages  Asus *
    public Ghost(char symbol, Level level) {
        super(symbol,level);
        this.isAlive = true;
        this.movements = new ArrayList<>();
        inSpawn = true;
    }
}
```

Existe uma classe para cada fantasma tendo em conta que todos se movem de forma diferente. De momento apenas o Blinky se encontra implementado, sendo que os outros fantamas replicam o mesmo metodo de movimento.



O pacman é representado na sua propria classe, sendo que esta serve para o movimentar, gerir as suas vidas e os pontos. De momento apenas se encontra implementado o movimento do pacman.

```java
package pt.isec.pa.a2019128044.tinypac.model.data.maze.elements;

import pt.isec.pa.a2019128044.tinypac.model.data.maze.Level;
import pt.isec.pa.a2019128044.tinypac.model.data.maze.elements.inanimateelements.Empty;
import pt.isec.pa.a2019128044.tinypac.model.data.maze.elements.inanimateelements.Portal;
import pt.isec.pa.a2019128044.tinypac.model.data.maze.elements.inanimateelements.SmallBall;
import pt.isec.pa.a2019128044.tinypac.model.data.maze.elements.inanimateelements.Warp;

2 usages  Asus *
public class Pacman extends Element {
    2 usages
    private long lastMovedTime = 0;
    1 usage
    int lives;

    1 usage  Asus *
    public Pacman(Level level) {
        super( symbol: 'P', level);
        lives = 3;
    }

    new *
    @Override
    public void evolve(long currentTime) {

        if (currentTime - lastMovedTime >= 10) {

            Level.Position myPos = level.getPositionOf( element: this);
            if (myPos == null)
                return;

            Level.Position neighboorPosition = level.getNeighboorPosition(myPos, level.getDirection());

            if (level.getElement(neighboorPosition) instanceof Portal || level.getElement(neighboorPosition)
                    instanceof Warp) {

            } else {
                boolean moved = level.setElementPosition( element: this, neighboorPosition);
                if (moved) {
                    level.setElementPosition(new Empty(level), myPos);
                }
            }
            lastMovedTime = currentTime;
        }

    }
}
```

A classe GameData contem toda a informação do jogo, incluindo o nivel atual e o mapa atual. Esta classe serve para gerir a criação do mapa atravez de um txt e ligação entre a

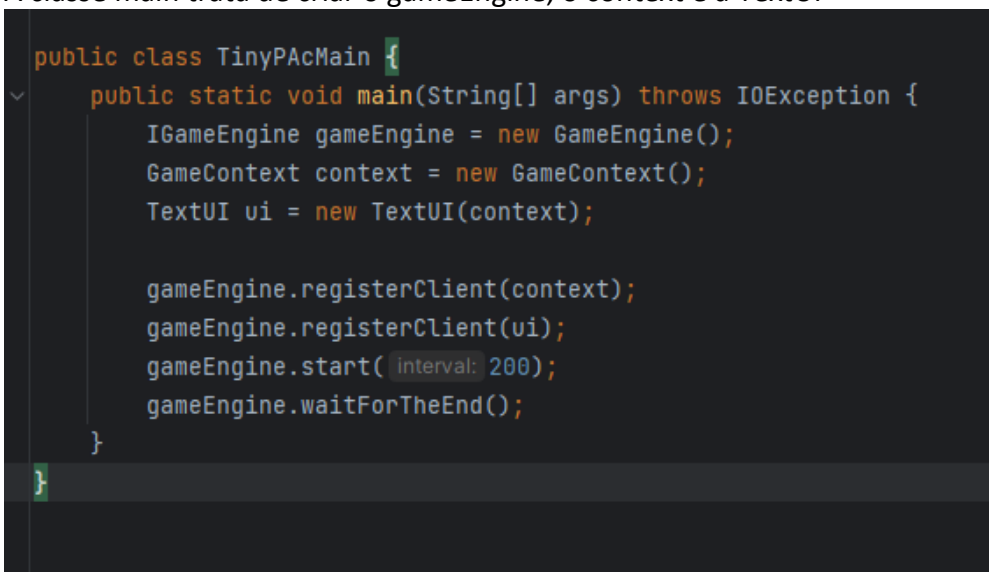
Classe Level. A classe context contem um objeto do tipo GameData de modo à maquina de estados poder comunicar com os dados de jogo

```java
18 usages  Asus *
public class GameData {
    2 usages
    private final int FIRSTLEVEL = 1;
    1 usage
    private final int MAXLEVEL = 20;
    4 usages
    private StringBuilder currentLevel;
    18 usages
    private Level level;
    1 usage
    private int levelNumber;
    no usages  Asus
    static final Map<Character, Class<? extends Element>> elements = new HashMap<>() {{
        put('y', Cavern.class);
        put('F', FruitZone.class);
        put('M', PacmanSpawn.class);
        put('Y', Portal.class);
        put('O', PowerUp.class);
        put('o', SmallBall.class);
        put('x', Wall.class);
        put('W', Warp.class);
    }};

    1 usage  Asus
    public GameData() {
        currentLevel = new StringBuilder();
        currentLevel.append("files/Level").append(String.format("%02d", FIRSTLEVEL)).append(".txt");
        level = createLevel();
        if(level == null){
            return;
        }
        level.spawnLiveElements();
        level.getLevel();
    }
```

A classe TextUi trata-se da classe que mostra ao utilizador o estado atual do jogo e de receber o input do utilazor de modo a se poder movimentar o pacman utilizando a biltioteca lanterna.

```
3 usages  Asus *
public class TextUI implements IGameEngineEvolve {

    6 usages
    GameContext context;
    6 usages
    Screen screen;

    1 usage  Asus *
    public TextUI(GameContext  context) throws IOException {
        this.context = context;
        screen = new DefaultTerminalFactory().setInitialTerminalSize(new TerminalSize( columns: 80,  rows: 40)).createSc
        screen.setCursorPosition(null);

        menu();
        show();
    }

    1 usage  new *
```

A classe main trata de criar o gameEngine, o context e a TextUI



```
public class TinyPAcMain {
    public static void main(String[] args) throws IOException {
        IGameEngine gameEngine = new GameEngine();
        GameContext context = new GameContext();
        TextUI ui = new TextUI(context);

        gameEngine.registerClient(context);
        gameEngine.registerClient(ui);
        gameEngine.start( interval: 200);
        gameEngine.waitForTheEnd();
    }
}
```

A classe Level trata de gerir todos os elementos do mapa, contento um obejto do tipo Maze com a informação sobre os elementos do mapa.

```java
package pt.isec.pa.a2019128044.tinypac.model.data.maze;

import pt.isec.pa.a2019128044.tinypac.model.Vunerablity;
import pt.isec.pa.a2019128044.tinypac.model.data.DIRECTION;
import pt.isec.pa.a2019128044.tinypac.model.data.maze.elements.Element;
import pt.isec.pa.a2019128044.tinypac.model.data.maze.elements.Elements;
import pt.isec.pa.a2019128044.tinypac.model.data.maze.elements.Pacman;
import pt.isec.pa.a2019128044.tinypac.model.data.maze.elements.ghosts.*;

Asus *
public class Level {
    5 usages
    private int height, width;
    14 usages
    private Maze maze;
    5 usages
    private DIRECTION direction;
    2 usages
    private DIRECTION lastDirection;
    2 usages
    private Vunerablity vulnerableElement;

    25 usages  new *
    public record Position(int y, int x) {}

    Asus *
    public Level(int height, int width) {
        this.height = height;
        this.width = width;
        this.maze = new Maze(height, width);
        direction = null;
    }

    1 usage  Asus
    public void addElement(Element element, int y, int x) { maze.set(y, x, element); }

    Asus *
    public void evolve(long currentTime) {
        for (int y = 0; y < height; y++) {
            for (int x = 0; x < width; x++) {
                if (maze.get(y, x) instanceof Element element) {
                    element.evolve(currentTime);
                }
            }
        }
    }

    9 usages  new *
    public Position getPositionOf(Element element) {...}

    17 usages  new *
    public Position getNeighboorPosition(Position currentPosition, DIRECTION direction) {...}


    1 usage  Asus
    public void spawnLiveElements() {

        int nGhosts = 0;
        boolean pacmanExists = false;
        for (int i = 0; i < height; i++) {
            for (int j = 0; j < width; j++) {
                if (maze.get(i, j).getSymbol() == Elements.CAVERN.getValue() && nGhosts < 4) {
                    switch (nGhosts) {
                        case 0 -> maze.set(i, j, new Blinky( level: this));
                        case 1 -> maze.set(i, j, new Clyde( level: this));
                        case 2 -> maze.set(i, j, new Inky( level: this));
                        case 3 -> maze.set(i, j, new Pinky( level: this));
                    }
```

Implementação do GameEngine

O game engine trata do desenvolver do programa, sendo que o context e UI implementam a interface IGameEngineEvolve. O game context tem utiliza método evolve que permite com que todo os jogo se desenvolva tendo sendo que este método é chamado pelo gameEngine a cada 200 milissegundos, tanto o pacman como os fantasmas movem-se através da chamada deste método no context, sendo que este acaba por permitir o decorrer do jogo fazendo ate mesmo em certas situações que o context transite de estado. A UI implementa o método evolve de modo poder atualizar o display mostrando a informação atual do jogo.

O game engine e o seu método evolve é o que permite com que o jogo se desenvolva de modo a os elementos dos jogo se possam mover e também permitindo com que o jogo transite de estados em certas ocasiões.