UNIVERSIDADE DE ÉVORA
CURSO DE ENGENHARIA INFORMÁTICA
2023/2024

Programação 2

→ Relatório ←

Miguel Aleixo - 51653
Tiago Morgado - 51717
Mariana Cavaco - 51820

1. Introdução

Este projeto tem como objetivo a utilização da linguagem de programação Java, para desenvolver a simulação de um prado habitado por coelhos e cenouras.

O jogo consiste na interação entre essas duas entidades, onde os coelhos se alimentam das cenouras presentes no prado. A sobrevivência desses mesmos está associada ao tempo que eles conseguem permanecer sem se alimentarem. Caso esse tempo persista, os coelhos morrem.

2. Metodos de cada classe

- Classe Grassland

```
public Grassland (int i, int j, int starveTime)
```

Este é o construtor para a classe Grassland. O objetivo é inicializar uma instância da classe com as dimensões especificadas, configurar o tempo de fome, criar matrizes para armazenar objetos da classe e valores de terreno.

```
public void addCarrot(int x, int y)
```

Este metodo adiciona cenouras à matriz `grassland` em uma posição específica, considerando a possibilidade de as coordenadas fornecidas estarem fora ou não dos limites do prado

```
public void addRabbit(int x, int y)
```

Este metodo adiciona coelhos à matriz `grassland` em uma posição específica, considerando a possibilidade de as coordenadas fornecidas estarem fora ou não dos limites do prado. O coelho adicionado terá o starve time resetado.

```
private void moverCoelhos(Grassland grass)
```

Este metodo é responsavel por copiar os objetos coelho de uma grassland para outra quando esta é atualizada a cada timeStep

```
 private int vizinhos(int x, int y, int type)
```

Este metodo auxiliar conta os 8 vizinhos adjacentes a uma celula de acordo com o seu tipo de modo a ser possivel implementar as regras no metodo abaixo.

```
public int regrasJogo(int x, int y)
```

Esta metodo define as regras para a evolução do jogo, especificamente para cada célula na matriz `grassland` durante um timestep.

```
public Grassland timeStep()
```

Esta função realiza um timestep no jogo, atualizando a matriz `grassland` com base nas regras específicas do jogo permitindo que seja possivel ver a evolução da grassland ao longo do tempo.

- Classe Coelho

```
public class Coelho {
    private int rabbitStarveTime;
    private int fome;

    public Coelho() {
        rabbitStarveTime = 0;
        fome = 0;
    }
    public int getRabbitStarveTime() {
        return rabbitStarveTime;
    }
    public int aumentarStarveTime() {
        return rabbitStarveTime++;
    }
    public boolean isHungry(){
        return fome >= rabbitStarveTime;
    }
    public int eat(int cenoura){
        rabbitStarveTime = 0;
        fome = Math.max(0, fome - cenoura);
        return rabbitStarveTime;
    }
}
```

Esta classe gera o objeto Coelho e define as suas principais “traits” como o rabbitStarveTime que define o starveTime de cada instancia desta de modo a ser possivel a cada coelho ter o seu proprio starveTime.Define tambem a “Fome” que é peça essencial para determinar o outcome de cada iteração do timeStep.
Em suma está classe permite a individualização de cada coelho permitindo que cada instancia de coelho contenha os seus valores proprios ao invez de valores standards para cada coelho.

- Classe SimText

```
rivate static void adicionarEntidades(Grassland gra, int x, int y){
    Random random = new Random();
    int numCarrots = random.nextInt(2);
    for(int c = 0; c < numCarrots; c++){
        gra.addCarrot(x, y);
    }
    int numRabbits = random.nextInt(2);
    for(int r = 0; r < numRabbits; r++){
        gra.addRabbit(x, y);
```

Este metodo cria as entidade Coelho e Cenoura atraves do uso de um random e chama o metodo addRabbit() e addCarrot() de modo a que seja possivel colocar estas entidades na grassland.

```
private static void inicializarPrado(Grassland meadow, int width, int
height){

    int grupo = 4;
    int espaço = 4;

    for(int x = 0; x < width; x+= grupo + espaço){
        for(int y = 0; y < height; y += grupo + espaço){
            for(int i = 0; i < grupo; i++){
                for(int j = 0; j < grupo; j++){
                    int cx = x +i;
                    int cy = y +j;
                    if(cx < width && cy < height){
                        adicionarEntidades(meadow, cx, cy);
                            }
                        }
                    }
                }
            }
      }
```

Este metodo inicializa o prado e define qual o tamanho de cada agrupamento de entidades e o espaço entre elas e atraves do chamamento do metodo anterior ela permite a que se encha o Grassland.

```
private static void printarPrado(Grassland meadow){
```

```
    for(int y = 0; y < meadow.height(); y++){
        for(int x = 0; x < meadow.width(); x++){
            switch(meadow.cellContents(x, y)){
                case Grassland.EMPTY:
                System.out.print("E");
                break;

                case Grassland.CARROT:
                System.out.print("C");
                break;

                case Grassland.RABBIT:
                System.out.print("R");
                break;

            }
            System.out.print(" ");

        }
        System.out.println();
    }
    System.out.println();
}
```

Este metodo produz o output dos metodos anteriores de acordo com as regras ja definidas na classe Grassland permitindo então a visualização textual da evolução do prado a cada timeStep.

```
public static void main(String[] args) {
    int width = 20;
    int height = 20;
    int starveTime = 4;

    Grassland meadow = new Grassland(width, height, starveTime);
    inicializarPrado(meadow, width, height);
    int timeStep = 20;

    for(int i = 0; i < timeStep; i++){
        printarPrado(meadow);

        meadow = meadow.timeStep();
```

```
        try{
            Thread.sleep(2000);
        }catch(InterruptedException e){
            e.printStackTrace();
        }
    }
```

Por fim esta é a main da classe que define o width, height e starveTime do prado e configura o tempo entre cada timeStep.

.

3. Classes

- Classe Grassland

Esta classe define as principais variáveis da simulação e inclui as funções que manuseiam as diferentes entidades encontradas nesta de acordo com as regras definidas.

- Classe Coelho

Esta classe define o objeto coelho e permite atribuir e manusear o objeto coelho ao longo da simulação e na classe Grassland permitindo que cada coelho tenha um starveTime, nivel de fome unico que é parte essencial da simulação

- Classe Simulation

Esta classe foi nos dada pela professora de modo a testar as outras duas e como tal permite fazer a representação gráfica da Simulação.

- Classe SimText

Esta classe foi criada de modo a ter um output textual da simulação de modo a termos uma visão mais concreta de como a simulação se densenvolve atraves de elementos textuais ao contrario dos elementos graficos que se encontram expostos na classe Simulation.

4. Conclusão

Ao finalizar este trabalho, destacamos a implementação bem-sucedida da simulação do jogo, utilizando a linguagem Java. A criação de um prado virtual, habitado por coelhos e cenouras.Apontamos algumas dificuldades que encontramos como a implementação correta das regras do jogo e do objeto coelho.

Em suma, a simulação do jogo, com coelhos e cenouras, não apenas atendeu aos requisitos propostos, mas também proporcionou uma oportunidade valiosa para explorar e aprimorar habilidades de programação. Este projeto representa uma contribuição significativa para o entendimento prático, aplicando vários conceitos fundamentais da programação orientada a objetos explorada durante todo este semestre.