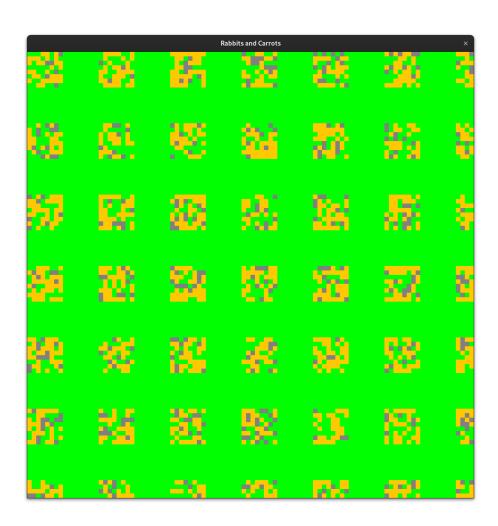


Relatório de Programação II

David Marinho 54560 Axel Amoroso Carapinha 55248 Joana Pimenta 51737



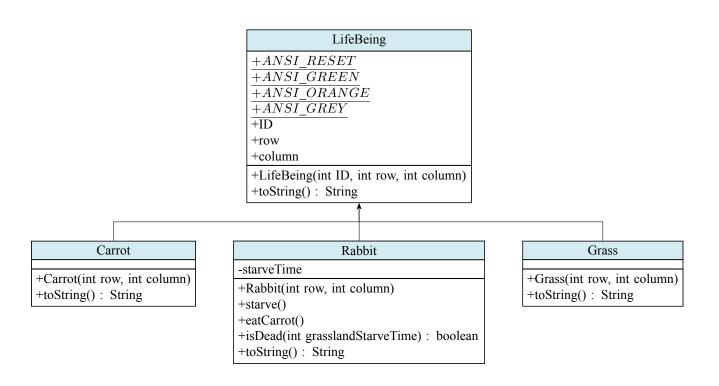
Procedimentos

- Na execução do programa podem ser passados 3 argumentos pela linha de comandos:
 - $i \rightarrow O$ comprimento do prado.
 - $j \rightarrow A$ largura do prado.
 - starveTime → Número de timesteps a que os coelhos sobrevivem sem comer qualquer cenoura. Por defeito, se não passados quaisquer argumentos, temos i = 100, o j = 100 e starveTime = 5.
- 2. Um prado de comprimento i x j é preenchido com objetos do tipo 'Grass' e são gerados aleatoriamente outros objetos, objetos de tipo 'Rabbit' e 'Carrot'.
- 3. A cada timestep é gerado um novo prado respeitando as regras a que a geração anterior esteve sujeita.

Design Patterns

Recorremos aos três pilares da Programação Orientada a Objetos, encapsulamento herança e polimorfismo, e criamos quatro classes para representar entidades específicas:

- LifeBeing → Classe mãe e abstrata das classes 'Rabbit', 'Carrot' e 'Grass'.
- Rabbit → Classe filha que herda 'LifeBeing'. Tem como função organizar todos os dados a respeito dos coelhos.
- Carrot → Classe filha que herda 'LifeBeing'. Organiza toda a estrutura de dados a respeito das cenouras.
- Grass → Classe filha que herda 'LifeBeing' e organiza toda a estrutura de dados a respeitos das ervas.
- Grassland → Classe responsável por organizar todos os dados a respeito do campo, marioritariamente a posição dos objetos no campo e o tempo de longevidade dos mesmos.
- Simulation → Classe principal do programa. Responsável pela renderização e execução do programa.



```
Grassland
+GRASS
+RABBIT
+CARROT
+NEWLINE
-meadowArr
-width
-height
-starveTime
+Grassland(int i, int j, int starveTime)
-fillWithGrass()
+startGrasslandLife()
+width(): int
+height(): int
+starveTime(): int
+addCarrot(int column, int row)
+addRabbit(int column, int row)
+timestep(): Grassland
-grasslandRules(LifeBeing currentCell, ArrayList<LifeBeing> neighbors): int
-grassRules(LifeBeing currentCell, ArrayList<LifeBeing> neighbors): int
-rabbitRules(LifeBeing currentCell, ArrayList<LifeBeing> neighbors): int
-carrotRules(LifeBeing oldGen, ArrayList<LifeBeing> neighbors): int
-getOccurrences(ArrayList<LifeBeing> lifeBeings, int lifeId): LifeBeing[] +cellContents(int column, int row): int
-getCell(int column, int row): LifeBeing
-collectCellNeighbors(int x, int y) : ArrayList<LifeBeing>
+printGrassland()
```

Figure 1: Diagrama de classes