Instituto Superior
de Engenharia
Politécnico de Coimbra

DEPARTAMENTO DE / DEPARTMENT OF
INFORMÁTICA E SISTEMAS

Trabalho Prático POO – Meta 1

Relatório de Licenciatura em Engenharia Informática

Autores

a2024126292 – João Carvalho

a2024113113 – Leonardo Ferreira

INSTITUTO POLITÉCNICO
DE COIMBRA

INSTITUTO SUPERIOR
DE ENGENHARIA
DE COIMBRA

Coimbra, Abril de 2025

ÍNDICE

1 INTRODUÇÃO

O objetivo deste trabalho prático é aplicar os conceitos fundamentais da disciplina de Programação Orientada por Objetos (POO), através do desenvolvimento de um simulador de jardim virtual. O projeto visa consolidar o uso de classes, herança, polimorfismo, encapsulamento e composição, integrando múltiplos objetos que interagem entre si num ambiente simulado.

O simulador de jardim representa um espaço retangular composto por células de solo, onde o utilizador pode plantar, colher e gerir plantas de diferentes tipos, utilizando comandos textuais.

Cada célula do jardim mantém o seu próprio estado e o sistema evolui ao longo do tempo, simulando o crescimento ou a morte das plantas consoante as condições do solo. Além disso, o utilizador controla um jardineiro virtual, que pode entrar, mover-se e atuar dentro do jardim, realizando ações como plantar, colher ou regar.

2 ESTRUTURA DAS CLASSES

2.1 Classe Interface

Gere a interação com o utilizador e o fluxo principal do programa.

É responsável por receber, interpretar e encaminhar os comandos introduzidos.

2.1.1 Atributos principais

- Jardim* jardim – Representa um ponteiro para o jardim atual
- bool ativo – Indica se o simulador está em execução

2.1.2 Métodos principais

- executar() – Ciclo principal de leitura e execução de comandos.
- processarComando(const string & linha) – Interpreta o comando introduzido pelo utilizador e invoca o comportamento correspondente.

2.2 Classe Comando

Implementa as ações concretas associadas aos diferentes comandos do utilizador. Cada método representa um comando do simulador.

2.2.1 Métodos principais

- comandoJardim() – Cria um jardim com dimensões definidas
- comandoAvanca() – Avança o tempo de simulação
- comandoListarInfoSolo() – Mostra as informações detalhadas de uma célula do solo.
- comandoPlantar() – Planta uma nova planta numa posição indicada
- Outros comandos complementares, como colher, pega, larga, etc.

2.3 Classe Jardim

Representa o ambiente principal do simulador que contém a grelha de solo e o jardineiro. É responsável por gerir o avanço da simulação, as interações e a impressão visual do jardim.

2.3.1 Atributos principais

- int nLinhas – Número de linhas da grelha
- int nColunas – Número de colunas da grelha
- Solo** grid – Matriz bidimensional de células de solo.
- Jardineiro* jardineiro - Representa um ponteiro para o jardineiro atual

2.3.2 Métodos principais

- imprimir() const – Mostra o estado atual do jardim no ecrã
- avanca(int n) – Faz avançar o tempo da simulação atualizando o estado das plantas.
- getSolo(int, int) - Devolve o ponteiro para a célula de solo nas coordenadas indicadas
- planta(int, int, char) – Planta uma planta de um tipo específico numa célula
- listarPlantas() – Mostra todas as plantas existentes no jardim
- listarPlanta(int, int) – Mostra uma planta especifica de uma determinanda posição
- colhe(int, int) – Colhe a planta de uma posição
- getJardineiro() – Acede ao objeto jardineiro associado ao jardim.

2.4 Classe Solo

Representa uma célula do jardim com propriedades locais (água, nutrientes, planta, ferramenta). Funciona como intermediário entre o ambiente e os objetos contidos.

2.4.1 Atributos principais

- int agua – Quantidade de água presente no solo
- int nutrientes – Quantidade de nutrientes disponíveis
- Planta * planta – Ponteiro para a planta presente numa célula
- Ferramenta * ferramenta – Ponteiro para a ferramenta presente numa célula

2.4.2 Métodos principais

- imprimir() const – Representa visualmente o estado do solo
- adicionaAgua (int) – Aumenta a quantidade de água na célula
- retiraNutrientes(int) – Reduz a quantidade de nutrientes do solo.
- criarPlanta(Planta*) – Define a planta presente na célula
- obterPlanta() – Devolve o ponteiro para a planta atual
- removerPlanta() – Remove a planta da célula
- imprimirDetalhado() const – Representa visualmente o estado do solo, mas de forma mais detalhada

2.5 Classe Planta (abstrata)

Classe base que define o comportamento comum a todas as plantas do simulador. Fornece uma interface padrão para o ciclo de vida das plantas.

2.5.1 Atributos principais

- int agua – Quantidade de água presente no solo
- int nutrientes – Quantidade de nutrientes disponíveis
- bool estaViva – Estado e vida da planta

2.5.2 Métodos principais

- virtual void agir (Solo&) = 0 – Define o comportamento de crescimento e absorção
- virtual void morrer (Solo&) = 0 – Executa a ação de morte da planta.
- virtual char getSimbolo() const = 0 – Retorna o símbolo representativo da planta.
- bool estaViva() – Indica se a planta continua viva.
- int obterAgua() – Obtém a quantidade de água atual da planta
- int obterNutrientes() – Obtém a quantidade de nutrientes atual da planta
- void adicionaNutrientes(int) – Adiciona nutrientes à planta

2.6 Classe Ferramenta (abstrata)

Classe base para todas as ferramentas usadas pelo jardineiro. Define a estrutura e o comportamento

2.6.1 Atributos principais

- int capacidade – Representa a utilização da ferramenta
- int unidades – Número de ferramentas disponíveis
- int nSerie – Identificador único da ferramenta

2.6.2 Métodos principais

- virtual void utilizar(Solo &) = 0 – Define a ação da ferramenta sobre o solo
- virtual void reduzCapacidade(Solo &) = 0 – Reduz a capacidade da ferramenta
- virtual char getSimbolo() const = 0; - Indica o símbolo da ferramenta a utilizar

2.7 Classe Jardineiro

Entidade controlada pelo utilizador, responsável por interagir com o jardim. Pode mover-se, plantar, colher e manipular ferramentas.

2.7.1 Atributos principais

- int linha – Linha atual no jardim

- int coluna – Coluna atual no jardim
- bool dentro – Indica se o jardineiro está dentro ou fora do jardim.
- Ferramenta* naMao – Ferramenta atualmente em uso.

2.7.2 Métodos principais

- void entrar(int, int) – Coloca o jardineiro dentro do jardim numa posição específica
- void sair() – Faz o jardineiro sair do jardim
- void mover(char) – Move o jardineiro em determinada direção (cima, baixo, esquerda, direita)
- bool temFerramenta() const – Verifica se o jardineiro se tem uma ferramenta na mão
- Ferramenta* obterFerramentaNaMao() const – Indica a ferramenta que tem na mão

3 COMANDOS DO SIMULADOR

3.1 Comando jardim <linhas> <colunas>

Cria um jardim com as dimensões especificadas. Se já existir um jardim, não é possível criar outro.

3.2 Comando executa <ficheiro>

Executa automaticamente uma sequência de comandos a partir de um ficheiro de texto. Permite automatizar operações do simulador.

3.3 Comando avanca [n]

Faz avançar a simulação n instantes de tempo (por defeito 1). Atualiza o estado de todas as plantas vivas no jardim.

3.4 Comando planta <linha> <coluna> <tipo>

Planta uma nova planta do tipo especificado na célula indicada.

3.5 Comando colhe <linha> <coluna>

Colhe a planta existe na posição indicada

3.6 Comando lplantas

Lista todas as plantas atualmente presentes no jardim

3.7 Comando lplanta <linha> <coluna>

Mostra as informações detalhadas da planta existente na posição indicada

3.8 Comando larea

Mostra todas as posições ocupadas no jardim

3.9 Comando lsolo <linha> <coluna> [n]

Mostra toda a informação do solo na posição indicada. Se for fornecido o parâmetro [n], também mostra as células vizinhas.

3.10 Comando lferr

Lista todas as ferramentas disponíveis no simulador

3.11 Comando entra <linha> <coluna>

Coloca o jardineiro dentro do jardim na posição indicada

3.12 Comando sai

Faz o jardineiro sair do jardim.

3.13 Comandos de movimento (e, d, c, b)

Movem o jardineiro numa célula. Esquerda (e), direita (d), cima (c), baixo (b).

3.14 Comando larga

Larga a ferramenta que o jardineiro tem na mão.

3.15 Comando pega <n>

Pega na ferramenta com o número de série n se esta estiver na célula atual ou no inventário.

3.16 Comando compra <tipo>

Adquire uma nova ferramenta do tipo especificado.

3.17 Comando grava <nome>

Guarda o estado atual do simulador sob o nome especificado.

3.18 Comando recupera <nome>

Restaura o estado do simulador gravado anteriormente.

3.19 Comando apaga <nome>

Remove uma gravação anteriormente criada.

3.20 Comando ajuda

Mostra a lista completa de comandos e explicações básicas de utilização.

3.21 Comando fim

Termina a execução do simulador.

3.22 Comando “” (vazio)

Ignorado pelo sistema. Usado para fazer *enters* na consola sem gerar erros.

4 FUNCIONAMENTO DO PROGRAMA

Ao iniciar o programa, a classe Interface é responsável por apresentar uma mensagem de boas-vindas e aguarda a introdução de comandos por parte do utilizador. Cada linha introduzida é interpretada pelo método processarComando(), que identifica o tipo de instrução e executa a ação correspondente. A Interface não contém a lógica da simulação em si, apenas gere a comunicação entre utilizador e classe Jardim, à qual delega as operações principais do sistema, através da classe Comando.

A classe Jardim representa o ambiente principal do simulador, sendo composta por uma grelha de células (Solo) que formam o espaço do jardim. Cada célula do solo pode conter uma planta e mantém o seu próprio estado, com valores de água e nutrientes que variam ao longo do tempo. Além disso, o jardim contém um objeto do tipo Jardineiro, que atua como o agente controlado pelo utilizador, pondendo entrar, sair e mover-se entre as diferentes posições da grelha para plantar ou colher.

A classe Planta, define o comportamento comum a todas as plantas do simulador, incluindo a absorção de água, o consumo de nutrientes e a morte em condições adversas. As subclasses implementam comportamentos específicos de crescimento e multiplicação baseando-se nos parâmetros definidos na classe Settings. Por sua vez a classe Solo funciona como intermediária entre o jardim e as plantas, armazenando o seu estado e fornecendo métodos para adicionar água, retirar nutrientes ou associar uma nova planta.

5 CONCLUSÃO

Este trabalho, apesar de incompleto, demonstra de forma prática os principais conceitos da programação orientada por objetos, como herança, polimorfismo, encapsulamento e composição.

Através da interação entre o utilizador, o jardineiro e os diferentes elementos do jardim, o programa recria um ambiente dinâmico e modular, permitindo a fácil expansão e manutenção de código.

Instituto Superior
de Engenharia
Politécnico de Coimbra