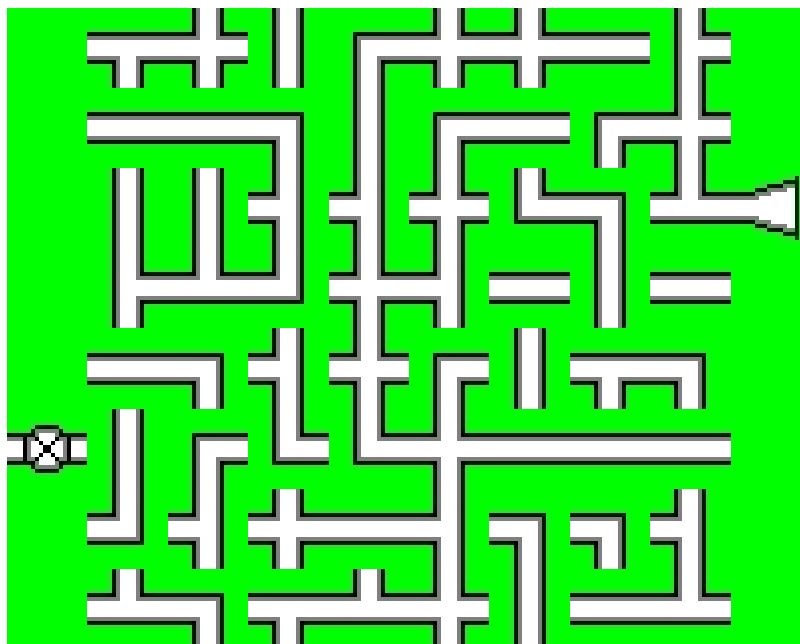


## **Programação Orientada por Objectos**

### **Sistema de Canalização**



**Ano Lectivo: 2010/2011**

**Época Especial**

## Introdução

O objectivo deste projecto é desenvolver, utilizando a linguagem Java e a metodologia de Programação Orientada por Objectos, um programa que permita a resolução de puzzles.

O problema em estudo consiste na criação de um sistema de canalização que permita levar a água de um ponto até ao extremo oposto.

Imagine que tem um jardim rectangular, onde existe um ponto de saída de água (☒), sempre localizado do lado esquerdo, e um ponto de destino da água (☒), sempre localizado do lado direito:



Entre estes dois pontos existem 4 tipos de canos:

- Linear .....
- Em L .....
- Em T .....
- Em Cruz .....

que estão dispersas por todo o terreno (excepto nos extremos):



O objectivo de resolução consiste em alterar o sistema de canalização, apenas através da rotação de cada tipo de cano, até construir um caminho, fechado, desde o ponto origem até ao ponto destino:



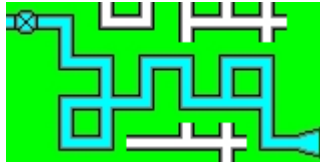
Os quatro tipos de canos podem ser rodados, nunca movidos, conseguindo-se as seguintes disposições:

- Linear ..... ⇒
- Em L ..... ⇒ ⇒ ⇒
- Em T ..... ⇒ ⇒ ⇒
- Em Cruz .....

A rotação de cada tipo de cano é conseguida através do simples clique do rato sobre ele. Cada clique do rato provoca uma rotação de 90°, por exemplo para o tipo Em L:



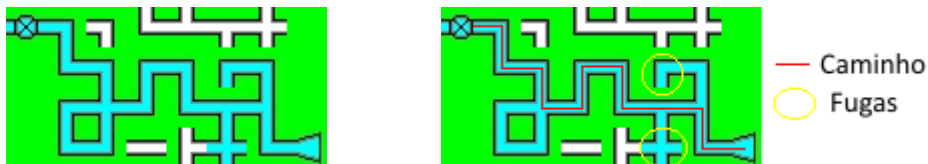
Quando o trabalho está concluído, após indicação do utilizador, o sistema deve validar a sua correcta execução:



Não são consideradas válidas situações onde no caminho existem aberturas (fugas na canalização):



No exemplo anterior, apesar de existir um caminho completo desde a origem até ao destino, temos pontos de fugas de água:

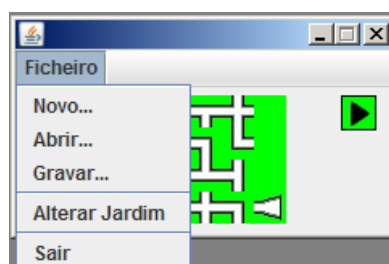


## Funcionalidades

O programa deve implementar as seguintes funcionalidades:

- Criar um puzzle, permitindo a colocação dos diversos tipos de canos no terreno;
- Alterar/corrigir o puzzle actual;
- Gravar o puzzle, criando um ficheiro com a caracterização do puzzle;
- Importar um puzzle gravado;
- Deixar o utilizador rodar os canos para resolver o puzzle, devendo, no final, verificar a sua correcção.

Caso opte por uma interface baseada em menus será algo do tipo:

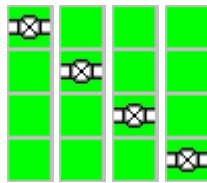


## Criação de puzzles

A opção de criação de puzzles deverá começar por pedir o número de linhas e colunas pretendido para a dimensão do puzzle. Às colunas indicadas deverão ser acrescentadas mais 2 colunas para conter a origem e o destino. Para um caso de 4 linhas por 6 colunas, teremos:

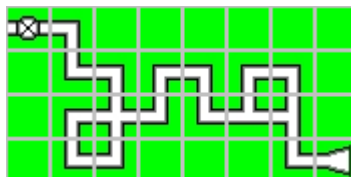


Na zona da Origem deverá ser possível colocar a origem em qualquer posição:



de forma análoga, se deverá poder colocar o Destino em qualquer das posições.

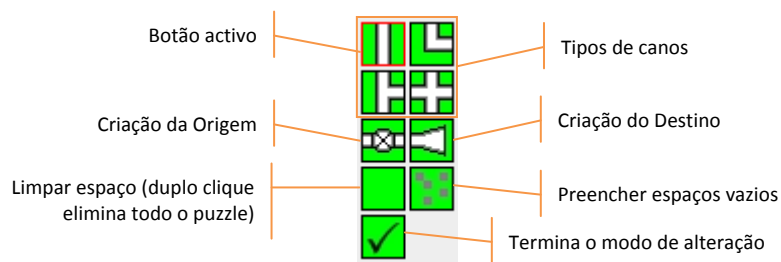
O programa deverá permitir colocar os diversos tipos de canos na zona central do puzzle. A forma de colocação em cada casa de um tipo de cano não deve colidir com a possibilidade de rotação, para que, à medida que se vai construindo o puzzle, se possa testar da sua validade:



No final deverão ser preenchidos os restantes espaços vazios, para que o puzzle fique completo. Poder-se-á criar um comando que executa esse trabalho, introduzindo, aleatoriamente, diversos tipos de cano nos espaço vazios:



Caso opte por uma interface baseada numa paleta de botões será algo do tipo:



Ao terminar o modo de alteração o puzzle deve ser “baralhado” (todos os canos devem ser rodados aleatoriamente) para se garantir que não é apresentada a solução ao utilizador.

## Alterar puzzle

Durante a criação do puzzle deve ser possível voltar ao modo de alteração do puzzle para efectuar correcções/adaptações.

## Gravar puzzle

O sistema deverá permitir a gravação do puzzle actual.

Deverá ser pedido um nome para o puzzle e ser criado um ficheiro, com esse nome, e com o conteúdo que descreva por completo o puzzle. Não é relevante guardar o estado da rotação dos tipos de canos pois quando ele for importado ou de cada vez que se tente resolver o puzzle todos os canos deverão ser rodados aleatoriamente.

O ficheiro criado poderá ser copiado para outra pasta e/ou máquina para permitir a criação de bibliotecas de puzzles.

## Importar puzzle

O sistema deverá permitir ler um ficheiro com o descritivo de um puzzle e colocá-lo como puzzle a resolver.

Deverá ser possível indicar o nome e, eventualmente, a directoria do ficheiro que se pretende importar.

Após a importação o puzzle deverá ser apresentado com todas as peças rodadas aleatoriamente, para se garantir que nunca se apresenta a solução do mesmo.

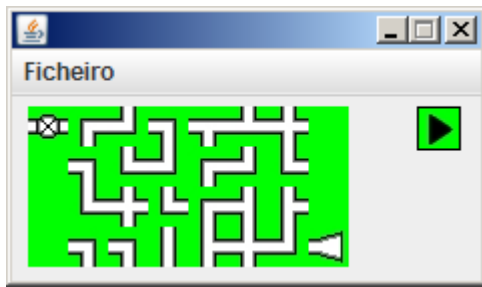
## Resolução do puzzle

Apenas deverá ser possível rodar os canos do puzzle em modo de resolução.

Neste modo podemos rodar qualquer dos canos (com excepção dos extremos) e, quando desejado, pedir para testar a solução (parando a possibilidade de rotação).

Caso se re-inicie, após testar uma eventual resolução, o puzzle deverá ser novamente “baralhado”, para não permitir a resolução cumulativa.



Deverá existir uma forma de indicar a mudança de estado “em resolução”/”resolvido”. Utilizando, por exemplo, botões, poderá ser algo do tipo:



Antes de entrar no modo de resolução  
(não é possível rodar os canos)



Em resolução  
(é possível rodar os canos)

O botão  dá início ao modo de resolução. O botão  termina o modo de resolução, validando e apresentando a indicação de sucesso ou insucesso e assinalando o caminho utilizado.

## Sugestões de Desenvolvimento

- Crie uma classe abstracta que represente um tipo de cano e permita:
  - guardar a sua informação (quantas ligações dispõe, em que orientação é que está e quem são os seus “vizinhos”, etc.);
  - realizar as suas operações (rodar, rodar aleatoriamente, verificar se tem alguma ligação aberta, testar se está na solução, etc.);
- Crie classe concretas que herdem da classe anterior e representem um tipo concreto de cano (linear, em L, em T e em cruz), com um número concreto de ligações;
- A classe Jardim deverá armazenar uma tabela bi-dimensional de tipos de canos;
- Implemente graficamente a zona que representa o Jardim através de um GridLayout, onde cada elemento será um botão que corresponde ao tipo de cano respectivo;
- Crie icons para cada situação possível: disposição do cano, orientação e situação (em resolução/em fase de teste). A escolha do icon a apresentar, no botão respectivo, depende dessas três variáveis: do tipo de cano, da sua orientação e da situação (resolução/teste). Por exemplo, para representar as diversas hipóteses de um cano em L, será preciso criar 8 icons:



## Regras Gerais de Desenvolvimento

- O trabalho deve ser desenvolvido em duas etapas: Modelização e Implementação.
- Sobre a etapa de modelização:
  - Deverá ser realizado um relatório com a análise e o design do programa. Neste relatório deve constar um **diagrama de classes em UML** onde se representam as classes e interfaces e relações entre as mesmas e que façam parte da lógica do programa (sem considerar interface com o utilizador). Separadamente, e que será complementada pelo JavaDoc, deve existir uma **descrição de cada uma das classes**

que fazem parte do diagrama onde se incluam a **explicação dos atributos e métodos** e das **responsabilidades de cada classe**.

- Sobre a etapa de Implementação:
  - O programa deve ser desenvolvido utilizando a linguagem Java e os métodos de programação orientada por objectos.
  - Para os identificadores siga as convenções adoptadas normalmente para a linguagem Java:
    - A notação camelCase para o nome das variáveis locais e identificadores de atributos e métodos;
    - A notação PascalCase para os nomes das classes e interfaces;
    - Maiúsculas para os nomes das constantes e dos valores enumerados;
    - Não utilize o símbolo '\_' nos identificadores (excepto nas constantes), nem abreviaturas.
  - Não existem restrições quanto às características e/ou funções da linguagem Java utilizadas, sendo no entanto requerido que o programa seja desenvolvido num ambiente gráfico usando o Swing.
  - Toda a interface com o utilizador fica ao critério do programador, sendo premiadas a facilidade de utilização e a apresentação.
  - É necessário que o projecto cumpra o que é pedido no seu enunciado, sendo deixado ao critério do programador qualquer pormenor de implementação que não seja referido no mesmo.

## Regras de Entrega do Projecto

- Cada projecto deverá ser elaborado em **grupos de dois alunos**, podendo eventualmente ser elaborado individualmente. Não serão permitidos em nenhum caso grupos de mais do que dois alunos.
- O projecto deve ser entregue até à data limite especificada por **via exclusivamente electrónica utilizando o Moodle** ou em último caso enviado por correio electrónico para o endereço do professor responsável. Todos os ficheiros que compõem o projecto deverão estar guardados num único ficheiro compactado em **formato ZIP ou RAR**.
- Todos os materiais do projecto devem ser devidamente identificados com nome, número, turma, curso e endereço de email dos alunos.
- **Projecto Final** – O projecto final deve incluir:
  - Um **Manual Técnico** onde conste uma breve descrição do programa, incluindo:
    - O **diagrama de classes UML** (apresentação da etapa de Modelação do projecto).
    - A documentação do programa em **JavaDoc** (não converta o documento gerado automaticamente em HTML para .doc)
  - O **código fonte do programa** na forma de projecto em NetBeans.

## Regras de Avaliação do Projecto

- A entrega do projecto depois do prazo estipulado implica a penalização de um valor por dia de atraso até um máximo de três valores. Depois disso o projecto não será aceite.
- A classificação do programa terá em conta a qualidade da programação (factores de qualidade do software), a estrutura do código criado segundo os princípios da programação orientada por objectos, tendo em conta conceitos como a coesão de classes e métodos, o grau de acoplamento entre classes e o desenho de classes orientado pela responsabilidade, e a utilização/conhecimento da linguagem Java.
- Serão premiadas a facilidade de utilização, a apresentação, a imaginação e a criatividade.
- O projecto terá uma componente de avaliação oral obrigatória com classificação individual dos elementos do grupo. Os alunos que não comparecerem à discussão serão classificados com zero. Nesta discussão com os alunos será apurada a capacidade do aluno de produzir o código apresentado. Nos casos em que essa capacidade não for demonstrada a nota atribuída será zero.
- A avaliação oral é realizada pelo professor responsável e irá ser feita uma marcação prévia para cada grupo de trabalho.
- Todos os projectos serão submetidos a um sistema de detecção de cópias automático. Os projectos que forem identificados como cópias e que se verificar serem cópias serão anulados.

## Critérios de Avaliação

O projecto será avaliado segundo os seguintes critérios:

- Diagrama de classes em UML ..... **3**
  - Identificação das Classes Principais ..... 1
  - Identificação correcta das relações “is-a” ..... 0,5
  - Identificação correcta das relações “has-a” ..... 0,5
  - Identificação dos atributos principais ..... 0,3
  - Identificação dos métodos principais ..... 0,3
  - Uso correcto da Notação UML ..... 0,4
- Manual Técnico ..... **1**
  - Descrição das classes e aspectos particulares da implementação ..... 0,5
  - Documentação JavaDoc ..... 0,5
- Avaliação do programa ..... **16**
  - Utilização do ambiente gráfico Swing ..... 4
  - Criação de puzzles com tamanhos variáveis ..... 1
  - Possibilidade de indicar a posição da Origem e do Destino ..... 0,5
  - Possibilidade de apagar todo o puzzle ..... 0,5
  - Criação de comando para preencher os espaços vazios ..... 1
  - Gravação dos puzzles em ficheiro ..... 1
  - Importação de puzzles gravados ..... 1
  - Detecção do sucesso/insucesso da solução ..... 1
  - Capacidade de lidar com circularidades ..... 1



- Destacar o caminho na solução final ..... 1
- Estilo de programação (avaliação subjectiva, por parte do docente, da qualidade da programação) ..... 4

Caso deseje poderá implementar as seguintes funcionalidades extras (com respectivas avaliações):

- Extras..... 4
  - Criação de um temporizador que controla o modo de resolução. Findo o prazo do temporizador o programa força a avaliação da solução existente. Caso o utilizador termine antes do tempo deve ser indicado o tempo total utilizado..... 1
  - Poder associar a cada puzzle o menor tempo de resolução até à data (necessita de implementar o temporizador anteriormente referido e gravá-lo no ficheiro em conjunto com a estrutura do puzzle). No final de cada resolução, deverá ser indicado, para além do tempo utilizado o respectivo melhor tempo ..... 1
  - Criar uma biblioteca de puzzles, ordenada por níveis de dificuldade (dimensão e tipo de canos utilizados), que serão apresentados por ordem crescente após a resolução de cada puzzle ..... 2

## Datas do projecto

Para o presente projecto destacam-se as seguintes datas

- Distribuição do Trabalho de Projecto – 18 de Julho de 2011.
- Entrega do Trabalho de Projecto - Antevéspera do Exame de Época Especial.
- Discussões do Projecto – Dia do Exame de Época Especial (imediatamente a seguir ao exame).