Enunciado do Trabalho Final de POO 2020

Introdução

O trabalho final consiste em criar um jogo do tipo conhecido como *Transport Puzzle* (esta variante em particular é conhecida como Sokoban). As principais características deste tipo de jogos são:

- É um jogo de estratégia.
- O personagem controlado pelo utilizador (neste caso uma empilhadora) faz um movimento de cada vez numa de quatro direções possíveis. O objetivo é arrumar um conjunto de caixas em locais pré definidos (marcados com "X"), empurrando-as e evitando que fiquem em posições de onde não é possível retirá-las. O jogador só consegue empurrar um caixote de cada vez.
- Normalmente, ao terminar a arrumação num nível do jogo, passa-se para um novo nível que tem um problema ligeiramente mais difícil. São valorizadas as soluções com o menor número de movimentos da empilhadora.

Objectivos

Usando o interface gráfico fornecido em anexo e respeitando o desenho de classes na figura 2 deve criar o conjunto de classes que constituem o "motor de jogo" para um jogo de Sokoban. O motor de jogo deve permitir jogar de um modo razoavelmente eficaz.

Ao ser resolvido um nível deve ser atribuída uma pontuação ao jogador, que será o número de movimentos (quanto menos movimentos, melhor) e deve ser mantido um registo das melhores pontuações em ficheiro.

Requisitos

Os mapas das salas devem ser lidos de ficheiro com um formato indicado na secção `Ficheiros de salas`.

Uma das classes do seu projeto deve registar-se como observador do interface gráfico para receber informação sobre as teclas (ver exemplo de código no pacote anexo ao enunciado). **Atenção, o código de exemplo serve apenas para exemplificar algumas das possibilidades de utilização da biblioteca**. O motor de jogo recebe do interface gráfico informações sobre as teclas pressionadas (chegam por invocação da função `update` como se pode ver no exemplo fornecido). As teclas de setas devem controlar a posição do personagem controlado pelo jogador.

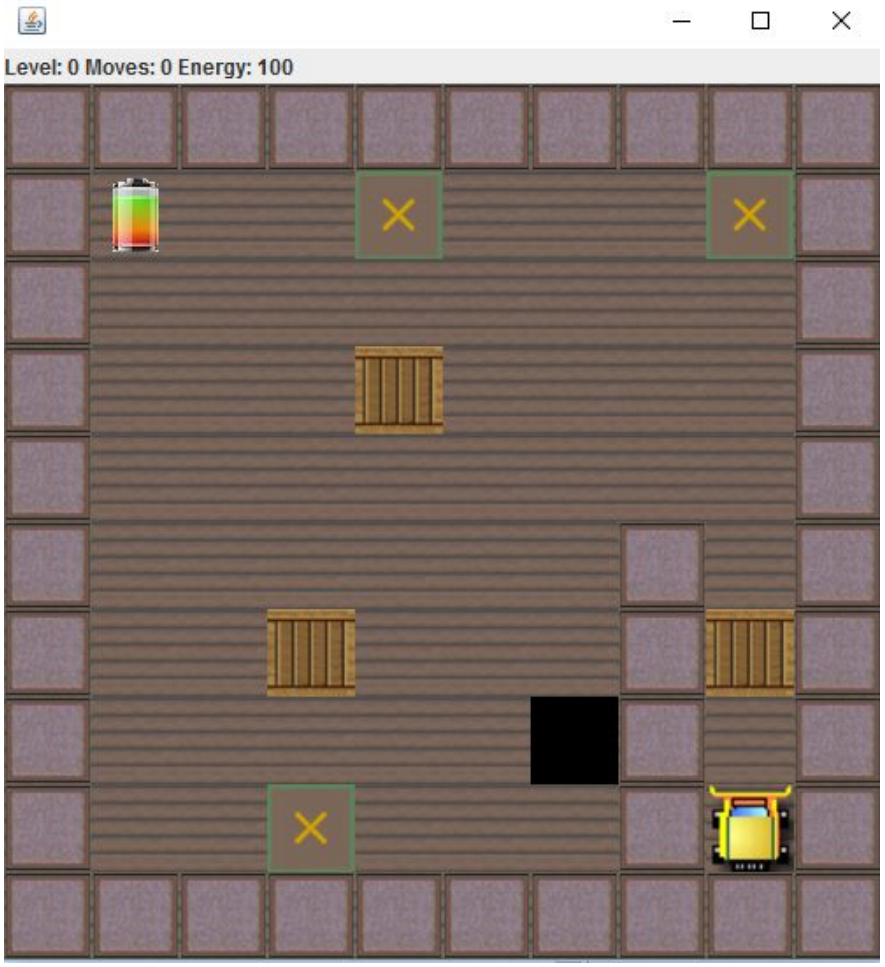

Fig. 1 Exemplo da aplicação no início de um nível, com a empilhadora controlada pelo jogador no canto inferior direito, três caixas que têm de ser empurradas para os locais assinalados com "X" e um buraco que irá "engolir" um caixote ou a empilhadora se passarem por cima, no primeiro caso o jogo continua até ao limite de movimentos (ficar sem energia), no segundo caso o jogador perde de imediato o jogo. Há também uma bateria para repôr a energia da empilhadora, que, caso chegue a zero, faz perder o jogo. A barra de estado indica o nível e pode também indicar o número de movimentos feitos até ao momento.
(crédito imagens: http://www.sokoban-online.de/help/sokoban/the-sokoban-game.html).

O motor de jogo pode enviar imagens para a janela usando implementações de `ImageTile` (como algumas classes do exemplo fornecido). De cada vez que há uma alteração (por exemplo, mudança das posições das imagens) deve ser chamado o update do interface gráfico. **Não é necessário re-enviar as imagens para o interface após cada alteração da posição, isso irá apenas sobrecarregar o interface gráfico e tornar o jogo mais lento à medida que a lista de imagens vai crescendo.**.

A interacção entre objectos deve ser tão autónoma quanto possível (passando o mínimo possível pela classe principal do motor de jogo).

O formato para os ficheiros onde são guardadas as melhores pontuações é livre.

As classes fornecidas no pacote `pt.iul.ista.poo.gui` **não devem ser alteradas** e são

fundamentais para a resolução (para garantir isso os docentes poderão substituir esse package pela versão "oficial" na apresentação / correcção do trabalho).

As classes no pacote `pt.iul.ista.poo.utils` podem ser complementadas caso ache necessário (mas as funcionalidades já existentes devem manter-se).

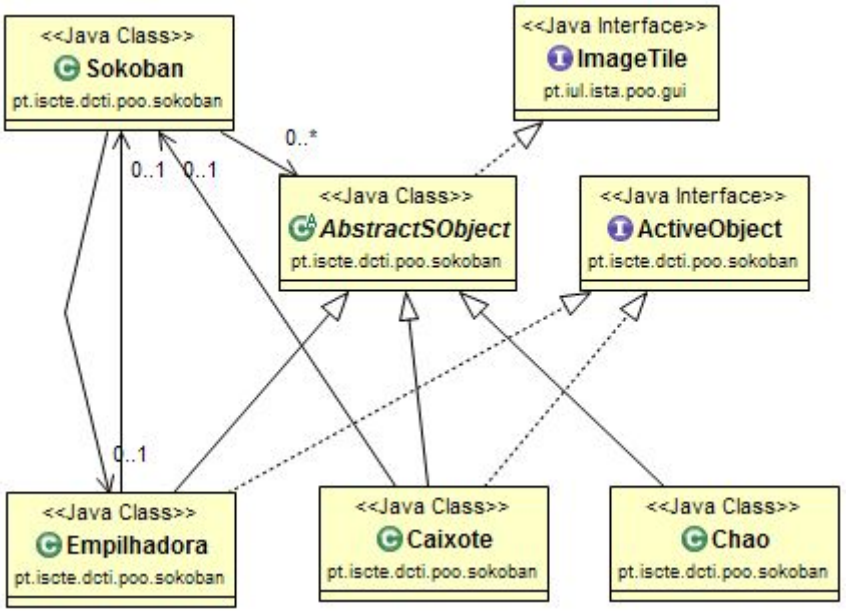

Figura 2. Desenho em UML genérico das principais classes

O objetivo geral é que, ao realizar o trabalho, consolide e explore os conceitos próprios de POO. Por isso, para além da componente funcional do trabalho, **é fundamental que demonstrem a utilização da matéria dada em POO**. Assim, é **obrigatória** a utilização de:

- ocultação de informação (boa utilização do **private** e manutenção de um interface independente do conteúdo das classes)
- herança de propriedades e sobreposição de métodos na interação entre objetos do jogo
- classe(s) abstracta(s)
- definição e implementação de interface(s)
- listas
- leitura de ficheiros com a configuração dos níveis
- escrita de ficheiros com as melhores pontuações de cada nível
- excepções (lançamento e tratamento) - tratamento de problemas que possam acontecer na leitura dos ficheiros, ou na passagem de informação entre as suas classes (argumentos nulos ou outros problemas). É necessário demonstrar que sabe usar excepções, não é obrigatório usá-las em todos os casos em que possam surgir problemas.

É ainda aconselhada a experimentação com estruturas de dados auxiliares diferentes das listas (mapas, conjuntos ou filas) em situações em que estas sejam adequadas.

Os objetos de jogo podem ainda incluir pedras pequenas e pedras grandes, ambas podem ser empurradas pela empilhadora, mas só uma de cada vez, tal como os caixotes. Um caixote não pode ir para cima de uma pedra nem empurrá-la. Atirar uma pedra pequena para o buraco faz

desaparecer a pedra. Se uma pedra grande cair no buraco bloqueia-o e a pedra deixa de se mexer (i.e. a pedra continua a ser intransponível e deixa de ser empurrável). O seu código deve ser flexível de modo a poder introduzir no jogo estes (ou outros) novos objetos, idealmente sem mexer em muitos pontos do código (além do ponto de criação do(s) objeto(s) que é inevitável). Se isso não acontecer, quer dizer que o seu código não está adequadamente modularizado para permitir a extensão a outros objetos, tente alterá-lo de modo a que a introdução de novos objetos no jogo seja tão simples quanto possível.

Ficheiro de configuração de níveis

Na figura 3 pode ver o ficheiro de configuração de um nível que contém paredes ('#'), três caixotes ('C'), uma empilhadora ('E'), três locais para arrumação ('X'), uma bateria ('b') no canto superior esquerdo e um buraco ('O'). Repare que o ficheiro tem 10x10 posições, exatamente o número de posições que é possível mostrar no interface gráfico. Se existirem, pedras pequenas são indicadas por p e pedras grandes por P.

```
##########
#b  X   X#
#        #
#   C    #
#        #
#      # #
#  C   #C#
#     O# #
#  X   #E#
##########
```

Figura 3. Exemplo de um nível simples

Interface Gráfico

O interface gráfico fornecido (classes `ImageMatrixGUI` e interface `ImageTile`, no `package pt.iul.ista.poo.gui` permite abrir uma janela como a apresentada na figura 1), que tem dentro uma área com 500x500 píxeis, vista como uma grelha bidimensional de 10x10 posições para imagens de 50x50 píxeis. Além dessa zona, o utilizador desta biblioteca dispõe de uma barra de estado acima desta, onde se podem mostrar mensagens simples com informações sobre o jogo. As imagens que são usadas fazem parte de uma "biblioteca" que se encontra na pasta "imagens" dentro do projeto. O *interface* `ImageTile` é usada para indicar qual a imagem a desenhar e qual a posição em que esta será desenhada dentro da grelha de 10x10 posições.

Ao implementar o interface `ImageTile` é obrigatório indicar o nome da imagem (sem extensão), a posição de desenho na grelha de 10x10 que constitui a janela visível e a sua camada para determinar quais as imagens que ficam por baixo ou por cima. São usadas as funções

```
public void addImages(final List<ImageTile> ...)
public void addImage(final ImageTile ...)
public void removeImage(final ImageTile ...)
Public void clearImages()
```

```
public void setStatusMessage(final String message)
```

Caso prefira, pode usar outras imagens para representar os elementos do jogo mas, como esse não é um dos objetivos do trabalho, não é contabilizado, nem serão prioritárias as dúvidas ou erros relacionados com a utilização de outras imagens. À partida não são esperados problemas desde que todas as imagens usadas tenham uma dimensão de 50x50 píxeis. Caso use imagens que não sejam da sua autoria, no trabalho devem ser dados os devidos créditos aos autores. Este *interface* será distribuído e explicado numa aula. Poderão vir a ser publicadas atualizações ao pacote de classes fornecido caso sejam detectadas falhas de funcionamento, por isso mantenha-se atento aos avisos.

Execução do Trabalho

O trabalho deve ser feito por grupos de dois alunos e espera-se que demore cerca de 40h a executar. Poderá também ser feito individualmente - nesse caso estima-se que o tempo de resolução seja um pouco maior, mas não substancialmente. Recomenda-se que seja feito por grupos de dois alunos com um nível de conhecimentos semelhantes porque a discussão das opções de implementação é em geral muito benéfica para a aprendizagem.

É encorajada a discussão entre colegas sobre o trabalho, mas é estritamente proibida a troca de código. Atenção que a partilha de código em trabalhos diferentes será seriamente penalizada. Serão usados os meios habituais de verificação de plágio e os trabalhos plagiados serão comunicados ao Conselho Pedagógico para procedimento disciplinar. Serão ainda feitas discussões do trabalho, das quais poderão ser dispensados os trabalhos que foram acompanhados pelo docente que está a avaliar.

Peça regularmente ao docente para que reveja o trabalho consigo, quer durante as aulas, quer em horário de dúvidas (com marcação). Desse modo:

1. pode evitar algumas más opções iniciais que normalmente conduzem a um grande aumento da quantidade de trabalho;
2. a discussão final do trabalho é menos crítica e poderá ser dispensada, dado que tanto o aluno como o docente foram discutindo o trabalho e as opções tomadas.

Entrega

Na primeira fase o seu projeto deve conter:

- Desenho (de preferência em UML) da proposta de organização de classes (basta que contenha os nomes das classes e as relações entre elas). Deve ser consistente com a Figura 2, caso tenha alterações estas devem ser justificadas.
- Leitura de um ficheiro com o formato descrito neste enunciado, contendo a configuração de um nível simples
- Mostrar esse nível no interface gráfico, e permitir ao utilizador controlar o personagem (empilhadora, caso use as imagens fornecidas) com as teclas

A demonstração da primeira fase deve ser **na aula de segunda-feira 20-abr-2020 na turma que frequenta habitualmente, ou, caso as aulas presenciais não estejam ainda a funcionar, entregue no e-learning até às 8:00**.

Na segunda faseo seu projeto deve conter:

- O personagem controlado não deve atravessar paredes nesta fase.
- Devem ter caixotes e poder empurrá-los

No final destas duas fases, a avaliação ditará se:

- Pode continuar diretamente (nota A ou B),
- precisa de apresentar de novo o trabalho com correcções na semana seguinte (C),
- tem de marcar uma sessão aconselhamento sobre o rumo a seguir nessa mesma semana (D),
- ou foi eliminado (E).

A demonstração da segunda fase deve ser **na aula de segunda-feira 04-mai-2020 na turma que frequenta habitualmente, ou, caso as aulas presenciais não estejam ainda a funcionar, entregue no e-learning até às 8:00**.

A entrega final será até às **08:00 do dia 25 de Maio de 2020**. Nesta entrega, além do código deverá ter um relatório sucinto **em PDF dentro do projeto** em que é obrigatório o desenho UML das principais classes do trabalho, uma declaração dos membros do grupo que atesta que o código entregue neste trabalho é integralmente da sua autoria. É opcional ainda uma breve discussão das opções tomadas e/ou um manual de utilização do jogo.

Para entregar (em todas as fases) deve:

- **Garantir que o seu projeto tem no seu nome o nome e número de aluno de todos os membros do grupo**
- Incluir no projeto todos os ficheiros que forem necessários para a entrega.
- Usar *File/Export/Archive File/*, selecionar o projeto onde tem o trabalho final e **entregar no *e-learning* na pasta do docente que acompanha nas aulas de segunda-feira**.

Tenha em atenção que:

- Todos os componentes devem estar dentro da pasta do projeto, incluindo as imagens.
- A exportação/importação para outro computador devem permitir o funcionamento normal do projeto. A importação e execução noutro computador devem ser testadas antes da entrega.
- Não devem usar caracteres acentuados no projeto.

Trabalhos que não sejam corretamente entregues poderão não ser vistos.

Avaliação

O trabalho será classificado com A, B, C ou D. A avaliação será feita do seguinte modo:

Serão excluídos (não aceites para discussão e classificados com zero valores) os trabalhos que :

- tenham erros de sintaxe;
- tenham um funcionamento muito limitado, sem que componentes essenciais do projeto estejam implementados. Por exemplo, se a gestão de colisões dos objetos não estiver

implementada é considerado que o trabalho não está utilizável;
- não usem herança ou interfaces (por exemplo no comportamento e características dos objetos) e coleções (por exemplo para guardar os vários objetos presentes numa cena);
- concentrem o comportamento todo do programa num método, numa classe ou não demonstrem que sabem usar corretamente classes e objetos para modularizar o código.

Serão classificados com C, os trabalhos que:
- forem jogáveis numa sala, com um personagem que se move de acordo com as decisões do utilizador; que permite empurrar caixas, interagir com baterias, buracos e paredes. Deve também assinalar o final do jogo quando todas as caixas estão no local devido;
- usem corretamente classes, herança, sobreposição de métodos e coleções;
- em que o comportamento do programa não está maioritariamente concentrado num ou dois métodos / classes e as classes estejam corretamente modularizadas e usem ocultação de informação por regra.
- Tenham um relatório de acordo com as indicações na secção Entrega.

Serão classificados com A ou B, os trabalhos com uma estrutura interna razoável, que cumpram todos os requisitos indicados na secção de Requisitos e que incluam o relatório.

São valorizados (têm maior possibilidade de serem classificados com A) trabalhos que:
- Tenham uma estrutura interna que, além dos pontos anteriores, facilite a implementação do jogo de modo realista (i.e. que resista bem ao escalamento caso se pretendam acrescentar mais elementos de jogo e muitos níveis distintos).
- Usem os conceitos dados em POO de um modo totalmente adequado à resolução do problema.
- Tenham uma boa modularização (funções e classes pequenas e com pouco código).

A discussão final é individual e todos os alunos terão que demonstrar ser capazes de fazer um trabalho com o nível de qualidade igual ao que assinaram. Ao entregar o trabalho, os alunos implicitamente afirmam que são os únicos responsáveis pelo código entregue e que todos os membros do grupo participaram de forma equilibrada na sua execução, tendo ambos adquirido os conhecimentos necessários para produzir um trabalho do mesmo tipo individualmente. Não demonstrar essa capacidade pode implicar a não aceitação do trabalho para um dos membros do grupo ou mesmo a acusação de plágio e processo disciplinar, caso haja suspeita de intenção de entregar como seu um trabalho em que não participaram.

Referências

[1] https://en.wikipedia.org/wiki/Sokoban