Enunciado do Trabalho Final de POO 2016-2017 (1º Semestre) - V1.0

Simulador de tráfego automóvel

Introdução

O trabalho final consiste em criar um simulador de tráfego automóvel que permita inserir ruas horizontais e verticais numa janela de simulação. Cada uma dessas ruas tem uma probabilidade de geração de automóveis a cada ciclo determinada no ficheiro de definição de ruas. A simulação deve gerar automóveis de diferentes tipos até ser parada (veja vídeo exemplo).

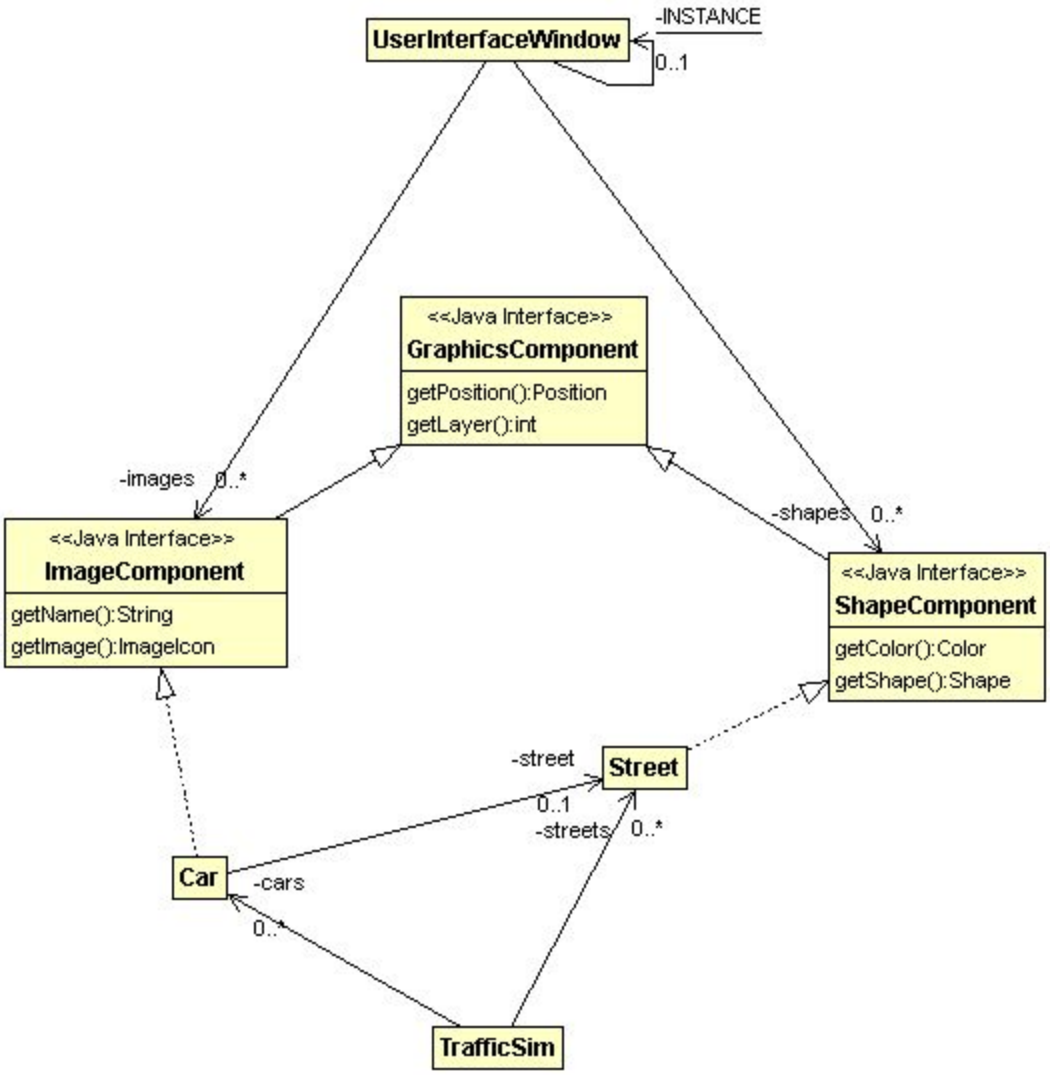

Figura 1. Desenho de alto nível do sistema

A simulação deve correr até que seja pressionado o botão para parar (dá origem à chamada da rotina `update()` no simulador, como se pode ver no exemplo fornecido). Em cada iteração da simulação deve ser dada a possibilidade a todos os carros de se movimentarem (se puderem) e o movimento dos carros deve garantir que não há colisões, nem nas ruas nem nos cruzamentos.

A biblioteca gráfica usada tem de ser a fornecida com o pacote associado ao enunciado e o desenho de alto nível do sistema deve ser semelhante ao exemplo dado na figura 1, embora, obviamente, a sua resolução irá ter outras classes que não são apresentadas aqui. Note que `UserInterfaceWindow` é um exemplo de um Solitão / Singleton (iremos estudar mais tarde) o que implica que está disponível em qualquer ponto do programa a única instância da classe usando o método `UserInterfaceWindow.getInstance()`.

Objectivos

O objectivo funcional é ter um programa que lê de um ficheiro um conjunto de ruas (apenas horizontais e verticais) que além do seu nome e da sua posição têm informação sobre a sua extensão, a quantidade de tráfego que passa nessa rua (na forma de uma probabilidade de geração de um carro em cada iteração por exemplo) e a velocidade máxima nessa rua (esta última pode estar na unidade que for mais conveniente). Deve depois criar uma simulação a partir desses dados que faça fluir o tráfego nas ruas de acordo com a informação dada, i.e. ruas que forem configuradas com uma maior quantidade de tráfego têm mais carros a entrar e ruas com menor quantidade, menos carros. Cada ftipo de carro exibe claramente o comportamento definido abaixo. Ver vídeo de exemplo.

Há um botão no interface gráfico que deve fazer terminar a simulação, este botão chama o método update() do simulador como é mostrado no exemplo fornecido.

Os carros têm um de três tipos possíveis (que pode ser distribuído aleatoriamente na sua criação):

- Carro Rápido (vermelho no vídeo): tenta ir sempre à sua velocidade máxima independentemente da velocidade máxima permitida na faixa. Quando bloqueado se tiver uma outra faixa ao lado passa para outra faixa.
- Carro Normal (azul no vídeo): tenta ir sempre um pouco abaixo da velocidade máxima permitida na faixa. Devem ser os mais frequentes. Raramente muda de faixa.
- Carro Lento (verde no vídeo): vai sempre abaixo de uma velocidade fixa, claramente inferior à velocidade máxima da faixa. Nunca muda de faixa.

No exemplo em vídeo não são demonstrados os comportamentos de mudança de faixa.

Requisitos

O objetivo geral é que ao fazer o trabalho aprenda os conceitos do POO por isso, mais do que a componente funcional é importante que demonstrem a utilização da matéria dada em POO. Assim, é **obrigatória** a utilização de:

- herança de propriedades e sobreposição de métodos (pode ser na hierarquia dos carros)
- classe(s) abstracta(s) (pode ser na hierarquia dos carros)
- implementação de interface(s) (pode ser na implementação dos interfaces `ImageComponent` e `ShapeComponent`)
- listas (há várias possibilidades, mas são fundamentais listas de carros, que devem conter carros de vários tipos)
- leitura de ficheiros (configurações das ruas)
- excepções (lançamento e tratamento) - tratamento de problemas de configuração na leitura dos ficheiros, ou na passagem de informação entre as suas classes (argumentos nulos ou outros problemas). Basta demonstrar que sabe usar excepções, não é necessário usá-las em todos os casos em que possam surgir problemas.

Para os melhores trabalhos é necessário ainda demonstrar que sabe usar:

- JUnit - Testes de uma das suas classes, uma pequena classe
- Javadoc - Documentação de uma das suas classes, uma pequena classe, pode ser a mesma que a anterior

Formato do ficheiro de definição de ruas

O ficheiro de definição de ruas **pode ter o formato que entenderem dar-lhe**, desde que possa ter informação sobre um número variável de ruas e, para cada rua, saber a sua posição no ecrã e a taxa de aparecimento de novos carros e o limite de velocidade para essa rua. Podem incluir informação adicional no ficheiro sobre cruzamentos e ruas adjacentes, mas não é obrigatório.
O ficheiro usado no vídeo de demonstração (que pode ver abaixo) está no formato JSON e foi lido com ajuda da biblioteca simple-json que se encontra em anexo ao seu trabalho. Pode usá-la, mas não é obrigatório que o faça. Pode usar qualquer modo de leitura e um formato de ficheiros ao seu gosto.

```
{"streets":[
     {"name":"A", "startPoint": {"x" : 0, "y": 100}, "streetLength" : 690,
"orientation" : "EAST", "car_probability": 0.02, "max_speed" : 3},
     {"name":"B","startPoint": {"x" : 120, "y": 0}, "streetLength" : 460,
"orientation" : "SOUTH", "car_probability": 0.01, "max_speed" : 1},
     {"name":"C", "startPoint": {"x" : 0, "y": 150}, "streetLength" : 690,
"orientation" : "WEST","car_probability": 0.01, "max_speed" : 4}
     {"name":"D","startPoint": {"x" : 180, "y": 0}, "streetLength" : 460,
"orientation" : "NORTH", "car_probability": 0.04, "max_speed" : 3},
]}
```

Interface Gráfico

O interface gráfico fornecido é composto por 3 packages:

- pt.iul.ista.poo.gui: Neste package encontra a classe que implementa a interface gráfica (`UserInterfaceWindow`) e as interfaces que usa, **esta classe e os seus interfaces não podem ser alteradas**.

- pt.iul.ista.poo.guitest: neste pacote encontra as classes usadas para o teste, em geral não têm utilidade a não ser como demonstração do modo de ligação entre a sua aplicação e o interface gráfico, caso use alguns excertos deste código é fundamental que perceba integralmente cada linha do código que usar dado que há várias soluções usadas na demonstração que não devem ser usadas no seu trabalho.
- pt.iul.ista.poo.utils: Contém classes utilitárias às quais pode acrescentar código se precisar, mas que foram usadas tal como estão na resolução oficial e por isso não deveriam necessitar de alteração para ajudar na resolução do trabalho. Não é penalizado se não usar estas classes.

Todo o código que lhe será útil foi comentado de modo a ser tão compreensível quanto possível. Foram também gerados os JavaDoc correspondentes que se encontram no projeto.

Pode, se quiser, usar outras imagens embora esse não seja um dos objetivos do trabalho, por isso não é contabilizado, nem serão prioritárias as dúvidas ou erros gerados pela utilização de outras imagens. Aconselha-se que todas as imagens usadas tenham um dos seguintes tamanhos:

- 20x40 pixels para os carros na vertical
- 40x20 para os carros na horizontal.

Os tamanhos podem ser definidos ao definir os métodos getWidth() e getHeight() das implementações de ImageComponent.

Veja com atenção o exemplo e ponha dúvidas no fórum.

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 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, que poderão ser dispensadas para os trabalhos que foram acompanhados pelo docente que está a avaliar.

Entrega

O enunciado do Trabalho Final (TF) sairá antes de 18-Out e **a primeira fase de entrega será a 7-Nov**, a **entrega final será até 12-Dez**.

Na primeira fase o seu projeto deve ter:

- Desenho (de preferência em UML) da proposta de organização de classes (apenas os nomes

das classes e as suas ligações). Baseado na figura 1, mas atualizado com a versão atual do seu projeto. Pode ser facilmente produzido com o ObjectAid se as suas classes e atributos principais já estiverem em código.
- Leitura de um ficheiro com o formato que escolher, contendo a configuração de um conjunto de estradas simples (pelo menos uma horizontal e uma vertical) e o seu desenho na interface gráfica (a sua classe que representar a rua deve para isso implementar ShapeComponent, da mesma foram que o carro implementa ImageComponent).
- Deve demonstrar um carro que passe numa das estradas. O carro deve acompanhar a estrada caso seja alterada a configuração e mudada a posição da estrada. Sugere-se que a posição do carro seja relativa à estrada e não uma posição absoluta.

A entrega final será até às 08:00 do dia 12 de Dezembro (segunda-feira). Nesta entrega, além do código deverá ter **no projeto** um relatório sumário em que é obrigatório apenas o desenho UML das principais classes do trabalho.

Para entregar (quer na primeira, quer na segunda fase) deve:
- Garantir que o nome do seu projeto tem o nome e número de todos os membros do grupo
- Incluir no projeto todos os ficheiros que forem necessários para a entrega (inclusive o relatório).
- Usar *File/Export/Archive File/*, selecionar o projeto onde tem o trabalho final e entregar tanto no e-learning como por mail.
    a. na plataforma de *e-learning*, na página de Entrega de Trabalhos (na área de conteúdos da UC), a disponibilizar brevemente
    b. por *e-mail* para o docente que acompanhou o trabalho. Caso não consiga enviar o zip, mude a extensão do ficheiro para .xip e envie

Tenha em atenção que todos os componentes devem estar dentro da pasta do projeto, incluindo as imagens, e que a exportação/importação para outro computador devem permitir o funcionamento normal do projeto. A importação e execução noutra máquina devem ser testadas antes da entrega.

Não devem usar caracteres acentuados no projeto.

Trabalhos que não sejam corretamente entregues poderão mesmo 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 leitura do ficheiro não estiver implementada é considerado que o trabalho não está utilizável;
- não usem herança (por exemplo no comportamento e características dos diferentes carros) ou coleções (por exemplo para guardar os carros);
- concentrem o comportamento todo do programa num método, numa classe ou não demonstrem que sabem usar corretamente classes e métodos para modularizar o código.

Serão classificados com C, os trabalhos que:
- Permitirem simular duas ruas que se cruzam com pelo menos dois tipos diferentes de carros e que os impedem de chocar travando um abruptamente se o cruzamento estiver ocupado.
- usem corretamente classes, protecção de atributos (private), herança e coleções;
- em que o comportamento do programa não está maioritariamente concentrado num ou dois métodos / classes.

Serão classificados com B, os trabalhos com uma estrutura interna razoável, nomeadamente:
- classes desenhadas de acordo com as boas práticas;
    - utilização das coleções adequadas para cada situação (listas, filas prioritárias, etc.);
    - boa proteção dos atributos de cada classe (em geral private);
    - métodos de tamanho razoável (muitos com menos de 10 linhas, muito raramente com mais de 50 linhas) e com um único objetivo;
- que permitam, em geral, correr o programa até ao fim sem situações anómalas
- devem demonstrar saber usar adequadamente exceções para tratar erros de programação e situações de exceção;
- e devem apresentar um relatório sucinto cuja única componente obrigatória é o desenho em UML, apenas com os nomes das principais classes e suas relações.

Serão classificados com A os trabalhos que além de todas as características anteriores a maioria das seguintes:
- Tenham uma estrutura interna bem construída;
- Tenham todas as funcionalidades descritas no enunciado, incluindo mudanças de faixa para outras no mesmo sentido sem acidentes e diferentes comportamentos dos condutores;
- Permitam parar e recomeçar, gravando a posição de todos os carros;
- demonstrem que sabem usar JUnits e Javadoc numa pequena classe.

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.

Referências

[1] Imagens usadas no projeto foram encontradas em freepik.com