Programação Concorrente Orientada por Objetos
Catch the Thief

Joel Pinheiro, 65151
joelpinheiro@ua.pt

3 de Janeiro de 2016

Resumo

Este documento descreve, de uma forma breve, o trabalho final de *Programação Concorrente Orientada por Objetos* do aluno Joel Pinheiro. Este relatório contem um pequeno tutorial de como usar a aplicação, decisões relativamente a concorrência, um diagrama com as principais classes e, por fim, alguns problemas e soluções encontrados.

1 Apresentação do tema

O projeto, com o nome *Catch the Thief*, consiste na simulação de um ambiente onde existem transeuntes, polícias e ladrões. Os ladrões são entidades que começam no seu esconderijo e que saem dele para procurar objectos. Esses objectos são trazidos para o seu esconderijo se, no caminho para o esconderijo, os ladrões não forem apanhados. Os transeuntes são pessoas que circulam na cidade e, nesta simulação, a sua tarefa é no caso de se cruzarem com um ladrão informarem imediatamente a polícia da última posição onde encontrou o ladrão. Esta informação é passada por telefone. O papel dos polícias é estar na esquadra preparados para o caso de haver um roubo. No caso disso acontecer, os polícias saem da sua esquadra e procuram o ladrão. Se o conseguirem encontrar antes de ele chegar ao esconderijo dos ladrões, este é trazido para a esquadra.

2 Entidades

2.1 Entidades ativas

Neste projeto existem as seguintes entidades ativas:

- Transeunte: pessoa que passeia pelo mapa e que, no caso de se cruzar com um ladrão, informa a polícia por telefone (na esquadra);
- Ladrão: indivíduo que sai do seu esconderijo à procura do objecto a roubar. Assim que o encontra, volta ao esconderijo para largar o objecto roubado;
- Polícia: os polícias começam na esquadra e ficam lá preparados para sair até que sejam informados (por telefone) que houve um roubo. Assim que

sejam informados estes saem do seu local à procura do ladrão. Caso o encontrem, trazem-no para a esquadra para ser preso.

2.2 Entidades passivas

Neste projeto existem as seguintes entidades passivas:

- Objecto a ser roubado: o objecto a ser roubado estará num sítio aleatório.

2.3 Regiões partilhadas

Neste projeto existem as seguintes regiões partilhadas:

- Board: O mapa da cidade onde todas as entidades activas se movimentam é uma região partilhada;
- GPSMonitor: O GPS é uma região partilhada entre todas as entidades. Esta classe é responsável por dar as direcções de um dado lugar x para outro y. Exemplo, polícias vão para lugar onde o transeunte viu pela última vez o ladrão. Ou, caminho de volta do ladrão para o esconderijo depois de ter encontrado o objecto.
- InformationCentralMonitor: Centro de informações sobre as posições actuais das entidades activas relevantes no mapa e interações entre elas.

3 Guia de Utilização

Para correr a simulação deve navegar até /CatchThief/build/classes e executar o comando java -ea catchthief.Main. Vários parâmetros são pedidos como o número de entidades activas que vão haver na simulação, mapa da simulação (existem dois: labyrinth, city map) e os simbolos.

```
Usage:
java -ea catchthief.Main
[NUMBER OF COPS]
[NUMBER OF PASSERBIES]
[NUMBER OF THIEFS]
[MAP FILE]
[PRISON SYMBOL]
[HIDINGPLACE SYMBOL]
[PASSERBY SYMBOL]
[OBJECT TO STEAL SYMBOL]

EXAMPLE:
java -ea catchthief.Main 2 2 1 ./board/labyrinth.txt 100 P H T $
```

Os mapas entregues são o labyrinth.txt (Figura 1) e o citymap.txt (Figura 2).

Podem ser adicionados outros.

Quando a aplicação é executada aparece a janela da simulação (Figura 1). Nesta janela será mostrado o mapa previamente configurado pelo parâmentro de execução. De seguida, sairão tantos transeuntes, do símbolo definido, quantos os estipulados pela sua configuração. Os transeuntes caminharão pelo mapa desde o início da simulação até ao final.

Após todos os transeuntes sairem de sua casa para o mapa, o ladrão vai sair do esconderijo para procurar o objecto a roubar. De seguida, o ladrão vai procurar pelo mapa até que encontre o objecto a roubar. Caso encontre o objecto a roubar, este volta para o seu esconderijo pedindo ao GPSMonitor o caminho de volta mais rápido para que este não tenha que encontrar o caminho de volta aleatóriamente.

Caso algum transeunte se cruze com o ladrão, este notifica a esquadra e todos os polícias irão sair para chegar ao sítio onde foi visto o ladrão pela última vez. Se os polícias se cruzarem com o ladrão, polícias e ladrões vão para a esquadra permanecendo apenas os transeuntes no mapa.


Catch The Thief Simulator
P
H
T
$
1. ~/Documents/GitHub/CatchThief/build/classes (java)
Usage: [NUMBER_OF_COPS] [NUMBER_OF_PASSERBIES] [NUMBER_OF_THIEFS] [MAP_FILE] [PRISON_SYMBOL] [HIDINGPLACE_SYMBOL
] [PASSERBY_SYMBOL] [OBJECT_TO_STEAL_SYMBOL]
EXAMPLE:
java -ea catchthief.Main 2 2 1 ./board/labyrinth.txt 100 P H T $
joelpinheiro at joelpinheiros-MacBook-Pro in ~/Documents/GitHub/CatchThief/build/classes on master [!]
$ java -ea catchthief.Main 20 20 1 ./board/labyrinth.txt P H T $
Simulation output:
PasserBy found Thief!
PasserBy found Thief!
PasserBy found Thief!
PasserBy found Thief!
PasserBy found Thief!
PasserBy found Thief!
PasserBy found Thief!
PasserBy found Thief!
PasserBy found Thief!

Vídeo de funcionamento:

https://www.youtube.com/watch?v=bFvfBsKo904

4 Estrutura e Decisões de concorrência

O projecto está separado por dois conjuntos. As entidades activas e os objectos partilhados. A classe main instancia o maze e este é passado a cada uma das entidades activas (singleton): ladrões, polícias e transeuntes.

O ciclo de vida dos transeuntes é passear aleatóriamente e infinitamente pelo mapa. Sempre que estes viajam no mapa são guardadas as suas posições para evitar ciclos, indo este para as posições ainda não visitadas ou para as posições visita. Caso este encontre o um ladrão notifica os polícias (que se encontram em Wait) e estes saem da esquadra para ir até ao ponto de onde foi encontrado pela última vez o ladrão.

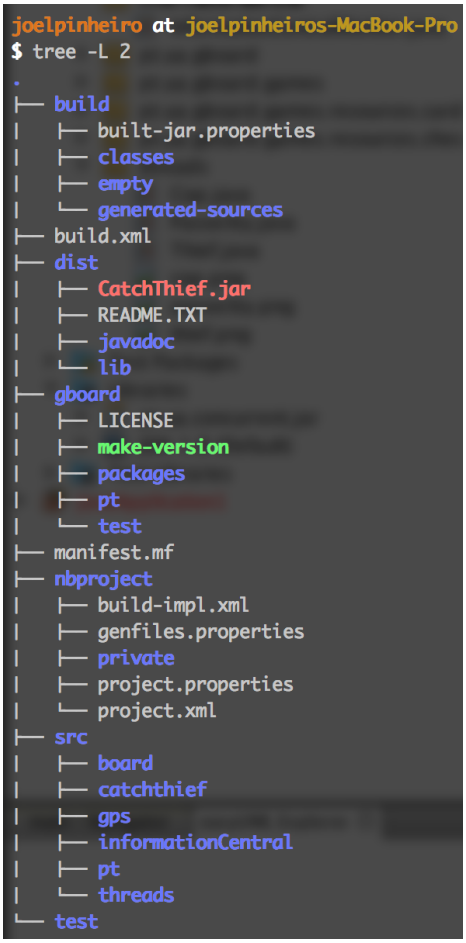
O ladrão, inicialmente, tem um ciclo de vida parecido ao dos transeuntes, este passeia pelo mapa para encontrar o objecto a roubar. Quando o encontra, volta para o seu esconderijo com o objecto usando o GPSMonitor para voltar até ao esconderijo pelo caminho mais rápido. Se entretanto se cruzar com um polícia, ladrão e polícias vão para a prisão utilizando o GPSMonitor.

O polícia começa a simulação permanecendo na prisão (wait) até que seja notificado. Quando for, este usando o GPSMonitor vai para o sítio onde foi visto o ladrão. Depois de chegar ao ponto onde foi visto o ladrão, o polícia vai procurar aleatóriamente pelo ladrão. Se se cruzar com ele, polícias e ladrões vão para a prisão.

No InformationCentralMonitor estão algumas informações como a posição actual do ladrão, última posição onde foi visto um ladrão pelo transeunte, ladrão encontrado, polícia encontrou o ladrão ou ladrão está na prisão. Todas estas informações são acedidas em exclusão mútua (synchronized).

Outro acesso partilhado está no maze onde cada uma das entidades activas concorrem pelo acesso ao mapa.

A estrutura do projecto é a seguinte:



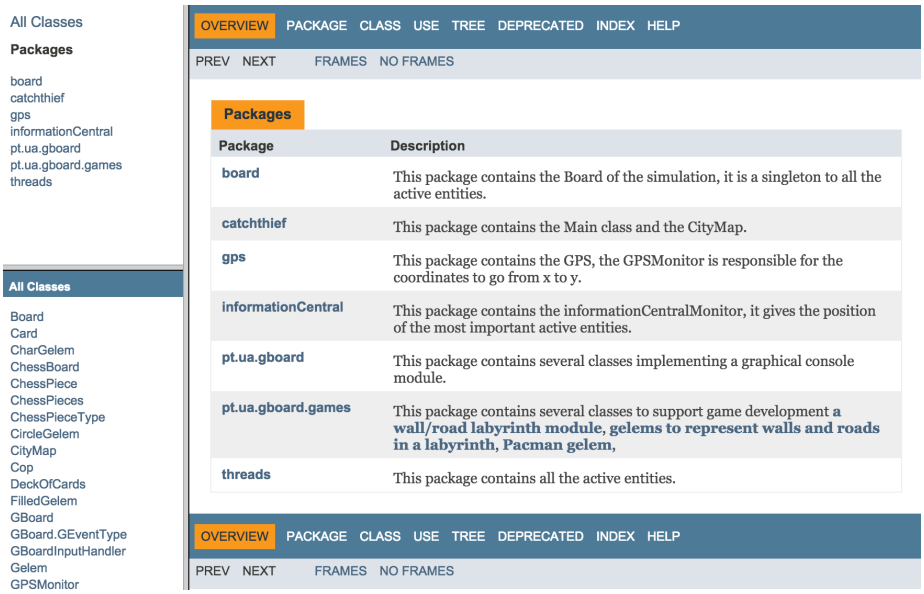
- build: Directoria onde estão as classes compiladas (main: catchthief.Main);
- dist/javadoc: Directoria onde está o javadoc gerado do projecto;
- gboard: Projecto externo;
- src: Directoria onde está o código fonte da simulação.

5 Problemas e Soluções

Houveram vários problemas na sincronização das entidades activas no mapa e interacão entre estes. No entanto, a solução foi prever melhor o acontecimento de deadlocks no ciclo de vida de cada entidade.

Outro problema foi o de visualmente as entidades activas parecerem cruzar-se mas, como estas se movimentam tão rápido, nem sempre isso acontece.

6 Javadoc



Todo o código desenvolvido tem para além de comentários gerais para uma maior percepção do código, os comentários javadoc.

7 Possibilidade de Extenção Futura

No futuro esta simulação pode ser melhorada contendo um ambiente ainda mais real onde existam gangues de ladrões, vários tipos de polícias ou vários tipos de ladrões.