Universidade de Aveiro

Arquiteturas de Software

Trabalho Prático I

Relatório

Autores:
84735 Pedro Ferreira
84746 Rafael Teixeira

Professor:
Óscar Pereira

23 de Março de 2020

Conteúdo

Lista de Tabelas

Lista de Figuras

1 Introdução

Inserido no plano curricular da disciplina de Arquitetura de Software, do curso de Mestrado em Engenharia Informática, da Universidade de Aveiro e leccionada pelo professor Óscar Pereira, este relatório é proveniente da execução do primeiro projeto da cadeira.

O principal objetivo deste projeto é compreender a formulação arquitetural de uma solução, onde a concorrência e a comunicação via sockets são aspetos fundamentais. Posto isto, este relatório tem como propósito expor a solução criada com base nos requisitos estabelecidos.

O relatório está organizado do seguinte modo: na Secção 2 estão definidos os objetivos para o projeto, na Secção 3 encontra-se uma explicação da comunicação realizada entre os dois sistemas da solução, o Control Center e a Farm Infrastructure, na Secção 4 é realizada uma análise da arquitetura da Farm Infrastructure e da utilização de ReentrantLock, respetivas condições e de métodos Synchronized nos diversos monitores e, por fim, na Secção 5 está explicito a contribuição de cada membro na elaboração do projeto.

2 Objetivos

Como forma de orientação, os seguintes objetivos foram definidos para a concretização deste estudo:

- Compreender a diferença entre o ReentrantLock e Synchronized;
- Disponibilizar Sockets de Comunicação;
- Criar Threads para o tratamento de pedidos;
- Compreender as implicações do uso de condições e de concorrência;
- Criar uma solução robusta e escalável.

3 Comunicação entre Control Center e Farm Infrastructure

Esta secção tem como propósito definir e explicar todos os componentes que constroem e permitem a comunicação entre o Control Center e a Farm Infrastructure.

3.1 Servidores

Para estabelecer um canal de comunicação entre as duas principais entidades, o Control Center e a Farm Infrastructure, é imperativo ter um servidor e um cliente. Assim sendo, ambas as entidades implementam um servidor e um cliente.

Posto isto, desenvolvemos a classe ServerCom, comum a ambas as entidades, que é responsável pela instanciação de um servidor. Esta classe tem como atributos um ServerSocket, um Socket, a porta do servidor e duas streams de comunicação, uma de entrada (ObjectInputStream) e outra de saída (ObjectOutputStream).

O ServerSocket existe para que o servidor fique à escuta na porta definida, onde permanece à espera que um cliente comunique com ele. Assim que um cliente comunique com o servidor, este aceita a ligação e cria um Socket de comunicação, assim como as streams de entrada e saída de mensagens, de forma a conseguir ler as mensagens enviadas pelo cliente e responder às mesmas.

As mensagens trocadas entre os clientes e os servidores nos canais de comunicação são do tipo Object, mas, posteriormente, são convertidas para a classe Message. Explicamos a implementação da classe Message na Subsecção 3.3.

3.1.1 Threads

Para cumprir com o objetivo de ter uma solução escalável, a implementação do nosso servidor faz uso de threads para responder aos clientes. Ou seja, sempre que um cliente envia uma mensagem para o servidor, este aceita a ligação e lança uma thread responsável por ler e processar a mensagem recebida, executar os métodos internos apropriados ao tipo da mensagem, e enviar uma resposta ao cliente.

Desta forma, o servidor encontra-se sempre disponível para aceitar novos pedidos de clientes, o que diminuí, substancialmente, o tempo de espera por parte do cliente.

Assim sendo, as threads lançadas para ajudar o servidor são do tipo TFICom ou TCCCom, estando dependente da entidade que as lança.

3.2 Clientes

Tal como referido anteriormente, ambas as entidades são clientes, e por isso criámos a classe ClientCom que, novamente comum a ambas as entidades, implementa um cliente. Esta conta como atributos um Socket, o endereço e porta do servidor ao qual o cliente se vai ligar e as duas streams de comunicação.

Uma vez que é o cliente quem toma a "iniciativa"de falar com o servidor, este não necessita de ter um ServerSocket, e por isso apenas tem um Socket para a comunicação com o servidor.

O propósito do Socket, assim como das duas streams de comunicação, é igual ao do servidor apenas com a lógica invertida, ou seja, agora a stream de entrada são as mensagens recebidas do servidor e a stream de saída são as mensagens enviadas ao servidor.

3.3 Mensagem

Uma vez que existe a necessidade de trocar mais do que um tipo de informação entre o Control Center e a Farm Infrastructure, decidimos implementar a classe Message, que tem como objetivo ser a estrutura das mensagens trocadas entre estas entidades. Assim sendo, esta classe possuí como atributos um corpo, um tipo, o número de espigas de milho, o número total de farmers, o número máximo de passos e o timeout.

O corpo da mensagem serve apenas como forma de contexto, e para atualizar o estado da Farm Infrastructure na interface gráfica do Control Center. O corpo da mensagem, para o processamento e validação das mensagens não é importante, porque existe o tipo de mensagem. Isto porque, em vez de validarmos o texto de

mensagens optámos por validar o tipo da mensagem, que é mais prático, direto e eficiente. Assim sendo, o tipo da mensagem está diretamente relacionado com o estado da simulação em curso na Farm Infrastructure.

Por fim, o número total de espigas de milho que cada farmer tem de colher, o número total de farmers que procede com a colheita, o número máximo de passos que cada farmer pode dar e o timeout associado ao tempo que um farmer demora a avançar no path e a colher uma espiga de milho, são variáveis que fazem parte da configuração da simulação, e são apenas definidos quando é enviada uma mensagem de preparação do Control Center para a Farm Infrastructure.

3.3.1 Tipo de Mensagem

O tipo da mensagem, tal como anteriormente referido, está diretamente associado ao estado da simulação quando a mensagem fora enviada. Posto isto, o tipo da mensagem é um dos possíveis estados presentes no enumerado HarvestState.

Abaixo estão as descrições e objetivos de cada mensagem, consoante o tipo.

3.3.1.1 WaitToStart

Mensagem enviada da Farm Infrastructure para o Control Center quando todos os farmers estiverem posicionados na storehouse, e serve para permitir o utilizador, do lado da interface gráfica do Control Center, modificar a configuração da simulação e enviá-la para a Farm Infrastructure.

3.3.1.2 Prepare

Mensagem enviada a partir do Control Center e transporta a configuração da simulação a ser usada pela Farm Infrastructure.

3.3.1.3 WaitToWalk

Mensagem enviada pela Farm Infrastructure e avisa o Control Center que os farmers todos estão posicionados na standing area à espera de avançar para o path. Posto isto, na interface gráfica do Control Center, apenas os botões start, stop e exit estão disponíveis.

3.3.1.4 Start

Mensagem enviada pelo Control Center e tem como objetivo desbloquear os farmers na standing area, para que possam deslocar-se para a granary, atravessando o path.

3.3.1.5 WaitToCollect

Mensagem enviada pela Farm Infrastructure e serve para avisar o Control Center que os farmers todos já estão na granary. Na interface gráfica do Control Center, apenas os botões collect, stop e exit estão disponíveis.

3.3.1.6 Collect

Mensagem enviada pelo Control Center para dar ordem aos farmers que já podem começar a colher espigas de milho.

3.3.1.7 WaitToReturn

Mensagem enviada pela Farm Infrastructure com o objetivo de alertar que os farmers todos já colheram as espigas de milho todas. Na interface gráfica do Control Center, apenas os botões return, stop e exit estão disponíveis.

3.3.1.8 Return

Mensagem enviada pelo Control Center para mandar os farmers todos regressarem à storehouse.

3.3.1.9 Update

Mensagem enviada pela Farm Infrastructure para atualizar o estado da simulação na interface gráfica do Control Center.

3.3.1.10 Stop

Mensagem enviada pelo Control Center para interromper a simulação em cuso e, consequentemente, o estado atual dos farmers, de forma a que estes voltem para o estado inicial na storehouse, e uma nova simulação possa ter lugar.

3.3.1.11 Exit

Mensagem enviada, primeiramente, pelo Control Center para dar ordem de termino à Farm Infrastructure, com o objetivo de desligar todos os serviços instanciados por esta. Posteriormente, mensagem enviada pela Farm Infrastructure para avisar que todos os seus serviços estão desligados e, por isso, o Control Center também pode desligar todos os seus serviços.

Aqui já não é possível realizar uma nova simulação, uma vez que ambas as entidades são desligadas.

3.3.1.12 FarmerTerminated

Mensagem enviada pela Farm Infrastructure para atualizar o estado dos farmers na interface gráfica do Control Center.

3.3.1.13 Error

Mensagem enviada pela Farm Infrastructure ou pelo Control Center sempre que o tipo da mensagem recebida não é reconhecido.

3.3.1.14 Ok

Mensagem enviada pela Farm Infrastructure ou pelo Control Center sempre que o tipo da mensagem recebida é reconhecido.

4 Farm Infrastructure

Nesta secção analisamos como os diferentes componentes da quinta interagem entre si, assim como as decisões de implementação tomadas no uso de métodos Synchronized e ReentrantLocks.

4.1 Interação entre Componentes

Antes de analisarmos a interação entre componentes é importante sabermos quais os componentes que compõem a infraestrutura da quinta.

A quinta é composta por 4 monitores, a Storehouse, a Standing Area, o Path e a Granary, sendo que cada um deles controla o acesso a uma zona da quinta. Contamos ainda com um FIController, que é usado como interface de comunicação entre os monitores, a interface gráfica da Farm Infrastructure e o Control Center, um CCStub que encapsula o envio de mensagens do FIController para o servidor do Control Center e um TCCCom que recebe mensagens do Control Center e processa-as.

Tendo em conta estes componentes, o diagrama, observável na Figura 1, representa a interação num processo de execução normal, ou seja, sem ocorrer qualquer tipo de paragem voluntária, ou seja, um stop ou um exit. Este diagrama não apresenta também a comunicação realizada com o Control Center, uma vez que esta é analisada, anteriormente, na Secção 3.

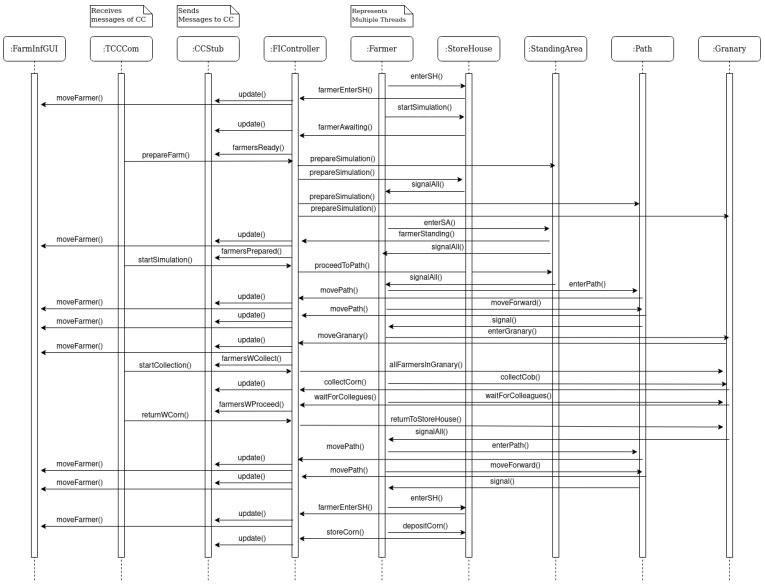

Figura 1: Diagrama de Interação da Farm Infrastructure.

4.2 Monitores

Nesta subsecção analisamos a implementação dos diversos monitores, em específico, a utilização de ReentrantLocks e respetivas condições, e do método Synchronized.

4.2.1 StoreHouse

Na storehouse implementamos tanto ReentrantLocks como Synchronized. O ReentrantLock contem uma condição que é usada para bloquear os farmers, enquanto a Farm Infrastructure espera o sinal "Prepare Farm"vindo do Control Center. Quanto ao Synchronized, este é usado para controlar a concorrência na obtenção de uma posição na storehouse e para controlar o depósito de espigas de milho, para que apenas um farmer deposite uma espiga de cada vez.

4.2.2 Standing Area

Na standing area usamos apenas o ReentrantLock com uma condição, uma vez que esta área serve apenas para que os farmers esperem para avançar para o path. A condição usada bloqueia os farmers até que o Control Center envie uma mensagem do tipo Start, pois todos os farmers à espera são libertados.

4.2.3 Path

No path recorremos ao ReentrantLock com uma condição que guarda a ordem de entrada das threads. Aqui é imprescindível preservar a ordem de chegada dos farmers, pois é necessário que, no path, a ordem de avanço dos farmers seja a mesma que a ordem de chegada.

4.2.4 Granary

No granary usamos tanto o ReentrantLock como o synchronized. O ReentrantLock é usado para que os farmers esperem pela ordem do Control Center para começar a apanhar as espigas de milho e, para que quando acabem de as colher, esperem pela ordem de retorno para a storehouse, vinda também do Control Center.

O Synchronized é usado na colheita das espigas de milho, de forma a garantir que apenas um farmer apanha uma espiga de milho de cada vez.

5 Distribuição do Trabalho

Nesta secção é apresentada a distribuição da carga de trabalho realizada por cada estudante, sendo esta especificada abaixo em termos de pacotes e, quando necessário, classes.
Trabalho realizado por Pedro Ferreira:

- Control Center;
- Path;
- Granary;
- Communication.

Trabalho realizado por Rafael Teixeira:

- Farm Infrastructure;
- StoreHouse;
- Standing Area;
- Communication.