UNIVERSIDADE NOVA DE LISBOA
FCT - Departamento de Engenharia Electrotécnica
Secção de Robótica e Manufatura Integrada

Integração de Sistemas

2020 / 2021

Trabalho 1

Programação de Sockets e XML

Duração: 4 aulas acompanhadas por docente

Entrega: 10 de abril de 2021

1. INTRODUÇÃO

É frequente em integração de sistemas a interligação de aplicações escritas em linguagens distintas. Um dos problemas tradicionais é a integração de bibliotecas já existentes na aplicação que se está a desenvolver.

Assim, neste trabalho estudar-se-ão técnicas de integração de outras diferentes aplicações e tecnologias utilizando Sockets, sendo pedido que duas aplicações (Uma codificada em C# e outra em Java) troquem informação utilizando Sockets.

2. Apresentação do Problema

Quinta de integração de Sistemas

Neste trabalho é pedido que se modele e implemente um ambiente simulado que deverá representar o comportamento de dois tipos de animais diferentes, num ambiente também ele replicando um ambiente real. O ambiente é criado por duas diferentes entidades com responsabilidades diferentes, responsáveis por entidades que interagem com o ambiente simulado.

Tabela 1- Agentes e as suas responsabilidades

Entidade	**Responsabilidade**
Cow	Esta entidade é responsável por abstrair uma vaca presente no ambiente simulado. Esta entidade deve garantir que a entidade se desloca de forma a procurar comida (*Grass*) e fugir dos lobos (*Wolfs*) de forma a garantir a sobrevivência.
Wolf	Esta entidade é responsável por abstrair um lobo presente no ambiente simulado. Este deve garantir que a entidade se desloca de forma a procurar comida que neste caso são as vacas (*Cows*).

O objetivo da infraestrutura é a dinâmica adaptação do sistema com base nas decisões de cada uma das entidades.

O ambiente simulado é já disponibilizado pelo corpo docente, e todas as vezes que uma entidade precisa de se movimentar, o ambiente simulado invoca um serviço que responde com a movimentação que a Vaca ou o Lobo deve fazer.

3. Troca de Mensagens

A estrutura que permite ao ambiente simulado enviar o estado do ambiente e receber as decisões das entidades do mesmo é fornecido pelo corpo docente, sobre a forma de *schema*, num ficheiro "*.xsd"*. A estrutura TMyPlace tem uma lista de TPlace's.

Cada lista TMyPlace envia uma lista com 9 elementos, estes novos elementos representam o estado do sítio onde se encontra o elemento e as posições envolventes, da seguinte forma:

Figure 1 - Representação do vector que é trocado entre o EnvironmentAgent e os restantes

- Grass, do tipo int:
  - Este valor pode ir de 0 (valor mínimo) até 3.
- Wolf, do tipo boolean:
  - Indica se nesta posição se encontra um lobo.
- Cow, do tipo boolean:
  - Indica se nesta posição se encontra uma vaca.
- Obstacle, do tipo boolean:
  - Indica se nesta posição se encontra um obstáculo.
- Entity, do tipo String:
  - Indica o nome da entidade (Cow ou Wolf) que se encontra nesta posição (*null* caso esteja vazia).
- Position, do tipo TPosition:

- Contem as coordenadas desta posição (xx e yy do tipo int).

Este *schema* será também utilizado para serializar o objeto para uma String e assim poder ser trocado quando os serviços forem invocados, nas diversas mensagens.

4. Implementação

Na implementação pedida será utilizado o seguinte material:

- Linguagem JAVA no IDE Netbeans:
  - JDK: https://www.oracle.com/pt/java/technologies/javase-jdk15-downloads.html
  - Netbeans: https://netbeans.apache.org/download/nb123/nb123.html
- Linguagem C# no IDE Visual Studio;
- JAXB: utilizado para criar as classes TMyPosition, TPosition e TPosition, através do *schema* e serializar objetos destes tipos;
- Windows 10 recomendado;
- Código fornecido pelos docentes da cadeira.

5. Planeamento das Aulas

Aula 1 – Explorar o projeto fornecido e modelar a lógica dos lobos e vacas

NOTA: Copiar a pasta *Images* que está na pasta *src* do projeto para c:/

Aula 2 – Serialização e deserialização

1. Implementar os métodos de serialização e de deserialização que estão no package *Common*, no ficheiro *MessageManagement*;

```
public static String createPlaceStateContent(TMyPlace myPlace);

public static TMyPlace retrievePlaceStateObject(String content).
```

Aula 3 e 4 – Implementação dos “servidores socket” que decidem qual o movimento de uma vaca ou lobo

1. Utilizar o projeto existente (em Java) para implementar a primeira versão dos servidores socket (Java);

2. Criar os servidores sockets (um para os lobos e outro para as vacas) necessários para movimentar a Vaca ou Lobo;

3. Deserializar o conteúdo que vem em String para o objeto TMyPlace;

4. Implementar a lógica do movimento;

5. Serializar a resposta para String;

6. Criar um projeto em C# para implementar um dos servidores anteriores em C#.

6. Avaliação

A avaliação do trabalho tem a seguinte ponderação:

- Correta implementação e demonstração de funcionamento do trabalho previsto para as aulas 2, 3 e 4 tudo na mesma tecnologia (JAVA):
  - 16 valores
- Correta implementação e demonstração de funcionamento do trabalho previsto para as aulas 2, 3 e 4 utilizando tecnologias diferentes (JAVA e C#):
  - 18 valores
- Correta implementação e demonstração de funcionamento do trabalho previsto para as aulas 2, 3 e 4 utilizando tecnologias diferentes (JAVA e C#) e funcionalidades extra:
  - 20 valores

Docentes

André Rocha andre.rocha@uninova.pt

Ricardo Peres ricardo.peres@uninova.pt

José Barata jab@uninova.pt


THE COW THAT JUMPED OVER THE MOON FAILS TO CONSIDER RE-ENTRY
MOO?
10-24