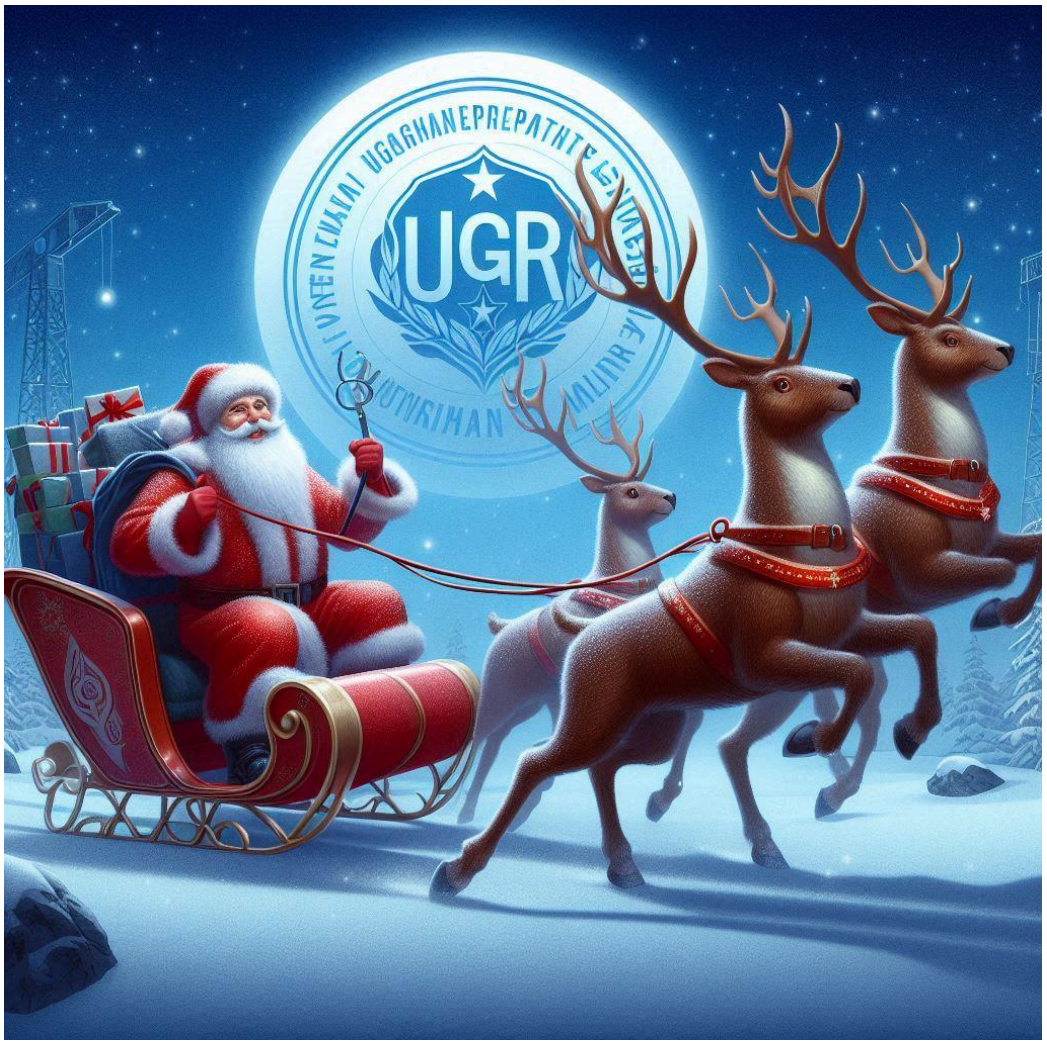
MEMORIA PRÁCTICA 3



Integrantes:
Ángela María Garrido Ruiz
Adrián Romero Vilchez
Álvaro Ruiz Luzón
Francisco de Asís Rivas Fernández

Grupo: A3

Asignatura: DBA

ÍNDICE

1. INTRODUCCIÓN
2. ¿CÓMO INTERACCIONAN EL ELFO, SANTA, EL AGENTE Y RUDOLF?
3. INTERFAZ GRÁFICA
4. CLASE AGENTE
5. CLASE ELFO
6. CLASE SANTA
7. CLASE RENO
8. CONCLUSIÓN

1. INTRODUCCIÓN

En esta práctica hemos trabajado en el desarrollo de un sistema de comunicación y colaboración entre agentes autónomos en un entorno simulado.

Se nos planteó el reto de diseñar un conjunto de agentes capaces de coordinarse para rescatar a los renos perdidos tras una ventisca y asegurar su retorno al establo antes de Navidad.

Nuestro objetivo principal ha sido implementar un protocolo de comunicación eficiente, basado en los estándares FIPA, que permita a los agentes comunicarse a pesar de las barreras lingüísticas entre Santa Claus, que se comunica en Fines, y el agente que se comunica en Generación Z. Para ello, hemos diseñado un agente llamado Elfo que actúa como intermediario lingüístico.

Nuestro solución ha requerido de la implementación de las siguientes clases:

- Clase Interfaz Gráfica: sigue la base de la creada para la Práctica 2 pero hemos hecho modificaciones que se exponen en el punto 3.
- Clases para los agentes: contaremos con 4 agentes Santa, Elfo, Rudolf y Agente
- Resto de clases que se usaron en la práctica anterior (Mundo, Entorno…)

Antes de ponernos a implementar hemos pensado el diagrama de secuencia para establecer cómo se harán las comunicaciones. En el siguiente punto comentamos cómo será el flujo del intercambio de mensajes entre nuestros agentes.

2. ¿CÓMO INTERACCIONAN EL ELFO, SANTA, EL AGENTE Y RUDOLF?

La interacción de Santa, Elfo y Agente será:

- El agente envía el mensaje que quiere mandar a Santa al Elfo con el formato de la Generación Z
- El Elfo traduce ese mensaje a Fines y reenvía el mensaje traducido al agente
- El agente recibe el mensaje traducido y se lo envía a Santa
- Santa recibe el mensaje y le manda la respuesta al Elfo, para que lo pase de Fines a Generación Z
- El Elfo lo traduce al idioma de la Generación Z y se lo manda a Santa
- Santa reenvía el mensaje al agente

De esta manera el Elfo solo funciona como un traductor que permite la comunicación entre Santa y el agente, pero no modifica el contenido de los mensajes ni los envía directamente a sus destinatarios.

La interacción de Rudolf y el Agente será más sencilla ya que no necesitamos traducción de mensajes:

- El Agente envía el código proporcionado por Santa a Rudolf junto con la petición de las coordenadas

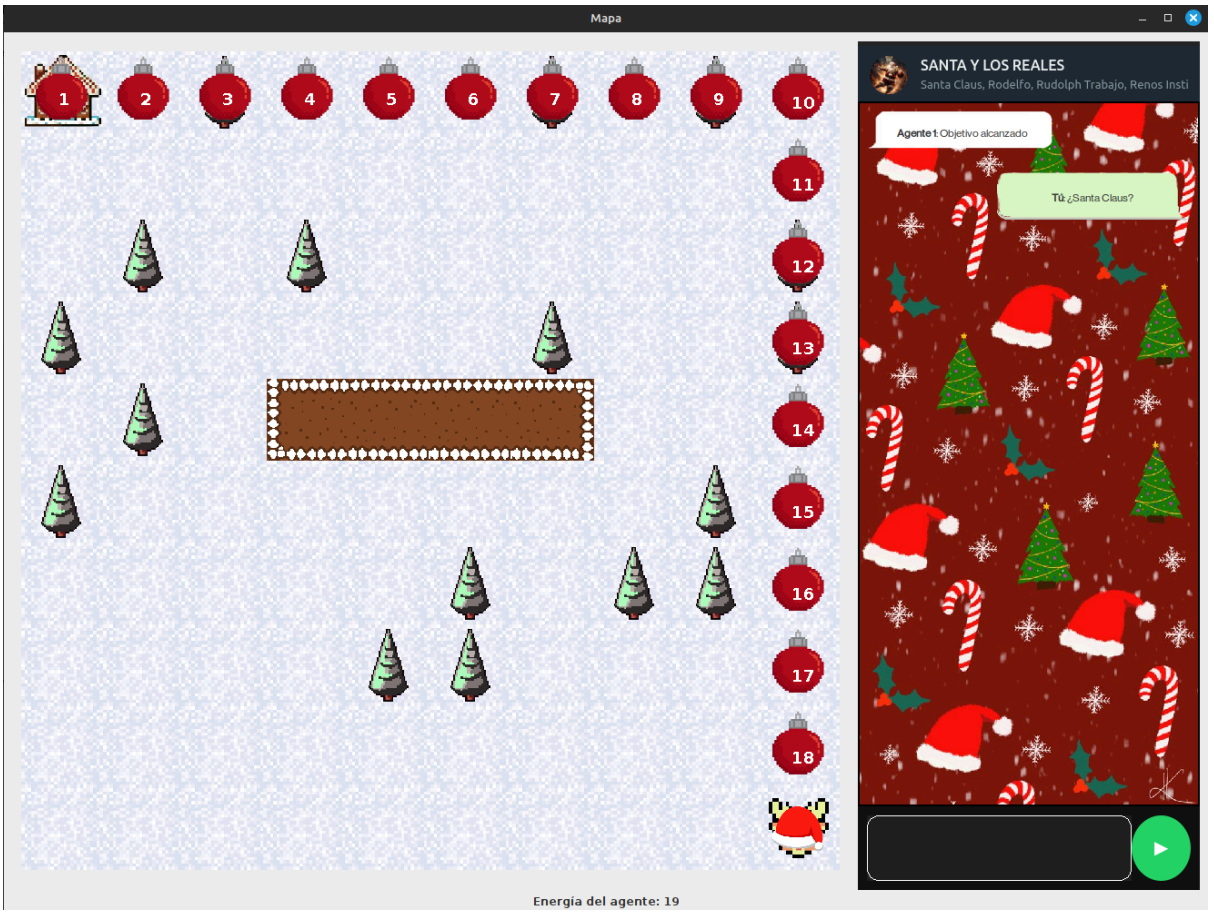
- Si el código es correcto Rudolf aceptará la petición y nos mandará las coordenadas de un de los renos perdidos
- El Agente se dirige hacia esas coordenadas
- El Agente estará pidiendo coordenadas hasta que Rudolf le informe de que ha terminado

Cuando Rudolf informe que ha terminado volveremos a tener comunicaciones entre Santa, el Elfo y el Agente (como se ha descrito anteriormente) ya que el Agente le pide a Santa su ubicación para encontrarse con él y confirmar que todo ha ido bien.

Todas estas interacciones las encontramos en el diagrama de secuencia que se adjunta en la entrega.

3. INTERFAZ GRÁFICA

En esta práctica hemos decidido dotar a nuestra interfaz gráfica de una temática navideña. Visualizandose de la siguiente manera:



Ejemplo de interfaz gráfica (Figura 1)

Cada celda con un cero se representa con un fondo de nieve; hay un 80% de fondo de nieve y un 20% aleatorio en el cual se dibujan unos pequeños pinos. Esto simplemente forma parte del fondo y no es un obstáculo.

Los obstáculos se representan como una especie de galleta de jengibre en el que se detectan si es una esquina un fin o un muro normal. Los pasos ahora se representan con bolas de Navidad rojas.

Como en esta práctica lo que se intenta buscar es la implementación de la comunicación entre agentes, hemos decidido crear una especie de chat en el que irán apareciendo los mensajes que se bailarán comunicando los distintos agentes.

Toda esta interfaz está creada con Java Swing. Simplemente son diferentes componentes de Swing que se colocan en el interfaz en la ventana para conseguir un efecto de teléfono móvil o de chat. El fondo de pantalla y las texturas son imágenes que se importan para colocar el texto sobre ellas.

La interfaz gráfica se compone de dos partes la parte del mapa usando un una clase llamada mapPanel y otra clase llamada ChatPanel para la parte del chat.

ChatPanel se compone de tres componentes Swing:

- **chatPanel:** un JPanel para el chat, con fondo de pantalla, cabecera
- **inputField:** un input para el texto . Está personalizado con fondo negro y bordes redondeados.
- **sendButton:** Un botón de enviar que también se presiona cuando pulsamos enter. Está personalizado para que tenga los bordes redondeados tanto como para ser un círculo.

Existe un método **addMessage** que lo que hace es colocar un mensaje que envía un agente. Dependiendo del agente que envía el mensaje, el globo de texto cambia de color y de posición, para reflejar quién está hablando en ese momento. Esta función aplica la fuente al texto y lo incluye dentro del globo de texto correspondiente.

4. CLASE AGENTE

Será el agente encargado de buscar los renos para Santa (si es aceptado para el trabajo), y avanzar hacia este último una vez los haya encontrado. Se comunicará con el reno para obtener las coordenadas de los diferentes renos que debe localizar.

Además de los atributos de la práctica anterior, tenemos otros como codigoSanta, que es el código que debemos pasarle al reno para que sepa que vamos de parte de Santa, step, que nos indicará el paso de flujo de trabajo del agente, y finish, que indica si se ha terminado el comportamiento.

Hemos hecho uso de un switch-case para saber qué acción tiene que realizar en cada momento:

- **Paso 0** - El agente buscador (AB para abreviar) quiere realizar la misión, por lo que le manda el mensaje al Elfo para que se lo traduzca.
- **Paso 1** - AB espera el mensaje del Elfo, y se lo manda a Santa con un QUERY_IF.
- **Paso 2** - Espera la confirmación de Santa.
    - Si la performativa es CONFIRM, quiere decir que Santa nos ha dado la misión, nos ha mandado un código para comunicarnos con el reno y avanzamos al paso 3.
    - Si no lo es, avanzamos al paso 14.
- **Paso 3** - Le enviamos un QUERY_IF al reno con el código de Santa para que nos mande las coordenadas de un reno perdido.
- **Paso 4** - Esperamos una respuesta del reno. Si hemos recibido unas coordenadas no nulas, actualizamos las posiciones del agente y del objetivo y priorizamos las direcciones hacia las que nos moveremos,
- **Paso 5** - Nos dirigiremos hacia el objetivo. Una vez lo encontremos, aumentaremos el contador de renos.
    - Si hemos encontrado todos los renos, avanzamos al paso 6.
    - Si no, volvemos al paso 3.
- **Paso 6** - Le pide al elfo que le traduzca el mensaje de que ha encontrado todos los renos. Y espera un INFORM.
- **Paso 7** - Espera el mensaje del Elfo, y se lo manda a Santa con un REQUEST.
- **Paso 8** - Le mandamos el mensaje de “Donde estás Santa” al elfo para que nos lo traduzca con un INFORM.
- **Paso 9** -Esperamos la traducción del elfo y le mandamos el mensaje a Santa con un REQUEST.
- **Paso 10** - Si Santa nos responde con un INFORM con sus coordenadas, actualizamos la posición del TargetPos con las coordenadas de Santa.
- **Paso 11** - Comenzamos a movernos hacia Santa.
    - Cuando lleguemos, avanzaremos al siguiente paso.
- **Paso 12** - Mandamos un mensaje al elfo con el mensaje “ya he llegado donde está Santa” para que lo traduzca.
- **Paso 13** - Esperamos el mensaje traducido del Elfo y se lo enviamos a Santa con un REQUEST.
- **Por defecto** - si hay algún error se invoca al método doDelete para finalizar el proceso

5. CLASE ELFO

Esta clase se encargará de traducir mensajes, como se ha explicado anteriormente, los que manda el Agente a Fines y los que manda Santa a generación Z. Tiene por tanto una lógica sencilla:

- Recibe el mensaje
- Mira quien ha sido el emisor
- Traduce el contenido del mensaje siguiendo ciertas reglas
- Envía una respuesta, en este caso la respuesta siempre es el mensaje traducido

Hemos creado dos métodos auxiliares para llevar a cabo la traducción de los mensajes:

- translateMesgAgente - nos reemplaza las palabras “Bro” y “En plan” que conforman el inicio y final del mensaje por "Rakas Joulupukki" y “Kiitos”.
- translateMesgSanta - sigue la lógica inversa al método anterior.

Ambas funciones usan métodos de la clase String para llevar a cabo el reemplazo de las palabras clave.

El funcionamiento como tal del Elfo ocurre dentro del método setup, en el hemos creado el comportamiento del Elfo.

- Recibe el mensaje usando el método blockingRecive que bloquea la ejecución del hasta que el Elfo recibe el mensaje, bien de Santa o del Agente. Dicho mensaje lo almacenamos en la variables msg
- Validación del mensaje, aquí se comprueba que el mensaje sea INFORM o CONFIRM.
    - Si es uno de esos tipos, procesamos el mensaje según quien lo envía
        - Si lo envía Santa se traduce el mensaje usando el método translateMesgSanta y envía la respuesta
        - Si lo envía el Agente se usa translateMesgAgente y envía la respuesta
    - Si no es ninguno de esos tipos imprime un mensaje de error

Los mensajes que recibe deben ser de tipo INFORM o CONFIRM, ya que cuando el Agente o Santa le envían un mensaje simple será de tipo INFORM. Sin embargo, cuando tenemos el caso de que Santa acepta o no que el Agente se encargue de recuperar a los renos el tipo de mensaje será CONFIRM ya que se responde a una consulta específica y el Agente está esperando una confirmación a una petición previa.

6. CLASE SANTA

Como sabemos Santa tendrá dos tareas fundamentales: aceptar o rechazar al Agente como el encargado de buscar a sus renos y proporcionar su ubicación al Agente cuando este ha terminado el trabajo si se le ha sido asignado.
Tenemos dos atributos: step (define el paso del flujo de trabajo del agente) y finish (indica si el comportamiento ha terminado)

Tenemos un comportamiento principal en el que se ha desarrollado un switch-case para saber qué acción se tiene que hacer en cada momento:

- **Paso 0** - es donde Santa confirma si confía en el Agente para que encuentre a sus renos. Como sabemos tenemos una probabilidad de un 80% en que acepte y un 20% de que no. Para determinar esa aceptación usamos un número aleatorio.
    - Si es confiable enviamos un mensaje de aceptación al Elfo para que lo traduzca con el código
    - Si no lo es se envía un mensaje al Elfo rechazando la misión
- **Paso 1** - el Elfo manda a Santa el mensaje traducido para que este informe al Agente de su decisión.
    - Si le llega el mensaje, lo reenvía al Agente con un CONFIRM
    - Si no, envía un error
- **Paso 2** - Santa espera el mensaje del agente buscador.

    - Si es una performativa REQUEST, avanza al siguiente paso.
    - Si no, termina la ejecución con un mensaje de error
- **Paso 3** - Santa le manda las coordenadas de su posición al Elfo con un INFORM para que las traduzca y avanza al siguiente paso.
- **Paso 4** - Santa espera un mensaje del Elfo a través de un INFORM con las coordenadas traducidas para poder enviárselas al agente con un CONFIRM.
- **Paso 5** - Es el paso final. Santa espera un mensaje de confirmación del Agente buscador indicando que ha llegado a su posición, y responde al Agente con un CONFIRM.
- **Por defecto** - si hay algún error se invoca al método doDelete para finalizar el proceso

Una vez que tenemos la comunicación entre Santa y el Agente y comprobamos que nos envía el código si somos confiables podemos pasar a implementar la clase Reno.

7. CLASE RENO

Rudolph es el reno de Santa, y será el que nos proporcione las coordenadas de los renos perdidos. Tiene un ArrayList con las posiciones de los renos, y otro ArrayList con las casillas que están libres, para que no pueda poner un reno encima de un muro (Lo que haría imposible para el agente encontrarlo), y una variable con el gui y otra con el entorno, aparte de un controlador para saber si hemos encontrado todos los renos.

Primero, inicializamos el gui, el entorno y las casillas libres. Luego, en un bucle, si el array de casillas libres no está vacío, seleccionamos una posición al azar y posicionamos un reno. También tenemos la funcionalidad para poner los renos en posiciones específicas.

El reno sigue un RequestCodeBehaviour, donde esperamos un mensaje de nuestro Agente.

- Tras comprobar que el que le ha enviado el mensaje es el agente esperado, manda un mensaje a la interfaz confirmando que ha recibido el mensaje, y luego le responde al agente con una performativa INFORM de las coordenadas de un reno, y actualiza la interfaz gráfica.
- Si el agente que se comunica con el reno no es el esperado, manda una performativa de DISCONFIRM
- Su comportamiento terminará cuando se hayan encontrado los 8 renos.

8. CONCLUSIÓN

Tras la realización de todo lo anterior, logramos implementar una comunicación básica entre diferentes agentes con las performativas vistas en clase.

A lo largo de la práctica nos hemos enfrentado a diferentes situaciones complicadas, sobre el orden de los mensajes que salían por el chat, o cómo actualizar la priorización de las dirección hacia la que tenía que ir el agente una vez encontrado un reno.

Nuestro trabajo cumple con los requisitos planteados en el guión y nos ha permitido familiarizarnos con las performativas de comunicación entre agentes.