Universidad Nacional de Educación a Distancia
E.T.S.I. Informática
Dpto. de Sistemas de Comunicación y Control

PRÁCTICA
DE
Sistemas Distribuidos

(Grado en Ingeniería Informática)

Implementación distribuida del juego batalla naval (hundir la flota) usando Java RMI

Curso 2023 - 2024

INTRODUCCIÓN

Este documento contiene el enunciado de la práctica obligatoria de laboratorio correspondiente a la asignatura de Sistemas Distribuidos del Grado de Ingeniería Informática. Lea detenidamente toda la memoria varias veces con el fin de que sus dudas queden resueltas. Si aun así sigue teniendo alguna duda, por favor consúltela a través del foro de la asignatura en el curso virtual.

El software mínimo necesario para la realización de la práctica es el siguiente:

- Kit de desarrollo de Java 11 JDK. Disponible en el servidor de la empresa Oracle, (https://www.oracle.com/java/technologies/downloads/#java11) actual propietaria de la tecnología. En esa dirección pueden encontrar, además de la documentación oficial de la API del lenguaje, valiosa información como tutoriales, artículos técnicos, etc.
- Entorno de desarrollo IDE. Aunque no es necesario (se pueden editar los ficheros fuentes con un sencillo editor como el Notepad y compilar/ejecutar desde la línea de comandos) pero sí se recomienda su uso para acelerar el desarrollo. Además, los editores poseen potentes depuradores que prestan un excelente servicio a la hora de depurar las aplicaciones. Se puede recurrir a cualquier entorno siempre que éste genere código 100% Java. El equipo docente recomienda:

  - Eclipse (http://www.eclipse.org/downloads/)

Les recordamos que la entrega de la práctica es **obligatoria** para todos los estudiantes. Además, es imprescindible aprobar la práctica para aprobar la asignatura. Le recomendamos la lectura detenida de la guía de curso para aclarar cualquier duda sobre los criterios de evaluación.

Una vez entregada la práctica con todo el material requerido, el equipo docente someterá a los programas que se entreguen a un test para comprobar su correcto funcionamiento. Por tanto, se recomienda probar todo el software antes de enviarlo, a ser posible en otra máquina física o virtual diferente en la que se ha desarrollado.

Si en la convocatoria de febrero supera la práctica obligatoria pero no supera el examen presencial, se le conservará la calificación de la práctica para la convocatoria extraordinaria del curso actual pero **nunca** para cursos posteriores. Análogamente sucederá con la nota del examen presencial.

No se conservan calificaciones de ninguna prueba, ya sea teórica o práctica, para cursos posteriores.

Las prácticas se realizan **individualmente**, no se aceptan grupos de trabajo. **La detección de una práctica copiada** de cualquier fuente incluida Internet, obligará al

equipo docente **a ponerlo en conocimiento del Servicio de Inspección de la UNED (CPRI)** para que proceda a la apertura de expediente académico. Para ello, se utilizarán herramientas para detectar este tipo de fraudes como *Turnitin*. Evidentemente, pueden consultar cualquier tipo de dudas, cuestiones, mañas, etc. a través de los foros en el curso virtual de la asignatura.

Lea detalladamente el resto de la memoria ya que se le indica claramente la forma de realizar la práctica, el material que hay que entregar, y los plazos para hacerlo. Para cualquier aclaración, no dude en ponerse en contacto con los miembros del equipo docente de la asignatura utilizando los medios que se indican en la Guía del curso.

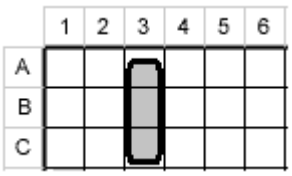
ENUNCIADO

El propósito de la práctica es el desarrollo de un software distribuido que implemente el conocido juego de batalla naval, también conocido como hundir la flota o los barquitos. Para aquellos estudiantes que no conozcan la mecánica del juego, pueden consultar el siguiente enlace.

Así pues, vamos a implementar un juego de tablero para dos jugadores. El tablero tendrá unas dimensiones 10x10. Las filas se notarán con una letra y las columnas con un número, de forma que podemos identificar cualquier casilla del tablero con una dupla. Por ejemplo: C3.

La implementación es muy simple, un jugador se conecta, abre el juego y espera a que se conecte otro jugador que será su contrincante. Una vez conectados los dos, ambos deben elegir en primer lugar donde colocar sus 2 barcos. Para simplificar, los dos barcos serán iguales en longitud (3 casillas) y sólo se podrán poner de forma vertical u horizontal sobre el tablero, nunca en diagonal.

Así pues, el programa cliente de cada jugador preguntará la casilla donde colocar la proa (parte delantera del barco) y su orientación (vertical u horizontal). Por ejemplo, si elige A3 vertical, el barco debe colocarse en las casillas A3, B3 y C3, como podemos observar en la imagen siguiente:



Este procedimiento se repetirá hasta completar la colocación de los 2 barcos del jugador. Nota: Comprobar que ningún barco se sale de las dimensiones del tablero.

Cuando ambos jugadores hayan terminado de poner sus barcos, el sistema, a través de la entidad Servidor, les indicará que empieza el juego mediante un evento. Para simplificar el primer disparo lo hará el jugador que abre el juego (el que se conecta primero), después será el turno del otro jugador que no podrá disparar hasta que lo haya hecho el

primer jugador. Esta mecánica se repite hasta que uno de los jugadores hunda todos los barcos del oponente. No hay posibilidad de empate.

Una vez que el sistema detecte que la partida ha terminado, le comunicara a cada uno de los jugadores su puntuación, la cual se distribuirá de la siguiente manera:

1 punto por cada disparo que impacte en un barco del contrincante.
4 puntos por hundir todos los barcos del contrincante.

Finalizada la partida, los jugadores de nuevo podrán iniciar una nueva partida o unirse a una ya creada por otro jugador y que está en espera de que se conecte un contrincante.

En este sistema actuaran tres tipos de actores cuyas funciones se listan a continuación:

1.- Servidor: La entidad Servidor se encarga de controlar y gestionar las partidas y el proceso de autenticación de los usuarios (jugadores) del sistema. Para ello, hace uso de dos servicios:

- Servicio Autenticación: Se encarga de registrar y de autenticar a los jugadores del sistema. La operación de registro consiste en que los jugadores introducen en el sistema por primera vez sus datos personales (nombre y password), siendo el nombre el identificador único de usuario. Es decir, no pueden existir dos jugadores en el sistema con el mismo nombre. Una vez registrado, el jugador puede acceder al sistema haciendo login utilizando este servicio de Autenticación.
- Servicio Gestor: Este servicio se encarga de gestionar todas las operaciones de los jugadores en relación con el juego y la emisión de eventos. Crea todas las estructuras necesarias de datos para jugar la partida, gestiona la lógica de juego, informa a los usuarios del fin de partida y de sus puntuaciones, etc.

2.- Base de datos: Esta entidad es la encargada de almacenar todos los datos del sistema: Jugadores, tableros de juego,...; Sólo la entidad Servidor puede consumir el servicio que suministra esta entidad y cuya funcionalidad se describe a continuación.

- Servicio Datos: Este servicio hará las funciones de una base de datos que relacione jugadores-partidas-tableros. Es decir, mantendrá la lista de jugadores registrados y/o conectados al sistema, junto con los jugadores que han iniciado juego y se encuentran a la espera de contrincante. También guardará toda la información relativa a la partida: tablero, posición de los barcos, disparos efectuados. Los dos servicios anteriores (Servicio Autenticación y Servicio Gestor) harán uso de este servicio para realizar las operaciones sobre el estado de los jugadores del sistema y sus partidas. Aunque podría usarse un sistema de gestión de bases de datos (SGBD) para implementar este servicio, esto haría muy complejo el desarrollo de la práctica y no atendería a los objetivos de la asignatura. Así pues, **el equipo docente recomienda** para la implementación del servicio las clases *List* y *HashMap* de Java.

3.- Jugadores (Usuarios): Son los actores principales del sistema, los jugadores se enfrentan entre ellos en partidas cuyo objetivo es hundir la flota del contrario. Los jugadores se registran en el sistema a través de la entidad Servidor. Una ver registrados se logean usando también esta entidad Servidor para poder iniciar una partida y esperar

contrincante o unirse a una partida creada por otro jugador y enfrentarse a él. La entidad Jugador debe implementar un servicio para recibir los mensajes (eventos) que le llegan del servidor que son: partida iniciada permanezca a la espera, otro jugador se ha unido a su partida, colocar los barcos en el tablero, comienza el juego, disparo realizado por el jugador contrario (tocado/agua), disparo realizado por ti (tocado/agua), has hundido un barco, te han hundido un barco, has hundido la flota de tu contrincante (has ganado), te han hundido tu flota (has perdido).

- Servicio CallbackJugador. Es el único servicio que arranca la entidad jugador y tiene los métodos necesarios para recibir los mensajes (eventos) que se han descrito anteriormente de forma automática.

Una vez que un jugador ha creado una partida, esta se registra en el servidor con un **identificador único** que puede ser un número consecutivo. El jugador contrincante usará este identificador de partida para unirse a ella.

Operativa

Inicialmente la entidad Base de datos levanta su servicio Datos. Posteriormente, la entidad Servidor levanta sus dos servicios: Autenticación y Gestor.

Por último, los jugadores arrancan su aplicación cliente y se registran/autentican en el sistema mediante el servicio Autenticación del servidor. El jugador puede decidir desloguearse (logout) del sistema y el sistema tiene que registrar ese evento convenientemente.

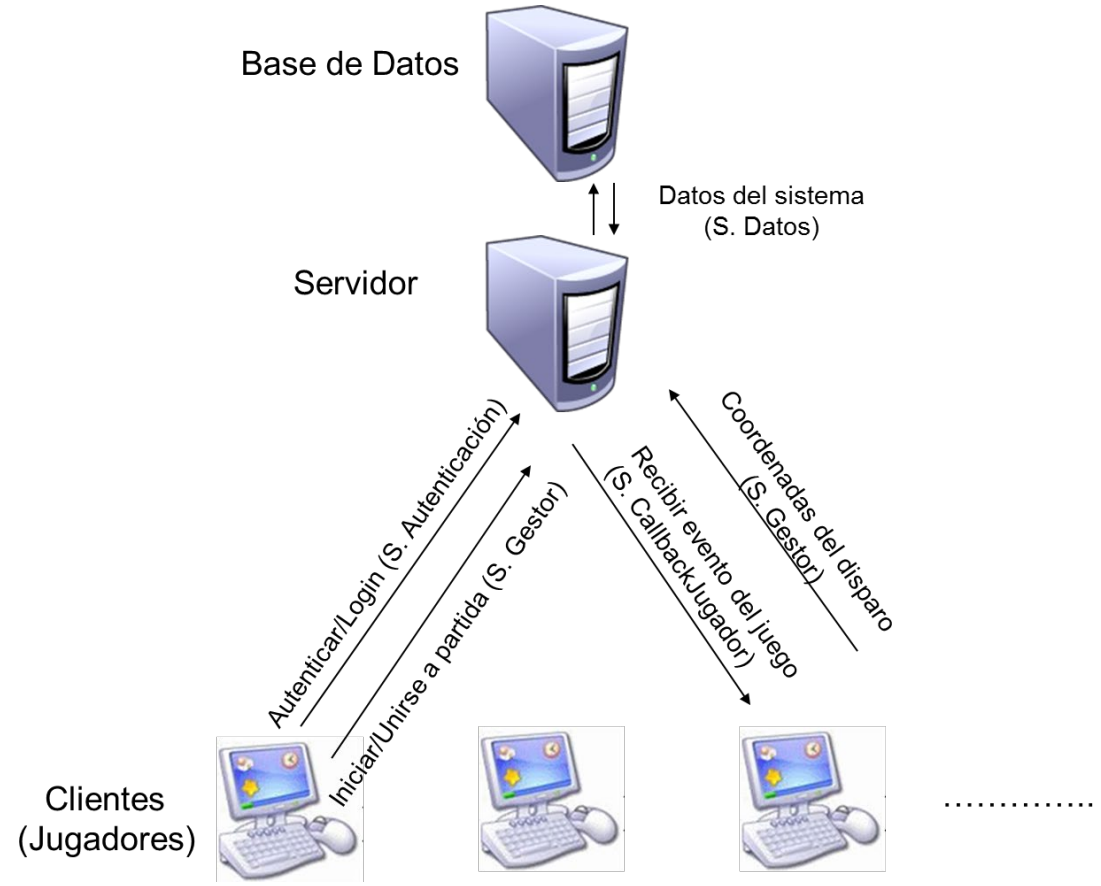
Una vez que el jugador esté logueado en el sistema, podrá realizar las operaciones de iniciar una partida, unirse a una partida creada por otro jugador, consultar sus puntuaciones y, por supuesto, todo lo relativo al juego, colocar sus barcos, disparar contra el enemigo y recibir los disparos del enemigo, cuando el juego se haya iniciado. Estas operaciones son gestionadas por la entidad Servidor.

Cuando un jugador realiza un disparo, éste se procesa por la entidad Servidor, concretamente por su servicio Gestor que una vez procesado envía **directa y automáticamente** el resultado del evento a ambos jugadores. Es decir, les dice si ha habido impactado con uno de los barcos o el proyectil ha caído al agua.

Para simplificar, la operación de unirse a una partida creada por otro jugador no necesita confirmación del jugador que ha creado la partida. Es decir, una vez que se une a la partida, empieza ésta.

Por otro lado, se propone como **tarea opcional** implementar la funcionalidad de que un jugador pueda abandonar el juego una vez comenzada una partida (escribiendo un código en lugar de una coordenada de disparo. Se debe escribir “CAPITULO”. En este caso, se enviará un mensaje al otro jugador de que el usuario se rinde y le concede todos los puntos de la partida.

En la figura siguiente podemos observar de forma esquemática la operativa del sistema:



Interfaz

- El Servidor debe permitir mediante su interfaz de texto las siguientes operaciones:
  1.- Información del Servidor.
  2.- Estado de las partidas que se están jugando en este momento.
  3.- Salir.

- La Base de Datos debe permitir mediante su interfaz de texto las siguientes operaciones:
  1.- Información de la Base de Datos.
  2.- Listar jugadores registrados (Sus puntuaciones).
  3.- Salir.

- Los Clientes (Jugadores) deben permitir mediante su interfaz de texto las siguientes operaciones:
  1.- Información del jugador (consultar puntuación histórica).
  2.- Iniciar una partida.
  3.- Listar partidas iniciadas a la espera de contrincante.
  4.- Unirse a una partida ya iniciada.
  5.- Salir "Logout".

IMPORTANTE: Cuando un jugador arranca la aplicación cliente debe aparecer inicialmente un menú con las siguientes opciones antes del menú anterior. Éste debe permitir el registro de un nuevo jugador en el sistema y/o hacer login:

1.- Registrar un nuevo jugador.
2.- Hacer login.
3.- Salir

Por favor, para facilitar la corrección de las prácticas, aténganse a presentar los menús en formato texto con las mismas opciones y orden que se le han presentado anteriormente.

Por último, como información del Servidor y Base de datos debe aparecer en pantalla el nombre con el que se ha nombrado al/los objeto/s remoto/s que implemente el servicio que corresponda en cada caso (Ver apartado recomendaciones más abajo).

Mecánica

Cuando un jugador inicia una partida, le aparece el mensaje "A la espera de contrincante" y se queda ahí hasta que otro jugador elija enfrentarse a él. Una vez que comienza la partida aparece un mensaje a ambos jugadores:

"Introduzca coordenadas barco 1: "

Y después

"Introduzca coordenadas barco 2: "

Las coordenadas se codifican como YXZ, donde:

Y=fila de la proa (A – J).
X=columna de la proa (1 – 10).
Z=orientación (V=vertical, H=horizontal).

Recuerde que los barcos siempre tienen tamaño 3.

Una vez introducidas estas coordenadas por ambos jugadores, el servicio Gestor le mandará al primer jugador un evento para que inicie la partida realizando su disparo. En la consola del jugador debe aparecer el mensaje:

"Introduzca coordenadas de tiro [YX]: "

Las coordenadas de tiro se codificarán como se ha descrito anteriormente sin orientación, ya que el tiro no la necesita.

Una vez que realice esta acción le aparecerá en la pantalla un mensaje que le diga el efecto de su acción, después del que el servidor haya procesado la acción del disparo. A continuación, se procederá de igual manera con el otro jugador hasta que se acabe la partida.

En la siguiente tabla, se describe la mecánica del juego resumida por turnos:

Paso	Jugador que inicia la partida	Jugador que se une a la partida
1	Inicia la partida y se mantiene a la espera	
2		El jugador se une a la partida
3	Colocar los barcos en el tablero	Colocar los barcos en el tablero
4	Introduce las coordenadas del disparo	
5	El servicio Gestor devuelve la información relacionada con el disparo	El servicio Gestor informa del disparo del otro jugador y sus consecuencias
6		Introduce las coordinadas del disparo
7	El servicio Gestor informa del disparo del otro jugador y sus consecuencias	El servicio Gestor devuelve la información relacionada con el disparo
…		
n	El servidor informa del fin de partida (Has ganado o has perdido)	El servidor informa del fin de partida (Has ganado o has perdido)

Como podéis ver, los clientes (jugadores) deben mantener una espera condicionada a la ocurrencia de un evento. Por tanto, arquitecturalmente debemos tener en cuenta esta cuestión y codificarla adecuadamente en Java. Una solución para ello es utilizar una lista sincronizada de eventos utilizando la capacidad de sincronizar evento de Java con la palabra clave *synchronized.* Como podemos observar en la documentación del fabricante (enlace), esta permite implantar una estrategia de acceso a datos compartidos por dos procesos, previniendo interferencias y errores de consistencia de memoria.

Más adelante en este enunciado podéis encontrar un epígrafe donde se explica este tipo de lista.

Especificaciones Generales

- Se debe utilizar código 100% Java JDK. No se admiten librerías de terceros.

- La sintaxis de llamada desde la línea de comandos tiene que ser:
  - basededatos (para iniciar el programa Base de Datos y su servicio).
  - servidor (para iniciar el programa Servidor y sus servicios).
  - jugador (para iniciar un programa cliente del jugador y su servicio).

- Para cumplir con esta sintaxis de llamada hay que crear ficheros en batería (.bat o .sh) para arrancar las aplicaciones. Se recomienda que dentro de estos ficheros haya sólo una línea: *java -jar nombre_fichero.jar*. No incluir dentro de los ficheros en batería comandos para cambiar el path del sistema operativo, esto no es limpio y no suele funcionar por temas de permisos.

- Por motivos de claridad, cada clase Java debe almacenarse en un fichero *.java* Todos los ficheros deben incluir, mediante comentarios al inicio de los mismos, todos los datos del autor: nombre, apellidos y correo electrónico.

- Antes de empezar a programar nada, le recomendamos que se lea varias veces esta memoria y que se haga un pequeño esquema planteando las relaciones entre cada una de las partes funcionales de la aplicación que se le pide.

- Se recomienda encapsular el código dentro de ficheros *.jar*. Esta es una forma elegante, eficiente y compacta de presentar la aplicación final.

- Pueden tomarse las decisiones de diseño que se consideren oportunas, siempre y cuando vayan en consonancia con el enunciado de la práctica y queden perfectamente comentadas en la memoria de la práctica que tiene que entregar.

- En los foros de la asignatura **no se pueden hacer preguntas del tipo**: No me funciona el programa porque me sale este error/excepción. ¿Podrían indicarme dentro de mi código fuente donde está el error o qué funciona mal?

- La forma correcta de hacer la retrollamada (*Callback*) para esta práctica se explica en la página 187 del libro base de la asignatura (Coulouris *et al.*). También tenéis un desarrollo más amplio en el capítulo 8 "RMI Avanzado" del libro recomendado (Computación Distribuida: Fundamentos y Aplicaciones, Autor M.L. Liu., Ed. Pearson).

CONSIDERACIONES DE DESARROLLO

RMI se estudia en el libro de texto básico, aunque es muy recomendable mirar el contenido de los siguientes enlaces:

-https://docs.oracle.com/javase/8/docs/technotes/guides/rmi/index.html, página Web oficial de RMI, donde se proporciona una documentación muy amplia (en inglés) sobre la utilización de RMI y las clases asociadas.

-http://docs.oracle.com/javase/tutorial/rmi/index.html, tutorial que ayuda a entender y comprender la arquitectura de RMI. Es muy recomendable su lectura.

- Los capítulos sobre Java RMI del Libro de la bibliografía complementaria: ISBN: 978-0201796449, Título: Distributed Computing: Principles and Applications., Autor M.L. Liu., Editorial Addison Wesley Higher Education. (Existe también versión en **castellano**).

- Sesiones 5,6,7 y 8 de las tutorías Intercampus de este curso.

- Vídeo tutorial del alumno Fermín Silva sobre Java RMI en Eclipse. http://www.youtube.com/watch?v=vnWBrCSjb44

De esta manera, se obtienen los conocimientos mínimos necesarios para instalar el entorno de desarrollo Java y ejecutar una aplicación distribuida RMI de ejemplo que sirva de base para el desarrollo de esta práctica.

Por favor, a la hora de implementar vuestra aplicación**, no preguntéis en todas las opciones de los programas ¿Está seguro (s/n)?** como sale en el vídeo de Fermín, ya que es muy farragoso a la hora de corregir.

Detalles sobre las clases de la práctica

Con el objetivo de tener cierto orden, unificación y coherencia en el código que se entrega y para facilitar su posterior corrección, se tienen que nombrar obligatoriamente las clases/interfaces principales del programa de la siguiente manera:

`-Servidor`: Clase que contiene el *main* de la entidad Servidor.
`-Basededatos`: Clase que contiene el *main* de la entidad Base de Datos.
`-Jugador`: Clase que contiene el *main* de la entidad Jugador.
`-ServicioAutenticacionInterface`: contiene la interfaz remota del servicio de autenticación que depende de la entidad Servidor.
`-ServicioAutenticacionImpl`: clase que implementa la interfaz remota anterior.
`-ServicioGestorInterface`: contiene la interfaz remota del servicio Gestor que depende de la entidad Servidor.
`-ServicioGestorImpl`: clase que implementa la interfaz remota anterior.
`-ServicioDatosInterface`: contiene la interfaz remota del servicio Datos que depende de la entidad Base de Datos.
`-ServicioDatosImpl`: clase que implementa la interfaz remota anterior.
`-CallbackJugadorInterface`: contiene la interfaz remota del servicio que le permite recibir al jugador los eventos que le manda el servicio Gestor sobre el estado de la partida.
`-CallbackJugadorImpl`: clase que implementa la interfaz remota anterior.

Aparte de éstas, se pueden utilizar todas las clases que sean necesarias, pero no olvidar describir detalladamente su función en la memoria de la practica, así como, su lugar en el diagrama de clases.

RECOMENDACIONES:

-Nombrar los objetos remotos usando una *URL* que recoja toda la información sobre su dirección-puerto, servicio que presta e identificador de su proveedor; como se hace en el libro de M. L. Liu:

```
URL_nombre="rmi://"+ip+":"+rmiport+"/"+nombre_servicio+"/"+identificador_unico;
Naming.rebind(URL_nombre, objExportado);
```

- Desarrollar el código poco a poco, e ir compilando conforme se vayan obteniendo unidades funcionales.

- No es necesario cargar rmiregistry desde la línea de comandos, es posible crear una instancia del mismo mediante:

LocateRegistry.createRegistry(port)

- No se pide la instalación del gestor de seguridad de Java. Por tanto, no es necesario instanciar la clase *SecurityManager*.

LISTA SINCRONIZADA

En esta lista, uno de los procesos (el programa principal del cliente), debe ejecutar el método de lectura de la lista (el cual deja en espera la ejecución hasta que haya datos en la lista). Por otro lado, el proceso que se encarga de recibir los eventos del servicio Gestor del Servidor, debe escribir estos eventos en la lista, ejecutando el método de escritura en esta lista. El objeto lista debe ser creado mediante su instancia de clase en el programa principal del cliente y éste debe ser pasado como parámetro al proceso que se encarga de recibir los eventos del servidor.

La lista sincronizada debe de contener el comando *wait*() en el método de lectura y el comando *notifyAll*() en el de escritura para desbloquear convenientemente los estados de espera. Ambos métodos se implementan en la clase raíz de Java (Object), por tanto, ya están implementados "de serie" en el lenguaje (enlace).

A continuación, tenéis el código de la clase `ListaSincronizada` que implementa la lista sincronizada comentada. Os recomendamos su implementación.

```
import java.util.ArrayList;
import java.util.List;
public class ListaSincronizada {
   private List <String> lista;

   protected ListaSincronizada ()
   {
       lista= new ArrayList<String>();
   }

   public synchronized int NdatosPendientes()
   {
       return (lista.size());
   }

   public synchronized void AñadirEvento(String dato)
   {
      lista.add(dato);
      notifyAll();
   }
   public synchronized String ObtenerEvento() throws InterruptedException
   {
      if (lista.size()==0)
         wait();
      String dato = lista.get(0);
      lista.remove(0);
      return dato;
   }
}
```

NORMAS DE ENTREGA

Deberá entregar un único fichero comprimido en formato ZIP que contendrá una carpeta denominada PRACTICA que contendrá los ficheros fuente *.java*, los ejecutables *.class* *.jar* de la práctica **además de los ficheros *.bat* o *.sh* para arrancar las aplicaciones según la sintaxis de llamada comentada anteriormente**. Este fichero ZIP debe contener en su raíz un fichero PDF que contendrá la memoria de la práctica.

El fichero tiene que llamarse "Nombreestudiante_Apellidosestudiante.zip", donde Nombreestudiante es el nombre del estudiante y Apellidosestudiante son los apellidos del estudiante completos sin usar acentos y usando guiones bajos (_) en vez de espacios en blanco entre nombre y apellidos.

La memoria debe constar de los siguientes apartados:

- Portada con nombre, apellidos, DNI y correo electrónico del estudiante.
- Memoria descriptiva en la que se debe explicar el trabajo realizado. Recurra a esquemas o gráficos para plantear el funcionamiento de los programas y su funcionalidad.
- Diagramas de clases descriptivas y explicativas de la estructura del sistema.
- Pantallazos de ejemplos de funcionamiento con sus explicaciones.
- Conclusiones, opiniones, y mejoras (si fuese el caso), relacionadas con el trabajo realizado en la práctica.

El tamaño máximo de este documento memoria es **50 hojas** y el formato *.pdf*

La entrega de la práctica se hará a través del curso virtual y los plazos de entrega son:

- Plazo 1 (convocatoria ordinaria): prácticas recibidas entes del 15 de enero.
- Plazo 2 (convocatoria extraordinaria): prácticas recibidas con posterioridad al 15 de enero y antes del 15 de junio.