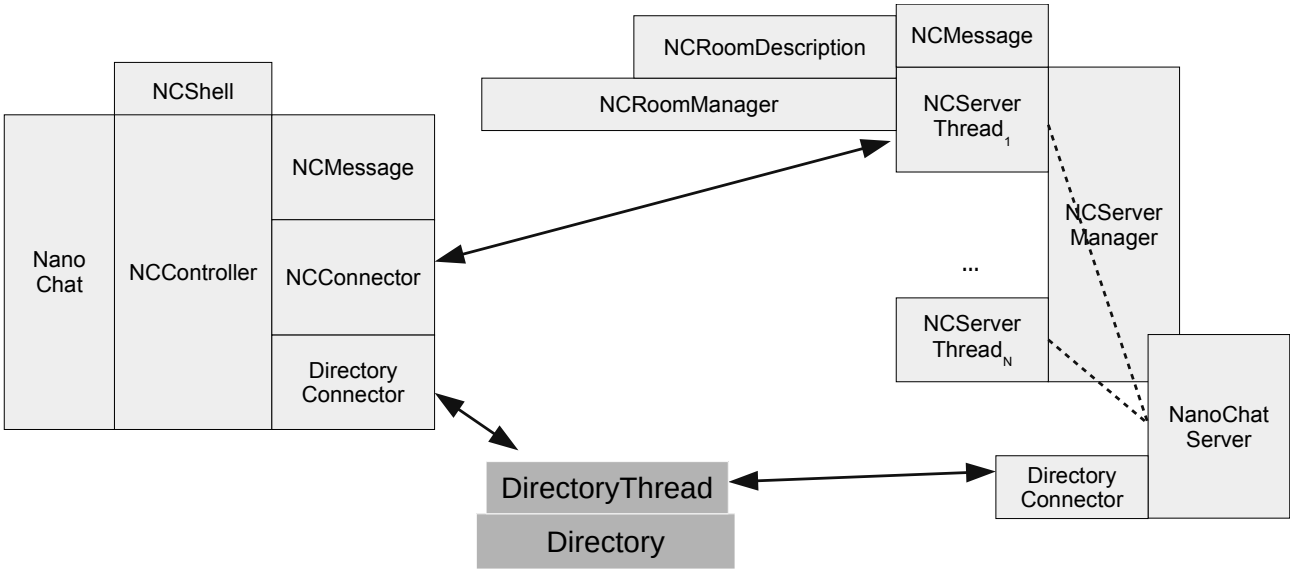
1. Introducción.

El objetivo de este boletín de prácticas es describir los componentes que conforman el proyecto de Eclipse `NanoChatStudents` que se proporciona a los estudiantes para desarrollar la práctica a partir de él.

2. Estructura del proyecto Eclipse.

La siguiente figura ilustra cómo se organiza el código del proyecto y qué entidades llevan a cabo la comunicación entre sí.



El código proporcionado a los estudiantes está formado por los siguientes paquetes y clases:

- **Paquete `es.um.redes.nanoChat.client.application`**
  - **`NanoChat.java`**: Clase main del cliente que tiene la secuencia de llamadas al controlador. Esta clase ya está terminada.
  - **`NCController.java`**: Implementación del controlador que estará encargado de coordinar la comunicación con el directorio y con el servidor de chat en función de la fase en la que nos encontremos o en función de lo que el usuario introduzca a través del shell.

- **Paquete `es.um.redes.nanoChat.client.shell`**
  - **`NCShell.java`**: Implementación del shell conforme a lo establecido en el documento de prácticas. Aunque está completamente terminada, puede ser modificada para incorporar mejoras.
  - **`NCCommands.java`**: Clase de apoyo para facilitar la implementación del shell. Contiene la definición de los tipos de comandos y de los parámetros aceptados por los mismos. Puede ser modificada para incorporar mejoras.

- **Paquete `es.um.redes.nanoChat.client.comm`**
  - **`NCConnector.java`**: Plantea la funcionalidad necesaria para comunicar el cliente con el servidor de chat. En ella deberán programarse los intercambios de mensajes iniciados principalmente por el cliente de chat.
- **Paquete `es.um.redes.nanoChat.directory.connector`**
  - **`DirectoryConnector.java`**: Implementación incompleta de clase que proporcionará los métodos necesarios para la comunicación con el Directorio usando sockets UDP. Será utilizada tanto por el cliente como por el servidor de chats. Se presupone ya completada en las prácticas anteriores.
- **Paquete `es.um.redes.nanoChat.directory.server`**
  - **`Directory.java`**: Implementación incompleta de la clase principal que procesa parámetros de llamada al servidor de directorio y se encarga de lanzar el proceso en segundo plano que atenderá solicitudes. Ya ha sido completada por cada grupo.
  - **`DirectoryThread.java`**: Implementación incompleta de la funcionalidad relacionada con el directorio. Se presupone ya completada en semanas anteriores.
- **Paquete `es.um.redes.nanoChat.messageFV / es.um.redes.nanochat.messageML`**
  - **`NCMessage.java`**: Superclase que proporciona la abstracción de un mensaje del protocolo entre el cliente y el servidor de chat. Contiene métodos de utilidad para generar o interpretar mensajes, algunos completamente desarrollados. De esta clase se tendrán que derivar los diferentes tipos de mensajes que se necesiten. Como ejemplo, se proporciona la implementación de uno de ellos.
  - **`NCRoomMessage.java`**: Subclase de ejemplo que proporciona la implementación concreta de un tipo de mensaje específico.
- **`Paquete es.um.redes.nanoChat.server`**
  - **`nanoChatServer.java`**: Clase main del servidor de chat. Se encarga de crear el socket de servidor, aceptar conexiones y generar los threads correspondientes para atender las solicitudes entrantes. Se proporciona implementada salvo algunos detalles.
  - **`NCServerThread.java`**: Clase que implementa el hilo que debe atender cada conexión entrante procedente de un cliente. Sólo se encuentra esbozada.
  - **`NCServerManager.java`**: Clase utilizada para gestionar las distintas salas que hay en el servidor así como los distintos usuarios que están conectados al servidor. Un objeto de esta clase será compartido entre todos los hilos. Su implementación está incompleta.

- **Paquete `es.um.redes.nanoChat.server.roomManager`**

  - **`NCRoomManager.java`**: Clase abstracta (superclase) que se utilizará como base a la hora de implementar la lógica concreta de cada chat. Se heredará de ella implementando los métodos y atributos necesarios.
  - **`NCRoomDescription.java`**: Clase que representa la descripción del estado actual de una sala, como puede ser su nombre, la lista de miembros y la fecha del último mensaje enviado. Contiene también un método para transformar esta información a un formato imprimible por pantalla.