# Evento de Discord para preparación del Examen Junio 2024 - 2025

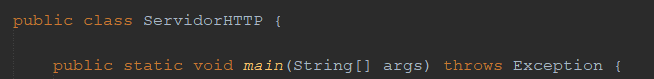
**El evento tendrá lugar el sábado 31 de mayo a las 18:00 (hora española) por Discord (enlace de descarga** [**aquí**](https://discord.com/download)**), una aplicación gratuita que permite videoconferencias grupales, así como compartir pantalla y chat de voz. Quién les escribe será el encargado de moderar y llevar a cabo la explicación, aunque contaremos con la colaboración de otros compañeros respondiendo dudas.**

**La invitación del evento a la que tendrán que unirse una vez tengan la aplicación instalada es este** [**Enlace a la clase**](https://discord.gg/srPHDEzd?event=1377700678285262960) **y el evento tendrá una duración aproximada de 2 horas.**

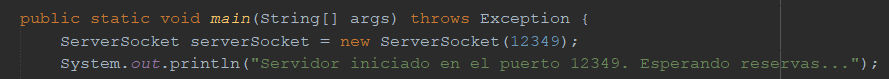
**La idea de este evento es tratar de dar una explicación del proceso que habrá que realizar para llevar a cabo el examen. Se enfoca en un acercamiento práctico a la realización de los ejercicios. Se tratarán temas repaso de concurrencia en Java y profundizaremos en la creación paso a paso de un servidor HTTP.**

# Pasos para crear un servidor HTTP en Java:

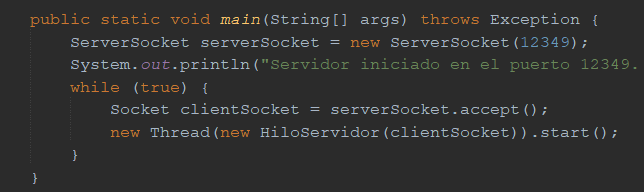
1. **Crear clase principal ServidorHTTP**



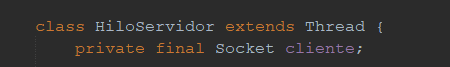
1. **Crear el método main con la creación de ServerSocket y el puerto**



1. **Crear el bucle que va a escuchar peticiones y lanzar el hilo**

****

1. **Crear clase HiloServidor que contendrá la lógica para manejar las peticiones**



1. **Hilo servidor recibe el socket donde envía los mensajes el cliente, se lo guarda(a través del constructor)**

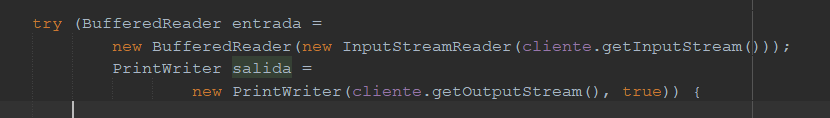
Texto

El contenido generado por IA puede ser incorrecto.

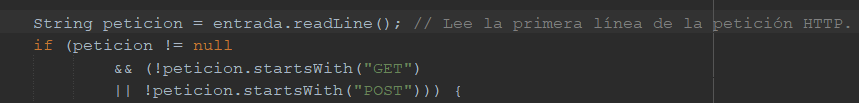
1. **La clase HiloServidor es una clase que extiende Thread o implementa Runnable por tanto utiliza su método run**



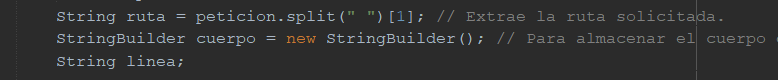
1. **En HiloServidor, se crean los BufferedReader y PrintWriter que serán los encargados de leer o escribir los mensajes del cliente. Siempre utilizaremos una try-with-resources o en su defecto try/catch**



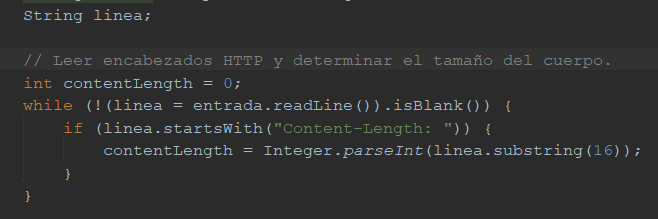
1. **Empezamos tratando las peticiones y comprobamos que si la petición es nula o no corresponde a una petición de tipo GET o POST devolvemos que no se ha encontrado**



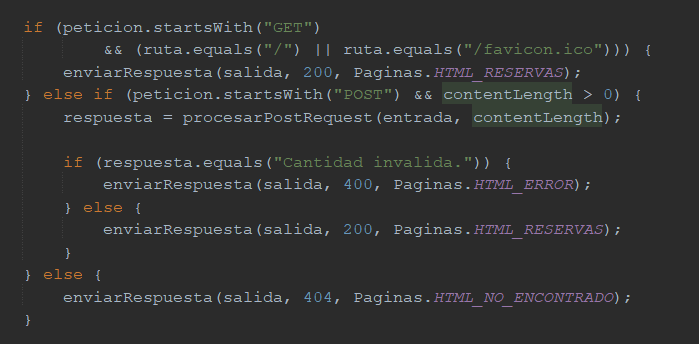
1. **Opcional: Obtendríamos entonces la ruta a la que se dirige la petición. Puede darse el caso de que nos pidan acceso una ruta diferente al inicio /**



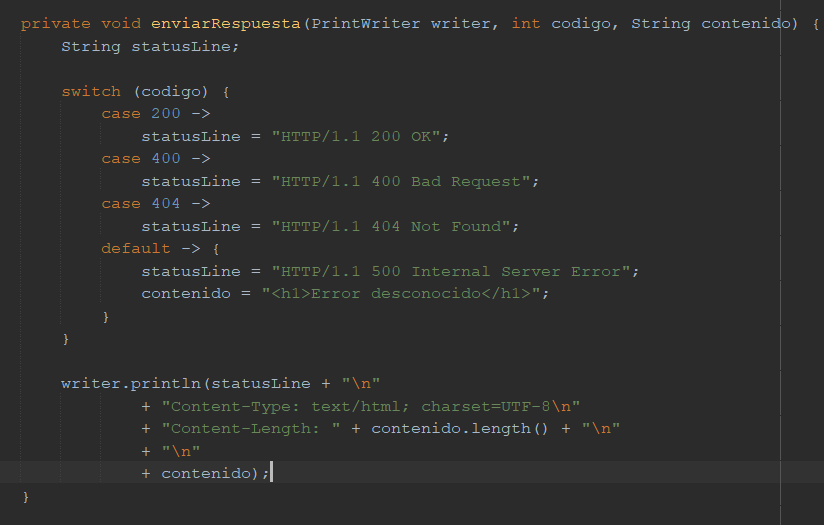
1. **Opcional: Obtendríamos el content-length(sólo nos sevirá en el caso de una petición POST, una petición GET no tiene cuerpo, por tanto, el content-lenght siempre será 0)**



1. **Comenzamos a clasificar la petición a través de if/else si son GET o POST(si nos los pidieran también clasificaríamos la petición por la ruta o recurso al que se dirige)**



1. **Según cada caso se devolverá un código de respuesta HTTP y una respuesta en HTML**



**Por último, comentaremos los recursos adicionales que deberíamos llevar al examen, así como se responderán todas las dudas que surjan.**

**¡¡¡¡ Les esperamos a todos !!!!**