**Programación Bluetooth en Java**

1. **Utiliza la documentación de la API y los métodos de la clase *LocalDevice* para obtener información acerca de la configuración de tu dispositivo Bluetooth local ¿Qué información puedes obtener?**

**A screen shot of a computer program

Description automatically generated**

En mi caso, se puede mostrar información como la dirección bluetooth del dispositivo, su nombre, si está encendido y algunas propiedades.

A screen shot of a computer

Description automatically generated

1. **Implementación de una aplicación que permita descubrir todos los dispositivos bluetooth visibles cercanos. Muestra su nombre y su dirección Bluetooth.**

Usaremos 2 clases de java, una clase principal donde declarar las clases necesarias y una clase *MyDiscoveryListener*, donde vamos a sobrescribir algunos métodos de la clase *DiscoveryListener*.

(En la captura se puede ver la explicación de cada una de las partes)

**A screen shot of a computer program

Description automatically generated**

Dentro de la clase *MyDiscoveryListener* se usa una lista para almacenar los dispositivos encontrados y un objeto *inquiryCompletedEvent* (junto con un monitor) para sincronizar la clase con la clase principal.

**A screen shot of a computer code

Description automatically generated**

Una vez hemos encontrado un dispositivo, el método *deviceDiscovered* nos informa, mostrando por pantalla la dirección bluetooth y su nombre, y guarda el dispositivo en la lista de dispositivos.

**A computer screen with colorful text

Description automatically generated**

Posteriormente, una vez que la búsqueda de dispositivos ha finalizado, el método *inquireCompleted* notifica al programa principal (libera el hilo con notifyAll) para que la ejecución del objeto en la clase principal continue.

A screen shot of a computer code

Description automatically generated

Si hacemos una búsqueda…

A computer screen shot of a black screen

Description automatically generated

1. **Implementación de una aplicación que permita descubrir un dispositivo concreto (por dirección Bluetooth y/o friendly name).**

**A screen shot of a computer code

Description automatically generated**

Creamos un objeto (String movil) en la clase principal, que pasaremos por el objeto “listener” para la clase *MyDiscoveryListener*, modificando a la vez el constructor de este, permitiendo que el método *deviceDiscovered* busque los dispositivos con el nombre de la variable “movil” o por otro lado por su dirección Bluetooth.

A computer screen shot of text

Description automatically generatedA screen shot of a computer

Description automatically generated

1. **Implementación de una aplicación que permita descubrir todos los servicios (de clase público) de todos los dispositivos cercanos.**

Para buscar los servicios necesitamos primero buscar los dispositivos, si hemos encontrado algún dispositivo, estarán guardados en la lista de dispositivos remotos.

A computer screen shot of a program code

Description automatically generated

Encontrados los dispositivos, declaramos los UUIDs (en nuestro solo será 1, el 0x1002, que es para mostrar servicios públicos), enteros que hacen referencia al nombre de servicio e ID y mostraremos una lista de todos los dispositivos y sus servicios con el siguiente método de la clase *MyDiscoveryListener*.

A screen shot of a computer code

Description automatically generated

Una vez la búsqueda finaliza, informa a la clase principal que la

búsqueda de servicios ha finalizado.

El método *servicesDiscovered* recibe una lista de servicios. Con esos servicios buscamos obtener las URLs de los mismos con el método *getConnectionURL*.

Si la URL no es null, la añadimos a la lista y conseguimos el nombre del servicio, si es que lo tiene. Si lo tiene, mostramos por pantalla el nombre y la URL del servicio. Si no, solo mostramos la URL.

A computer screen with colorful text

Description automatically generatedA screenshot of a computer

Description automatically generated

La lista de servicios, dada la cantidad de dispositivos encontrados

es muy larga, muestro alguno de los muchos servicios.

1. **Implementación de una aplicación que permita descubrir todos los servicios (de clase público) de un dispositivo concreto (especificado por dirección Bluetooth y/o friendly name).**

Aplicamos la misma idea que el ejercicio 4, solo que esta vez, al igual que en el ejercicio 3, declaramos un objeto de tipo String con el nombre (o dirección Bluetooth) del dispositivo que queramos encontrar y la pasamos por el constructor de la clase *MyDiscoveryListener*.

A screenshot of a computer

Description automatically generated

1. **Implementación de una aplicación que permita descubrir un servicio con un nombre concreto en un dispositivo concreto.**

Primero, debemos añadir un nuevo String (String service) a la clase *MyDiscoveryListener* para guardar el servicio a buscar y añadirlo al constructor.

A screen shot of a computer program

Description automatically generated

En el método *servicesDiscovered* comprobamos si el servicio descubierto es el que estamos buscando. Si lo es, mostramos por pantalla el servicio en cuestión y la URL de este, y lo añadimos a la lista de URLs.

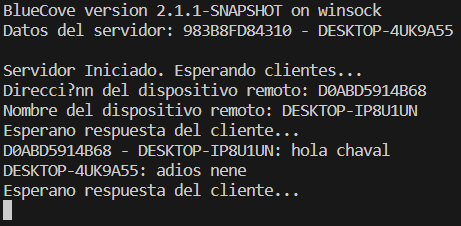
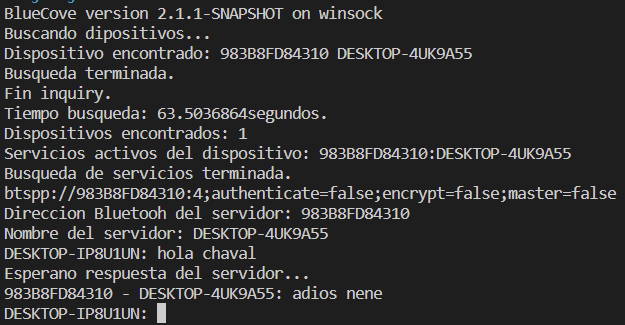
1. **Implementa una aplicación cliente-servidor que permita el envío de mensajes de texto entre ellos (como un chat textual). Ambos procesos (cliente y servidor) son procesos Java distintos que se ejecutan en máquinas diferentes para comprobar la comunicación inalámbrica.**

Extendiendo lo dicho en el ejercicio 4, una vez encontrada la URL del servicio “chat”, creamos una conexión, mostramos la dirección Bluetooth y nombre del servidor y comenzamos el intercambio de mensajes hasta que el servidor nos responda con “FIN”.

A computer screen shot of a program code

Description automatically generated

Ejemplo de comunicación entre dos dispositivos Cliente – Servidor:



**Cliente Servidor**

1. **Ejercicio opcional: una vez que tengas desarrollada la funcionalidad del cliente, integra su funcionalidad (y la del servidor) con una interfaz gráfica para probar la comunicación entre ambos interlocutores.**

Implementaremos un objeto *chatWindow* para que tanto el cliente como el servidor muestren su conversación en una ventana emergente y no por consola.

A screen shot of a computer program

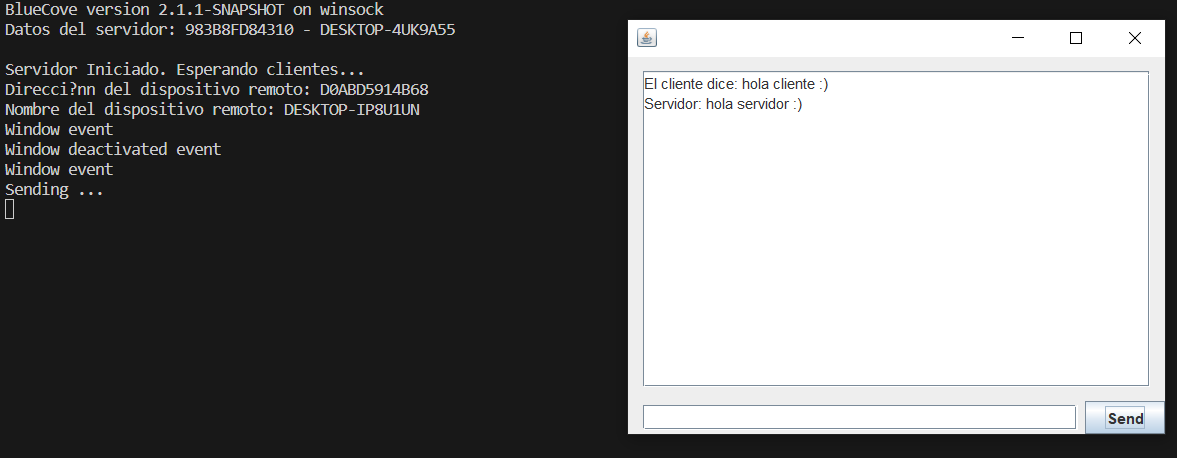
Description automatically generated

Implementación necesaria tanto para el Cliente como para el Servidor

Un ejemplo de ejecución seria: A screenshot of a computer

Description automatically generated

*Cliente*



*Servidor*