EJERCICIO 2.- SERVICIOS MÓVILES

1. Definir en sintaxis ABNF los mensajes del protocolo.

Ver documento denominado “MENSAJES DEFINIDOS EN ABNF”.

Nota:

Para obtener más información sobre los siguientes apartados y ver como se han realizado consultar el proyecto declarado como Practica2_Servicios.

2. Definir e implementar una clase para los mensajes del protocolo. Deberá incluir además todos aquellos parámetros o constantes necesarios.

Para ello, crearemos una clase Peluqueria. En ella definiremos los distintos constructores, métodos get y set, y un método toString que escribirá un mensaje.

```java
public class Peluqueria {
    //Variables
    private String nombre_peluqueria;
    private int codigo_peluqueria;
    private long fecha;

    //Constructor vacio
    public Peluqueria(){
        this.nombre_peluqueria="";
        this.codigo_peluqueria=0;
        this.fecha=0;
    }

    //Constructor
    public Peluqueria(String nombre_peluqueria, int codigo_peluqueria, long fecha) {
        this.nombre_peluqueria = nombre_peluqueria;
        this.codigo_peluqueria = codigo_peluqueria;
        this.fecha=fecha;
    }
```

```java
//Metodo de sobrescritura
@Override
public String toString() {
    return "Peluqueria{\n" + "nombre_peluqueria:" + nombre_peluqueria + "\ncodigo_peluqueria:"
            + codigo_peluqueria + "\nfecha:" + fecha + "\n}";
}
```

Y así con las diferentes clases que hemos realizado.

El método que visualiza el mensaje de la cita es el método siguiente:

```
 //Clase para los mensajes del protocolo
public String visualizaCita() {

    String texto="";
   //Recorremos la lista de citas
     for(int pos=0;pos<this.lcitas.size();pos++){
     texto+= this.lcitas.get(pos).toString();
}
     return texto;
}
//Lista las citas en una fecha
 public String visualizaCita(int dia, int mes, int anio){
     long fechaBuscar=dia + 100*mes + 10000*anio;
     //Inicializamos texto a vacío
     String texto="";
     //Recorremos la lista de citas
      for(int pos=0;pos<this.lcitas.size();pos++){

         //Comparamos si la fechabuscar es igual a la fecha
         if(this.lcitas.get(pos).getFechaCompara()==fechaBuscar){
                 texto+= this.lcitas.get(pos).toString();
         }else texto="No hay partidos en la fecha\n";
     }
     return texto;
 }
```

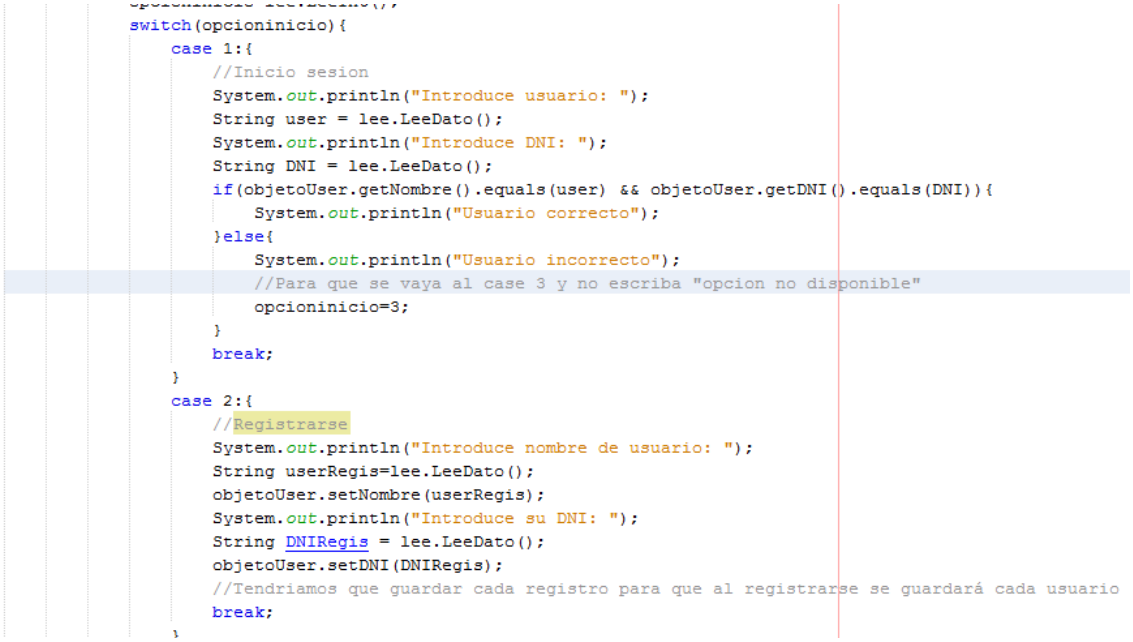
3. Definir e implementar una clase para los datos que se envían en el protocolo. Deberá incluir además todos aquellos parámetros o constantes necesarios.

Para el envío de los diferentes datos hemos creado en la clase principal un switch con sus diferentes cases y que permitirá al usuario registrarse, iniciar sesión, incluir una cita…

Veamos algún ejemplo:

```
//Menu inicio sesión
    public String textoMenuInicio(){
        String texto="Menu Inicio sesión\n";
        texto+="1. Inicio sesion \n";
        texto+="2. Registrarse\n";
        return texto;
    }

//Menu
    public String textoMenu(){
        String texto="MENU\n";
        texto+="0.- Cerrar Sesion.\n";
        texto+="1.- Alta cita.\n";
        texto+="2.- Baja cita.\n";
        texto+="3.- Mostrar informacion de citas.\n";
        texto+="4.- Mostrar informacion de citas dada una fecha.\n";
        return texto;
    }
```



```
switch(opcioninicio){
    case 1:{
        //Inicio sesion
        System.out.println("Introduce usuario: ");
        String user = lee.LeeDato();
        System.out.println("Introduce DNI: ");
        String DNI = lee.LeeDato();
        if(objetoUser.getNombre().equals(user) && objetoUser.getDNI().equals(DNI)){
            System.out.println("Usuario correcto");
        }else{
            System.out.println("Usuario incorrecto");
            //Para que se vaya al case 3 y no escriba "opcion no disponible"
            opcioninicio=3;
        }
        break;
    }
    case 2:{
        //Registrarse
        System.out.println("Introduce nombre de usuario: ");
        String userRegis=lee.LeeDato();
        objetoUser.setNombre(userRegis);
        System.out.println("Introduce su DNI: ");
        String DNIRegis = lee.LeeDato();
        objetoUser.setDNI(DNIRegis);
        //Tendriamos que guardar cada registro para que al registrarse se guardará cada usuario
        break;
    }
```

Cabe destacar que tenemos una clase donde se realizan los métodos de añadir citas, borrar citas, listar citas…

Para el caso de añadir citas tenemos:

```
public Peluqueria addCita(String nomb_pelu, int cod_pelu, long fecha){
    //Declaramos un jugador, utilizando el constructor de la clase Jugador
    Peluqueria nuevaCit= new Peluqueria(nomb_pelu,cod_pelu, fecha);
    //damos de alta en la hashmap
    lcitas.add(nuevaCit);

    return nuevaCit;
}
```

4. Definir e implementar una clase abstracta que implemente los métodos básicos para conseguir dar el servicio (esa clase sería la que emplearía el servidor). Métodos a implementar:

a. Identificación, autenticación o inicio de sesión.

b. Liberación de recursos o cierre de sesión.

c. Petición básica de servicio 1: modelar la petición a una operación concreta dentro del servicio.

Como podemos ver en las imágenes del punto 3, hemos realizado un do{}while con un switch que permitirá registrarse, iniciar sesión, cerrar sesión…

Más información: Ver proyecto.

5. Definir e implementar una interfaz con los métodos que debería tener un cliente, para poder acceder a la operación prevista. Dicha interfaz deberá incluir también todos los parámetros y constantes del servicio.

El interfaz que hemos realizado es el siguiente:

```
//Menu inicio sesión
    public String textoMenuInicio(){
        String texto="Menu Inicio sesión\n";
        texto+="1. Inicio sesion \n";
        texto+="2. Registrarse\n";
        return texto;
    }

 //Menu
    public String textoMenu(){
        String texto="MENU\n";
        texto+="0.- Cerrar Sesion.\n";
        texto+="1.- Alta cita.\n";
        texto+="2.- Baja cita.\n";
        texto+="3.- Mostrar informacion de citas.\n";
        texto+="4.- Mostrar informacion de citas dada una fecha.\n";
        return texto;
    }
```

6. Comentar adecuadamente todo el código

Todo el código comentado se encuentra en el proyecto enviado como "Practica2_Servicios". Este archivo podrá ser consultado para ver como se ha realizado este ejercicio.

Breve introducción del proyecto:

Hemos definido 6 clases.

1 → Peluqueria (Contiene datos de la peluqueria)

2 → usuario (Contiene los datos de usuario)

3 → Peluquero (Contiene los datos del peluquero)

4 → Lectura (Nos permite leer)

5 → Practica2_Servicios (clase principal)

6 → Practica2_Servicios (Tendrá los métodos de añadir cita, dar de baja una cita, visualizar las citas…)

Apunte:

Lo único que nos faltaría sería dar de alta peluquerías. Pero como esa parte no la define el usuario no hemos realizado un case para dar de alta una peluquería.

Las peluquerías estarían dadas de alta en el servidor. Por lo tanto, en nuestro código no se podría realizar una cita ya que no existen peluquerías dadas de alta.

Este problema lo podríamos “arreglar” definiendo una peluquería por defecto o simplemente introduciendo en el código un método que de alta peluquerías. Y las almacene en un arraylist de peluquerías.

Otro apunte sería guardar los datos de todos los usuarios que se registran, para no tener que registrarse cada vez que inicien nuestra app.

Antonio Osuna Melgarejo

Servicios Móviles

3º Curso

Telemática