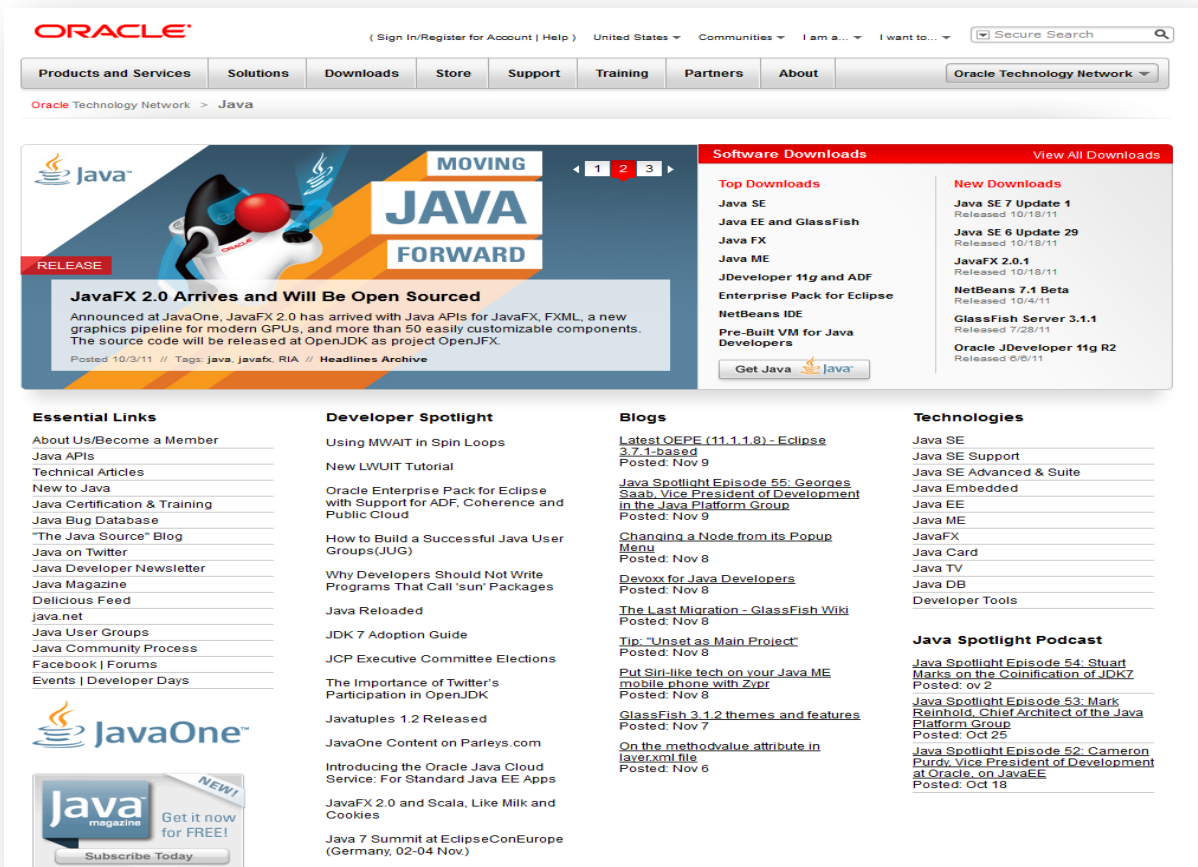


Tema 3: Java y los Sockets (2ª parte)



Bibliografía:

- ❑ [Kurose10] Apartados 2.1, 2.7, 2.8
- ❑ <https://docs.oracle.com/en/java/javase/15/docs/api/index.html>
- ❑ <http://java.com/es/about/>

- **InetAddress** es la clase que se utiliza para almacenar direcciones IP en Java
- Algunos métodos para visualizar información
 - static InetAddress **getByName**(String nombre)
 - Obtiene la dirección IP asociada a un nombre
 - String **getHostAddress**()
 - Devuelve la dirección IP en formato "aa.bb.cc.dd"
 - Ejemplo:

```
InetAddress inet = InetAddress.getByName("www.mit.edu");  
System.out.println("IP : " + inet.getHostAddress());
```
 - String **getHostName**()
 - Devuelve el nombre del host
 - Ejemplo:

```
InetAddress address = InetAddress.getLocalHost();  
String sHostName = address.getHostName();  
System.out.println(sHostName);
```



Obteniendo información de la conexión

```
public class EjemploInetAddress {
    public static void main(String args[]) {
        try{
            InetAddress zoltar = InetAddress.getByName("zoltar.redes.upv.es");
            Socket s=new Socket(zoltar, 7);
            System.out.println("Conectado");

            System.out.print("Host local:");
            System.out.println(s.getLocalAddress().getHostName());
            System.out.print("IP local:");
            System.out.println(s.getLocalAddress().getHostAddress());
            System.out.print("Puerto local:"); System.out.println (s.getLocalPort());
            System.out.print("Host remoto:");
            System.out.println(s.getInetAddress().getHostName());
            System.out.print("IP remota:");
            System.out.println(s.getInetAddress().getHostAddress());
            System.out.print("Puerto remoto:"); System.out.println(s.getPort());

            s.close();
            System.out.println("Desconectado");
        } catch (UnknownHostException e) {
            System.out.println("Host desconocido");
            System.out.println(e);
        } catch (NoRouteToHostException e) {
            System.out.println("No se puede conectar");
            System.out.println(e);
        } catch (IOException e) {
            System.out.println("Fallo al cerrar el socket");
            System.out.println(e);
        }

    } // main
} // class
```



- Argumentos de un programa desde la línea de órdenes:
 - `public static void main(String args[])`
 - `Args[0]`, `args[1]`, ...
 - Si hay argumentos de entrada, `args.length` debe ser mayor o igual a uno



■ Metodo `equals()`:

```
- String str1="El lenguaje Java";  
- String str2=new String("El lenguaje Java");  
- if(str1==str2){  
-     System.out.println("Los mismos objetos");  
- }else{  
-     System.out.println("Distintos objetos");  
- }  
- if(str1.equals(str2)){  
-     System.out.println("El mismo contenido");  
- }else{  
-     System.out.println("Distinto contenido");  
- }
```

} Compara objetos

} Compara strings

■ Metodo `compareTo()`

- Devuelve un entero menor que cero si el objeto string es menor (en orden alfabético) que el string dado, cero si son iguales, y mayor que cero si el objeto string es mayor que el string dado.
- `String str="Tomás";`
`int resultado=str.compareTo("Alberto");`
- La variable entera resultado tomará un valor mayor que cero, ya que Tomás está después de Alberto en orden alfabético.



Strings: obtener información

- Para obtener la longitud, número de caracteres que guarda un string se llama a la función miembro *length*.
 - `String str="El primer programa";`
 - `int longitud=str.length();`
- Podemos conocer si un string comienza con un determinado prefijo, llamando al método *startsWith*, que devuelve *true* o *false*, según que el string comience o no por dicho prefijo
 - `String str="El primer programa";`
 - `boolean resultado=str.startsWith("El");`
 - En este ejemplo la variable resultado tomará el valor *true*.
- De modo similar, podemos saber si un string finaliza con un conjunto dado de caracteres, mediante la función miembro *endsWith*.
 - `String str="El primer programa";`
 - `boolean resultado=str.endsWith("programa");`
- Si se quiere obtener la posición de la primera ocurrencia de la letra p, se usa la función *indexOf*.
 - `String str="El primer programa";`
 - `int pos=str.indexOf('p');`
- Para obtener las sucesivas posiciones de la letra p, se llama a otra versión de la misma función
 - `pos=str.indexOf('p', pos+1);`
 - El segundo argumento le dice a la función *indexOf* que empiece a buscar la primera ocurrencia de la letra p a partir de la posición pos+1.
- Otra versión de *indexOf* busca la primera ocurrencia de un substring dentro del string.
 - `String str="El primer programa";`
 - `int pos=str.indexOf("pro");`

