# PRÁCTICAS – TEMA 3

## Ejercicios

1. Instancia un objeto del tipo InetAdress con getLocalHost y analiza lo que te devuelve:
   1. getHostName ()
   2. getHostAddress()
2. Crea una conexión sobre una dirección web que corresponda a un API, y extrae el contenido utilizando URLConnection.

Has de hacerlo, al menos, sobre 3 endpoints.

Como ayuda se adjunta la documentación de un API gratuita <https://pokeapi.co/>

1. Usando el protocolo TCP, crea un programa cliente que solicite al usuario un texto, lo encripte y se lo envíe al servidor. El servidor ha de desencriptar el mensaje y mostrarlo. (puedes utilizar funciones que encuentres en internet, como ayuda se adjuntan dos funciones muy simples.)

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Output cliente:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Output servidor:

Texto

Descripción generada automáticamente

Una vez esté programado imprime y explica:

* En el cliente
  + El puerto y la ip del socket.
* En el servidor
  + El puerto y la ip del server socket
  + El puerto y la ip del socket generado en la comunicación

1. Usando el protocolo TCP, crea un programa cliente que solicite al usuario un número y se lo envíe al servidor. El servidor ha de devolver el cubo del número.

Output cliente:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Output Servidor:

Texto

Descripción generada automáticamente

1. Modifica el ejercicio anterior para que:
   1. Una vez finalice de hacer una cuenta y se la mande al cliente ha de seguir esperando nuevos clientes.

Output Servidor:

Texto

Descripción generada automáticamente

1. Usando el protocolo TCP, vamos a simular un proceso calculadora en un servidor.

Se mostrará un menú al usuario:

Introduzca una opción:

* 1 - Sumar
* 2 - Restar
* 3 - Multiplicar
* 4 - Dividir
* 9 - Salir
  1. Crea un programa cliente que:
     1. Le muestre un menú al usuario

Introduzca una opción:

* 1 - Sumar
* 2 - Restar
* 3 - Multiplicar
* 4 - Dividir
* 9 - Salir
  + 1. El usuario le comunica la opción elegida al Servidor.
    2. Si se han seleccionado las opciones 1, 2, 3 o 4:
       1. Le pide un número al usuario y se lo envía al servidor.
       2. Le pide otro número al usuario y se lo envía al servidor.
       3. Recibe el resultado de la operación del servidor y la muestra.
       4. Vuelve a mostrar el menú y queda a la espera de que el usuario introduzca una opción.
    3. Si el usuario ha seleccionado la opción 9:
       1. Mostrará el mensaje: “Saliendo del programa” y finalizará su ejecución.
    4. Si el usuario ha introducido otra opción se le indicará: “Opción errónea y se le volverá a mostrar el menú y pedir operación”
  1. Crea un programa servidor que:
     1. Abra un severSocket y se quede a la espera de clientes.
     2. Cada vez que reciba un cliente espere a que este le envíe la opción seleccionada.
     3. Si la opción es una operación:
        1. Quedará a la espera de recibir dos números.
        2. Realizará el cálculo correspondiente.
        3. Le envíe el resultado al cliente.
        4. Quedará a la espera de recibir otra operación.
     4. Si la opción es salir:
        1. Finalizará la conexión con el cliente y mostrará el mensaje: “Cliente finalizado”
     5. Una vez finalice la conexión con un cliente quedará a la espera de otro.
     6. Si se da, a lo largo de la ejecución, un error de conexión con el cliente se mostrará el mensaje “Conexión perdida” y el servidor se quedará a la espera de conectarse con un nuevo cliente.

1. Haz el ejercicio 3 utilizando el protocolo UDP.

Si no queremos que se rellenen con espacios vacíos los espacios del buffer que no se han llenado al recibir el mensaje, es necesario utilizar este constructor.



1. Haz el ejercicio 4 utilizando el protocolo UDP.
2. Haz el ejercicio 5 utilizando el protocolo UDP.
3. Haz el ejercicio 6 utilizando el protocolo UDP.
4. Paraleliza el ejercicio 6, de tal manera, que el servidor pueda gestionar las peticiones de varios clientes a la vez.
5. Modifica el ejercicio 11 para que, el cliente, en vez de mandar tres datos (primero el signo, después el num1 y después el num2) envíe un objeto que contenga ya toda esta información.
6. Crea un juego de preguntas y respuestas:
   1. El servidor:
      1. Tendrá almacenadas 5 preguntas con sus correspondientes respuestas.
      2. Le realizará las 5 preguntas al cliente y cotejará sus respuestas.
      3. Indicará al usuario su resultado.
      4. Almacenará una lista con los 5 jugadores que hayan acertado más preguntas.
   2. El cliente:
      1. Verá la lista con los 5 mejores jugadores.
      2. Recibirá, por orden, las cinco preguntas del servidor.
      3. Responderá a las preguntas y enviará las respuestas al servidor.
      4. Recibirá el resultado por parte del servidor.
   3. El servidor ha de poder comunicarse con varios clientes de manera simultánea.
   4. La lista de los mejores jugadores solo estará activa mientras el servidor no se caiga, cuando se reinicie, se perderán los valores.

Ejemplo de ejecución del cliente:

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

1. Ahorcado en red V1:
   1. Utilizando comunicación TCP vamos a crear un juego del ahorcado (de una manera un poco particular).
   2. Lo iremos haciendo a lo largo de varios ejercicios para que sea más cómodo y sencillo para el estudiante.
   3. Esta versión consistirá en lo siguiente:

* El servidor generará una palabra a adivinar.
* Habrá dos jugadores que colaborarán para adivinar la palabra siguiendo el formato ahorcado.
* En esta primera versión un jugador será un humano (cliente) y el otro será la máquina (un jugador simulado por la cpu).
* El juego consistirá en 5 fases y cada fase constará de un turno de cada jugador.
* En cada turno los jugadores dirán una letra y se desvelarán las casillas que coincidan con esa letra.

Texto

Descripción generada automáticamente

Ilustración 1: Vista del servidor

* Si los jugadores consiguen adivinar la palabra antes de que terminen los turnos los jugadores ganan.
* En caso contrario pierden.
  1. Clase Servidor:
     1. Atributos:
        1. Necesita tener control de los dos jugadores.
        2. Necesita controlar el turno y la fase para llamar al jugador correspondiente.
        3. Necesita la palabra a adivinar.
        4. Necesita una lista con todas las letras dichas por los distintos jugadores.
        5. Necesita una lista de posibles palabras.
     2. Ejecución
        1. Selecciona, de manera aleatoria, una de las posibles palabras y la establece como palabra a adivinar.
        2. Genera la palabra adivinanza (llamaremos palabra adivinanza a la palabra que se muestra al usuario desvelando las letras adivinadas por el usuario). Por ejemplo:



* + - 1. Muestra la palabra adivinanza al jugador al que le corresponda y espera a recibir una letra por parte de este.
      2. Se incrementa el turno.
      3. Se comprueba si ha finalizado el juego (se han consumido todas las fases/turnos y/o se ha finalizado la construcción de la palabra) .

En el caso de que no haya finalizado se vuelve al punto 2.

En el caso de que haya finalizado se indica el resultado y finaliza la partida.

* + 1. La idea es que el servidor no muera, sino que una vez finalice la partida quede a la espera de una nueva partida.
  1. Clase Cliente (jugadorHumano):
     1. Funcionamiento
        1. Recibirá la palabra adivinanza del servidor y la mostrará.
        2. Pedirá por teclado una letra y la enviará al servidor.
        3. Esperará su turno.
        4. Si el juego a ha finalizado recibirá un mensaje indicándole el resultado.
        5. Si el juego no ha finalizado recibirá un mensaje que le indicará que continue y volverá al punto 1.
  2. jugadorMáquina:
     1. Será un jugador que colaborará con el humano para intentar adivinar la palabra.
     2. Funcionamiento
        1. Cuando sea su turno pasará una letra aleatoria al servidor.
        2. Al finalizar el juego finalizará su ejecución
  3. Ejemplo de Funcionamiento:
     1. Output Jugador Humano (Cliente):

Texto

Descripción generada automáticamente

Es la primera fase del jugador, ha introducido la letra d y se le muestra las posiciones en las que se encuentran la letra d.

Por lo que parece, el jugador máquina ha dicho una letra que no se encuentra en la palabra.

* + 1. Output Servidor:

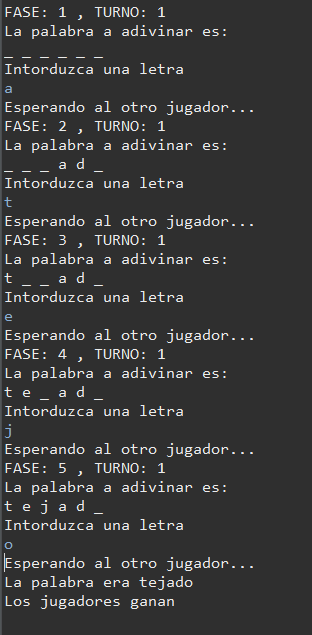
Texto

Descripción generada automáticamente

El servidor ha generado la palabra a adivinar que es saludo.

En la primera fase se han consumido los dos turnos

* El jugador humano ha dicho la letra d y se ha desvelado en la palabra.
* El jugado máquina ha dicho la letra y que no se encuentra en la palabra a adivinar.
  + 1. Output Jugador Humano (Cliente):



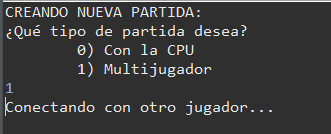
Aquí se puede ver la ejecución completa de una partida ganada por parte del cliente.

* + 1. Output Servidor:

Texto

Descripción generada automáticamente

Esa misma partida desde el punto de vista del servidor.

1. Ahorcado en red V2:
   1. Modifica el ejercicio anterior para que el servidor pueda gestionar varias partidas a la vez (paralelización).
2. Ahorcado en red V3:
   1. Mejoraremos el ejercicio anterior creando otra opción en la que el usuario pueda jugad con otro compañero online en vez de con un bot.
   2. Al comenzar la partida el cliente podrá elegir si quiere jugar con la máquina o con un usuario humano.
   3. 
   4. El funcionamiento, a nivel de normas, será similar a la versión anterior pero con un jugador humano en vez de un jugador simulado.
   5. Output del jugador 1:

Texto

Descripción generada automáticamente

* 1. Output del jugador 2:

Texto

Descripción generada automáticamente

1. Output del servidor:

Texto

Descripción generada automáticamente