

MASTERMIND
PROYECTO DE PROGRAMACIÓN DE SERVICIOS Y PROCESOS

Índice

1. Objetivos.....	2
2. Herramientas.....	2
3. Diseño.....	2
3.1. MastermindServer (funciones que ejecuta).....	2
3.2. MastermindClient (funciones que ejecuta).....	2
4. Funcionamiento.....	3
4.1.Ejecución del programa (Servidor).....	3
4.2.Ejecución del programa (Cliente).....	4
5. Conclusiones.....	5

1. Objetivos

Se propone la implementación del juego de Mastermind mediante una arquitectura cliente/servidor, que permita a los usuarios jugar desde la consola y enviar los datos en formato JSON. Además, se contempla como mejora incluir la funcionalidad de almacenamiento de las mejores puntuaciones, lo que permitiría a los jugadores competir y comparar sus resultados con los de otros usuarios.

2. Herramientas

En la implementación del juego se utiliza la librería org.json de Java, la cual incluye la clase JSONObject para formatear los datos de acuerdo a los requisitos establecidos. Para conectar el cliente y el servidor se emplean las clases Socket y ServerSocket, mientras que la comunicación entre ambos se realiza a través de las clases BufferedReader y PrintWriter.

Para generar el código de colores, se utiliza la clase Random para crear una secuencia de números aleatorios que se corresponden con los distintos colores disponibles en el juego. Por otro lado, para recibir la información que el usuario introduce, se utiliza la clase Scanner.

3. Diseño

El programa consta de dos clases: MastermindServer y MastermindClient, las cuales se encargan de intercambiar información entre sí.

3.1. MastermindServer (funciones que ejecuta)

1. Espera a que un cliente se conecte
2. Generación del código de colores
3. Espera a recibir datos del cliente
4. Procesamiento de la información recibida
5. Envío de una respuesta al cliente
6. Comprobar si la secuencia coincide
 - Sí: enviar un aviso al cliente de que la partida ha finalizado
 - No: enviar la respuesta al cliente y volver al punto 3
7. Comprobar el número de intentos (< 6)
 - Sí: almacenar los datos
8. Esperar respuesta del cliente (ver o no las puntuaciones)
 - Sí: leer la información y enviarla al cliente

3.2. MastermindClient (funciones que ejecuta)

1. Conexión al servidor
2. Solicitud de información al usuario (nombre y código de colores)
3. Envío de la información al servidor
4. Espera de la respuesta del servidor
5. Comprobar si la respuesta del servidor indica la victoria
 - Sí: proceder a la finalización del programa
 - No: solicitud del código de colores al usuario
6. Volver al paso 5
7. Fin de la ejecución

4. Funcionamiento

El funcionamiento comienza con la ejecución de la clase MastermindServer, la cual espera a que un cliente se conecte. Una vez establecida la conexión, el servidor genera un código de colores y espera a recibir datos del cliente. Tras procesar la información recibida, se envía una respuesta al cliente. Si la secuencia de colores completa coincide, se envía un aviso al cliente de que la partida ha finalizado. En caso contrario, se envía la respuesta y se continúa con la ejecución del programa.

Si al finalizar la partida el número de intentos es igual o inferior a cinco, el servidor almacena los datos relevantes de la partida. En caso de que el cliente desee ver esos datos, el servidor leerá la información del archivo de almacenamiento y la enviará al cliente. Tras la desconexión del cliente, el servidor espera a que un nuevo cliente se conecte para iniciar una nueva partida.

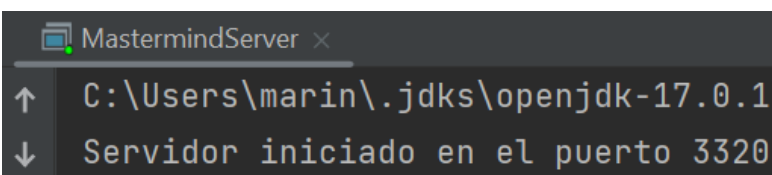
Por su parte, la clase MastermindClient se encarga de conectarse al servidor y solicitar información al usuario para enviarla al servidor. Tras ello, espera la respuesta del servidor y, dependiendo de la misma, sigue ejecutando la orden de solicitar el código de colores al usuario o procede a la finalización del programa.

Antes de terminar, se solicita al usuario que indique si desea ver los mejores resultados y, en caso afirmativo, se recibe la información del servidor sobre ellos y se muestra al usuario.

En cuanto al intercambio de datos, tanto en el cliente como en el servidor se crean dos JSONObject, uno para formatear la información que se va a enviar y otro para adecuarse al formato de la información recibida. Estos JSONObject, se pasan entre cliente y servidor mediante un PrintWriter utilizado para enviar los datos y un BufferedReader para recibirlos.

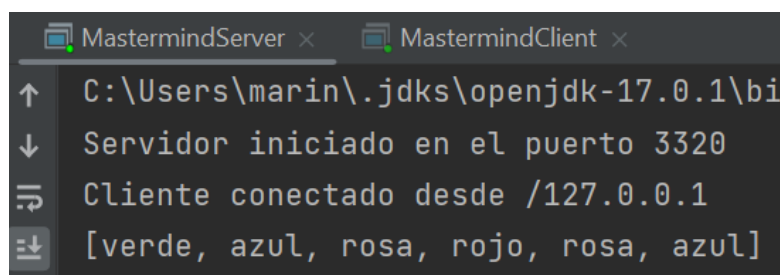
Para el fichero empleado para almacenar las puntuaciones, se utiliza un FileWriter y PrintWriter para su creación y escritura y a la hora de leerlo, se lee con Scanner y FileReader y se envía mediante el mismo PrintWriter que envía los JSONObject.

4.1.Ejecución del programa (Servidor)



```
MastermindServer x
C:\Users\marin\.jdk\openjdk-17.0.1\bin
Servidor iniciado en el puerto 3320
```

Se inicia el servidor



```
MastermindServer x  MastermindClient x
C:\Users\marin\.jdk\openjdk-17.0.1\bin
Servidor iniciado en el puerto 3320
Cliente conectado desde /127.0.0.1
[verde, azul, rosa, rojo, rosa, azul]
```

Cliente conectado +
generación de código

```
MastermindServer x MastermindClient x
C:\Users\marin\.jdk\openjdk-17.0.1\bin\java.exe
Servidor iniciado en el puerto 3320
Cliente conectado desde /127.0.0.1
[verde, azul, rosa, rojo, rosa, azul]
El jugador ha adivinado el código.
```

Fin de la partida

4.2.Ejecución del programa (Cliente)

```
MastermindServer x MastermindClient x
C:\Users\marin\.jdk\openjdk-17.0.1\bin\java.exe "-javaagent:C:\Program Files\Jet
Conectado al servidor
Instrucciones:
  Introduce los datos que se piden en el formato que se pide.
  Al introducir los colores recibirás una de estas respuesta:
    'No': El color no forma parte del código
    '--': El color forma parte del código pero la posición no es la correcta
    'Si': El color forma parte del código y está en la posición correcta
  Colores: [rojo, verde, azul, amarillo, rosa, naranja]
  -----
  Introduce tu nombre:
```

Conexión establecida

```
Introduce tu nombre: Marina
-----
Formato del código: color1 color2 color3 color4 color5 color6
Introduce código de colores separado por espacios: rojo verde azul amarillo rosa naranja
[--, --, --, No, Si, No]
Introduce código de colores separado por espacios:
```

Datos enviados al servidor, información procesada por el servidor, respuesta recibida en el cliente,
respuesta incorrecta

```
Introduce código de colores separado por espacios: verde azul rosa rojo rosa azul
[Si, Si, Si, Si, Si, Si]
Código adivinado, has ganado!
Marina: has tardado 2 intentos en adivinar el código.
¿Deseas ver los mejores resultados? [S/N]
```

Datos enviados al servidor, información procesada por el servidor, respuesta recibida en el cliente,
respuesta correcta

```
¿Deseas ver los mejores resultados? [S/N]
S
Nombre: Marina, Intentos: 5 Fecha: 2023-02-17 Hora: 9:20
Nombre: Marina, Intentos: 4 Fecha: 2023-02-17 Hora: 9:27
Nombre: Marina, Intentos: 2 Fecha: 2023-02-17 Hora: 13:35
Nombre: Marina, Intentos: 3 Fecha: 2023-02-17 Hora: 13:58
Nombre: Felix, Intentos: 2 Fecha: 2023-02-17 Hora: 14:4
Nombre: Marina, Intentos: 2 Fecha: 2023-02-23 Hora: 18:24
Fin del juego
Desconectado del servidor

Process finished with exit code 0
```

Mostrando las mejores puntuaciones

5. Conclusiones

El proyecto funciona de la manera esperada y se ajusta a los requisitos básicos aunque a parte de almacenar las mejores puntuaciones sería interesante desarrollar una interfaz gráfica para mejorar la experiencia de juego o implementar un sistema multijugador. Además, también se podría implementar el guardado de las partidas almacenando el nombre del jugador y el código que ha generado el servidor para esa partida.

A parte de esas mejoras, el guardado de los resultados se debería implementar también con JSONObject y escribiendo un fichero .json y no un archivo de texto, ya que al estar utilizando la clase sería bastante sencillo de realizar.

En cuanto a las herramientas, la elección de la librería org.json de Java para el formateo de datos en formato JSON es adecuada y permite una fácil comprensión de los datos que se envían y reciben.

El diseño del programa en dos clases, MastermindServer y MastermindClient, es sencillo. Se hace uso de estructuras de control como bucles y condicionales para manejar el flujo del programa y controlar la lógica del juego.

En conclusión, el programa tiene mucho margen de mejora dado que sus funcionalidades actualmente se encuentran muy limitadas.