Orientación a Objetos II 2025

Explicación de práctica
Semana del 26 de mayo

Ejercicios de la semana

En el cuadernillo de Frameworks

- Ejercicio 4: tcp.server.reply

Ejercicio 4: tcp.server.reply

A partir del código compartido en teoría de tcp.server.reply.

i) Reimplementar el servidor PasswordServer empleando un enfoque basado en composición de objetos utilizando los componentes del framework tcp.server.reply vistos en teoría

Recordando el ejercicio 1…

El servidor debe generar una password a partir de los tres argumentos que recibe en el mensaje enviado por el cliente.

- Arg[0]: cadena de caracteres (letras) permitidas para utilizar en la password
- Arg[1]: cadena de caracteres (números de 0 a 9) permitidos para utilizar en la password
- Arg[2]: cadena de caracteres especiales permitidos para utilizar en la password

Las passwords deben ser generadas de forma aleatoria y complir con las siguientes reglas:

Tener una longitud de 8 caracteres

Contener letras, al menos un número y un solo carácter especial

Dominio: TCP Servers

SingleThreadTCPServer (whitebox)

Cookbook:

1. Subclasificar SimpleThreadTCPServer
   a. Debe implementar Main(String[])
      i. crear una instancia
      ii. enviar método startLoop(String[])
   b. Debe implementar handleMessage(String) ⇒ hook

tcp.server.reply (blackbox)

Cookbook

1. En un objeto “contexto”
   a. instanciar un MessageHandler
      i. Echo,
      ii. Void
   b. Instanciar ConnectionHandler con el MessageHandler
      i. SimpleConnectionHandler
      ii. MultiConnectionHandler
   c. Instanciar TCPControlLoop con ConnectionHandler
   d. Enviar método startLoop() al TCPControlLoop

Ejercicio 4: tcp.server.reply

```java
public class TCPControlLoop {

    private IConnectionHandler connectionHandler;
    public TCPControlLoop() {
        this.connectionHandler = new SingleConnectionHandler();
    }

    public TCPControlLoop(IConnectionHandler connHandler) {
        this.connectionHandler = connHandler;
    }

    public final void startLoop(String[] args) {
        checkArguments(args);

        int portNumber = Integer.parseInt(args[0]);

        try (ServerSocket serverSocket = new ServerSocket(portNumber)) {
            displaySocketInformation(portNumber);
            while (true) {
                connectionHandler.handleConnection(serverSocket);
            }
        } catch (IOException e) {
            displayAndExit(portNumber);
        }
    }
```

Ejercicio 4: tcp.server.reply

```java
public interface IConnectionHandler {
    void handleConnection(ServerSocket serverSocket)  throws IOException;
}
```

```java
public interface IMessageHandler {

    void handleMessage(String message, PrintWriter out);

}
```

Ejercicio 4: tcp.server.reply

```
public class SingleConnectionHandler implements IConnectionHandler {
    private IMessageHandler messageHandler;

    public SingleConnectionHandler(IMessageHandler messageHandler, /*EndSessionPolicy endSessionPolicy, */
            ITCPSession sessionHandler) {
        this.messageHandler = messageHandler;

    }

    public SingleConnectionHandler(IMessageHandler messageHandler /*, EndSessionPolicy endSessionPolicy */) {
        this.messageHandler = messageHandler;

    }
```

Ejercicio 4: tcp.server.reply

```java
public class EchoHandler implements IMessageHandler {
    public void handleMessage(String message, PrintWriter out) {
        out.println(message);
    }
}
```

```java
public class EchoApp  {
    Run | Debug
    public static void main(String[] args) {

        new TCPControlLoop(new SingleConnectionHandler(new EchoHandler())).startLoop(args);

    }
}
```

Ejercicio 4: tcp.server.reply

ii) Modifique el framework para que la condición de cierre de una conexión sea configurable con strings provistos por las instanciaciones.

Recordando el ejercicio 1…

a. Una variable en SingleThreadTCPServer que se configura desde el método main() de las subclases.
b. Un método (hook) que retorna un booleano resultado de evaluar la condición.
c. Un método (hook) que retorna un String que es la palabra de término de sesión.
d. Una jerarquía de Strategies que implementan cada una de las condiciones de cierre de sesión.

Ejercicio 4: tcp.server.reply

iii) Respecto a las dos formas vistas para implementar los servidores (PasswordServer ejercicio 1) iii) y 4) i):

- ¿Qué debe hacer un desarrollador para extender el framework en cada una de las formas? Especifique qué clases debe subclasificar o implementar, qué métodos debe definir.
- ¿Cuánto conocimiento necesita tener el desarrollador sobre la estructura interna del framework para instanciarlo? ¿Y para extenderlo?
- ¿Qué técnica usarías si tuviera que ofrecer muchas configuraciones posibles para el servidor? ¿Por qué?
- Identifique los hotspots y frozen spots en cada una de las implementaciones.
- Considerando las dos formas de implementación del servidor PasswordServer, los programadores pueden asegurar que hay inversión de control? Justifique su respuesta identificando en qué parte se produce la inversión de control en cada uno de los casos.

Examen Parcial

Patrones	Documentar los roles del patrón. Repasar UML
Refactoring	Enfocarse en los smells más evidentes. Indicar de **forma breve los pasos** del refactoring elegido. Dar el código final, indicando donde se aplicó cada paso.
Frameworks	Diferenciar “Extender una clase” (extends), subclasificar “Extender un framework” / “Modificar un framework” “Usar un framework”

Examen Parcial

- **Primera fecha**: **Sábado 7 de junio, 13,00 horas**
  aula 5, 4, 9, 10a y 11

- 1er Recuperatorio: Sábado 28 de junio, 13,00 horas
  aula 5, 4, 9, 10a y 11

- 2do Recuperatorio: Sábado 12 de julio, 13,00 horas
  aula 5, 4, 9, 10a y 11
  + (aula a definir) + promoción