Modificación Patrón State

Objetivo

Crear una nueva estado “Apagado seguro”, el cual al ser activado, deberá esperar que se procesen todos los mensajes recibidos para continuar con el proceso de apagado.

Resolución

Código Fuente: https://github.com/abuestanv/PatronComportamiento

A continuación, se detallarán los pasos realizados para completar el objetivo de la tarea

i. Definición del nuevo estado

```
public class ShutdownSecureState extends AbstractServerState {
    private Thread monitoringThread;

    public ShutdownSecureState(final Server server){
        server.getMessageProcess().start();
        monitoringThread = new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    while (true) {
                        Thread.sleep(1000 * 5);
                        if (server.getMessageProcess().countMessage() == 0) {
                            server.setState(new StopServerState(server));
                            break;
                        }
                        System.out.println("Procesing data.....");
                    }
                } catch (Exception e) { System.out.println(e.getMessage());  }
            }
        });
        monitoringThread.start();
    }

    @Override
    public void handleMessage(Server server, String message) {
        System.out.println("Cant´t process request, server in process of safe
shutdown");
    }

}
```

ii. Validamos que no se pueda realizar el cambio de estado, mientras se tenga una cola activa y el estado actual será de tipo Apagado Seguro

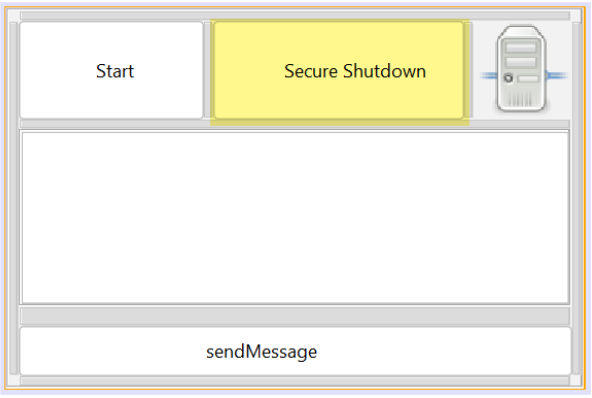
```
public class Server {
    [ … More code …]
    public void setState(AbstractServerState state) {
        if (this.state instanceof ShutdownSecureState &&
this.messageProcess.countMessage() > 0) {
            System.out.println("Server is in secure shutdown, cannot change
state");
            return;
        }
        if (this.state instanceof StartingServerState && state instanceof
StopServerState) {
            System.out.println("Server is starting, cannot change state");
            return;
        }
        this.state = state;
```

```
        System.out.println("Server change state > " +
this.state.getClass().getSimpleName());
    }
}
```

iii. Implementación del nuevo estado, sobre el formulario ServerPanel

Diseño. Se crea un nuevo botón “Secure Shutodown”, el evento a controlar solo se ejecutará si la aplicación está en los estados:

- Start o iniciado
- Starting o iniciando
- Saturated o saturado



La función que va a realizar, será la de asegurarse de que todos los mensajes, si es que existen, se envíen; posterior a eso quedara en estado detenido (Stop)

iv. Configuración dentro de la clase del formulario o JPanel

```
public class ServerPanel extends javax.swing.JPanel {
    private Server server;
    private int messageCouter;

    public ServerPanel() {
      [… MORE CODE …]
    }

    private class TextAreaPrinter extends PrintStream {
      [… MORE CODE …]
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
      [… MORE CODE …]
    }// </editor-fold>

    private void sendMessageEvent(java.awt.event.ActionEvent evt) {
        server.handleMessage("Send Message + " + ++messageCouter);
    }

    private void startAction(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        AbstractServerState state = server.getState();
        if (state instanceof ShutdownSecureState) {
            System.out.println("Secure shutdiwn in progress....");
            return;
        }
        if (state instanceof StopServerState) {
            btnStart.setText("Stop");
```

```
            server.setState(new StartingServerState(server));
        } else {
            if (state instanceof StartingServerState) {
                server.setState(new StopServerState(server));
            } else {
                btnStart.setText("Start");
                server.setState(new StopServerState(server));
            }
        }

    }


    private void shutdownSecureAction(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        AbstractServerState state = server.getState();
        if (state instanceof StopServerState) {
            server.handleMessage("Event not invalid...");
        } else {
            btnStart.setText("Start");
            server.setState(new ShutdownSecureState(server));
        }
    }



    // Variables declaration - do not modify
    private javax.swing.JButton btnSendMessage;
    private javax.swing.JButton btnShutDownSecure;
    private javax.swing.JButton btnStart;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTextArea jTextArea1;
    // End of variables declaration


}
```

Resultado

Ejemplo

a. Iniciar la aplicación
b. Enviar 4 mensajes
c. Apagar de forma segura la maquina