TRABAJO PRÁCTICO GRUPAL - PROGRAMACIÓN C

Grupo 8 - integrantes:

- Mariano Andrés Horianski
- Fernando Daniel Rosal
- Jazmín Milagros Piriz
- Jennifer Beltrán Flores

Este informe buscará dar una breve explicación y justificación de la implementación de las clases, interfaces y estructuras utilizada para la implementación de la **segunda entrega** del trabajo práctico.

Nuevos Participantes

Asociado

Tipo: Clase concreta que extiende de Persona e implementa Runnable.

Responsabilidades:

- Simular pedidos al sistema (atención a domicilio / traslado).

Justificación:

- Extiende de persona ya que comparte atributos generales de una persona.
- Implementa Runnable para poder ejecutarse en un hilo durante la simulación.

Operario

Tipo: Clase concreta que implementa Runnable.

Responsabilidades:

- Brindarle al usuario la oportunidad de intervenir en la simulación gestando de operario al solicitar el mantenimiento de la ambulancia.

Justificación:

- Implementa Runnable para poder ejecutar sus pedidos en un hilo independiente, de lo contrario la vista quedaría congelada hasta que se efectúe la solicitud.

Ambulancia

Tipo: Clase concreta que cumple la función de recurso compartido.

Patrón aplicado: Patrón State

Responsabilidades:

- Recibir solicitudes de los hilos concurrentes.
- Cambiar de estado siguiendo el patrón.
- Sincronizar acceso a los métodos con synchronized, wait() y notifyAll().

Justificación:

- La ambulancia debe cambiar de comportamiento según su estado, por lo que el patrón State es ideal.
- Actúa como recurso compartido ya que si se efectuaran de manera concurrente se corrompería la zona crítica (el cambio de estado).

Estados de la Ambulancia (IAmbulanciaState + subclases)

Tipo:

- IAmbulanciaState: Interfaz que modela el comportamiento de los estados.
- AmbulanciaStateDisponible, AmbulanciaStateEnTaller, etc: Clases concretas que implementan IAmbulanciaState.

Responsabilidades:

- Recibir solicitudes de los hilos concurrentes.
- Cambiar de estado siguiendo el patrón.
- Sincronizar acceso a los métodos con synchronized, wait() y notifyAll().

Justificación:

- Efectuar los cambios de estado según correspondiese.
- Determinar si se puede realizar o no cada acción.

Retorno Automático

Tipo: Clase concreta que implementa Runnable.

Responsabilidades:

- Generar periódicamente y de manera concurrente solicitudes de retorno automático, evitando deadlocks.

Justificación:

- Implementa Runnable ya que debe funcionar en un hilo de forma independiente y paralela a los asociados.

Colección de Asociados

Implementación: Se utiliza la clase HashMap<String, Asociado> como atributo en SingletonClínica.

Responsabilidades:

- Almacenar, buscar y eliminar asociados de manera eficiente.
- Evitar duplicados mediante la clave (DNI).

Justificación:

- Se utiliza un HashMap ya que la clave primaria del asociado es el DNI, único por definición, además de que el orden de los asociados es irrelevante.

SingletonClinica

Tipo: Clase concreta implementada como Singleton.

Responsabilidades:

- Se reutiliza la clínica para almacenar los nuevos participantes y centralizarlos.
- Gestionar las simulación (activar / finalizar).

Justificación:

- Garantiza consistencia y acceso global desde distintos hilos.

Persistencia

BDConexion

Tipo: Clase concreta implementada como Singleton.

Responsabilidades:

- Provee conexiones JDBC a la base de datos del sistema.

Justificación:

- Se utiliza singleton ya que debe existir una conexión única de la BD para todo el programa.

IAsociadoDAO

Tipo: Interfaz.

Responsabilidades:

- Modela el comportamiento que debe tener una clase que implemente el patrón DAO.

Justificación:

- Facilita el cambio de motor de base de datos.

AsociadoDAOMySQL

Tipo: Clase concreta que implementa el patrón DAO y la interfaz IAsociadoDAO.

Responsabilidades:

- Se encarga de separar la lógica de negocio de la lógica de acceso a datos.

Justificación:

- El patrón DAO nos permite aislar la lógica de acceso a la base de datos y, a su vez, facilita los cambios de motor de base de datos sin modificar el resto del sistema.

AsociadoDTO

Tipo: Clase concreta que implementa Serializable.

Responsabilidades:

- Se utilizará como el objeto plano que se utilizará para transferir o recibir datos en el DAO.

Justificación:

- Debe implementar Serializable ya que los objetos tendrán que viajar por la red.

Vista

Paneles, frames, botones, etc

Tipo: Clases concretas que aportan a la vista.

Responsabilidades:

- Le da una interfaz gráfica al usuario con la cual interactuar.

Justificación:

- Seguir el modelo MVC

Controladores

ActionListenerXXX, XXXListener

Tipo: Clases concretas que implementan ActionListener, MouseAdapter, etc.

Responsabilidades:

- Actuar de intermediario entre el modelo y la vista.

Justificación:

- Seguir el modelo MVC.

controlador.Asociados

Tipo: Paquete dentro de Controlador donde se encuentran todos los Listener relacionados a las interacciones de los asociados.

controlador.Simulacion

Tipo: Paquete dentro de Controlador donde se encuentra el Listener relacionado a la simulación.