Guía para Monitoreos nº 3: Capa Lógica y Persistencia- Tareas a realizar:

Leer capítulos 4 y 5 del material teórico y hacer projects para probar los distintos ejemplos que se muestran en el teórico.

Manejo de Excepciones:

Definir excepciones para manejo de errores. Cada excepción representa un posible error que puede ocurrir en la ejecución de un requerimiento. Las excepciones sustituyen a los booleanos y enumerados que retornábamos en las fachadas de C++. Por ejemplo:

```
public class PersonaException extends Exception
{
  …
}
```

La **Fachada** será la clase encargada de lanzar las excepciones hacia la capa gráfica para notificar la ocurrencia del error. Por ejemplo:

```
public class Fachada
{   ...
    public VOPersona listarPersona (int ced) throws PersonaException
    /* internamente busca la persona a listar, llamando a los métodos de la(s)
       colección(es) que sean necesarios, lanzando una PersonaException en caso
       de que la persona no esté registrada */
```

Puede suceder que haya métodos que necesiten lanzar más de una excepción. En tal caso, las mismas se colocan separadas con coma luego del **throws**.

Manejo de Persistencia:

Hacer una clase Persistencia.java (similar a la del capítulo 3 del material teórico) que respalde/recupere toda la estructura de objetos de la lógica en un archivo en disco. Se sugiere encapsular toda la estructura a respaldar/recuperar en un único objeto, a efectos de evitar duplicar información. Todo lo que se respalde debe ser **serializable**.

Monitor de Lectura y Escritura:

Es una clase que sirve para manejar el acceso concurrente a la capa lógica. Cuenta con dos atributos: cantLectores (int) y escribiendo (boolean) y los siguientes métodos:

- comienzoLectura
- terminoLectura
- comienzoEscritura
- terminoEscritura

La idea es tener una instancia del monitor como un nuevo atributo de la fachada y usarlo para controlar el acceso a cada requerimiento. Según lo que haga cada requerimiento, se debe razonar si es necesario utilizar el monitor para lectura o para escritura.

Por ejemplo, el siguiente requerimiento realiza únicamente tareas de lectura de datos, por lo que el monitor será usado para lectura. Tener **mucho** cuidado con en el pedido y la liberación del monitor. No liberarlo adecuadamente puede producir **deadlock**.

```
public VOPersona listarPersona (int ced) throws PersonaException
{
        monitor.comienzoLectura( );
        busco a la persona en el diccionario de personas
        if (la persona no existe en el diccionario)
        {
                monitor.terminoLectura( );
                throw new PersonaException ("la persona no está registrada");
        }
        else
        {
                Obtengo los datos de la persona y armo el value object
                monitor.terminoLectura();
                Retorno el value object
        }
}
```

Manejo de RMI (capítulo 5 del material teórico):

La **Fachada** es el único objeto que será accedido en forma remota (mediante RMI) desde los equipos de los usuarios. Se debe adaptar el código fuente de la clase Fachada (de acuerdo a lo que dice el material teórico) a efectos de que se transforme en un **objeto remoto**. Asegurarse de que los value objects que se transfieran entre clientes y servidor sean **serializables**.

Luego hacer un programa **`main`** que se ejecutará en el servidor central de la aplicación. Dicho programa **`main`** se encargará únicamente de:

1. Crear la fachada con las colecciones vacías
2. Invocar al método de la fachada que internamente levante de archivo la información de las colecciones. Si no pudo hacerlo (porque es la primera vez que se ejecuta) entonces quedan las colecciones vacías creadas en el punto anterior.
3. Publicar la fachada como objeto remoto, para que quede disponible para los programas cliente que luego se conectarán remotamente.

Luego hacer otro programa **`main`** que se ejecutará en c/u de los equipos de los usuarios. Por el momento, este será un programa main de prueba que se conectará remotamente (vía RMI) con el objeto remoto que reside en el equipo Servidor, y luego llamará a alguno(s) de sus métodos para comprobar que la comunicación remota funciona bien.