Para cada uno de los ejercicios debe realizar (en el orden indicado)

1) Diagrama de clases que modele la solución.
2) Indicar aquellos patrones de diseño utilizados con sus respectivos roles.

Luego de completar el paso 1) y 2) para TODOS los ejercicios

3) Implementar en Java cada uno de ellos (en los casos que detecte el uso del patrón de diseño Observer utilice en su implementación Observer/Observable y en otra alternativa la implementación propia con Listeners).

ToDo - Item

Vamos a modelar un objeto tarea (ToDoItem) de un manejador de tareas (ToDoList). Un ToDoItem tiene un texto que describe una tarea a realizar, una prioridad (1 a 10) y un estado.

Defina la clase ToDoItem que entienda los siguientes mensajes:

- **getText** "Retorna el texto"
- **setText:** "Setea el texto"
- **incrementPriority** "Incrementa la prioridad en uno. Si ya es 10, no hace nada"
- **decrementPriority** "Decrementa la prioridad en uno. Si ya es 0, no hace nada"
- **isCompleted** "Retorna true si la tarea ya fue completada, false en caso contrario"
- **toggle** "Cambia el estado de completada a no completada y viceversa"

Adicionalmente, se pide manejar dos nuevos tipos de tareas: las tareas compuestas, y la tarea con múltiples alternativas.

Tarea compuesta:

1. Una tarea compuesta tiene un texto general, y una o varias sub-tareas. Las subtareas pueden ser de cualquier tipo (inclusive, tareas compuestas o con múltiples alternativas). Una tarea compuesta está terminada (**isCompleted**) si "todas" sus subtareas han sido completadas.
2. La prioridad de una tarea compuesta se toma del máximo entre las prioridades de las subtareas.
3. Una tarea compuesta no entiende el mensaje **toggle** ni se le puede cambiar la prioridad.

Tarea con múltiples alternativas:

- Una tarea con múltiples alternativas tiene un texto general, y una o varias sub-tareas. Las subtareas pueden ser de cualquier tipo (inclusive, tareas compuestas o con múltiples alternativas). Una tarea con alternativas esta terminada (**isCompleted**) si "al menos una" sus subtareas han sido completadas.
- La prioridad de una tarea con múltiples alternativas se toma del máximo entre las prioridades de las subtareas.

- Una tarea con múltiples alternativas no entiende el mensaje **toggle** ni se le puede cambiar la prioridad.

Modele ambos tipos de tareas y especifique el comportamiento de las clases.

Filtros de imágenes

Debe desarrollarse una aplicación que permita el procesamiento de imágenes (al estilo photoshop, GIMP, etc). Específicamente, vamos a diseñar el subsistema de aplicación de filtros.

Los filtros son funciones que se aplican a nivel de píxel a una o varias imágenes y como resultado producen una nueva imagen. Cada imagen producida puede ser tomada como entrada para otro filtro.

Por ejemplo un filtro OR entre dos imágenes (del mismo tamaño) produce una tercera imagen donde el pixel X1,Y1 es el resultado de aplicar la operación logica OR a los píxeles en la posicion X1,Y1 en la primera imagen y X1,Y1 de la segunda imagen.

a) Diseñe una estructura de clases que permita construir filtros de cualquier complejidad utilizando operaciones primitivas ya implementadas. (AND, OR, XOR, NOT).

b) Implemente el mensaje **apply** para los filtros, que permite aplicar un filtro no primitivo.

Importante: Ud. no debe implementar las operaciones, suponga que existen, sólo debería definir los mensajes mediante los cuales espera comunicarse con ellos.

Figuras

Modele un editor de figuras (tipo corel draw). Una figura puede ser simplemente una figura geométrica. Otras figuras son grupos. Dado un conjunto de figuras simples debe ser posible agrupar las figuras en una figura compuesta o grupo. Un grupo puede estar formado por otros grupos y figuras.

Implemente el mensaje #area que retorna el área de una figura que se calcula como:

- Circulo: Pi * r^2
- Rectángulo: base * altura
- Triangulo: base * altura / 2

El área de un grupo debe ser la suma de las áreas de las figuras y grupos que lo componen.

Muro de Facebook

Facebook implementa una serie de aplicaciones sobre una red social. En Facebook, los usuarios pueden relacionarse entre ellos estableciendo relaciones de amistad. Una de las cualidades de ser “amigo” de otro usuario incluye poder ver en forma instantánea las publicaciones en el muro del usuario amigo en el muro propio. Por ejemplo, si Juan es amigo de Pedro, cuando Pedro publique un elemento en su muro, Juan instantáneamente podrá ver en su muro la publicación de Pedro.

Calculadora

Se necesita que modele una calculadora simple. Esta calculadora pasa por distintos estados. Cuando se enciende esta esperando el primer operando, sigue en este estado mientras el usuario ingresa dígitos. Una vez que el usuario ingresa un operador la calculadora pasa esperar el segundo operando. Se mantiene en este estado hasta que el usuario presiona el botón igual, en ese momento la calculadora realiza el calculo y lo muestra.
La calculadora puede pasar a estado de error si se divide por 0. Cuando esta en este estado ignora cualquier input hasta que el usuario presiona el botón CE. El usuario puede presionar dicho botón para cancelar la operación en cualquier momento.

Logger

En un sistema informático es importante poder tener registrados los sucesos que van ocurriendo a lo largo de la ejecución del mismo. Para ello, debe ser importante modelar un objeto que permita registrar un texto ingresado como parámetro con la descripción del evento y la fecha y hora en que se realizó el registro. El objeto diseñado, se utilizará como herramienta de programación y debugging por lo tanto es posible que se utilice en cualquier parte del sistema, en cualquier clase que cada programador pueda estar trabajando. Lo importante, es que toda la información que se registre, se guarde en la misma bitácora.

Fechas

Se está desarrollando una aplicación de edición de textos que será utilizada tanto en Estados Unidos como en Europa. Como se sabe la construcción de las fechas es diferente según donde nos encontremos, en Estados Unidos es mes / día / año, mientras que en Europa es día / mes / año. Desarrollar utilizando patrones de diseño teniendo en cuenta que el sistema cuenta la clase que crea fechas en la forma utilizada en Estados Unidos.

Expresiones aritméticas

Modele una solución para que sea posible representar expresiones aritméticas simples con los operadores de suma, resta, multiplicación y división y también la particularidad de utilizar paréntesis para agrupar sub expresiones.

Por otro lado, es necesario imprimir la expresión aritmética de tres formas: prefija (notación polaca), postfija (notación polaca inversa) e infija.