Organizador de Tareas

Es común en un grupo de trabajo, como los de nuestra Facultad, llevar a cabo varias tareas diariamente. Para esto le han pedido a usted que implemente un módulo de un organizador de tareas cumpliendo con las especificaciones siguientes:

Un **recurso** de una tarea se define como:

```
// Tipos de Recursos
public enum TipoRecurso
{
    // Persona en particular
    Persona,
    // Proyector
    Proyector,
    // Local
    Local
}
// Interfaz que debe de cumplir un Recurso
public interface IRecurso
{
    // Nombre o identificador del Recurso en particular
    string Nombre { get; set; }
    // Tipo de Recurso
    TipoRecurso TipoRecurso { get; set; }
}
```

Como puede ver un recurso puede ser: una persona, un proyector o un local, y además va a tener un **Nombre** que es por el que se va a distinguir.

Una **tarea** se define como:

```
// Interfaz que debe de tener una Tarea
public interface ITarea
{
    // Fecha de Inicio de la Tarea
    DateTime Inicio { get; set; }
    // Fecha de Vencimiento de la Tarea
    DateTime Vencimiento { get; set; }
    // Asunto o Tema de la Tarea
    string Asunto { get; set; }
    // Prioridad de la Tarea
    int Prioridad { get; set; }
    // Porciento de completado de la Tarea
    int PorcientoCompletado { get; set; }
    // Recursos que tiene asignados la Tarea
    IEnumerable<IRecurso> Recursos { get; set; }
    // Sub Tareas de la Tarea actual.
    IEnumerable<ITarea> Subtareas { get; set; }
}
```

La implementación de esta interfaz `ITarea` debe de cumplir:

- La fecha de *Inicio* no puede ser mayor que la de Vencimiento. De manera predeterminada la fecha de *Inicio* es la fecha de cuando se crea la *Tarea*.
- Siempre una tarea tiene un *Asunto* que no puede ser `null` ni la cadena vacía.
- El *PorcientoCompletado* no puede ser mayor que 100 ni menor que 0.
- Cada tarea debe de tener al menos un `IRecurso` asignado, no puede tener el mismo `IRecurso` asignado más de una vez.
- Ninguna de las Subtareas puede tener una fecha de inicio anterior a la fecha de inicio de la tarea que la contiene.
- Una tarea es igual a otra si tienen el mismo asunto, fecha de inicio, fecha de vencimiento, prioridad, porciento de cumplimiento, recursos y subtareas.

Cualquier violación de los requerimientos anteriores debe expresarse lanzando una Excepción.

Un organizador de tarea se define como:

```
    // Interfaz que debe de cumplir un organizador de Tareas
    public interface IOrganizadorTareas
    {
        /// <summary>
        /// Inserta una nueva Tarea. De no ser posible por
        /// las restricciones retorna
        /// false. Si se insertó sin problemas retorna true.
        /// O(min(n, k*log(n))).
        /// </summary>
        /// <param name="tarea">Nueva Tarea a insertar</param>
        /// <returns>True si se insertó la nueva Tarea,
        /// retorna False en cualquier otro caso</returns>
        bool AdicionaTarea(ITarea tarea);
        /// <summary>
        /// Elimina una Tarea. De no existir la Tarea
        /// dispara una Excepción. O(lg(n))
        /// </summary>
        /// <param name="tarea">Tarea a eliminar</param>
        void EliminaTarea(ITarea tarea);
        /// <summary>
        /// Elimina todas las Tareas que cumplen con el
        /// Filtro. Menor orden posible.
        /// </summary>
        /// <param name="filtro">Filtro de las Tareas</param>
        void EliminaTareas(Filtro filtro);
        /// <summary>
        /// Método que retorna un IEnumerable de todas las
        /// Tareas que hay.
        /// </summary>
        /// <returns> IEnumerable de todas las Tareas.</returns>
        IEnumerable<ITarea> Tareas();
        /// <summary>
        /// Método que retorna un IEnumerable de todas
        /// las Tareas que cumplen con el Filtro. Menor orden posible.
        /// </summary>
        /// <param name="filtro">Filtro de las Tareas</param>
        /// <returns>IEnumerable de todas las Tareas que
        /// cumplen con el Filtro</returns>
        IEnumerable<ITarea> Tareas(Filtro filtro);
    }
```

La implementación debe cumplir:

Cuando se adiciona una nueva tarea hay que chequear si ésta no entra en contradicción con las ya existentes. Para esto no puede haber dos tareas o más que requieran de un mismo recurso y que se solapen en el tiempo designado a la tarea. Tener en cuenta que:

- Un mismo proyector no puede estar asignado a dos o más tareas que sean en diferentes locales.
- La misma persona no puede estar asignada a diferentes locales.
- Un local no puede estar asignado a dos personas diferentes o más, si el asunto de las tareas a las que pertenecen no es el mismo.

Vale señalar que hay que chequear esto para las subtareas de la tarea que se está añadiendo. Cualquier violación de estos requerimientos debe de lanzar una Excepción cuando se inserta la nueva tarea.

Cuando se enumeran las tareas éstas deben darse ordenadas por la fecha de inicio, luego por la prioridad (tomando como orden de mayor prioridad a menor), luego por el asunto y por último por porciento de cumplimiento.

A partir del proyecto que se entrega debe completar la implementación de las interfaces que se les brindan así como deben de cumplir con los requerimientos antes mencionados.

La operación para saber las tareas que se solapan dada una tarea en específico tiene que tener un costo de **O (min(n, k lg n))**, donde k es el número de las tareas que se solapan con la tarea en cuestión.

Para lograr la eficiencia requerida con esta tarea usted deberá utilizar un árbol rojo-negro como estructura de datos base para la organización y gestión de las tareas.