Ejercicios sobre POO

Filtrado genérico

Se tiene la interfaz genérica `IFilter` que encapsula el concepto de un criterio de filtrado. Aquellos elementos para los que el método `Apply` devuelve `true` se consideran que cumplen el criterio asociado.

```
interface IFilter<T> {
    bool Apply(T item);
}
```

A partir de esta interfaz, se quiere implementar la interfaz `IFiltering` que encapsula operaciones a realizar con filtros.

```
interface IFiltering<T> {
    int Count(T[] items, IFilter<T> filter) {
        // Devolver la cantidad de elementos en `items`
        // que cumplen con el criterio de filtro.
    }

    T[] Select(T[] items, IFilter<T> filter) {
        // Devolver los elementos en `items` para los que
        // se cumple el criterio.
    }

    IFilter<T> Complement(IFilter<T> filter) {
        // Devolver una implementación de `IFilter`
        // que encapsule el criterio complementario
        // de `filter`.
    }
}
```

Por ejemplo, una posible implementación es la clase `OddFilter` que representa el criterio de un número que es impar.

```
class OddFilter: IFilter<int> {
    public bool Apply(int item) {
        return item % 2 != 0;
    }
}
```

A partir de este filtro y una implementación sensible de `IFiltering` se podría hacer lo siguiente:

```
IFiltering<int> filtering = new MyFiltering<int>(); // su implementación
IFilter<int> oddFilter = new OddFilter();

int[] items = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
```

```
filtering.Count(items, oddFilter); // Devolvería 5

filtering.Select(items, oddFilter);
// Devolvería {1, 3, 5, 7, 9}

filtering.Select(items, filtering.Complement(oddFilter));
// Devolvería {0, 2, 4, 6, 8}
```