Ejercicios propuestos patrones de diseño

Ejercicio patrón Factory-Method

1. Crear nuevos formatos para las fuentes 1 y 2, siendo respectivamente "Roman-Baseline" y "TrueType-Font" y volver a ejecutar la aplicación.
2. Cómo añadirías un nuevo botón de formato a la ventana?

Ejercicio patrón Adapter

Nuestra clase Sorting tiene un método "sortingSort"[1] que a partir de un objeto Iterator y un objeto Comparator, devuelve otro Iterator, donde los elementos de Iterator inicial están ordenados en base al criterio establecido en el Comparator. Su implementación en la siguiente:

```
public class Sorting {
       public static Iterator sortedIterator(Iterator it, Comparator comparator) {
             List list = new ArrayList();
             while (it.hasNext()) {
                 list.add(it.next());
             }

             Collections.sort(list, comparator);
             return list.iterator();
         }

}
```

Donde Iterator y Comparator son dos interfaces definidas en la librería de java.util. Su definición es la siguiente:

Method Summary

Interface Iterator

Modifier and Type	Method and Description
boolean	**hasNext**() Returns true if the iteration has more elements.
E	**next**() Returns the next element in the iteration.

Interface Comparator

Modifier and Type	Method and Description
int	**compare**(**T** o1, **T** o2) Compares its two arguments for order.
boolean	**equals**(**Object** obj) Indicates whether some other object is "equal to" this comparator.

[1] Método extraído de https://stackoverflow.com/questions/16434526/sort-an-iterator-of-strings. Octubre 2016.

Ejercicios propuestos:

1- Crear un programa principal, con un Vector de objetos de tipo Person (nombre, ciudad) y los visualice ordenados por nombre, y a continuación por ciudad.
2- Dada la clase AddressBook

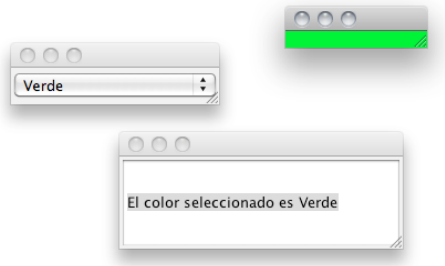
```
public class AddressBook {
	Vector<Person> personList=new Vector<Person>();

	public AddressBook(){
		personList.add(new Person("Jon", "Donostia"));
		personList.add(new Person("Ane", "Irun"));
		personList.add(new Person("Izaskun", "Tolosa"));
		personList.add(new Person("Mikel", "Hernani"));
	}
	public int getSize(){
		return personList.size();
	}
	public Person getPerson(int pos){
		return  personList.elementAt(pos);
	}
}
```

Crear un programa principal, que de igual forma que el punto anterior, visualice sus contactos ordenados por nombre, y a continuación por ciudad. Para desarrollar este apartado, **NO SE PUEDE modificar el código de la clase AddressBook.**

Ejercicio patrón Observer:

1. Queremos añadir una nueva ventana que únicamente nos indique cuál es color seleccionado tal y como se muestra en la siguiente figura:



2. Crear una aplicación que tenga a la vez 3 desplegables y donde las vistas de la derecha correspondiesen a cada desplegable tal y como se muestra en la siguiente figura.