Examen de Programación Orientada a Objetos (Plan 2014)

10 de diciembre de 2018

Nº matrícula:_____ **Grupo:**_____ **Nombre:** _____

Apellidos: _____

Práctica de Genéricos. (10 puntos)

1.- Codificar la interfaz IMatrimonio (la propia interfaz, no la clase que implementa la interfaz), que permite establecer parejas entre cualquier tipo de `Persona` o sus clases derivadas. El interface debe contener los métodos `set`, `get` (que devuelve un `IMatrimonio`), `getPareja1` y `getPareja2`. Documentar el interface para que pueda ser leído por `javadoc`.

```
/**
 * Alberga los objetos que forman un matrimonio, y que deben ser personas.
 * @param <P1> Una de las personas que conforma el matrimonio.
 * @param <P2> La otra persona que conforma el matrimonio.
 */
public interface IMatrimonio <P1 extends IPersona,P2 extends IPersona> {

        /**
         * @param pareja1 Una de las personas que conforma el matrimonio.
         * @param pareja2 La otra persona que conforma el matrimonio.
         */
        public void set(P1 pareja1, P2 pareja2);

        /**   @return un objeto matrimonio, conformado por dos personas.  */
        public IMatrimonio<P1,P2> get();

        /**   @return Uno de los dos objetos que definen el matrimonio.   */
        public P1 getPareja1();

        /**   @return Otro de los dos objetos que definen el matrimonio.  */
        public P2 getPareja2();
    }
```

Práctica de Colecciones. (10 puntos)

1.- Implementar el interface `IFiltro`, que permite saber si una palabra es vacía (si se encuentra en los ficheros `verbos.txt` o `palabrasVacias.txt`). **SOLO** debe hacerse una lectura de los ficheros; posteriormente a esa lectura se podrá preguntar por cada palabra que se desea conocer si es vacía. Se podrá utilizar la estructura `java.util` deseada.

```
public interface IFiltro {

   /**
    * Indica si una palabra es vacia.
    *
    * @param s Palabra que se deasea saber si es o no vacía.
    * @return true: la palabra aportada es vacia.
    */
    public boolean contains(String s);

}

public class Filtro {

       private TreeSet<String> verbos,vacias;

       public Filtro() {
         verbos = new TreeSet<String>();
         vacias = new TreeSet<String>();
         verbos=toTreeSet("verbos.txt");
         vacias=toTreeSet("palabrasVacias.txt");
       }

       private TreeSet<String> toTreeSet(String ficheroPalabrasVacias) {
         Scanner in=null;
         try {
             in = new Scanner(new FileReader(ficheroPalabrasVacias));
         } catch (FileNotFoundException e) {
             System.out.println("Error abriendo el fichero "+ficheroPalabrasVacias);
         }
         TreeSet<String> treeSet = new TreeSet<String>();
         while (in.hasNext())
             treeSet.add(in.next().toUpperCase());
         in.close();
         return treeSet;
       }

       public boolean contains(String s) {
           return vacias.contains(s)||verbos.contains(s);
       }

}
```