Examen de Prácticas de Programación 1 - 31/ENE/2013

- Tiempo máximo para realizar el trabajo de programación propuesto: 75 minutos
- Entrega del trabajo: envío del **fichero de texto con el código Java** de la clase a desarrollar (el fichero **Febrero2013.java**) por correo electrónico a la dirección que se indique. Nadie debe levantarse de su puesto de trabajo ni abandonar la sala hasta que el profesor le confirme la recepción del trabajo.
- En esta prueba se valorará, como condición previa y esencial, que **el método pedido se comporte según sus especificaciones**. Sólo en el caso de que se satisfaga lo anterior, se valorará también: 1) el diseño algorítmico del código y 2) la legibilidad del código.

Especificación del trabajo a desarrollar en el turno 1º (9:00 horas)

En el trabajo obligatorio de esta asignatura se han diseñado las clases ***Ciudadano***, ***Socio***, ***Asociacion*** y ***FicheroCiudadanos*** en el **package *trabajoProg1***. Cada alumno va a volver a hacer uso de estas clases.

En este examen debe desarrollar la clase **trabajoProg1.Febrero2013** que ofrecerá un único método público, el método ***coincidencias(Asociacion, Asociacion)*** cuya especificación se muestra a continuación.

```
package trabajoProg1;

/**
 * Autor: no olvide escribir aquí su NOMBRE y APELLIDOS
 */
public class Febrero2013 {

    /**
     * Pre: asoc1!=null, asoc2!=null, en la asociación asoc1 no hay socios con idéntico DNI
     *      y en la asociación asoc2 no hay socios con idéntico DNI
     * Post: Crea una tabla de referencias a objetos Ciudadano con tantos elementos
     *       como socios de [asoc1] son también socios de [asoc2] y devuelve la referencia
     *       a la tabla creada. Cada uno de los objetos referenciados por la tabla
     *       gestiona la información de un ciudadano socio de ambas asociaciaciones.
     *       Si no hubiera ningún socio común devuelve una referencia null.
     */
    public static Ciudadano[] coincidencias (Asociacion asoc1, Asociacion asoc2)

}
```

Conviene tener presente que es posible que un mismo ciudadano sea socio de varias asociaciones y en ellas consten algunos datos de él con valores diferentes, por ejemplo, su nombre y apellidos en una asociación pueden ser *María Luisa Pérez Lacambra*, en otra *Mª Luisa Perez Lacambra* (el nombre abreviado y el apellido sin tilde) y en otra *Marisa Pérez Lacambra* y, por ejemplo, el estado civil que conste en unas de ellas sea *soltera* y, en otras, *casada*. Por ello los socios de una asociación se identifican de forma inequívoca mediante el número de su DNI, independientemente de que sus restantes datos personales en ellas puedan ser idénticos o presentar variaciones, como las comentadas anteriormente. A efectos de identificar socios, el único criterio válido es la comparación de sus números de DNI.

Antes de enviar el trabajo, conviene hacer cuantas pruebas sean necesarias sobre el comportamiento del método pedido. Para ello cada alumno diseñará uno o más programas de prueba. Este código no deberá ser entregado. Se recuerda que en la web de la asignatura hay disponibles varios ficheros con información de ciudadanos para facilitar la realización de pruebas (*ciu01.txt*, *ciu02.txt*, etc.): ***Materiales docentes*** → ***Código_Java*** → *carpeta* ***datos/trabajoProg1***.

Examen de Prácticas de Programación 1 - 31/ENE/2013

- Tiempo máximo para realizar el trabajo de programación propuesto: 75 minutos
- Entrega del trabajo: envío del **fichero de texto con el código Java** de la clase a desarrollar (el fichero **Febrero2013.java**) por correo electrónico a la dirección que se indique. Nadie debe levantarse de su puesto de trabajo ni abandonar la sala hasta que el profesor le confirme la recepción del trabajo.
- En esta prueba se valorará, como condición previa y esencial, que **el método pedido se comporte según sus especificaciones**. Sólo en el caso de que se satisfaga lo anterior, se valorará también: 1) el diseño algorítmico del código y 2) la legibilidad del código.

Especificación del trabajo a desarrollar en el turno 2º (11:00 horas)

En el trabajo obligatorio de esta asignatura se han diseñado las clases ***Ciudadano***, ***Socio***, ***Asociacion*** y ***FicheroCiudadanos*** en el **package *trabajoProg1***. Cada alumno va a volver a hacer uso de estas clases.

En este examen debe desarrollar la clase **trabajoProg1.Febrero2013** que ofrecerá un único método público, el método ***listaReplicados(Asociacion)*** cuya especificación se muestra a continuación.

```
package trabajoProg1;

/**
 * Autor: no olvide escribir aquí su NOMBRE y APELLIDOS
 */
public class Febrero2013 {

    /**
     * Pre: asoc != null
     * Post: Devuelve la referencia a una tabla que almacena los números de socio de miembros
     *       repetidos de [asoc] y que, por tanto, ya están registrados en dicha asociación
     *       con un número de socio más bajo (con mayor antigüedad). Si no hubiera ningún socio
     *       repetido entonces devuelve una referencia null.
     */
    public static int[] listaReplicados (Asociacion asoc)

}
```

Conviene tener presente que es posible que, por error, puede que un mismo ciudadano haya sido dado de alta como socio de una misma asociación más de una vez. También es posible que alguno de sus datos varíe al ser dado de alta cada vez , por ejemplo, pudo ser dado de alta la primera vez como *María Luisa Pérez Lacambra*, posterioemente como *Mª Luisa Perez Lacambra* (el nombre abreviado y el apellido sin tilde) y más adelante como *Marisa Pérez Lacambra* y, por ejemplo, puede que su estado civil, *soltera* o *casada*, fuera diferente al producirse cada alta. Por ello, los socios de una asociación se identifican de forma inequívoca mediante el número de su DNI, independientemente de que alguno de sus datos personales puedan presentar variaciones, como las comentadas anteriormente. A efectos de identificar socios replicados en una asociación, el único criterio válido es mediante su número de DNI, un dato que no puede variar.

Antes de enviar el trabajo, conviene hacer cuantas pruebas sean necesarias sobre el comportamiento del método pedido. Para ello cada alumno diseñará uno o más programas de prueba. Este código no deberá ser entregado. Se recuerda que en la web de la asignatura hay disponibles varios ficheros con información de ciudadanos para facilitar la realización de pruebas (*ciu01.txt*, *ciu02.txt*, etc.): ***Materiales docentes*** → ***Código_Java*** → *carpeta* ***datos/trabajoProg1***.

Una solución del problema propuesto en el turno 1º de Febrero/2013

```
package trabajoProg1;

/**
 * Autor: escriba aquí su NOMBRE y APELLIDOS
 */
public class Febrero2013 {

    /**
     * Pre: asoc1!=null, asoc2!=null, en la asociación asoc1 no hay socios con idéntico DNI
     *      y en la asociación asoc2 no hay socios con idéntico DNI
     * Post: Crea una tabla de referencias a objetos Ciudadano con tantos elementos
     *       como socios de [asoc1] son también socios de [asoc2] y devuelve la referencia
     *       a la tabla creada. Cada uno de los objetos referenciados por la tabla
     *       gestiona la información de un ciudadano socio de ambas asociaciaciones.
     *       Si no hubiera ningún socio común devuelve una referencia null.
     */
    public static Ciudadano[] coincidencias (Asociacion asoc1, Asociacion asoc2) {
        /*
         * Calcula, en cuenta, el número de socios de [asoc1] que también lo son
         * de [asoc2]
         */
        int cuenta = 0;
        for (int i=1; i<asoc1.numSocios(); ++i) {
            int dni = asoc1.socio(i).datosPersonales().dni();
            if (buscar(asoc2, dni)) {
                ++cuenta;
            }
        }
        if (cuenta==0) {
            /*
             * No hay socios comunes
             */
            return null;
        }
        else {
            /*
             * Crea la tabla con la información de los socios comunes
             */
            Ciudadano[] T = new Ciudadano[cuenta];
            cuenta = 0;
            for (int i=1; i<asoc1.numSocios(); ++i) {
                int dni = asoc1.socio(i).datosPersonales().dni();
                if (buscar(asoc2, dni)) {
                    T[cuenta] = asoc1.socio(i).datosPersonales();
                    ++cuenta;
                }
            }
            return T;
        }
    }

    /**
     * Pre: asoc !=null
     * Post: Devuelve true si y sólo si hay un socio de [asoc] cuyo número de DNI
     *       es igual al valor del segundo parámetro (dni)
     */
    private static boolean buscar (Asociacion asoc, int dni) {
        boolean encontrado = false;
        int j=1;
```

```
        while (!encontrado && j<=asoc.numSocios()){
            if(dni==asoc.socio(j).datosPersonales().dni()) {
                encontrado = true;
            }
            else {
                ++j;
            }
        }
        return encontrado;
    }

}
```

Una solución del problema propuesto en el turno 2º de Febrero/2013

```
package trabajoProg1;

/**
 * Autor: escriba aquí su NOMBRE y APELLIDOS
 */
public class Febrero2013 {

    /**
     * Pre: asoc !=null
     * Post: Devuelve la referencia a una tabla que almacena los números de socio de miembros
     *       repetidos de [asoc] y que, por tanto, ya están registrados en dicha asociación
     *       con un número de socio más bajo (con mayor antigüedad). Si no hubiera ningún socio
     *       repetido entonces devuelve una referencia null.
     */
    public static int[] listaReplicados (Asociacion asoc) {
        int numSocios = asoc.numSocios();
        /*
         * Cuenta en [cuenta] el número de sococios replicados
         */
        int cuenta = 0;
        for (int i=2; i<=numSocios; ++i) {
            Socio s = asoc.socio(i);
            if (buscar(asoc, s.datosPersonales().dni(), i)) {
                ++cuenta;
            }
        }
        if (cuenta>0) {
            /*
             * Almacena en [T] los números de los socios replicados
             */
            int[] T = new int[cuenta];
            cuenta = 0;
            for (int i=2; i<=numSocios; ++i) {
                Socio s = asoc.socio(i);
                if (buscar(asoc, s.datosPersonales().dni(), i)) {
                    T[cuenta] = s.numeroSocio();
                    ++cuenta;
                }
            }
            return T;
        }
        else {
            return null;
        }
    }

    /**
     * Pre: asoc !=null
     * Post: Devuelve true si y sólo si hay un socio de [asoc] que, con una antigüedad
     *       mayor al socio i-ésimo, tiene un número de DNI igual al valor del segundo
     *       parámetro (dni)
     */
    private static boolean buscar (Asociacion asoc, int dni, int i) {
        boolean encontrado = false;
        int j = 1;
        while (!encontrado && j<i) {
            Socio s2 = asoc.socio(j);
            encontrado = dni==s2.datosPersonales().dni();
            ++j;
```

```
        }
        return encontrado;
    }

}
```