	Programación Orientada a Objetos- Curso 2022/2023 Grado en Ingeniería Informática en Sistemas de Información **EXAMEN - ENSEÑANZAS BÁSICAS — 24/05/2023**	**Valoración: 5 puntos**

APELLIDOS:_____ **NOMBRE:**_____ **D.N.I.:**_____

Normas Generales:
- Es imprescindible que todas las hojas lleven el nombre, apellidos y DNI del alumno/a, incluidas las hojas de borrador. Si un examen no tiene nombre, quedará invalidado.
- Se permite el uso de lápiz para completar el examen.
- Se recomienda encarecidamente que se lea el examen completo antes de comenzar.
- No se admitirán preguntas durante los primeros 20 minutos de examen.
- La duración de examen es de 2 horas y media.

Se pretende completar la aplicación que modela un Gimnasio y sus abonados. El diagrama de clases proporcionado al final del enunciado contempla todas las clases de la aplicación. Las clases Fecha y Persona se consideran implementadas según el diagrama de clases, excepto aquellas partes que sean cuestiones a resolver en esta evaluación.

NOTA:
- Se tendrá muy en cuenta la reutilización de código y el aprovechamiento de la herencia.
- En el diagrama UML, para mayor claridad, los atributos de tipo referencia aparecen indicados por duplicado, tanto en el compartimento de atributos de la clase, como en la flecha que señala la delegación que se produce.

APARTADO A

1) Implemente el orden natural ascendente de la clase Fecha. Escriba sólo aquellos elementos necesarios en la clase Fecha para dotar a la clase de dicho orden natural.

2) Implemente la clase **Abonado,** hija de la clase Persona, según las siguientes especificaciones:
   - El **constructor** toma 4 parámetros: n (nombre), a (apellidos), d (DNI) y (s) número de abonado; e inicializa los atributos con estos valores. La fecha de alta se inicializa a la fecha actual (que es proporcionada por el método estático *hoy()* de la clase Fecha), mientras que la fecha de baja se inicializa a *null* y el atributo *activo* a *true*.
   - El método **baja** procesa la baja de un abonado. Para dar a un abonado de baja, éste debe estar activo y la fecha de baja (que se recibe como parámetro), debe ser mayor que la fecha de alta del abonado. Si se dan estas condiciones, se establece el valor de la fecha de baja según el parámetro de entrada y se cambia el atributo *activo* a *false*. Si alguna de las condiciones no se cumple, debe mostrar un mensaje por pantalla indicado el error concreto.
   - El método **isActivo** es el método consultor estándar del atributo *activo*, y devuelve el valor de este.
   - El método **toString** devuelve una cadena con el siguiente formato, que puede variar según si el abonado está activo o no. Por ejemplo:

     ```
     #23 - Ramirez Lopez, Luis - 11111111A (Fecha Alta: 27/5/2020 - Fecha Baja: 1/6/2020)
     ```

     Donde los elementos sombrados son los valores de los atributos del objeto y teniendo en cuenta que el formato de la fecha es el que devuelve el método *toString* de la clase *Fecha*. Si el abonado no está activo, no se mostrará la fecha de baja. Por ejemplo:

     ```
     #2 - Lozano Alonso, Carmen - 22222222B (Fecha Alta: 27/5/2020)
     ```

APARTADO B

1) Implemente un criterio de orden (**ComparadorAntiguedadAbonado**) para objetos de tipo Abonado, que establezca un criterio de ordenación por el campo *activo* de manera que, dados dos abonados, si uno es activo y otro no, el activo será el menor de los dos. En caso de igualdad del campo *activo*, el orden será ascendente por *apellidos* y *nombre*, respectivamente, y sin diferenciar entre mayúsculas y minúsculas.

2) Implemente la excepción propia **AbonadoException**, con un único constructor que recibe como parámetro el mensaje a almacenar en la excepción cuando se eleve.

APARTADO C

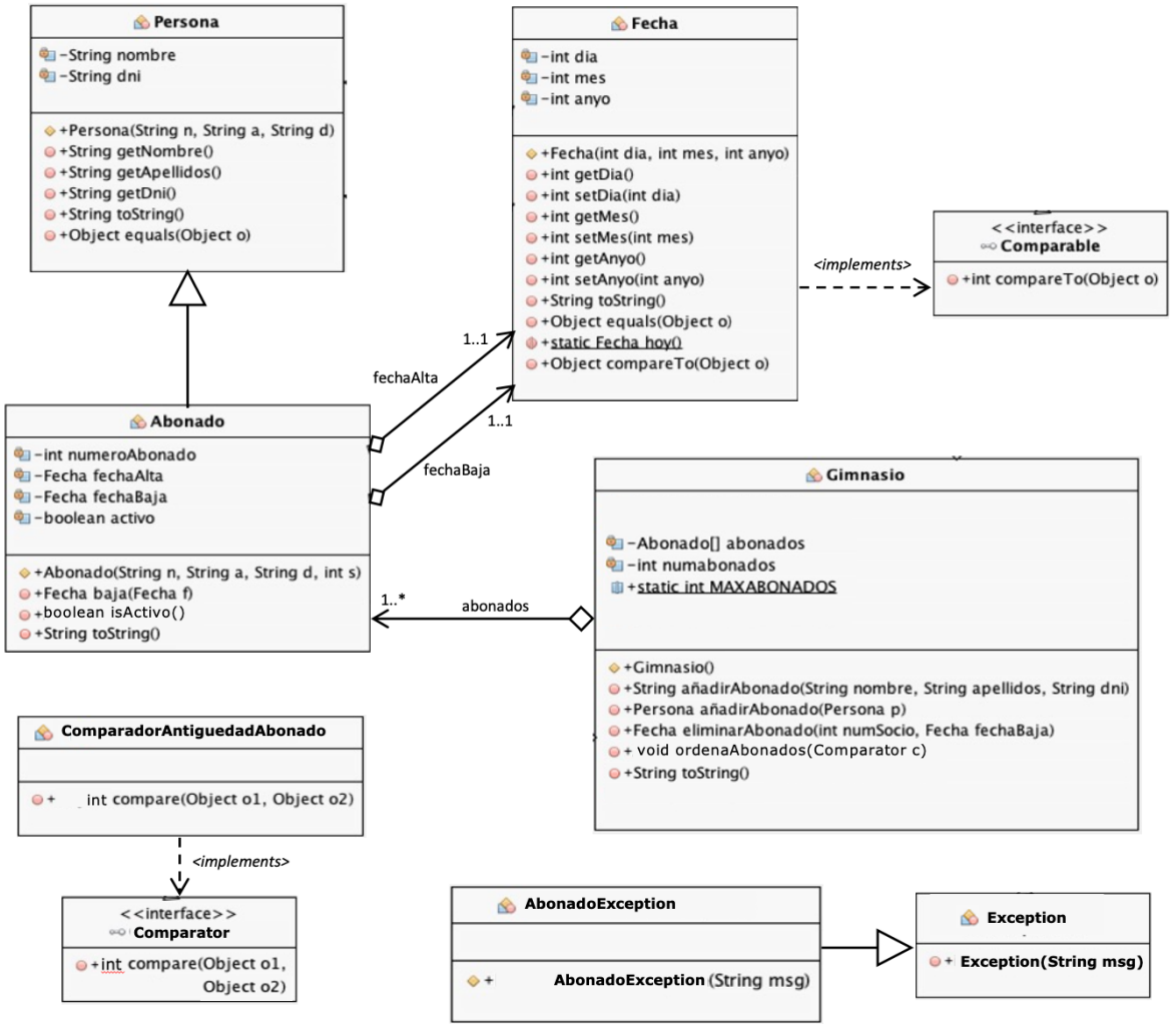
Implemente la clase **Gimnasio**, siguiendo el diagrama de clases y las siguientes especificaciones:

- La clase **Gimnasio** contiene una tabla unidimensional de objetos de tipo Abonado (*abonados*), una constante que determina el tamaño máximo de la tabla (*MAXABONADOS=100*) así como un atributo entero que indica los elementos válidos de la tabla (*numAbonados*).
- El constructor inicializa un Gimnasio vacío, es decir, sin elementos en la tabla.
- El método **añadirAbonado** está sobrecargado (véase el diagrama de clases) e inserta en la tabla abonados un nuevo abonado que se le pasa como parámetro de entrada, asignándole como número de abonado el siguiente entero al número de abonados (los abonados se numeran desde el 1 en adelante de manera correlativa). Si el abonado no existe en la tabla, se inserta al final de la tabla, y se actualiza el número de abonados válidos. Si el abonado ya existe en la tabla, se eleva una excepción de tipo **AbonadoException** con un mensaje indicativo. Esta excepción siempre debe ser propagada. El abonado (parámetros de entrada) puede venir expresado como un objeto Persona o con el nombre, apellidos y DNI por separado (observe la sobrecarga en el diagrama de clases).
- El método **eliminarAbonado** da de baja el abonado cuyo número de abonado recibe como parámetro, pero no lo borra de la tabla, sino que simplemente lo marca como no activo y establece la fecha de baja a la fecha que recibe como parámetro (observe el método *baja* de la clase *Abonado*). Note que, al no borrarse nunca un abonado de la tabla (aunque se dé de baja), el número de abonado debe coincidir con el índice en la tabla-1. Si el abonado a quien se pretende dar de baja no existe, se presentará un mensaje de error por pantalla.
- EL método **ordenaAbonados** recibe como parámetro un objeto de tipo *Comparator*, y ordena el vector de abonados según el criterio establecido por dicho comparador.
- El método **toString** devuelve una cadena con el siguiente formato:

```
* * * ABONADOS DEL GIMNASIO * * *
#1 - Lozano Alonso, Carmen - 22222222B (Fecha Alta: 23/5/2023)
#2 - Ramirez Lopez, Luis - 11111111A (Fecha Alta: 23/5/2023)
#3 - Martinez Garcia, Antonia - 33333333C (Fecha Alta: 23/5/2023)
```

APARTADO D

Implemente un programa principal que cree un Gimnasio con tres abonados, luego elimine (baja) uno de ellos, luego los ordene usando comparador *AbonadoException* definido en el apartado B y finalmente imprima lo por pantalla según el formato indicado en el apartado anterior. Capture las posibles excepciones, mostrando por pantalla el mensaje de dicha excepción.



Programación Orientada a Objetos- Curso 2022/2023
Grado en Ingeniería Informática en Sistemas de Información
EXAMEN - ENSEÑANZAD BÁSICAS — 24/05/2023

APARTADO A

PREGUNTA 1

```
public class Fecha implements Comparable{
...

    public int compareTo(Object o) {
       Fecha f = (Fecha)o;

        int cmp = anyo-f.anyo;
        if(cmp==0){
            cmp = mes-f.mes;
            if(cmp==0){
                cmp=dia-f.dia;
            }
        }
        return cmp;
    }
}
```

PREGUNTA 2

```
public class Abonado extends Persona {

    private int numeroAbonado;
    private Fecha fechaAlta;
    private Fecha fechaBaja;
    private boolean activo;

    public Abonado(String n, String a, String d, int s) {
        super(n, a, d);
        numeroAbonado = s;
        fechaAlta = Fecha.hoy();
        fechaBaja = null;
        activo = true;
    }

    public void baja(Fecha f) {
        if (isActivo()) {
            if (fechaAlta.compareTo(f) < 0) {
                fechaBaja = f;
                activo = false;
            } else {
                System.out.println("ERROR: La fecha de baja debe ser posterior a la de alta");
            }

        } else {
            System.out.println("ERROR: El asociado ya estaba dado de baja");
        }

    }

    public boolean isActivo() {
        return activo;
    }

    public String toString() {
        String s = "#" + numeroAbonado + " - " + super.toString() + " (Fecha Alta: " + fechaAlta;
        if (fechaBaja != null) {
            s += " - Fecha Baja: " + fechaBaja;

        }
        s += ")";
        return s;

    }

}
```

APARTADO B

PREGUNTA 1

```
import java.util.Comparator;

public class ComparadorAntiguedadAbonado implements Comparator {

    public int compare(Object o1, Object o2) {
        Abonado a1 = (Abonado) o1;
        Abonado a2 = (Abonado) o2;

        int cmp = 0;
        if (a1.isActivo() && !a2.isActivo()) {
            cmp = -1;
        } else if (!a1.isActivo() && a2.isActivo()) {
            cmp = 1;
        } else {
            cmp = a1.getApellidos().compareToIgnoreCase(a2.getApellidos());

            if (cmp == 0) {
                cmp = a1.getNombre().compareToIgnoreCase(a2.getNombre());
            }
        }

        return cmp;

    }

}
```

PREGUNTA 2

```
public class AbonadoException extends Exception{

    public AbonadoException(String m){
        super(m);
    }

}
```

APARTADO C

```
import java.util.Arrays;
import java.util.Comparator;

public class Gimnasio {

    private Abonado[] abonados;
    private int numabonados;
    public static int MAXABONADOS = 100;

    public Gimnasio() {
        abonados = new Abonado[MAXABONADOS];
        numabonados = 0;
    }

    public void añadirAbonado(String nombre, String apellidos, String dni) throws AbonadoException {
        Abonado s = new Abonado(nombre, apellidos, dni, numabonados + 1);
        int i;
        boolean enc;
        for (i = 0, enc = false; i < numabonados && !enc; i++) {
            if (abonados[i].equals(s)) {
                enc = true;
            }
        }
        if (enc) {
            throw new AbonadoException ("Este abonado ya existe");
        } else {
            abonados[numabonados] = s;
            numabonados++;
        }
    }

// Mejor llamar desde este método al anterior
    public void añadirAbonado(Persona p) throws AbonadoException {
       añadirAbonado(p.getNombre(), p.getApellidos(), p.getDni());
    }
```

```
    public void eliminarAbonado(int numSocio, Fecha fechaBaja) {
        if (numSocio > numabonados || numSocio < 0) {
            System.out.println("El socio número " + numSocio + " no existe.");
        } else {
            abonados[numSocio-1].baja(fechaBaja);
        }
    }

    public void ordenaAbonados(Comparator c){
        Arrays.sort(abonados, 0, numabonados, c);

    }

    public String toString(){
        String s = "* * * ABONADOS DEL GIMNASIO * * *%n";
        int i;

        for(i=0; i<numabonados;i++){
            s+=abonados[i]+"%n";
        }
       return s;
    }


}
```

APARTADO D

```
public class Principal {

    public static void main(String[] args) {
        Gimnasio g = new Gimnasio();

        try {
            g.añadirAbonado("Carmen", "Lozano Alonso", "22222222B");
            g.añadirAbonado(new Persona("Luis", "Ramirez Lopez", "11111111A"));
            g.añadirAbonado("Antonia", "Martinez Garcia", "33333333C");
            g.añadirAbonado("Carmen", "Lozano Alonso", "22222222B");
        } catch (AbonadoException e) {
            System.out.println(e.getMessage());

        }
        g.eliminarAbonado(2, new Fecha(28, 5, 2023));
        g.ordenaAbonados(new ComparadorAntiguedadAbonado());
        System.out.println(g);
    }
}
```