

Nombre y apellidos:

Fecha nacimiento:

Examen práctico (70% de la nota del examen)

EJERCICIO 1 (4 puntos)

En el Complejo Educativo de Cheste los estudiantes de primero de DAM se han cansado de no encontrar gente con quien jugar y han decidido crear una aplicación para buscar contrincantes de DAW y ASIR. Para ello, parten de los siguientes requisitos:

- De cada juego se tiene que saber su nombre, el número de jugadores, las copias de las que se dispone del mismo y si tenemos reglas o no. Estas reglas pueden estar en más de un idioma para un mayor entendimiento. También será necesario saber quién es el propietario del juego, y tipo de juego (rol, terror y miedo, Fantasía y magia, waregame...).
- De los jugadores necesitamos saber su nombre y apellidos, el teléfono para contactar con ellos si fuese necesario, la dirección y el correo electrónico. Cada jugador puede indicar en que juegos está interesado en participar.
- Las “quedadas” se organizan por algún jugador, siempre y cuando sea propietario de algún juego, indicando el juego al que se va a jugar, el lugar de la quedada, el día y la hora. Cuando algún jugador organiza una quedada, se envía un mensaje a todas las personas interesadas en el juego, las cuales pueden confirmar o no la asistencia a la quedada. El evento de completa cuando se tienen jugadores suficientes.
- Los jugadores pueden llevarse a casa las reglas del juego, en caso de que sea un juego bastante complejo y se necesite de una preparación previa.
- Para finalizar, también de dispondrá de una opción de “mercado”, donde los propietarios del juego lo podrán poner en venta siempre indicando el precio de venta y si se aceptan o no ofertas a la baja. El resto de jugadores de la plataforma podrán consultar todos los juegos que hay en venta y, si se aceptan ofertas, enviar ofertas por el juego, y el propietario podrá aceptarlas o no.
- La aplicación también tendrá un usuario administrador, encargado del mantenimiento de los juegos, los jugadores y las reglas.

EJERCICIO 2 (3 puntos)

Los alumnos de 1º de DAW han realizado una aplicación para verificar los datos de un array pero al no realizar las pruebas unitarias GitHub no deja subir el código ya que puede contener errores y puede perjudicar la existencia de la empresa. Crea las pruebas unitarias para cada uno de los métodos, incluido el constructor. El array para comprobar el funcionamiento contendrá vuestra fecha de nacimiento, por ejemplo para el 05/01/1987 será:

```
int[] datos = {0,5,0,1,1,9,8,7};
```

```
/**
 *
 * @author Antonio Camarena Ivars
 */
public class OpMatematicas {
    // campo llamado numeros que es una lista (o ArrayList en este caso) que contendrá objetos de tipo Integer.
    private java.util.ArrayList<Integer> numeros;

    /**
     * Constructor de OpMatematicas.
     *
     * @param datos arreglo con los números para la lista
     */

    public OpMatematicas(int[] datos) {
        // Verifica si el array de datos es nulo o si su longitud es igual a 0.
        if (datos == null || datos.length == 0) {
            // Si se cumple alguna de las condiciones anteriores, lanza la
            // excepción IllegalArgumentException -> indica que un método ha
            // recibido un argumento ilegal o inapropiado.
            throw new IllegalArgumentException();
        }
        // Inicializa un atributo "numeros" como una nueva instancia de ArrayList.
        this.numeros = new java.util.ArrayList<>();
        // Recorre cada elemento del array "datos"
        for (int elemento : datos) {
            // Agrega cada elemento del array dentro de "numeros".
            numeros.add(elemento);
        }
    }

    /**
     * @return la suma total de los números de la lista.
     */
    public int getSuma() {
        int suma = 0;
        for (int elemento : numeros) {
            suma += elemento;
        }
        return suma;
    }

    /**
     * @return el número mínimo de la lista.
     */
    public int getMinimo() {
        int min = Integer.MAX_VALUE;
        for (int elemento : numeros) {
            if (elemento < min) {
                min = elemento;
            }
        }
        return min;
    }
}
```

```
/**
 * @return el número máximo de la lista.
 */
public int getMaximo() {
    int max = Integer.MIN_VALUE;
    for (int elemento : numeros) {
        if (elemento > max) {
            max = elemento;
        }
    }
    return max;
}

/**
 * Retorna el índice de un número buscado.
 *
 * @param numero Número buscado.
 * @return Retorna la posición de un número dentro de la lista.
 * @throws java.util.NoSuchElementException Si el número no existe en la
 * lista.
 */
public int getIndiceDe(int numero) throws java.util.NoSuchElementException {
    int pos = 0;
    for (int elemento : numeros) {
        if (elemento == numero) {
            return pos;
        }
        pos++;
    }
    // Si el número no existe, lanzamos la excepción NoSuchElementException ->
    // indica que no se puede acceder a un elemento en una colección
    throw new java.util.NoSuchElementException(String.valueOf(numero));
}
```

1. Constructor (1 punto)

- Con un array a "null".
- Con una array vacío.

2. getMinimo (0,5 puntos)

- Comprueba que saca el número mínimo del array.

3. getMaximo (0,5 puntos)

- Comprueba que saca el número máximo del array.

4. getSuma (0,5 puntos)

- Comprueba que la suma total del array es correcta

5. getIndiceDe (0,5 puntos)

- Comprueba que con el índice 3 devuelva el número correcto (la posición del array)
- Prueba con el número 666, que no se encuentra en el array.

Resultado aproximado de las pruebas unitarias bien diseñadas.

✓ Default Suite	9 ms
✓ examen_dam_ed	9 ms
✓ OpMatematicasNGTest	9 ms
✓ testGetIndiceDe	6 ms
✓ testGetIndiceDe1	1 ms
✓ testMax	0 ms
✓ testMin	0 ms
✓ testOperacionesNumeros2	1 ms
✓ testOperacionesNumeros3	0 ms
✓ testSum	1 ms

Nota*: En el caso de tener que devolver una excepción, se deberá de implementar correctamente y lanzarla.*

Nota*: Recuerda que para sacar la máxima nota los test deben pasar correctamente y estar bien implementados.*

ENTREGA

EJERCICIO 1. Diagrama de clases mediante el programa Modelio.

- Hay que **exportar** el modelo con el formato **ejer1modelio_[Nombre_apellidos]** dentro de la carpeta **EJERCICIO_1**.
- Formato de imagen con el nombre **ejer1imagen_[Nombre_apellidos]** dentro de la carpeta **EJERCICIO_1**.

EJERCICIO 2. Pruebas unitarias

- Archivo con formato **Ejer2_procedimiento_[nombre_apellidos].txt** con el procedimiento para ejecutar las pruebas. Dentro de la carpeta **EJERCICIO_2**.
- Exportar el **proyecto completo desde IntelliJ IDEA** con el formato **Ejer2_Proyecto_[nombre_apellidos]**. Dentro de la carpeta **EJERCICIO_2**.