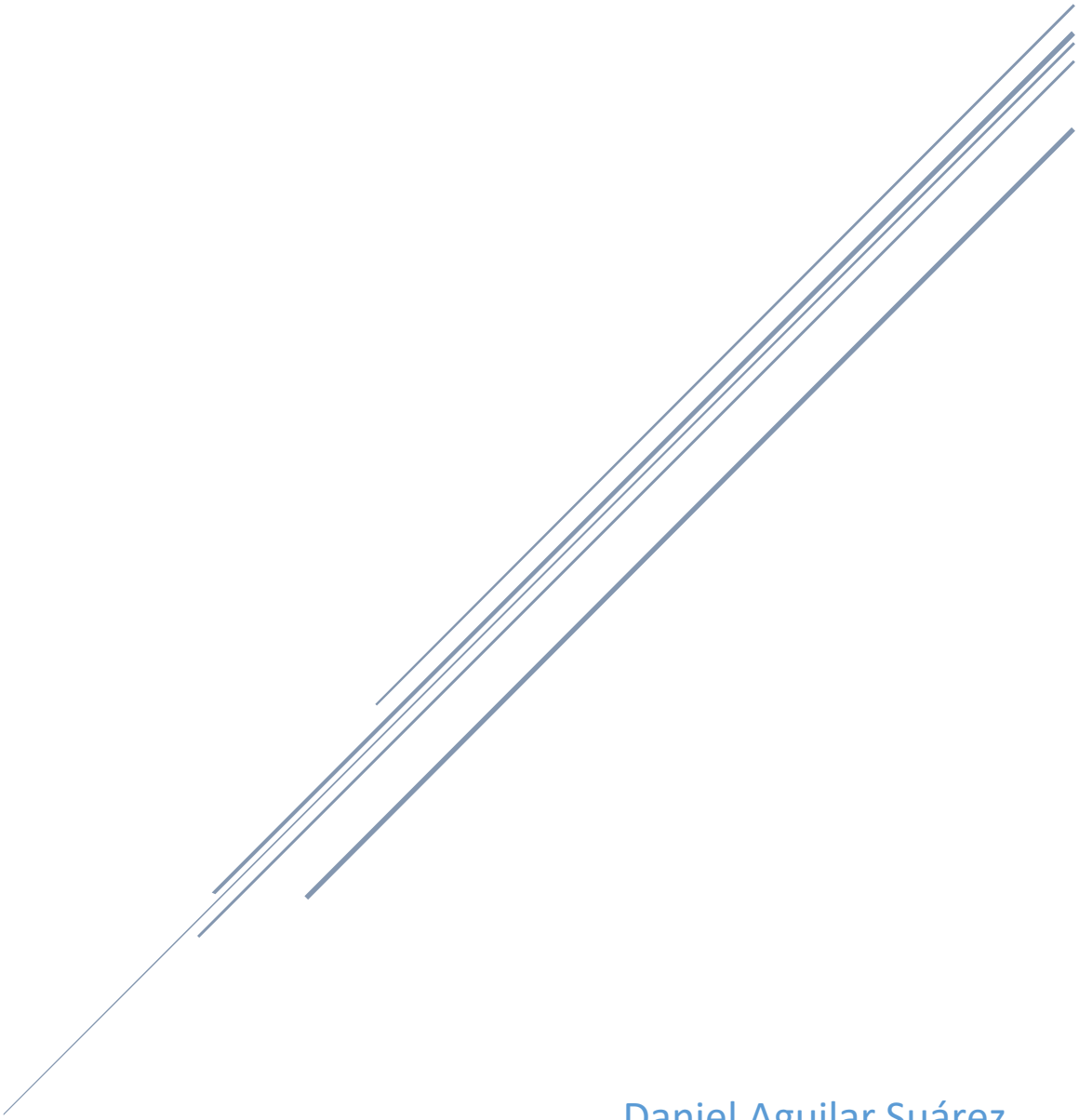
ESTRUCTURAS DE DATOS AVANZADAS Y CONTROL DE EXCEPCIONES

Programación II



Daniel Aguilar Suárez
DAW

Indice

Información del alumno

Nombre	Daniel
Apellidos	Aguilar Suárez
Módulo/Crédito	M03II
UF (solo ciclos LOE)	UF5
Título de la actividad	Estructuras de datos avanzadas y control de excepciones

ENUNCIADOS CREACIÓN DE CLASE

1. Crea una clase **Persona** con los siguientes campos: nombre , edad y dni. Se debe crear el constructor y los métodos get y set necesarios.
2. Crea una excepción propia **EdadIncorrecta** que se lance con el mensaje “La edad no puede ser menor a 1 ni mayor a 110” en el método **setEdad** de Persona cuando se intente establecer una edad menor a 1 o mayor a 110.
3. Crea una excepción propia **NombreIncorrecto** que se lance con el mensaje “El nombre debe tener mínimo 3 letras" o bien "El nombre no puede tener dígitos" en el método **setNombre** de Persona cuando el nombre tenga menos de 3 letras o contenga algún número respectivamente.
4. Crea una excepción propia **DniIncorrecto** que se lance con el mensaje "El DNI ha de ser una string de 9 valores" en el método **setDni** de Persona cuando el dni no sea una string de longitud 9 (no hace falta hacer más comprobaciones para el DNI).

CLASE PERSONA CREADA Y EXPLICADA

```
package domain;

import exceptions.DniIncorrecto;
import exceptions.EdadIncorrecta;
import exceptions.NombreIncorrecto;

public class Persona {
    private String das_nombre;
    private int das_edad;
    private String das_dni;

    // CONSTRUCTOR
    public Persona(String das_nombre, int das_edad, String das_dni) {
        this.das_nombre = das_nombre;
        this.das_edad = das_edad;
        this.das_dni = das_dni;
    }

    public Persona() {
    }

    // GETTERS & SETTERS
    public String getDas_nombre() {
        return this.das_nombre;
    }

    public void setDas_nombre(String das_nombre) throws NombreIncorrecto {
    // Comprueba si el nombre tiene menos de 3 letras, si tiene menos de 3
arroja una excepcion
        if (das_nombre.length()<3){
            throw new NombreIncorrecto("El nombre debe tener minimo 3
letras");
        } else if (tieneNumeros(das_nombre) == true){
            // Comprueba si en el nombre hay numeros, para ello se ayuda de
la funcion tieneNumeros()
            // si tiene arroja una excepcion
            throw new NombreIncorrecto("El nombre no puede tener dígitos");
        } else {
            this.das_nombre = das_nombre;
        }
    }

    public int getDas_edad() {
        return this.das_edad;
    }

    public void setDas_edad (int das_edad) throws EdadIncorrecta {
        // Comprueba si la edad es menor a 1 o mayor a 110, si es incorrecta
arroja una excepcion
        if (das_edad<1 || das_edad>110){
            throw new EdadIncorrecta("La edad no puede ser menor a 1 ni mayor
a 110");
        }else {
            this.das_edad = das_edad;
        }
```

```
    }

    public String getDas_dni() {
        return this.das_dni;
    }

    public void setDas_dni(String das_dni) throws DniIncorrecto {
        // Comprueba si el DNI tiene mas de 9 valores o menos de 9, si es
incorrecto arroja una excepcion
        if (das_dni.length()>9 || das_dni.length()<9) {
            throw new DniIncorrecto(("El DNI ha de ser una String de 9
valores"));
        }
        this.das_dni = das_dni;
    }

    // toString

    @Override
    public String toString() {
        return "Persona{" +
                "das_nombre='" + this.das_nombre + '\'' +
                ", das_edad=" + this.das_edad +
                ", das_dni='" + this.das_dni + '\'' +
                '}';
    }

    // METODOS
    // Comprueba si el nombre tiene números, usa un booleano para guardar las
respuestas.
    // pasa a un array de chars el String pasado por parámetro y lo recorre,
comprueba si es un dígito, y si
    // lo es, devuelve el true de la variable das_b. Si termina la ejecución
significa que en la cadena no tiene
    //  digitos asi que devuelve el false de la variable das_b
    public static boolean tieneNumeros(String cadena){
        boolean das_b = true;
        char[] das_v = cadena.toCharArray();
        for (int i = 0; i < das_v.length; i++) {
            if(Character.isDigit(das_v[i])){
                return das_b;
            }
        }
        das_b = false;
        return das_b;
        }
}
```

EXCEPCIONES

DniIncorrecto

```
package exceptions;

public class DniIncorrecto extends Exception{
    public DniIncorrecto(String message) {
        super(message);
    }
}
```

EdadIncorrecta

```
package exceptions;

public class EdadIncorrecta extends Exception {
    public EdadIncorrecta(String message) {
        super(message);
    }
}
```

NombreIncorrecto

```
package exceptions;

public class NombreIncorrecto extends Exception{
    public NombreIncorrecto(String message) {
        super(message);
    }
}
```

TEST CASE

Nombre incorrecto:

```
System.out.println("Daniel Aguilar Suarez");
Persona persona1 = new Persona("Dani",35,"49050958p");
// Pinto el estado actual de la persona creada
System.out.println(persona1.toString());
// Le añado un digito al nombre y compruebo la salida
persona1.setDas_nombre("Dan1a");
```

```
/Library/Java/JavaVirtualMachines/amazon-corretto-11.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea_rt.jar=49992:/Applications/
Daniel Aguilar Suarez
Persona{das_nombre='Dani', das_edad=35, das_dni='49050958p'}
Exception in thread "main" exceptions.NombreIncorrecto Create breakpoint : El nombre no puede tener dígitos
    at domain.Persona.setDas_nombre(Persona.java:34)
    at GestionaPersonas.main(GestionaPersonas.java:20)

Process finished with exit code 1
```

Edad incorrecta:

```
// Le añado un 0 a la edad y compruebo la salida
persona1.setDas_edad(0);
```

```
/Library/Java/JavaVirtualMachines/amazon-corretto-11.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea_rt.jar=50004:/Applications/
Daniel Aguilar Suarez
Persona{das_nombre='Dani', das_edad=35, das_dni='49050958p'}
Exception in thread "main" exceptions.EdadIncorrecta Create breakpoint : La edad no puede ser menor a 1 ni mayor a 110
    at domain.Persona.setDas_edad(Persona.java:47)
    at GestionaPersonas.main(GestionaPersonas.java:22)

Process finished with exit code 1
```

DNI incorrecto

```
// Le añado mas de 9 digitos al DNI y compruebo la salida
persona1.setDas_dni("1234567890");
```

```
/Library/Java/JavaVirtualMachines/amazon-corretto-11.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea_rt.jar=50011:/Applications/
Daniel Aguilar Suarez
Persona{das_nombre='Dani', das_edad=35, das_dni='49050958p'}
Exception in thread "main" exceptions.DniIncorrecto Create breakpoint : El DNI ha de ser una String de 9 valores
    at domain.Persona.setDas_dni(Persona.java:60)
    at GestionaPersonas.main(GestionaPersonas.java:24)

Process finished with exit code 1
```

EJERCICIO 5.a CREAR HASHMAP

Defina una **HashMap** de nombre "**trabajadores**" para almacenar un grupo de trabajadores según su DNI (String) y añada dos nuevas personas a "trabajadores" con valores establecidos por el programador.

```
HashMap<String,Persona> trabajadores = new HashMap<String,Persona>();
Persona trabajador1 = new Persona("Dani",35,"12345678p");
Persona trabajador2 = new Persona("Alicia",24,"87654321p");
trabajadores.put(trabajador1.getDas_dni(),trabajador1);
trabajadores.put(trabajador2.getDas_dni(),trabajador2);
```

TEST CASE

Recorriendo el HashMap:

```
System.out.println("Daniel Aguilar Suarez");
trabajadores.forEach((k,v) -> System.out.println("Key: " + k + ": Value: " +
v));
trabajadores.forEach((k,v) -> System.out.println(k + " -    " +
v.getDas_nombre() + " edad: " + v.getDas_edad()));
```

```
/Library/Java/JavaVirtualMachines/amazon-corretto-11.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea_rt.jar=50042:/Applications/
Daniel Aguilar Suarez
Key: 87654321p: Value: Persona{das_nombre='Alicia', das_edad=24, das_dni='87654321p'}
Key: 12345678p: Value: Persona{das_nombre='Dani', das_edad=35, das_dni='12345678p'}
87654321p -    Alicia edad: 24
12345678p -    Dani edad: 35

Process finished with exit code 0
```

EJERCICIO 5.b ARRAYLIST

Defina una **ArrayList** de nombre "**grupoTrabajo**" para almacenar las personas que formarán parte del grupo de trabajo.

```
ArrayList<Persona> grupoTrabajo = new ArrayList<Persona>();
```

TEST CASE

```
// TEST añadiendo personas al grupo de trabajo y recorriendolo
grupoTrabajo.add(trabajador1);
grupoTrabajo.add(trabajador2);
for (int i = 0; i < grupoTrabajo.size(); i++) {
    System.out.println("grupoTrabajo = " + grupoTrabajo);
}
```

```
/Library/Java/JavaVirtualMachines/amazon-corretto-11.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea_rt.jar=50061:/Applications/
grupoTrabajo = [Persona{das_nombre='Dani', das_edad=35, das_dni='12345678p'}, Persona{das_nombre='Alicia', das_edad=24, das_dni='87654321p'}]
grupoTrabajo = [Persona{das_nombre='Dani', das_edad=35, das_dni='12345678p'}, Persona{das_nombre='Alicia', das_edad=24, das_dni='87654321p'}]
```

EJERCICIO 5.c MENU

Muestre el siguiente menú:

- 1-Crea un trabajador
- 2- Borra un trabajador
- 3- Añade un trabajador al grupo de trabajo
- 4- Borra un trabajador del grupo de trabajo.
- Salir

```
// VARIABLES
boolean das_salir = false; // Usada para salir del bucle en el menu
String das_preg1 = "Introduce un valor entre 0 y 4"; // Usada para el case
// SELECCION
do {
    System.out.println("Daniel Aguilar Suarez");
    System.out.println("*********************************************");
    System.out.println("*************** Bienvenido *****************");
    Metodos.menu(); // Metodo menu muestra un array con todas las secciones
    switch (Metodos.pideEntero(das_preg1)) { // Metodo pideEntero es
reutilizable, controla la inserccion de enteros
        case 1:
            Persona nueva = Metodos.añadirTrabajador();
            trabajadores.put(nueva.getDas_dni(),nueva);
            Metodos.mostrarTrabajadores(trabajadores);
            Metodos.mostrarGrupo(grupoTrabajo);
            break;
        case 2:
            trabajadores.remove(Metodos.borrarPersona(trabajadores));
            Metodos.mostrarTrabajadores(trabajadores);
            Metodos.mostrarGrupo(grupoTrabajo);
            break;
        case 3:

            Metodos.mostrarTrabajadores(trabajadores);
            Metodos.mostrarGrupo(grupoTrabajo);

            break;
        case 4:

            Metodos.mostrarTrabajadores(trabajadores);
            Metodos.mostrarGrupo(grupoTrabajo);

            break;
        case 0:
            System.out.println("Gracias por usar el programa!");
            das_salir = true;
            break;
        default:
            System.out.println("Opción introducida invalida");
    }
} while (!das_salir);
```

METODO pideEntero

```
public static Integer pideEntero(String pregunta) {
    boolean das_salir = false;
    Integer das_entero = 0;
    Scanner scanner = new Scanner(System.in);
    do {
        try {
            System.out.println(pregunta);
            das_entero = scanner.nextInt();
            das_salir = true;
        } catch (Exception exc1) {
            System.out.println("Valor introducido invalido, vuelve a
intentarlo");
            scanner.next();
        }
    } while (!das_salir);
    return das_entero;
}
```

METODO menu

```
public static void menu() {
    String[] das_muestraMenu;
    das_muestraMenu = new String[7];
    das_muestraMenu[0] = "************* Qué quieres hacer? *************";
    das_muestraMenu[1] = "1. Añadir una persona";
    das_muestraMenu[2] = "2. Borrar una persona";
    das_muestraMenu[3] = "3. Añadir persona al grupo de trabajo";
    das_muestraMenu[4] = "4. Quitar persona del grupo de trabajo";
    das_muestraMenu[5] = "0. Salir";
    das_muestraMenu[6] = "*********************************************";

    for (String das_opcion : das_muestraMenu) {
        System.out.println(das_opcion);
    }
}
```

NOTA SOBRE EL MENÚ

Al terminar la práctica he decidido añadir al menú una opción más (Añadiéndola al Array en Métodos y al switch case) para mostrar únicamente los trabajadores y el grupo de trabajo, por que resulte mas cómodo de utilizar, por ello no aparecerá dicha opción en los pantallazos adjunto

TEST CASE

Se muestra el menú

```
/Library/Java/JavaVirtualMachines/amazon-corretto-11.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea_rt.jar=50092:/Applications/
Daniel Aguilar Suarez
**********************************************
*************** Bienvenido *****************
************* Qué quieres hacer? *************
1. Añadir una persona
2. Borrar una persona
3. Añadir persona al grupo de trabajo
4. Quitar persona del grupo de trabajo
0. Salir
**********************************************
Introduce un valor entre 0 y 4
```

Si no se introduce un Integer controla la excepción y pide de nuevo un valor

```
Introduce un valor entre 0 y 4
basdjkasbd
Valor introducido invalido, vuelve a intentarlo
Introduce un valor entre 0 y 4
```

EJERCICIO 5.d MOSTRAR LISTADO DENTRO DE TRABAJADORES Y GRUPO DE TRABAJO

Después de ejecutar cada opción se debe mostrar el listado de personas dentro de "**trabajadores**" y dentro de "**grupoTrabajo**".

```
switch (Metodos.pideEntero(das_preg1)) { // Metodo pideEntero es
reutilizable, controla la inserccion de enteros
    case 1:

        Metodos.mostrarTrabajadores(trabajadores);
        Metodos.mostrarGrupo(grupoTrabajo);
        break;
    case 2:

        Metodos.mostrarTrabajadores(trabajadores);
        Metodos.mostrarGrupo(grupoTrabajo);
        break;
    case 3:

        Metodos.mostrarTrabajadores(trabajadores);
        Metodos.mostrarGrupo(grupoTrabajo);

        break;
    case 4:

        Metodos.mostrarTrabajadores(trabajadores);
        Metodos.mostrarGrupo(grupoTrabajo);

        break;
    case 0:
        System.out.println("Gracias por usar el programa!");
        das_salir = true;
        break;
    default:
        System.out.println("Opción introducida invalida");
}
```

METODOS mostrarTrabajadores Y mostrarGrupo

```
// MOSTRAR TRABAJADORES Y GRUPOS DE TRABAJO
// Mostrar trabajadores, recibiendo por parametro un HashMap y recorriendolo
con un forEach
public static void mostrarTrabajadores(HashMap<String, Persona>
das_trabajadores){
    System.out.println("---Lista de trabajadores:---");
    das_trabajadores.forEach((k,v) -> System.out.println(k + " -   " +
v.getDas_nombre() + " edad: " + v.getDas_edad()));
}
// Mostrar grupo de trabajo, recibiendo por parametro un ArrayList y
recorriendolo con un for i
public static void mostrarGrupo(ArrayList<Persona> das_grupo) {
    System.out.println("---Grupo de trabajo:---");
    for (int i = 0; i < das_grupo.size(); i++) {
        System.out.println("["+i+"]  " + das_grupo.get(i).getDas_nombre() + "
" + "edad: " + das_grupo.get(i).getDas_edad());
    }
}
```

TEST CASE

Muestra el menú y al pulsar una opción muestra los trabajadores y grupos de trabajo.
Se han añadido dos personas al grupo de trabajo para hacer la prueba:

```
grupoTrabajo.add(trabajador1);
grupoTrabajo.add(trabajador2);
```

Resultado:

```
/Library/Java/JavaVirtualMachines/amazon-corretto-11.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea_rt.jar=50168:/Applications/
Daniel Aguilar Suarez
**********************************************
*************** Bienvenido ******************
************* Qué quieres hacer? *************
1. Añadir una persona
2. Borrar una persona
3. Añadir persona al grupo de trabajo
4. Quitar persona del grupo de trabajo
0. Salir
**********************************************
Introduce un valor entre 0 y 4
1
---Lista de trabajadores:---
87654321p -   Alicia edad: 24
12345678p -   Dani edad: 35
---Grupo de trabajo:---
[0]  Dani edad: 35
[1]  Alicia edad: 24
Daniel Aguilar Suarez
**********************************************
*************** Bienvenido ******************
```

EJERCICIO 5.e OPCION 1, CREAR UN TRABAJADOR

La opción "1-Crea un trabajador" debe de pedir los datos de un trabajador y añadirlo a "**trabajadores**" según su DNI. Si algún dato es incorrecto se debe capturar la excepción y volver a pedir el dato.

```
case 1:
    // Crea una nueva persona con el objeto recibido por el metodo
añadirTrabajador
    Persona nueva = Metodos.addTrabajador();
    // Añade la nueva persona al HashMap
    trabajadores.put(nueva.getDas_dni(),nueva);
    Metodos.mostrarTrabajadores(trabajadores); // Explicación en el metodo
    Metodos.mostrarGrupo(grupoTrabajo); // Explicación en el metodo
    break;
```

METODO addTrabajador

```
// CREAR TRABAJADOR
// Este metodo va recogiendo datos para crear un nuevo objeto de persona.
Cada dato esta controlado mediante
// Un try catch, cuando se introduce un dato incorrecto, se arrojara el
mensaje de su excepcion correspondiente
// y volverá a solicitar el dato de nuevo
public static Persona addTrabajador() throws NombreIncorrecto, EdadIncorrecta
{
    // VARIABLES Y CREACIÓN DEL OBJETO
    boolean das_salir = false;
    Scanner scanner = new Scanner(System.in);
    String das_nombre;
    String das_dni;
    Persona nueva = new Persona();
    int das_edad;

    // Introducir nombre
    System.out.println("Introduce nombre");
    do {
        das_nombre = scanner.next();
        try {
            nueva.setDas_nombre(das_nombre);
            das_salir = true;
        } catch (NombreIncorrecto ni){
            System.out.println(ni.getMessage()); // Mensaje de la excepcion
            System.out.println("Vuelve a introducirlo");
        }
    } while (!das_salir);

    // Introducir edad
    das_salir = false;
    do {
        das_edad = pideEntero("Introduce edad"); // Usando la funcion
pideEntero controla el valor
        try {
            nueva.setDas_edad(das_edad);
            das_salir = true;
        } catch (EdadIncorrecta ei) {
```

```
            System.out.println(ei.getMessage()); // Mensaje de la excepcion
            System.out.println("Vuelve a introducirla");
        }
    } while (!das_salir);

    // Introducir DNI
    das_salir = false;
    System.out.println("Introduce DNI");
    do {
        das_dni = scanner.next();
        try{
            nueva.setDas_dni(das_dni);
            das_salir = true;
        } catch (DniIncorrecto di) {
            System.out.println(di.getMessage()); // Mensaje de la excepcion
            System.out.println("Vuelve a introducirlo");
        }
    } while (!das_salir);

    // Retorno de la persona
    return nueva;
}
```

TEST CASE

Listado actual de trabajadores:

```
---Lista de trabajadores:---
87654321p -   Alicia edad: 24
12345678p -   Dani edad: 35
```

Añadir nuevo trabajador y prueba de excepciones:

```
Introduce nombre
da
El nombre debe tener minimo 3 letras
Vuelve a introducirlo
Leia
Introduce edad
0
La edad no puede ser menor a 1 ni mayor a 110
Vuelve a introducirla
30
Introduce DNI
adpkqj
El DNI ha de ser una String de 9 valores
Vuelve a introducirlo
12986576p
---Lista de trabajadores:---
87654321p -   Alicia edad: 24
12986576p -   Leia edad: 30
12345678p -   Dani edad: 35
---Grupo de trabajo:---
Daniel Aguilar Suarez
**********************************************
*************** Bienvenido ******************
************* Qué quieres hacer? *************
1. Añadir una persona
2. Borrar una persona
3. Añadir persona al grupo de trabajo
4. Quitar persona del grupo de trabajo
0. Salir
```

EJERCICIO 5.f OPCION 2, BORRAR UN TRABAJADOR

La opción "2- Borra un trabajador" debe de pedir el DNI del trabajador a borrar y borrarlo de "**trabajadores**" si existe (no hace falta borrarlo también de "grupoTrabajo").

```
case 2:
    // Borra una persona, se le pasa como parametro el HashMap
    Metodos.borrarTrabajador(trabajadores);
    Metodos.mostrarTrabajadores(trabajadores); // Explicación en el metodo
    Metodos.mostrarGrupo(grupoTrabajo); // Explicación en el metodo
    break;
```

METODO borrarTrabajador

```
// BORRAR TRABAJADOR
// Recibe por parametro el HashMap. Pide por consola el DNI de la persona, y
con el comprueba si esta en las Key
// del HashMap, si esta lo borra, si no muestra un mensaje de que no existe
public static void borrarTrabajador(HashMap<String,Persona> das_personaAnt) {
    Scanner scanner = new Scanner(System.in);
    String das_dni;
    System.out.println("Introduce el DNI de la persona");
    das_dni = scanner.next();
    if (das_personaAnt.containsKey(das_dni)) {
        das_personaAnt.remove(das_dni);
    } else {
        System.out.println("La persona no existe");
    }
}
```

TEST CASE

Borraremos el siguiente trabajador:
12345678p - Dani edad: 35

Si no existe:

```
Daniel Aguilar Suarez
*********************************************
*************** Bienvenido ******************
************ Qué quieres hacer? *************
1. Añadir una persona
2. Borrar una persona
3. Añadir persona al grupo de trabajo
4. Quitar persona del grupo de trabajo
0. Salir
*********************************************
Introduce un valor entre 0 y 4
2
Introduce el DNI de la persona
12349876o
La persona no existe
---Lista de trabajadores:---
87654321p -   Alicia edad: 24
12345678p -   Dani edad: 35
---Grupo de trabajo:---
```

Si el trabajador existe:

```
Daniel Aguilar Suarez
********************************************
*************** Bienvenido *****************
************* Qué quieres hacer? *************
1. Añadir una persona
2. Borrar una persona
3. Añadir persona al grupo de trabajo
4. Quitar persona del grupo de trabajo
0. Salir
********************************************
Introduce un valor entre 0 y 4
2
Introduce el DNI de la persona
12345678p
---Lista de trabajadores:---
87654321p -   Alicia edad: 24
---Grupo de trabajo:---
```

EJERCICIO 5.g OPCION 3, AÑADIR UN TRABAJADOR AL GRUPO DE TRABAJO

La opción “3- Añade un trabajador al grupo de trabajo” debe de pedir el DNI del trabajador a añadir en “grupoTrabajo”. Como máximo el ArrayList puede contener 2 personas, si se intenta añadir una tercera lanza una excepción propia del tipo **DemasiadoObjetos** y captúrala en el main.

```
case 3:
    // Añade un trabajador al grupo, uso try catch para controlarla desde el
main
    try {
        Metodos.addGroupTrabajador(grupoTrabajo,trabajadores); // Explicacion
en el metodo
    } catch (DemasiadosObjetos deOb) {
        System.out.println(deOb.getMessage());
    }
    Metodos.mostrarTrabajadores(trabajadores); // Explicación en el metodo
    Metodos.mostrarGrupo(grupoTrabajo); // Explicación en el metodo
    break;
```

METODO addGroupTrabajador

```
// AÑADIR TRABAJADOR AL GRUPO DE TRABAJO
// Metodo que recibe por parametro un ArrayList y un HashMap, después pide el
DNI. Primero comprobara si se ha
// alcanzado el maximo de trabajadores en el grupo, si es asi arroja la
excepcion de DemasiadosObjetos.
// Después comprueba si el trabajador existe, si no existe mostrara un
mensaje.
// Por último, añade el trabajador al grupo
public static void addGroupTrabajador (ArrayList<Persona> das_grupo ,
HashMap<String, Persona> das_trabajadores) throws DemasiadosObjetos {
    Scanner scanner = new Scanner(System.in);
    String das_dni;
    System.out.println("Introduce el DNI de la persona");
    das_dni = scanner.next();
    if (das_grupo.size() > 1) {
        throw new DemasiadosObjetos("Se ha alcanzado el limite de
trabajadores en el grupo de trabajo");
    } else if (!das_trabajadores.containsKey(das_dni)) {
        System.out.println("El trabajador no existe");
    } else {
        das_grupo.add(das_trabajadores.get(das_dni));
    }
}
```

EXCEPCION DemasiadosObjetos

```
package exceptions;

public class DemasiadosObjetos extends Exception {
    public DemasiadosObjetos(String message) {
        super(message);
    }
}
```

TEST CASE

Lista actual de trabajadores:

```
---Lista de trabajadores:---
37847625p -   prueba edad: 44
87654321p -   Alicia edad: 24
12345678p -   Dani edad: 35
```

Agregamos trabajadores al grupo:

```
Daniel Aguilar Suarez
**********************************************
*************** Bienvenido ******************
************* Qué quieres hacer? *************
1. Añadir una persona
2. Borrar una persona
3. Añadir persona al grupo de trabajo
4. Quitar persona del grupo de trabajo
0. Salir
**********************************************
Introduce un valor entre 0 y 4
3
Introduce el DNI de la persona
37847625p
---Lista de trabajadores:---
37847625p -   prueba edad: 44
87654321p -   Alicia edad: 24
12345678p -   Dani edad: 35
---Grupo de trabajo:---
[0]  prueba edad: 44
```

```
**********************************************
Introduce un valor entre 0 y 4
3
Introduce el DNI de la persona
87654321p
---Lista de trabajadores:---
37847625p -   prueba edad: 44
87654321p -   Alicia edad: 24
12345678p -   Dani edad: 35
---Grupo de trabajo:---
[0]  prueba edad: 44
[1]  Alicia edad: 24
```

Controlamos que falle al añadir otro mas:

```
*****************************************
Introduce un valor entre 0 y 4
3
Introduce el DNI de la persona
12345678p
Se ha alcanzado el limite de trabajadores en el grupo de trabajo
---Lista de trabajadores:---
37847625p -   prueba edad: 44
87654321p -   Alicia edad: 24
12345678p -   Dani edad: 35
---Grupo de trabajo:---
[0]  prueba edad: 44
[1]  Alicia edad: 24
Daniel Aguilar Suarez
*****************************************
```

Control al intentar añadir un trabajador que no existe:

```
Daniel Aguilar Suarez
*********************************************
**************** Bienvenido ******************
************* Qué quieres hacer? *************
1. Añadir una persona
2. Borrar una persona
3. Añadir persona al grupo de trabajo
4. Quitar persona del grupo de trabajo
0. Salir
*********************************************
Introduce un valor entre 0 y 4
3
Introduce el DNI de la persona
67345609o
El trabajador no existe
---Lista de trabajadores:---
87654321p -   Alicia edad: 24
12345678p -   Dani edad: 35
---Grupo de trabajo:---
```

EJERCICIO 5.h OPCION 4, BORRAR UN TRABAJADOR DEL GRUPO DE TRABAJO

La opción "4- Borra un trabajador del grupo de trabajo." debe pedir el índice dentro de la ArrayList y quitar ésa persona de "**grupoTrabajo"** pero no de "**trabajadores**".Si el usuario indica un índice fuera del rango posible, lanza desde la función de borrar una persona, una excepción PosicionIncorrecta y captúrala en el main.

```
case 4:
    // Quita un trabajador del grupo, uso try catch para controlarlo desde el
main
    try {
        int das_index = Metodos.borrarGroupTrabajador(grupoTrabajo); //
Explicacion en el metodo
        grupoTrabajo.remove(das_index); // Borrado del index recogido
    } catch (PosicionIncorrecta pos) {
        System.out.println(pos.getMessage()); // Mensaje de la excepción
    }
    Metodos.mostrarTrabajadores(trabajadores);
    Metodos.mostrarGrupo(grupoTrabajo);
    break;
```

METODO borrarGroupTrabajador

```
// BORRAR UN TRABAJADOR DEL GRUPO DE TRABAJO
// Metodo que recibe  un ArrayList, primero almacena en una variable de
indice la posición controlada por
// la función pideEntero. Si se ha introducido un valor permitido, comprueba
si esta fuera del tamaño del ArrayList
// En caso de estar mal, arroja la excepción PosicionIncorrecta.
// Si no hay problemas, devuelve el indice para su tratamiento desde el main
public static Integer borrarGroupTrabajador (ArrayList<Persona> das_grupo)
throws PosicionIncorrecta {
    Integer das_indice;
    das_indice = pideEntero("Indica el indice de un trabajador para
eliminarlo del grupo de trabajo");
    if (das_indice >= das_grupo.size() || das_indice < 0) {
        throw new PosicionIncorrecta("La posición indicada es incorrecta");
    }
    return das_indice;
}
```

EXCEPCIÓN PosicionIncorrecta

```
package exceptions;

public class PosicionIncorrecta extends Exception{
    public PosicionIncorrecta(String message) {
        super(message);
    }
}
```

TEST CASE

Teniendo a los siguientes trabajadores y personas en el grupo de trabajo:

```
---Lista de trabajadores:---
34876256p -   prueba edad: 45
87654321p -   Alicia edad: 24
12345678p -   Dani edad: 35
---Grupo de trabajo:---
[0]  Alicia edad: 24
[1]  Dani edad: 35
```

Quitamos el número 0:

```
Daniel Aguilar Suarez
**********************************************
*************** Bienvenido ******************
************* Qué quieres hacer? *************
1. Añadir una persona
2. Borrar una persona
3. Añadir persona al grupo de trabajo
4. Quitar persona del grupo de trabajo
0. Salir
**********************************************
Introduce un valor entre 0 y 4
4
Indica el indice de un trabajador para eliminarlo del grupo de trabajo
0
---Lista de trabajadores:---
34876256p -   prueba edad: 45
87654321p -   Alicia edad: 24
12345678p -   Dani edad: 35
---Grupo de trabajo:---
[0]  Dani edad: 35
Daniel Aguilar Suarez
**********************************************
```

Si intentamos quitar el índice 1 (que no existe):

```
---Lista de trabajadores:---
87654321p -   Alicia edad: 24
12345678p -   Dani edad: 35
---Grupo de trabajo:---
[0]  Dani edad: 35
Daniel Aguilar Suarez
**********************************************
*************** Bienvenido ******************
************* Qué quieres hacer? *************
1. Añadir una persona
2. Borrar una persona
3. Añadir persona al grupo de trabajo
4. Quitar persona del grupo de trabajo
0. Salir
**********************************************
Introduce un valor entre 0 y 4
4
Indica el indice de un trabajador para eliminarlo del grupo de trabajo
1
La posición indicada es incorrecta
```