Contents

1 Documentación da Clase `ContactosDAO`

1.1 O que é esta clase?

Imaxina que esta clase é coma un **caixón organizador** onde gardas todos os contactos da túa axenda. Cada contacto (con nome, teléfono, email, etc.) ten unha posición específica neste caixón. A clase `ContactosDAO` é a encargada de: - Gardar novos contactos - Buscar contactos existentes - Eliminar contactos - Organizar todo de forma ordenada

O termo **DAO** significa *Data Access Object* - é un patrón de deseño que separa a forma de gardar os datos do resto do programa. Así, se algún día quixésemos gardar os contactos nunha base de datos no canto dun array, só teríamos que cambiar esta clase, non todo o programa!

1.2 Como se crea unha axenda?

Primeiro temos que crear o "caixón" con espazo para os contactos:

```java
// Creamos unha axenda para 10 contactos
ContactosDAO axenda = new ContactosDAO(10);
```

O número 10 indica o **máximo de contactos** que poderemos gardar. Unha vez creado, este tamaño non se pode cambiar.

> **Atención para principiantes**: Se intentas gardar máis contactos do que o tamaño permitido, o programa dará un erro (`StorageFull`).

1.3 Métodos Principais Explicados

1.3.1 Gardar un contacto novo

```java
axenda.guardarContacto(unContacto);
```

O que fai: - Comproba se o contacto xa existe (para non ter duplicados) - Busca un espazo libre no array - Asigna un número de posición ao contacto - Garda o contacto nesa posición

Posibles erros: - `StorageFull`: Cando non hai espazo para máis contactos - `DuplicateElement`: Cando o contacto xa existe na axenda

Exemplo completo:

```java
try {
    Persoa persoa = new Persoa("12345678Z", "María", "López");
    Contacto contacto = new Contacto(persoa, "612345678", "maria@email.com", "Amiga");
    axenda.guardarContacto(contacto);
    System.out.println("Contacto gardado correctamente!");
} catch (StorageFull e) {
    System.out.println("Erro: A axenda está chea!");
} catch (DuplicateElement e) {
    System.out.println("Erro: Ese contacto xa existe!");
}
```

1.3.2 Buscar contactos

A clase permite buscar contactos de varias formas:

```java
Contacto[] resultados = axenda.getContactoNif("12345678Z");
```

1.3.2.1 Buscar por NIF

O que fai: Busca todos os contactos que teñan esa persoa asociada.

```
Contacto[] resultados = axenda.getContactoPhone("612345678");
```

1.3.2.2 Buscar por teléfono

O que fai: Busca todos os contactos con ese número de teléfono.

```
Contacto[] resultados = axenda.getContactoEmail("email@example.com");
```

1.3.2.3 Buscar por email

O que fai: Busca todos os contactos con ese enderezo de email.

```
Contacto[] resultados = axenda.getContactoDescripcion("amigo");
```

1.3.2.4 Buscar por descrición

O que fai: Busca todos os contactos que teñan esa palabra na súa descrición (busca sen distinguir maiúsculas/minúsculas).

Como usar os resultados:

```
Contacto[] resultados = axenda.getContactoNif("12345678Z");

if (resultados == null) {
    System.out.println("Non se atopou ningún contacto con ese NIF");
} else {
    System.out.println("Atopados " + resultados.length + " contactos:");
    for (Contacto c : resultados) {
        System.out.println("- " + c.getPersoa().getNome() + " " + c.getPhone());
    }
}
```

1.3.3 Obter un contacto específico

```
Contacto contacto = axenda.getContactoNumero(0); // Obtén o contacto da posición 0
```

O que fai: Devolve o contacto que está na posición indicada (comeza dende 0).

Importante: Se a posición non ten ningún contacto, devolve `null`.

1.3.4 Eliminar contactos

```
String resultado = axenda.eliminarContacto(0); // Elimina o contacto da posición 0
```

1.3.4.1 Eliminar un contacto específico

O que fai: Elimina o contacto da posición indicada e coloca `null` nesa posición.

Retorno: - `"Contacto Eliminado"` se todo foi ben - `null` se non había ningún contacto nesa posición

```
axenda.eliminarContactos();
```

1.3.4.2 Eliminar todos os contactos

O que fai: Borra todos os contactos da axenda, deixando todas as posicións baleiras.

1.3.5 Obter todos os contactos

```
Contacto[] todos = axenda.getAxenda();
```

O que fai: Devolve un array con todos os contactos que hai na axenda (só os que non están baleiros).

Como usalo:

```
Contacto[] todos = axenda.getAxenda();

if (todos == null) {
    System.out.println("A axenda está baleira");
} else {
    System.out.println("Contactos na axenda: " + todos.length);
    for (Contacto c : todos) {
        System.out.println(c.getPersoa().getNome() + " - " + c.getPhone());
    }
}
```

1.4 Estrutura Interna (Para Entender Mellor)

1.4.1 O Array de Contactos

```
private Contacto axenda[];
```

- É un array (lista ordenada) onde se gardan os contactos
- Cada posición pode ter un contacto ou estar baleira (`null`)
- O tamaño é fixo e defínese no construtor

1.4.2 Métodos Privados (Para uso interno)

Estes métodos non podes chamalos dende fóra da clase, pero son importantes para o funcionamento:

1.4.2.1 `exists(Contacto contacto)`

- **Propósito**: Comprobar se un contacto xa existe na axenda
- **Como funciona**: Recorre todo o array buscando coincidencias

1.4.2.2 `getPosicion()`

- **Propósito**: Atopar a primeira posición libre no array
- **Como funciona**: Busca a primeira posición que teña `null`
- **Erro posible**: `StorageFull` se non hai espazo libre

1.5 Excepcións que Debemos Coñecer

1.5.1 `StorageFull`

- **Cando ocorre**: Cando a axenda está chea e non hai espazo para máis contactos
- **Como evitar**: Comprobar o tamaño antes de gardar ou usar un tamaño maior ao crear a axenda

1.5.2 `DuplicateElement`

- **Cando ocorre**: Cando intentamos gardar un contacto que xa existe
- **Como evitar**: Comprobar antes se o contacto xa está na axenda

---

1.6 Boas Prácticas para Principiantes

1.6.1 1. Sempre manexa as excepcións

Nunca deixes sen controlar as excepcións que poden lanzar os métodos:

```
try {
    axenda.guardarContacto(contacto);
} catch (StorageFull e) {
    System.out.println("A axenda está chea!");
} catch (DuplicateElement e) {
    System.out.println("Ese contacto xa existe!");
}
```

1.6.2 2. Comproba os resultados nulos

Cando buscas contactos, sempre comproba se o resultado é `null`:

```
Contacto[] resultados = axenda.getContactoNif("12345678Z");
if (resultados != null) {
    // Hai resultados
} else {
    // Non hai resultados
}
```

1.6.3 3. Usa os métodos correctos para cada tarefa

- Para buscar por características específicas: `getContactoNif()`, `getContactoPhone()`, etc.
- Para obter un contacto específico: `getContactoNumero()`
- Para obter todos: `getAxenda()`

1.7 Limitacións Importantes (Para que o Saibas)

1. **Tamaño fixo**: Unha vez creada a axenda, non se pode cambiar o seu tamaño
2. **Sen persistencia**: Os datos pérdense cando se pecha o programa
3. **Rendemento**: Canto máis contactos teña a axenda, máis lento será buscar e gardar
4. **Sen ordenación**: Os contactos devólvense na orde en que foron gardados, non ordenados alfabeticamente

---

1.8 Exemplo Completo de Uso

```
public class ExemploAxenda {
    public static void main(String[] args) {
        // 1. Creamos a axenda para 5 contactos
        ContactosDAO axenda = new ContactosDAO(5);

        try {
            // 2. Creamos e gardamos contactos
            Persoa p1 = new Persoa("12345678Z", "Ana", "García");
            Contacto c1 = new Contacto(p1, "612345678", "ana@email.com", "Colega");
            axenda.guardarContacto(c1);

            Persoa p2 = new Persoa("87654321X", "Carlos", "Rodríguez");
            Contacto c2 = new Contacto(p2, "698765432", "carlos@email.com", "Traballo");
```

```
            axenda.guardarContacto(c2);

            System.out.println("Contactos gardados correctamente!");

            // 3. Buscamos por NIF
            Contacto[] resultados = axenda.getContactoNif("12345678Z");
            if (resultados != null) {
                System.out.println("\nContactos de Ana:");
                for (Contacto c : resultados) {
                    System.out.println("- Teléfono: " + c.getPhone());
                    System.out.println("- Email: " + c.getEmail());
                    System.out.println("- Descrición: " + c.getDescripcion());
                }
            }

            // 4. Eliminamos un contacto
            String eliminado = axenda.eliminarContacto(0);
            if (eliminado != null) {
                System.out.println("\n" + eliminado);
            }

            // 5. Amosamos todos os contactos restantes
            Contacto[] todos = axenda.getAxenda();
            System.out.println("\nContactos restantes: " + (todos != null ? todos.length : 0

        } catch (StorageFull e) {
            System.err.println("Erro: " + e.getMessage());
        } catch (DuplicateElement e) {
            System.err.println("Erro: " + e.getMessage());
        }
    }
}
```

Saída esperada:

```
Contactos gardados correctamente!

Contactos de Ana:
- Teléfono: 612345678
- Email: ana@email.com
- Descrición: Colega

Contacto Eliminado

Contactos restantes: 1
```

1.9 Consellos Finais para Principiantes

Paso a paso: Non intentes entender todo á vez. Primeiro apréndete os métodos básicos (`guardarContacto`, `getAxenda`), despois os máis complexos.

Practica con exemplos pequenos: Crea un programa simple que só garde 2-3 contactos e os amose.

Manexa ben os erros: Os principiantes adoitan esquecer as excepcións. Lembra que `guardarContacto` pode fallar!

Pensa en tamaño: Se non estás seguro de cantos contactos terás, pon un tamaño maior do necesario (por exemplo, 100 no canto de 10).

Pregunta: Se algo non queda claro, pregunta! Esta clase ten conceptos importantes como arrays, excepcións e encapsulamento.

Esta clase é un excelente exemplo de como organizar o código para que sexa reutilizable e fácil de manter. Cando entendás ben o seu funcionamento, terás dados un gran paso na túa aprendizaxe de Java!