

- Actividades de comprobación. Realizarlas todas. Copiar todas las preguntas y sus respuestas correctas.

8.1. Sobre una subclase es correcto afirmar que:

- a) Tiene menos atributos que su superclase.
- b) Tiene menos miembros que su superclase.
- c) Hereda los miembros no privados de su superclase.
- d) Hereda todos los miembros de su superclase.

C

8.2. En relación con las clases abstractas es correcto señalar que:

- a) Implementan todos sus métodos.
- b) No implementan ningún método.
- c) No tienen atributos.
- d) Tienen algún método abstracto.

D

8.3. ¿En qué consiste la sustitución u overriding?

- a) En sustituir un método heredado por otro implementado en la propia clase.
- b) En sustituir un atributo por otro del mismo nombre.
- c) En sustituir una clase por una subclase.
- d) En sustituir un valor de una variable por otro.

A

8.4. Sobre la clase `Object` es cierto indicar que:

- a) Es abstracta.
- b) Hereda de todas las demás.
- c) Tiene todos sus métodos abstractos.
- d) Es superclase de todas las demás clases.

D

8.5. ¿Cuál de las siguientes afirmaciones sobre el método `equals()` es correcta?

- a) Hay que implementarlo, ya que es abstracto.
- b) Sirve para comparar solo objetos de la clase `Object`.
- c) Se hereda de `Object`, pero debemos reimplementarlo al definirlo en una clase.
- d) No hay que implementarlo, ya que se hereda de `Object`.

C

8.6. ¿Cuál de las siguientes afirmaciones sobre el método `toString()` es correcta?

- a) Sirve para mostrar la información que nos interesa de un objeto.
- b) Convierte automáticamente un objeto en una cadena.
- c) Encadena varios objetos.
- d) Es un método abstracto de `Object` que tenemos que implementar.

D

8.7. ¿Cuál de las siguientes afirmaciones sobre el método `getClass()` es correcta?

- a) Convierte los objetos en clases.
- b) Obtiene la clase a la que pertenece un objeto.
- c) Obtiene la superclase de una clase.
- d) Obtiene una clase a partir de su nombre.

B

8.8. Una clase puede heredar:

- a) De una clase.
- b) De dos clases.
- c) De todas las clases que queramos.
- d) Solo de la clase `Object`.

A

**8.9. La selección dinámica de métodos:**

- a) Se produce cuando una variable cambia de valor durante la ejecución de un programa.
- b) Es el cambio de tipo de una variable en tiempo de ejecución.
- c) Es la asignación de un mismo objeto a más de una variable en tiempo de ejecución.
- d) Es la ejecución de distintas implementaciones de un mismo método, asignando objetos de distintas clases a una misma variable, en tiempo de ejecución.

D

**8.10. ¿Cuál de las siguientes afirmaciones sobre el método `super()` es correcta?**

- a) Sirve para llamar al constructor de la superclase.
- b) Sirve para invocar un método escrito más arriba en el código.
- c) Sirve para llamar a cualquier método de la superclase.
- d) Sirve para hacer referencia a un atributo de la superclase.

A

- Actividades de aplicación. Realizar las siguientes 8.11, 8.12, 8.14, 8.15, 8.16, 8.17.

**8.11.** Crea la clase `Campana` que hereda de `Instrumento` (definida en la Actividad resuelta 8.4).

```
1 package com.mycompany.aa8_11;
2
3 class Campana extends Instrumento{
4
5     public Campana() {
6         super();
7     }
8
9     @Override
10    public void interpretar() {
11        for (Nota nota : melodia) {
12            switch (nota) {
13                case DO:
14                    System.out.print(": plin-do ");
15                    break;
16                case RE:
17                    System.out.print(": plin-re ");
18                    break;
19                case MI:
20                    System.out.print(": plin-mi ");
21                    break;
22                case FA:
23                    System.out.print(": plin-fa ");
24                    break;
25                case SOL:
26                    System.out.print(": plin-sol ");
27                    break;
28                case LA:
29                    System.out.print(": plin-la ");
30                    break;
31                case SI:
32                    System.out.print(": plin-si ");
33                    break;
34            }
35        }
36        System.out.println(": ");
37    }
38 }
39
```

com.mycompany.aa8\_11.Campana > interpretar > for (Nota nota : melodia) > switch (nota) > case SI: >

Output - Run (Aa8\_11) x

```
--- exec:3.1.0:exec (default-cli) @ Aa8_11 ---
plin-do plin-si plin-sol plin-re plin-fa
-----
BUILD SUCCESS
-----
```

**8.12.** Las empresas de transporte, para evitar daños en los paquetes, embalan todas sus mercancías en cajas con el tamaño adecuado. Una caja se crea expresamente con un ancho, un alto y un fondo y, una vez creada, se mantiene inmutable. Cada caja lleva pegada una etiqueta, de un máximo de 30 caracteres, con información útil como el nombre del destinatario, dirección, etc. Implementa la clase `Caja` con los siguientes métodos:

- `Caja(int ancho, int alto, int fondo, Unidad unidad)`: que construye una caja con las dimensiones especificadas, que pueden encontrarse en «cm» (centímetros) o «m» (metros).
- `double getVolumen()`: que devuelve el volumen de la caja en metros cúbicos.
- `void setEtiqueta(String etiqueta)`: que modifica el valor de la etiqueta de la caja.
- `String toString()`: que devuelve una cadena con la representación de la caja.

```
1 package com.mycompany.aa8_12;
2
3 class Caja {
4     final int alto, ancho, fondo;
5     String etiqueta; // no puede ser privado porque en el main se accede a el
6     //atributo directamente sin el getter
7     Caja(int ancho, int alto, int fondo, Unidad unidad) {
8         this.alto = alto;
9         this.ancho = ancho;
10        this.fondo = fondo;
11    }
12
13    double getVolumen(){
14        return alto * ancho * fondo;
15    }
16
17    void setEtiqueta(String etiqueta) {
18        if( etiqueta.length() <= 30){
19            this.etiqueta = etiqueta;
20        }
21        else {System.out.println(x: "ERROR: Etiqueta demasiado larga!!");}
22    }
23
24    @Override
25    public String toString() {
26        return "\n--Caja: " + etiqueta + " \nMedidas: alto=" + alto + ", ancho="
27            + ancho + ", fondo=" + fondo;
28    }
29 }
30
```

com.mycompany.aa8\_12.Caja > toString >

Output - Run (Aa8\_12) x

--- exec:3.1.0:exec (default-cli) @ Aa8\_12 ---

6.0

45.0

Monitor

Raton

--Caja: Monitor

Medidas: alto=2, ancho=1, fondo=3

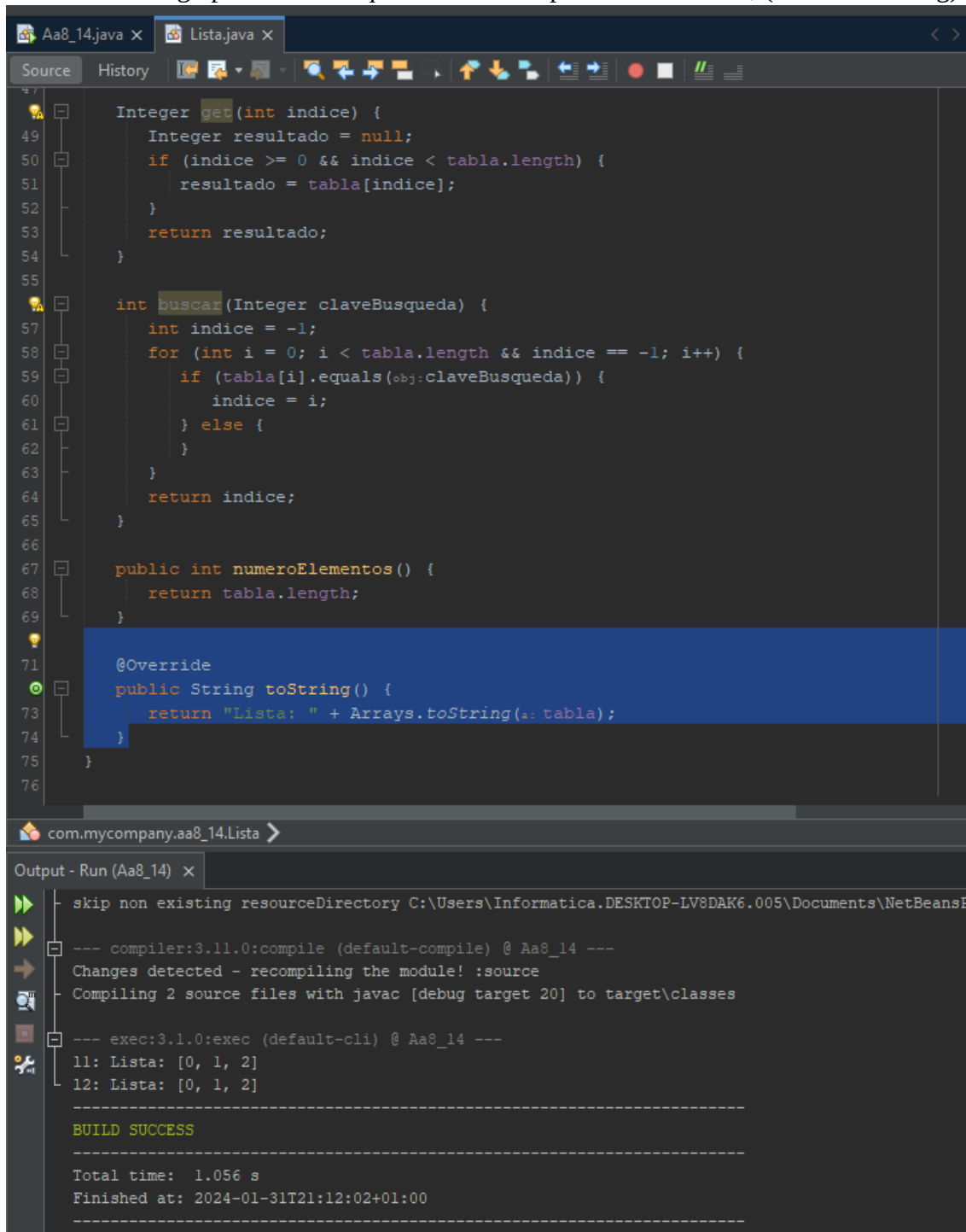
--Caja: Raton

Medidas: alto=3, ancho=3, fondo=5

-----  
BUILD SUCCESS  
-----

**8.14.** Reimplementa la clase `Lista` de la Actividad resuelta 7.11, sustituyendo el método `mostrar()` por el método `toString()`.

No se ve todo el código pero lo único que cambia es la parte seleccionada, (método `toString()`)



The screenshot shows an IDE with two tabs: `Aa8_14.java` and `Lista.java`. The `Lista.java` tab is active, displaying the source code of the `Lista` class. The code includes methods `get`, `buscar`, `numeroElementos`, and `toString`. The `toString` method is highlighted in blue. Below the code editor, the `Output - Run (Aa8_14)` window shows the execution results, including compilation and execution messages, and the output of the `toString` method.

```
Integer get(int indice) {
    Integer resultado = null;
    if (indice >= 0 && indice < tabla.length) {
        resultado = tabla[indice];
    }
    return resultado;
}

int buscar(Integer claveBusqueda) {
    int indice = -1;
    for (int i = 0; i < tabla.length && indice == -1; i++) {
        if (tabla[i].equals(obj:claveBusqueda)) {
            indice = i;
        } else {
        }
    }
    return indice;
}

public int numeroElementos() {
    return tabla.length;
}

@Override
public String toString() {
    return "Lista: " + Arrays.toString(a: tabla);
}

}
```

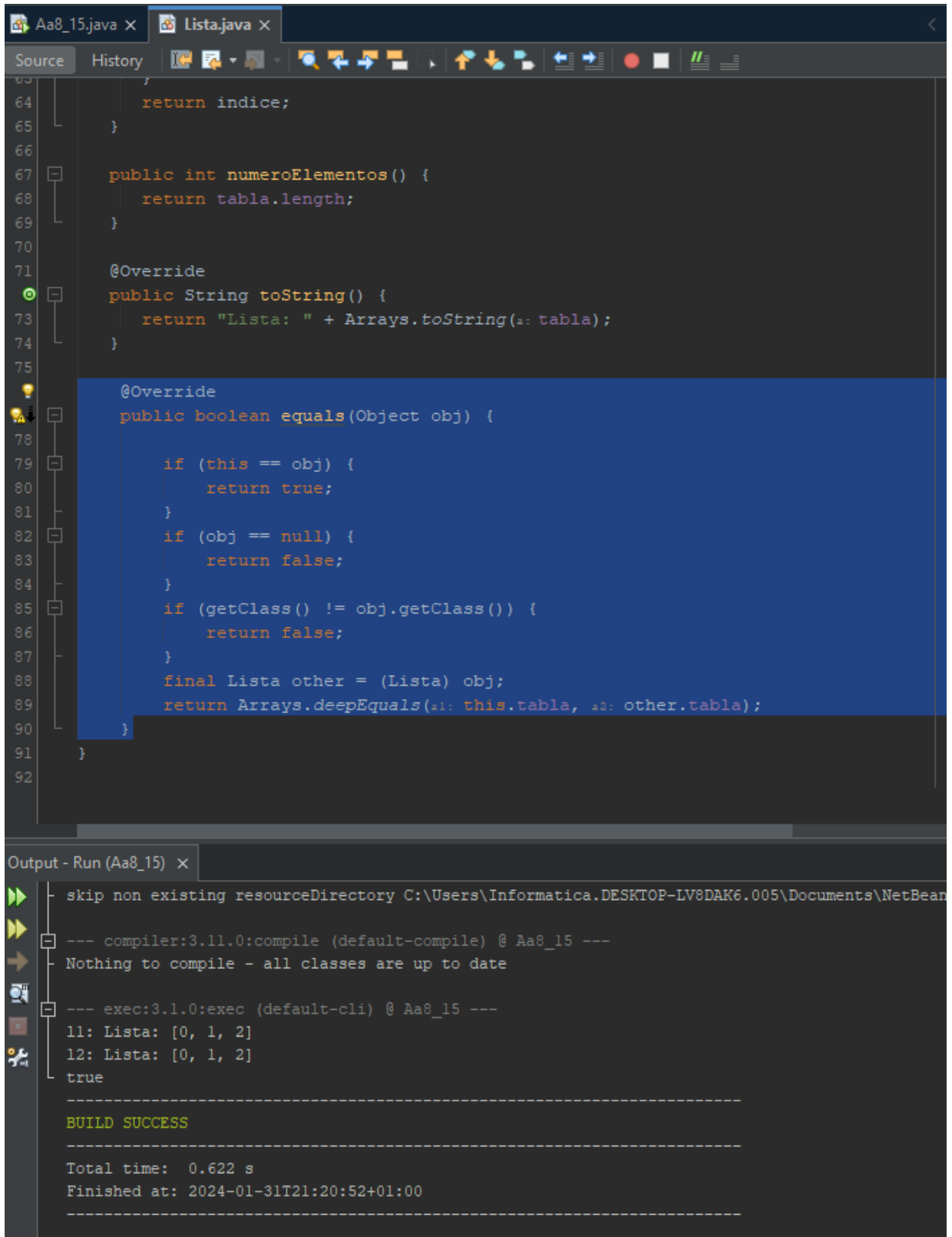
com.mycompany.aa8\_14.Lista

Output - Run (Aa8\_14)

```
skip non existing resourceDirectory C:\Users\Informatica.DESKTOP-LV8DAK6.005\Documents\NetBeansP
--- compiler:3.11.0:compile (default-compile) @ Aa8_14 ---
Changes detected - recompiling the module! :source
Compiling 2 source files with javac [debug target 20] to target\classes
--- exec:3.1.0:exec (default-cli) @ Aa8_14 ---
11: Lista: [0, 1, 2]
12: Lista: [0, 1, 2]
-----
BUILD SUCCESS
-----
Total time: 1.056 s
Finished at: 2024-01-31T21:12:02+01:00
-----
```

**8.15.** Escribe en la clase `Lista` un método `equals()` para compararlas. Dos listas se considerarán iguales si tienen los mismos elementos (incluidas las repeticiones) en el mismo orden.

No se ve todo el código pero lo único que cambia es la parte seleccionada, (método `equals`)



```
63     },
64     return indice;
65 }
66
67 public int numeroElementos() {
68     return tabla.length;
69 }
70
71 @Override
72 public String toString() {
73     return "Lista: " + Arrays.toString(a: tabla);
74 }
75
76 @Override
77 public boolean equals(Object obj) {
78     if (this == obj) {
79         return true;
80     }
81     if (obj == null) {
82         return false;
83     }
84     if (getClass() != obj.getClass()) {
85         return false;
86     }
87     final Lista other = (Lista) obj;
88     return Arrays.deepEquals(a1: this.tabla, a2: other.tabla);
89 }
90
91 }
92
```

Output - Run (Aa8\_15) x

```
skip non existing resourceDirectory C:\Users\Informatica.DESKTOP-LV8DAK6.005\Documents\NetBear
--- compiler:3.11.0:compile (default-compile) @ Aa8_15 ---
Nothing to compile - all classes are up to date
--- exec:3.1.0:exec (default-cli) @ Aa8_15 ---
11: Lista: [0, 1, 2]
12: Lista: [0, 1, 2]
true
-----
BUILD SUCCESS
-----
Total time: 0.622 s
Finished at: 2024-01-31T21:20:52+01:00
-----
```

**8.16.** Diseña la clase **Pila** heredando de **Lista** (ver Actividad resuelta 7.13).

Clase pila:

```
Aa8_16.java x Lista.java x Pila.java x
Source History
1 package com.mycompany.aa8_16;
2
3 public class Pila extends Lista{
4
5     public Pila() {
6         super();
7     }
8
9     void apilar(Integer elemento) {
10         super.insertarFinal(nuevo: elemento);
11     }
12
13     Integer desapilar() {
14         return super.eliminar(super.tabla.length - 1);
15     }
16
17     @Override
18     public String toString() {
19         return super.mostrar();
20     }
21 }
22
```

En la clase padre Lista, la declaro como abstracta y modifico el método mostrar

```
Aa8_16.java x Lista.java x Pila.java x
Source History
1 package com.mycompany.aa8_16;
2
3 import java.util.Arrays;
4
5 public abstract class Lista {
6     Integer[] tabla;
7 }

```

```
Aa8_16.java x Lista.java x Pila.java x
Source History
61 }
62 }
63 return indice;
64 }
65
66 void ordenar() {
67     Arrays.sort(=: tabla);
68 }
69
70 public String mostrar() {
71     return Arrays.toString(=: tabla);
72 }
73 }

```

## Resultado

```
- Compiling 3 source files with javac [debug target 20] to target\classes
--- exec:3.1.0:exec (default-cli) @ Aa8_16 ---
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
9
8
7
6
5
4
3
2
1
0
-----
BUILD SUCCESS
-----
Total time: 1.035 s
Finished at: 2024-01-31T21:35:56+01:00
```



**8.17. Escribe la clase Cola heredando de Lista (ver Actividad final 7.18).**

Main:

```
1 package com.mycompany.aa8_17;
2
3 public class Aa8_17 {
4     public static void main(String[] args) {
5         Cola c = new Cola();
6         for (int i = 0; i < 10; i++) {
7             c.encolar(num:i);
8         }
9
10        System.out.println(x: c);
11
12        Integer n = c.desencolar();
13
14        while (n != null) {
15            System.out.println(x: n);
16            n = c.desencolar();
17        }
18    }
19 }
20
```

Clase Cola

```
1 package com.mycompany.aa8_17;
2
3 public class Cola extends Lista{
4
5     public void encolar(int num){
6         super.insertar(posicion: 0, nuevo: num);
7     }
8
9     public Integer desencolar(){
10        Integer aEliminar = null;
11
12        if( super.tabla.length > 0){
13            aEliminar = super.tabla[0];
14            super.eliminar(indice: 0);
15        }
16        return aEliminar;
17    }
18
19
20    @Override
21    public String toString() {
22        return "Cola: " + super.mostrar();
23    }
24 }
25
```

## Clase Lista (sin tocar del ejercicio 7.18)

```
1 package com.myccompany.aa8_17;
2
3 import java.util.Arrays;
4
5 public abstract class Lista {
6     Integer[] tabla;
7
8     public Lista() {
9         tabla = new Integer[0];
10    }
11
12    void insertarPrincipio(Integer nuevo) {
13        tabla = Arrays.copyOf(original: tabla, tabla.length + 1);
14        System.arraycopy(src: tabla, srcPos: 0, dest: tabla, destPos: 1, tabla.length - 1);
15        tabla[0] = nuevo;
16    }
17
18    void insertarFinal(Integer nuevo) {
19        tabla = Arrays.copyOf(original: tabla, tabla.length + 1);
20        tabla[tabla.length - 1] = nuevo;
21    }
22
23    void insertarFinal(Lista otraLista) {
24        int tamIni = tabla.length; // tamaño inicial tabla
25        tabla = Arrays.copyOf(original: tabla, tabla.length + otraLista.tabla.length);
26        System.arraycopy(src: otraLista.tabla, srcPos: 0, dest: tabla, destPos: tamIni,
27            length: otraLista.tabla.length);
28    }
29
30    void insertar(int posicion, Integer nuevo) {
31        tabla = Arrays.copyOf(original: tabla, tabla.length + 1);
32        System.arraycopy(src: tabla, srcPos: posicion, dest: tabla, destPos: posicion + 1,
33            tabla.length - posicion - 1);
34        tabla[posicion] = nuevo;
35    }
36
37    Integer eliminar(int indice) {
38        Integer eliminado = null;
39        if (indice >= 0 && indice < tabla.length) {
40            eliminado = tabla[indice];
41            for (int i = indice + 1; i < tabla.length; i++) {
42                tabla[i - 1] = tabla[i];
43            }
44            tabla = Arrays.copyOf(original: tabla, tabla.length - 1);
45        }
46        return eliminado;
47    }
48
49    Integer get(int indice) {
50        Integer resultado = null;
51        if (indice >= 0 && indice < tabla.length) {
52            resultado = tabla[indice];
53        }
54        return resultado;
55    }
56
57    int buscar(Integer claveBusqueda) {
58        int indice = -1;
59        for (int i = 0; i < tabla.length && indice == -1; i++) {
60            if (tabla[i].equals(obj: claveBusqueda)) {
61                indice = i;
62            }
63        }
64        return indice;
65    }
66
67    void ordenar() {
68        Arrays.sort(a: tabla);
69    }
70
71    public String mostrar() {
72        return Arrays.toString(a: tabla);
73    }
74 }
75
```

Resultado:

```
--- exec:3.1.0:exec (default-cli) @ Aa8_17 ---
Cola: [9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
9
8
7
6
5
4
3
2
1
0
-----
BUILD SUCCESS
-----
Total time: 1.717 s
Finished at: 2024-02-01T19:16:49Z
-----
```