

# Programación orientada a objetos

# Contenido

- Implementación de clases
  - Constructores
  - Métodos modificadores y de acceso
  - Representación y comparación de objetos
  - Miembros estáticos
- Excepciones
- Paquetes

# Ejemplo: puntos en una cuadrícula

EjemploPuntos.java

```
// Solo se muestra parte del método main
```

```
Punto p = new Punto(x1, y1);
```

```
Punto q = new Punto(x2, y2);
```

```
System.out.println("Distancia entre p y q: " + p.distancia(q));
```

Creamos dos objetos  
de una clase **Punto**

Y calculamos la  
distancia entre ambos

# Ejemplo: puntos en una cuadrícula

EjemploPuntos.java

```
// Solo se muestra parte del método main
```

```
Punto p = new Punto(x1, y1);
```

```
Punto q = new Punto(x2, y2);
```

```
System.out.println("Distancia entre p y q: " + p.distancia(q));
```

Vamos a definir  
esta clase

# Declaración de una clase

Punto.java

```
public class Punto {  
    // Atributos  
    // Constructores  
    Métodos  
}
```

De momento solo  
«clases públicas»

# Declaración de una clase

Punto.java

```
public class Punto {
```

```
    // Atributos
```

```
    // Constructores
```

```
    // Métodos
```

```
}
```

Datos de «cada objeto»  
de la clase definida

# Declaración de una clase

Punto.java

```
public class Punto {  
    // Atributos  
    // Constructores  
    // Métodos  
}
```

Inician el  
objeto creado

# Declaración de una clase

Punto.java

```
public class Punto {  
    // Atributos  
    // Constructores  
    // Métodos  
}
```

Métodos aplicables a  
objetos de la clase



# Declaración de una clase

## Punto.java

```
public class Punto {  
    // Atributos  
  
    // Constructores  
  
    // Métodos  
  
}
```

¿main?

Clase que  
utiliza **Punto**

## EjemploPuntos.java

```
// Solo se muestra parte del método main  
Punto p = new Punto(x1, y1);  
Punto q = new Punto(x2, y2);  
// Otras instrucciones
```

# Declaración de atributos

Punto.java

```
public class Punto {  
    // Atributos  
    private int x;  
    private int y;  
    // Constructores  
    // Métodos  
}
```

Coordenadas de  
«cada objeto» **Punto**

Se inicializan al valor  
por defecto de su tipo

# Declaración de atributos

Punto.java

```
public class Punto {  
    // Atributos  
    private int x;  
    private int y;  
    // Constructores  
    // Métodos  
}
```

«Directamente» accesibles  
solo desde la propia clase

# Constructores

Punto.java

```
public class Punto {  
    // Atributos  
    // Constructores  
    public Punto(int x, int y) {  
        this.x = x;  
        this.y = y;  
    }  
    // Métodos  
}
```

Llamado «automáticamente»  
al crear un objeto de la clase

# Constructores

Punto.java

```
public class Punto {  
    // Atributos  
    // Constructores  
    public Punto(int x, int y) {  
        this.x = x;  
        this.y = y;  
    }  
    // Métodos  
}
```

Su nombre coincide  
con el de la clase

Sin tipo de retorno

# Sobrecarga de constructores

Punto.java

```
// Constructores
```

```
public Punto(int x, int y) {
```

```
    this.x = x;
```

```
    this.y = y;
```

```
}
```

```
public Punto() {
```

```
    this(0, 0);
```

```
}
```

Constructor con  
2 parámetros

Constructor sin  
parámetros

# Palabra reservada «this»

Punto.java

```
// Constructores
```

```
public Punto(int x, int y) {
```

```
    this.x = x;
```

```
    this.y = y;
```

```
}
```

```
public Punto() {
```

```
    this(0, 0);
```

```
}
```

Referencia al  
objeto actual

Permite acceder a atributos  
«ocultos» por parámetros  
con el mismo nombre

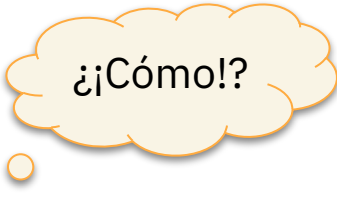
Llamada explícita a otro  
constructor de la clase

Solo si es la  
primera sentencia

# Constructor «por defecto»

Punto.java

```
public class Punto {  
    // Atributos  
    private int x;  
    private int y;  
  
    // Ningún constructor  
  
    // Métodos  
}
```



¿¡Cómo!?



# Constructor «por defecto»

Punto.java

```
public class Punto {  
    // Atributos  
    private int x;  
    private int y;  
  
    // Ningún constructor  
  
    // Métodos  
}
```

Disponible «automáticamente» un constructor sin parámetros

```
public Punto {  
}
```

Crearemos nuevos puntos así:  
`Punto p = new Punto();`

# Sin constructor por defecto

Punto.java

```
public class Punto {  
    // Atributos  
    // Constructor  
    public Punto(int x, int y) {  
        this.x = x;  
        this.y = y;  
    }  
    // Métodos  
}
```

Al haber declarado un constructor, ya no hay un constructor por defecto

¡No podemos crear nuevos puntos así!  
Punto p = new Punto();

# Métodos de acceso («Getters»)

## Punto.java

```
public class Punto {  
    // Atributos  
    // Constructores  
    // Métodos  
    public int getX() {  
        return x;  
    }  
}
```

Permite a otras clases «consultar»  
el valor del atributo privado x

## EjemploPuntos.java

```
// Solo se muestra parte del método main  
Punto p = new Punto(x1, y1);  
Punto q = new Punto(x2, y2);  
System.out.println("p.x = " + p.getX());  
System.out.println("q.x = " + q.getX());  
// Otras instrucciones
```

# Métodos modificadores («Setters»)

## Punto.java

```
public class Punto {  
    // Atributos  
    // Constructores  
    // Métodos  
    public void setX(int x) {  
        this.x = x;  
    }  
}
```

Permite a otras clases «modificar»  
el valor del atributo privado x

## EjemploPuntos.java

```
// Solo se muestra parte del método main  
Punto p = new Punto(x1, y1);  
Punto q = new Punto(x2, y2);  
p.setX(100 + p.getX());  
// Otras instrucciones
```

# Representación de objetos: «toString»

Punto.java

```
public class Punto {  
    // Atributos  
    // Constructores  
    // Métodos  
    public String toString() {  
        return "(" + x + ", " + y + ")";  
    }  
}
```

Devuelve una cadena con la representación del objeto **Punto**

# Comparación de objetos: «equals»

Punto.java

```
// Otros métodos
public boolean equals(Object o) {
    if (this == o)
        return true;
    if (!(o instanceof Punto))
        return false;
    Punto punto = (Punto) o;
    return x == punto.x && y == punto.y;
}
```

Parámetro de  
tipo **Object**

# Atributos estáticos

Punto.java

```
public class Punto {  
    // Atributos (variables de instancia)  
    private int x;  
    private int y;  
    // Atributos estáticos (variables de clase)  
    public static final int MAX_X = 100;  
    public static final int MAX_Y = 100;  
    // Constructores  
    // Métodos  
}
```

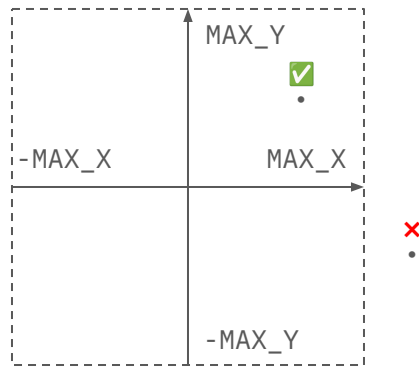
# Atributos estáticos

Punto.java

```
public class Punto {  
    // Atributos (variables de instancia)  
    private int x;  
    private int y;  
  
    // Atributos estáticos (variables de clase)  
    public static final int MAX_X = 100;  
    public static final int MAX_Y = 100;  
  
    // Constructores  
    // Métodos  
}
```

Atributos de «la clase»  
(no de cada objeto)

Define el intervalo para las  
coordenadas de cada punto





# Palabra reservada «final»

Punto.java

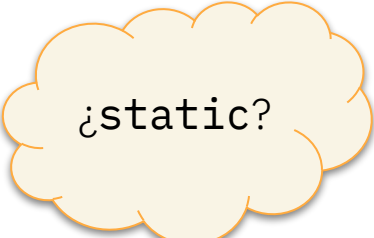
```
public class Punto {  
    // Atributos (variables de instancia)  
    private int x;  
    private int y;  
  
    // Atributos estáticos (variables de clase)  
    public static final int MAX_X = 100;  
    public static final int MAX_Y = 100;  
  
    // Constructores  
    // Métodos  
}
```

Impide asignar un nuevo valor a una variable

# Ejemplo: contador de objetos creados

Punto.java

```
public class Punto {  
    // Atributos  
    // Constructores  
    public Punto(int x, int y) {  
        this.x = x;  
        this.y = y;  
        puntosCreados++;  
    }  
    // Métodos  
}
```



¿static?

# Ejemplo: fechas en intervalo

Fecha.java

```
// Atributo (variable de instancia)
```

```
private int díasDesdeOrigen;
```

Días transcurridos desde el  
1 de enero de **PRIMER\_AÑO**

```
// Atributos estáticos (variables de clase)
```

```
public static final int PRIMER_AÑO = 2001;
```

```
public static final int ÚLTIMO_AÑO = 2100;
```

```
private static final int[] díasHastaPrimeroDeMes
```

```
    = {0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334};
```

```
private static final int[] díasHasta1Enero;
```

# Ejemplo: fechas en intervalo

Fecha.java

```
// Atributo (variable de instancia)
private int díasDesdeOrigen;

// Atributos estáticos (variables de clase)
public static final int PRIMER_AÑO = 2001;
public static final int ÚLTIMO_AÑO = 2100;

private static final int[] díasHastaPrimeroDeMes
    = {0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334};

private static final int[] díasHasta1Enero;
```

Fechas del siglo XXI

# Ejemplo: fechas en intervalo

Fecha.java

```
// Atributo (variable de instancia)
private int díasDesdeOrigen;

// Atributos estáticos (variables de clase)
public static final int PRIMER_AÑO = 2001;
public static final int ÚLTIMO_AÑO = 2100;
```

```
private static final int[] díasHastaPrimeroDeMes
    = {0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334};
private static final int[] díasHasta1Enero;
```

Facilitan el cálculo de  
díasDesdeOrigen

# Inicializador estático

Fecha.java

```
// Inicializar atributo estático díasHasta1Enero
```

```
static {
```

```
    díasHasta1Enero = new int[ÚLTIMO_AÑO - PRIMER_AÑO + 1];
```

```
    díasHasta1Enero[0] = 0;
```

```
    for (int i = 1; i <= díasHasta1Enero.length; i++) {
```

```
        díasHasta1Enero[i] = díasHasta1Enero[i - 1] + 365;
```

```
        if (esBisiesto((i - 1) + PRIMER_AÑO))
```

```
            díasHasta1Enero[i]++;
```

```
    }
```

```
}
```

Instrucciones ejecutadas  
al «cargar» la clase

# Métodos estáticos

Fecha.java

```
public class Fecha {  
    // Atributos  
    // Constructores  
    // Métodos  
    public static boolean esBisiesto(int año) {  
        return año % 4 == 0 && año % 100 != 0 || año % 400 == 0;  
    }  
}
```

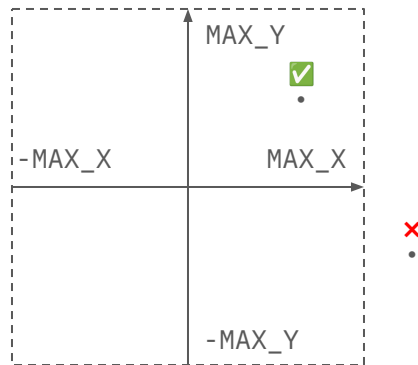
No podemos usar **this** ni acceder a miembros no estáticos

# Comprobaciones en un constructor

Punto.java

```
// Solo se muestran los constructores  
public Punto(int x, int y) {  
    this.x = x;  
    this.y = y;  
}  
public Punto() {  
    this(0, 0);  
}
```

¿Coordenadas  
no válidas?





# Lanzar una excepción

Punto.java

```
// Atributos  
// Constructores  
public Punto(int x, int y) {  
    if (Math.abs(x) > MAX_X || Math.abs(y) > MAX_Y)  
        throw new ExcepcionPuntoNoValido();  
    this.x = x;  
    this.y = y;  
}  
// Métodos
```

Si las coordenadas  
no son válidas

# Lanzar una excepción: «throw»

Punto.java

```
// Atributos  
  
// Constructores  
public Punto(int x, int y) {  
    if (Math.abs(x) > MAX_X || Math.abs(y) > MAX_Y)  
        throw new ExcepcionPuntoNoValido();  
    this.x = x;  
    this.y = y;  
}  
  
// Métodos
```

«Lanzamos» una excepción  
(no se crea el punto)

# Capturar una excepción

EjemploPuntos.java

Sentencia  
«try-catch»

```
Punto p = null;
do {
    System.out.print("Introduce el valor de x: "); int x = entrada.nextInt();
    System.out.print("Introduce el valor de y: "); int y = entrada.nextInt();

    try {
        p = new Punto(x, y);
    } catch (ExcepcionPuntoNoValido e) {
        System.out.println("Valores incorrectos. Inténtalo de nuevo");
    }

} while (p == null);
```

# Capturar una excepción

## EjemploPuntos.java

```
Punto p = null;
do {
    System.out.print("Introduce
    System.out.print("Introduce
    try {
        p = new Punto(x, y);
    } catch (ExcepcionPuntoNoValido e) {
        System.out.println("Valores incorrectos. Inténtalo de nuevo");
    }
} while (p == null);
```

Código que puede lanzar una excepción

```
x = entrada.nextInt();
y = entrada.nextInt();
```

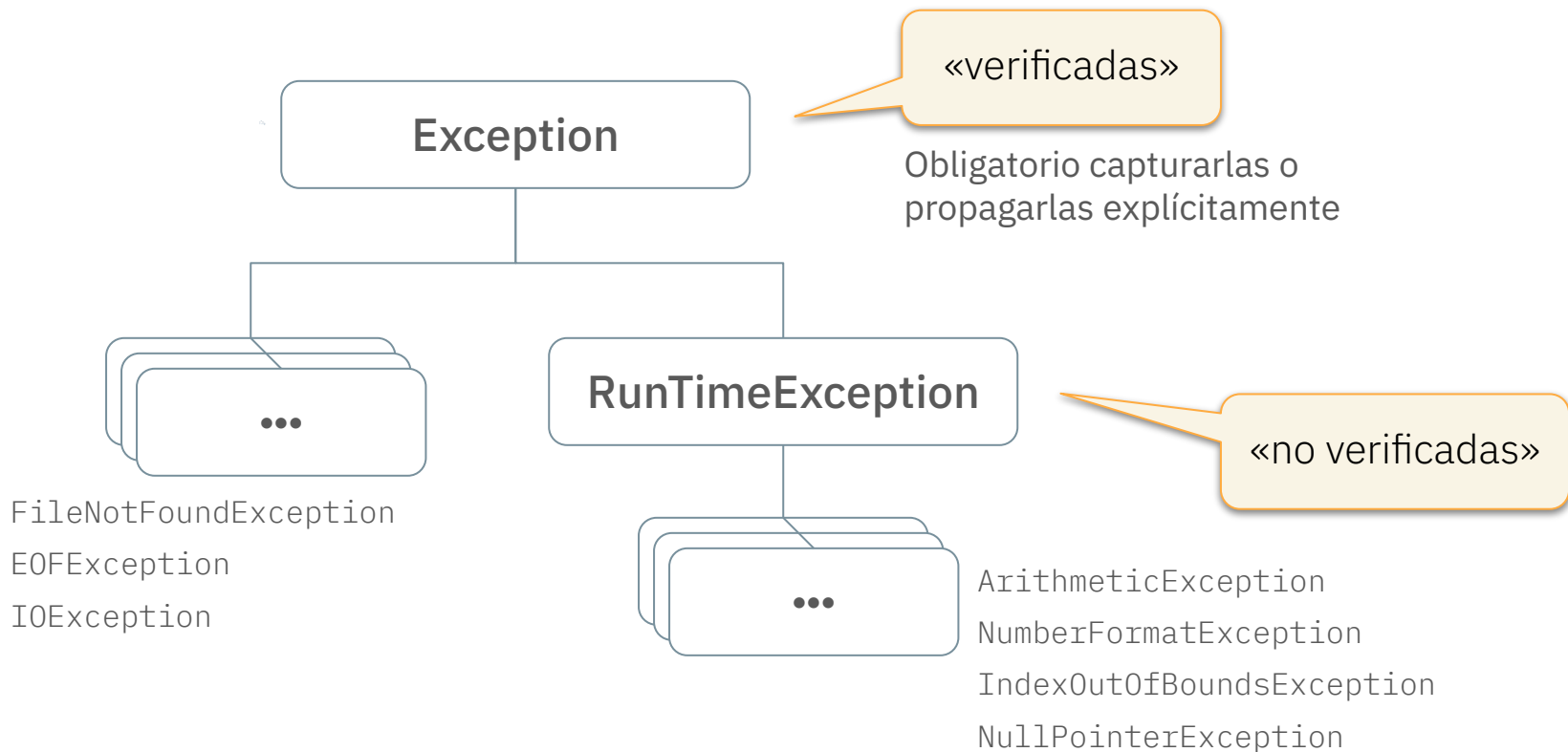
# Capturar una excepción

## EjemploPuntos.java

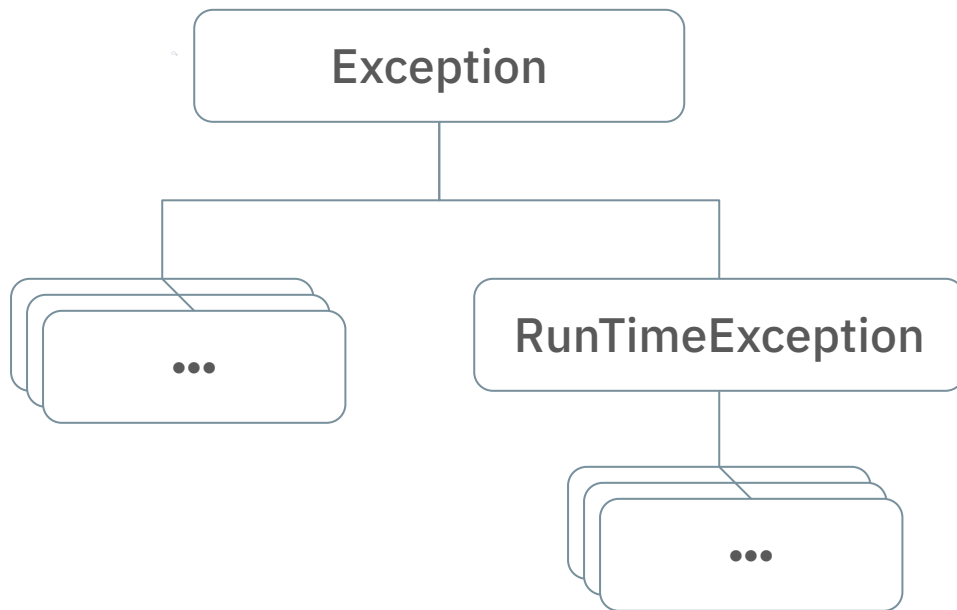
```
Punto p = null;
do {
    System.out.print("Introduce el valor de x: "); int x = entrada.nextInt();
    System.out.print("Introduce el valor de y: "); int y = entrada.nextInt();
    try {
        p = new Punto(x, y);
    } catch (ExcepcionPuntoNoValido e) {
        System.out.println("Valores incorrectos. Inténtalo de nuevo");
    }
} while (p == null);
```

«Manejador»  
de la excepción

# Tipos de excepciones



# Tipos de excepciones



¿De qué tipo?

ExcepcionPuntoNoValido

# Ejemplo: control previo a la creación

## EjemploPuntos.java

```
System.out.print("Introduce el valor de x: "); int x = entrada.nextInt();
System.out.print("Introduce el valor de y: "); int y = entrada.nextInt();
while ( Math.abs(x) > Punto.MAX_X || Math.abs(y) > Punto.MAX_Y ) {
    System.out.println("Valores incorrectos. Inténtalo de nuevo");
    System.out.print("Introduce el valor de x: "); x = entrada.nextInt();
    System.out.print("Introduce el valor de y: "); y = entrada.nextInt();
}
Punto p = new Punto(x, y);
```



# Ejemplo: control previo a la creación

## EjemploPuntos.java

```
System.out.print("Introduce el valor de x: "); int x = entrada.nextInt();
System.out.print("Introduce el valor de y: "); int y = entrada.nextInt();
while ( Math.abs(x) > Punto.MAX_X || Math.abs(y) > Punto.MAX_Y ) {
    System.out.println("Valores fuera de rango. Introduce de nuevo");
    System.out.print("Introduce el valor de x: "); x = entrada.nextInt();
    System.out.print("Introduce el valor de y: "); y = entrada.nextInt();
}
Punto p = new Punto(x, y);
```

Sabemos que no puede producirse la excepción

¿Verificada?

# Nueva excepción «no verificada»

ExcepcionPuntoNoValido.java

```
public class ExcepcionPuntoNoValido extends RuntimeException {  
    // De momento lo dejamos vacío  
}
```

# Propagación de excepciones

## ContarLineas.java

```
static int contarLineas(String nombreFichero) throws FileNotFoundException {  
    int cantidad = 0; // Cantidad de líneas  
    Scanner entrada = new Scanner(new File(nombreFichero));  
    while (entrada.hasNextLine()) {  
        String línea = entrada.nextLine();  
        cantidad++;  
    }  
    entrada.close();  
    return cantidad;  
}
```

Indicamos que  
sería «propagada»

Podría producirse una excepción  
FileNotFoundException

Obligatorio capturarla o propagarla  
explícitamente

# Sentencia «try-catch»

## Ejemplo

```
try {  
    // Código que puede lanzar alguna excepción  
} catch (NombreExcepción1 e) {  
    // Manejador de excepción  
} ... {  
} catch (NombreExcepciónN e) {  
    // Manejador de excepción  
} finally {  
    // Código ejecutado siempre (se produzca o no excepción)  
}  
// Código posterior
```

# Clases organizadas en «paquetes»

HolaMundo.java

Indica a qué paquete pertenece una clase

```
package practica0;
```

```
public class HolaMundo {
```

```
    public static void main(String[] args) {
```

```
        System.out.println("¡Hola, mundo!");
```

```
    }
```

```
}
```

# Clases organizadas en «paquetes»

HolaMundo.java

```
package practica0;  
  
public class HolaMundo {  
    public static void main(String[] args) {  
        System.out.println("¡Hola, mundo!");  
    }  
}
```

Relacionado con la estructura del directorio donde reside la clase

# Algunos paquetes predefinidos

Paquete	Algunas clases
java.lang	Math Object String System
java.util	Scanner Formatter Arrays Calendar
java.io	File

Importado de forma  
«automática»

# Mismo nombre en paquetes distintos

Complejo.java

```
package tema2.binomica;  
  
public class Complejo {  
    // Atributos  
    private double re;  
    private double im;  
  
    // Constructores  
  
    // Métodos  
  
}
```

Complejo.java

```
package tema2.polar;  
  
public class Complejo {  
    // Atributos  
    private double módulo;  
    private double argumento;  
  
    // Constructores  
  
    // Métodos  
  
}
```

Clases de igual nombre  
en paquetes distintos



# Directiva «import»

EjemploComplejos.java

```
package tema2;
import tema2.binomica.Complejo;
public class EjemploComplejos {
    public static void main(String[] args) {
        Complejo c1 = new Complejo(3, 5);
        // Otras instrucciones
    }
}
```

# «Nombre cualificado» de una clase

## CrearFicheroRaicesSinImport.java

```
// Crea un fichero con las raíces cuadradas de los números de 1 a n
static void crearFicheroRaicesCuadradas(int n, String nombreFichero)
    throws java.io.FileNotFoundException {
    java.util.Formatter salida = new java.util.Formatter(nombreFichero);
    for (int i = 1; i <= n; i++)
        salida.format("Raíz cuadrada de %d = %.3f %n", i, Math.sqrt(i));
    salida.close();
}
```

# «Nombre cualificado» de una clase

**CrearFicheroRaicesSinImport.java**

```
// Crea un fichero con las raíces cuadradas de los números de 1 a n
static void crearFicheroRaicesCuadradas(int n, String nombreFichero)
    throws java.io.FileNotFoundException {
    java.util.Formatter salida = new java.util.Formatter(nombreFichero);
    for (int i = 1; i <= n; i++)
        salida.format("Raíz cuadrada de %d = %.3f %n", i, Math.sqrt(i));
    salida.close();
}
```

¿java.lang?