Programación Lineal en Java

Curso Completo sin Programación Orientada a Objetos

Autor: [Sh4d0w$_E rr0r$]

Edición: 1.0

Fecha: 25 de diciembre de 2025

Material de estudio complementario para el curso de YouTube
Programación Lineal en Java

Objetivo

Objetivo del Libro

Este libro tiene como objetivo enseñar los fundamentos de la programación en Java utilizando un enfoque de **programación lineal** o **programación estructurada**. A diferencia de la mayoría de los cursos de Java, aquí **no utilizaremos Programación Orientada a Objetos (POO), o al menos todavía**, enfocándonos en desarrollar habilidades fundamentales de programación usando solo métodos estáticos y estructuras básicas.

¿A quién va dirigido?

- Estudiantes que inician en programación
- Programadores de otros lenguajes que quieren aprender Java
- Personas que necesitan reforzar conceptos básicos antes de abordar POO
- Autodidactas que prefieren un enfoque gradual

Estructura del Contenido

El libro está organizado en 10 capítulos que progresan desde lo más básico hasta proyectos prácticos:

1. **Introducción a Java**: Configuración del entorno y primer programa
2. **Variables y tipos de datos**: Fundamentos del almacenamiento de información
3. **Operadores y expresiones**: Manipulación de datos y cálculos
4. **Control de flujo**: Toma de decisiones y repetición de código
5. **Entrada y salida**: Interacción con el usuario por consola
6. **Métodos**: Organización y reutilización de código
7. **Ámbito de variables**: Entendiendo el alcance de las variables

8. **Arrays**: Trabajo con colecciones de datos
9. **Buenas prácticas**: Escribiendo código limpio y mantenible
10. **Proyectos prácticos**: Aplicación de todo lo aprendido

Requisitos Previos

- Conocimientos básicos de uso de un ordenador
- No se requiere experiencia previa en programación
- Motivación para aprender y practicar

Cómo Usar Este Libro

Nota Importante

Este libro está diseñado para ser estudiado secuencialmente. Cada capítulo construye sobre los conceptos aprendidos en los anteriores.

- **Lee los capítulos en orden**
- **Prueba todos los ejemplos** de código
- **Completa los ejercicios propuestos**
- **Experimenta** modificando los códigos para ver qué sucede
- **Consulta el material complementario** del curso en YouTube

Convenciones Utilizadas

- **Código Java**: Se muestra en recuadros con fondo gris claro
- `Texto en lnea`: Para referencias a código dentro del texto
- **Notas importantes**: En recuadros azules
- **Advertencias**: En recuadros rojos para evitar errores comunes
- **Ejercicios**: En recuadros verdes para práctica
- **Ejemplos**: En recuadros amarillos con casos prácticos

Software Necesario

- **Java Development Kit (JDK)** 8 o superior (Recomiendo la ultima version estable)
- Un editor de texto o IDE (recomendado: Antigravity, IntelliJ IDEA Community, o Eclipse)
- Terminal o línea de comandos para ejecutar programas (Opcional)

¡Comencemos nuestro viaje en la programación con Java!

Índice general

Capítulo 1

Introducción a Java

1.1. ¿Qué es Java?

Java es un lenguaje de programación creado por Sun Microsystems (ahora propiedad de Oracle) en 1995. Es uno de los lenguajes más populares y utilizados en el mundo debido a sus características:

- **Multiplataforma**: 'Escribe una vez, ejecuta en cualquier lugar'
- **Orientado a objetos**: Aunque en este curso no usaremos esta característica inicialmente
- **Robusto y seguro**: Con manejo automático de memoria y verificación de tipos
- **Extensa biblioteca estándar**: Con muchas funciones incorporadas

Nota Importante

En este curso, utilizaremos Java como un lenguaje de programación **lineal o estructurado**, similar a cómo se usan C o Pascal. Esto nos permitirá concentrarnos en los fundamentos de programación sin la complejidad inicial de la POO.

1.2. Instalación del Entorno de Desarrollo

1.2.1. Instalación del JDK

El Java Development Kit (JDK) es el software necesario para compilar y ejecutar programas Java.

1. Visita https://adoptium.net/ (recomendado) u https://www.oracle.com/java/technologies/
2. Descarga la versión más reciente de JDK para tu sistema operativo
3. Sigue las instrucciones de instalación

4. Verifica la instalación abriendo una terminal y escribiendo:

```
java -version
javac -version
```

Listing 1.1: Verificando la instalación de Java

1.2.2. Configuración de Variables de Entorno (Windows)

1. Busca "Variables de entorno."[en] el menú inicio
2. Haz clic en "Variables de entorno"
3. En "Variables del sistema", busca o crea la variable `JAVA_HOME`
4. Establece su valor a la ruta de instalación del JDK (ej: `C:\Program Files\Java\jdk-17`)
5. Edita la variable `Path` y añade `%JAVA_HOME%\bin`

¡Advertencia!

Si no configuras correctamente las variables de entorno, no podrás compilar programas Java desde la terminal. En macOS y Linux, generalmente Java se configura automáticamente.

1.3. Editores de Código Recomendados

- **Antigravity**: Gratuito, ligero y con excelente soporte para Java
- **IntelliJ IDEA Community**: IDE completo específico para Java
- **Eclipse**: Alternativa popular en el mundo Java

1.4. Estructura Básica de un Programa Java

Todo programa Java tiene una estructura mínima requerida:

```
// Comentario: Este es un programa Java b sico

public class NombreDelPrograma {

    // M todo principal - punto de entrada del programa
```

```
    public static void main(String[] args) {
        // Tu c digo va aqu
    }
}
```

Listing 1.2: Estructura mínima de un programa Java

Nota Importante

La línea `public static void main(String[] args)` es esencial. Es el punto de entrada de cualquier programa Java. Sin ella, el programa no puede ejecutarse. Cuando nosotros intentamos arrancar nuestro programa, este busca alguna coincidencia con un 'public static void main' para arreancar. Es el centro de nuestro programa.

1.4.1. Partes de la Estructura

- **Comentarios**: Líneas que comienzan con `//` o bloques `/* ... */`
- **Clase**: Definida con `public class NombreDelPrograma`
- **Método main**: Donde comienza la ejecución
- **Llaves { }**: Delimitan bloques de código

1.5. Primer Programa: Hola Mundo

Vamos a crear nuestro primer programa Java paso a paso:

1.5.1. Paso 1: Crear el Archivo

Crea un nuevo archivo llamado **`HolaMundo.java`** con el siguiente contenido:

```
// Programa: HolaMundo.java
// Autor: Estudiante de Programaci n Lineal
// Descripci n: Muestra un mensaje en la consola

public class HolaMundo {
    public static void main(String[] args) {
        // Mostrar mensaje en la consola
        System.out.println(" Hola , mundo!");
        System.out.println("Bienvenido a la programaci n en Java");

        // Tambi n podemos usar print (sin salto de l nea)
```

```
12          System.out.print("Este es mi ");
13          System.out.print("primer programa");
14          System.out.println(" en Java.");
15
16          // Mostrar informaci n sobre el sistema
17          System.out.println("Java version: " +
18                             System.getProperty("java.version"));
19      }
20  }
```

Listing 1.3: HolaMundo.java - Nuestro primer programa

1.5.2. Paso 2: Compilar el Programa

Abre una terminal, navega a la carpeta donde guardaste el archivo y ejecuta:

```
1  javac HolaMundo.java
```

Listing 1.4: Compilando el programa Java

Si no hay errores, se creará un archivo **`HolaMundo.class`**.

1.5.3. Paso 3: Ejecutar el Programa

Ejecuta el programa compilado:

```
1  java HolaMundo
```

Listing 1.5: Ejecutando el programa Java

Ejemplo Práctico

Salida esperada en la consola:

```
¡Hola, mundo!
Bienvenido a la programación en Java
Este es mi primer programa en Java.
Java version: 17.0.1
```

1.6. Explicación del Código

- `System.out.println()`: Muestra texto en la consola y añade un salto de línea

- `System.out.print()`: Muestra texto sin salto de línea
- `+`: Operador para concatenar (unir) textos
- `System.getProperty()`: Obtiene información del sistema

1.7. Errores Comunes

¡Advertencia!

Error: çlass HolaMundo is public, should be declared in a file named HolaMundo.java"
Solución: El nombre del archivo debe coincidir EXACTAMENTE con el nombre de la clase pública.

¡Advertencia!

Error: çannot find symbol"
Solución: Revisa la ortografía. Java es **case-sensitive** (distingue mayúsculas y minúsculas).

1.8. Convenciones de Nombres en Java

- **Clases**: Comienzan con mayúscula, estilo CamelCase (Ej: `HolaMundo`)
- **Métodos**: Comienzan con minúscula, estilo camelCase (Ej: `calcularPromedio()`)
- **Variables**: Comienzan con minúscula, estilo camelCase (Ej: `nombreUsuario`)
- **Constantes**: Todo en mayúsculas, con guiones bajos (Ej: `PI`, `MAX_INTENTOS`)

Ejercicio Propuesto

Ejercicio 1.1: Personalizar Hola Mundo
Crea un programa llamado **`Bienvenida.java`** que:

1. Muestre tu nombre
2. Muestre tu ciudad
3. Muestre una breve descripción de por qué quieres aprender Java
4. Use al menos 3 veces `System.out.println()` y 2 veces `System.out.print()`

Compila y ejecuta el programa para verificar que funciona correctamente.

2.4. Tipo String (Cadena de Texto)

Aunque `String` no es un tipo primitivo, es fundamental en Java:

```
public class StringsEjemplo {
    public static void main(String[] args) {
        // Declaraci n de Strings
        String nombre = "Ana";
        String apellido = "Garc a";
        String saludo = "Hola";

        // Concatenaci n de Strings
        String nombreCompleto = nombre + " " + apellido;
        String mensaje = saludo + ", " + nombre + "!";

        System.out.println("Nombre completo: " + nombreCompleto);
        System.out.println("Mensaje: " + mensaje);

        // Longitud de un String
        int longitud = nombreCompleto.length();
        System.out.println("Longitud del nombre: " + longitud + "
            caracteres");

        // M todos  tiles  de String
        System.out.println("En may sculas: " +
            nombreCompleto.toUpperCase());
        System.out.println("En min sculas: " +
            nombreCompleto.toLowerCase());
        System.out.println("Primer car cter: " +
            nombreCompleto.charAt(0));
        System.out.println("  Contiene  'Garc a'? " +
            nombreCompleto.contains("Garc a"));

        // Comparaci n de Strings (  IMPORTANTE  !)
        String str1 = "Hola";
        String str2 = "Hola";
        String str3 = new String("Hola");

        System.out.println("\nComparaciones:");
        System.out.println("str1 == str2: " + (str1 == str2));  //
            true (mismo objeto)
        System.out.println("str1 == str3: " + (str1 == str3));  //
            false (objetos diferentes)
        System.out.println("str1.equals(str3): " +
            str1.equals(str3));  // true (mismo contenido)
```

```
34      }
35  }
```

Listing 2.2: Trabajando con Strings

¡Advertencia!

¡Cuidado con la comparación de Strings! Usa siempre `.equals()` para comparar el contenido, no == que compara referencias de objetos.

2.5. Conversión entre Tipos de Datos

```
1  public class ConversionTipos {
2      public static void main(String[] args) {
3          // Conversi n impl cita (autom tica) - de menor a mayor
               precisi n
4          int entero = 100;
5          double decimal = entero;  // Conversi n autom tica
6          System.out.println("Entero a double: " + decimal);
7
8          // Conversi n expl cita (casting) - de mayor a menor
               precisi n
9          double precio = 99.99;
10         int precioEntero = (int) precio;  // Casting expl cito
11         System.out.println("Double a int: " + precioEntero);  //
               Pierde  decimales!
12
13         // Casting entre tipos num ricos
14         byte b = 100;
15         short s = 1000;
16         int i = 100000;
17         long l = 10000000000L;
18
19         // Necesario cuando hay riesgo de p rdida de datos
20         int grande = 1000;
21         byte peque o = (byte) grande;  //  Puede  haber
               desbordamiento!
22         System.out.println("1000 como byte: " + peque o);  //
               Resultado inesperado
23
24         // Conversi n String a n mero
25         String numeroTexto = "123";
26         int numero = Integer.parseInt(numeroTexto);
```

```
        double decimalTexto = Double.parseDouble("45.67");

        System.out.println("String a int: " + numero);
        System.out.println("String a double: " + decimalTexto);

        // Conversi n  n mero a String
        int valor = 456;
        String texto = String.valueOf(valor);
        String texto2 = valor + "";  // Otra forma

        System.out.println("int a String: " + texto);
        System.out.println("int a String (otra forma): " + texto2);
    }
}
```

Listing 2.3: Conversión de tipos (casting)

2.6. Ejemplo Práctico: Calculadora de Edad

```
public class CalculadoraEdad {
    public static void main(String[] args) {
        // Informaci n personal
        String nombre = "Carlos";
        int a oNacimiento = 1995;
        int a oActual = 2024;

        // C lculos
        int edad = a oActual - a oNacimiento;
        int meses = edad * 12;
        int semanas = edad * 52;
        int dias = edad * 365;  // Aproximado
        int horas = dias * 24;

        // Constantes
        final int DIAS_POR_A O = 365;
        final int HORAS_POR_DIA = 24;
        final int MINUTOS_POR_HORA = 60;

        // C lculo  m  s preciso (considerando minutos)
        int minutos = horas * MINUTOS_POR_HORA;

        // Mostrar resultados
        System.out.println("=== CALCULADORA DE EDAD ===");
```

```
25          System.out.println("Nombre: " + nombre);
26          System.out.println("A o de nacimiento: " + a oNacimiento);
27          System.out.println("A o actual: " + a oActual);
28          System.out.println("\nResultados:");
29          System.out.println("Edad: " + edad + " a os");
30          System.out.println("Meses: " + meses + " meses");
31          System.out.println("Semanas: " + semanas + " semanas");
32          System.out.println("D as: " + dias + " d as (aprox)");
33          System.out.println("Horas: " + horas + " horas");
34          System.out.println("Minutos: " + minutos + " minutos");
35
36          // Proyecci n a futuro
37          int edadEn5A os = edad + 5;
38          int a oJubilacion = a oNacimiento + 65;  // Suponiendo
               jubilaci n a los 65
39          int a osParaJubilacion = 65 - edad;
40
41          System.out.println("\nProyecciones:");
42          System.out.println("Edad en 5 a os: " + edadEn5A os + "
               a os");
43          System.out.println("A o de jubilaci n: " + a oJubilacion);
44          System.out.println("A os para jubilaci n: " +
               a osParaJubilacion);
45      }
46  }
```

Listing 2.4: Calculadora de edad con variables

Ejercicio Propuesto

Ejercicio 2.1: Calculadora de IMC

Crea un programa que calcule el Índice de Masa Corporal (IMC).

Fórmula: IMC = peso / (altura × altura)

Donde:

- peso en kilogramos
- altura en metros

El programa debe:

1. Declarar variables para peso y altura
2. Calcular el IMC
3. Mostrar el resultado con 2 decimales

4. Mostrar un mensaje según la clasificación:
    - Bajo peso: IMC <18.5
    - Normal: 18.5 IMC <25
    - Sobrepeso: 25 IMC <30
    - Obesidad: IMC 30

Ejercicio Propuesto

Ejercicio 2.2: Conversor de Unidades

Crea un programa que convierta:

1. Grados Celsius a Fahrenheit (F = C × 9/5 + 32)
2. Kilómetros a Millas (1 km = 0.621371 millas)
3. Kilogramos a Libras (1 kg = 2.20462 lbs)

Usa constantes para los factores de conversión.

Ejercicio Propuesto

Ejercicio 2.3: Calculadora de Interés Simple

Crea un programa que calcule el interés simple.
Fórmula: I = P × R × T
Donde:

- P = Principal (monto inicial)
- R = Tasa de interés anual (en decimal, ej: 5 % = 0.05)
- T = Tiempo en años

El programa debe calcular y mostrar:

1. El interés generado
2. El monto total (principal + interés)

2.7. Resumen del Capítulo

- Las variables son espacios de memoria para almacenar datos

- Java tiene 8 tipos primitivos: byte, short, int, long, float, double, char, boolean
- Los Strings se usan para texto y tienen métodos útiles
- Las constantes se declaran con `final` y no pueden cambiar
- El casting explícito es necesario para conversiones que pueden perder datos
- Usa `.equals()` para comparar Strings, no `==`

Capítulo 3

Operadores y Expresiones

3.1. Introducción a los Operadores

Los operadores son símbolos especiales que realizan operaciones sobre operandos (variables, valores o expresiones). Java tiene varios tipos de operadores para diferentes propósitos.

3.2. Operadores Aritméticos

Operador	Nombre	Ejemplo	Resultado
`+`	Suma	`5 + 3`	8
`-`	Resta	`10 - 4`	6
`*`	Multiplicación	`6 * 7`	42
`/`	División	`15 / 4`	3 (división entera)
`%`	Módulo (residuo)	`15 % 4`	3
`++`	Incremento	`x++`	`x + 1`
`-`	Decremento	`x-`	`x - 1`

Cuadro 3.1: Operadores aritméticos en Java

```
1  public class OperadoresAritmeticos {
2      public static void main(String[] args) {
3          int a = 15;
4          int b = 4;
5
6          System.out.println("a = " + a + ", b = " + b);
7          System.out.println("\nOperaciones b sicas:");
8          System.out.println("Suma: " + a + " + " + b + " = " + (a +
               b));
9          System.out.println("Resta: " + a + " - " + b + " = " + (a -
               b));
10         System.out.println("Multiplicaci n: " + a + " * " + b + " =
               " + (a * b));
```

```
11          System.out.println("Divisi n entera: " + a + " / " + b + "
               = " + (a / b));
12          System.out.println("Residuo: " + a + " % " + b + " = " + (a
               % b));
13
14          // Divisi n con n meros decimales
15          double x = 15.0;
16          double y = 4.0;
17          System.out.println("\nDivisi n decimal: " + x + " / " + y +
               " = " + (x / y));
18
19          // Operadores de incremento y decremento
20          int contador = 5;
21          System.out.println("\nOperadores de incremento:");
22          System.out.println("contador = " + contador);
23          System.out.println("contador++ = " + contador++);  //
               Post-incremento
24          System.out.println("Despu s: contador = " + contador);
25          System.out.println("++contador = " + ++contador);  //
               Pre-incremento
26          System.out.println("Final: contador = " + contador);
27
28          // Ejemplo pr ctico: calcular promedio
29          int nota1 = 85;
30          int nota2 = 90;
31          int nota3 = 78;
32          int nota4 = 92;
33
34          int suma = nota1 + nota2 + nota3 + nota4;
35          double promedio = (double) suma / 4;  // Casting para decimal
36
37          System.out.println("\nC lculo de promedio:");
38          System.out.println("Notas: " + nota1 + ", " + nota2 + ", " +
               nota3 + ", " + nota4);
39          System.out.println("Suma: " + suma);
40          System.out.println("Promedio: " + promedio);
41
42          // Operaciones combinadas
43          System.out.println("\nOperaciones combinadas:");
44          int resultado = 10 + 5 * 2 - 8 / 4;
45          System.out.println("10 + 5 * 2 - 8 / 4 = " + resultado);
46          System.out.println("(10 + 5) * (2 - 8) / 4 = " + ((10 + 5) *
               (2 - 8) / 4));
47      }
48  }
```

Listing 3.1: Ejemplos de operadores aritméticos

3.3. Operadores Relacionales

Los operadores relacionales comparan dos valores y devuelven un booleano (`true` o `false`).

Operador	Descripción	Ejemplo	Resultado
==	Igual a	5 == 5	true
!=	No igual a	5 != 3	true
>	Mayor que	10 >5	true
<	Menor que	10 <5	false
>=	Mayor o igual que	10 >= 10	true
<=	Menor o igual que	5 <= 10	true

Cuadro 3.2: Operadores relacionales en Java

```java
public class OperadoresRelacionales {
    public static void main(String[] args) {
        int a = 10;
        int b = 5;
        int c = 10;

        System.out.println("a = " + a + ", b = " + b + ", c = " + c);
        System.out.println("\nComparaciones:");
        System.out.println("a == b: " + (a == b));       // false
        System.out.println("a == c: " + (a == c));       // true
        System.out.println("a != b: " + (a != b));       // true
        System.out.println("a > b: " + (a > b));         // true
        System.out.println("a < b: " + (a < b));         // false
        System.out.println("a >= c: " + (a >= c));       // true
        System.out.println("b <= a: " + (b <= a));       // true

        // Comparaciones con n meros decimales
        double x = 3.1416;
        double y = 3.14;

        System.out.println("\nComparaciones decimales:");
        System.out.println("x == y: " + (x == y));       // false
        System.out.println("x > y: " + (x > y));         // true

```

```
25          // Comparaciones de caracteres (basado en c digo Unicode)
26          char letra1 = 'A';
27          char letra2 = 'B';
28          char letra3 = 'a';
29
30          System.out.println("\nComparaciones de caracteres:");
31          System.out.println("'A' == 'A': " + (letra1 == 'A'));     //
               true
32          System.out.println("'A' < 'B': " + (letra1 < letra2));    //
               true
33          System.out.println("'A' < 'a': " + (letra1 < letra3));    //
               true (A=65, a=97)
34
35          // Ejemplo pr ctico: verificar si un n mero est  en rango
36          int numero = 25;
37          int minimo = 0;
38          int maximo = 100;
39
40          boolean enRango = numero >= minimo && numero <= maximo;
41          System.out.println("\nVerificaci n de rango:");
42          System.out.println("N mero: " + numero);
43          System.out.println("Rango: " + minimo + " a " + maximo);
44          System.out.println("  Est   en rango? " + enRango);
45
46          // Verificar si un n mero es par
47          boolean esPar = numero % 2 == 0;
48          System.out.println("  " + numero + " es par? " + esPar);
49      }
50  }
```

Listing 3.2: Ejemplos de operadores relacionales

3.4. Operadores Lógicos

Los operadores lógicos se usan para combinar expresiones booleanas.

Operador	Nombre	Descripción	Ejemplo
`&&`	AND (Y)	`true` si ambos operandos son `true`	`(a >0) && (a <10)`
`||`	OR (O)	`true` si al menos un operando es `true`	`(edad <18) || (edad >65)`
`!`	NOT (NO)	Invierte el valor booleano	`!(esFinDeSemana)`

Cuadro 3.3: Operadores lógicos en Java

```java
public class OperadoresLogicos {
    public static void main(String[] args) {
        // Ejemplos b sicos
        boolean verdadero = true;
        boolean falso = false;

        System.out.println("Tablas de verdad:");
        System.out.println("true && true = " + (verdadero &&
            verdadero));
        System.out.println("true && false = " + (verdadero &&
            falso));
        System.out.println("false && true = " + (falso &&
            verdadero));
        System.out.println("false && false = " + (falso && falso));

        System.out.println("\ntrue || true = " + (verdadero ||
            verdadero));
        System.out.println("true || false = " + (verdadero ||
            falso));
        System.out.println("false || true = " + (falso ||
            verdadero));
        System.out.println("false || false = " + (falso || falso));

        System.out.println("\n!true = " + (!verdadero));
        System.out.println("!false = " + (!falso));

        // Ejemplo pr ctico 1: Validaci n de datos
        int edad = 25;
        boolean tieneLicencia = true;
        boolean estaSobrio = true;

        boolean puedeConducir = edad >= 18 && tieneLicencia &&
            estaSobrio;
        System.out.println("\nValidaci n para conducir:");
        System.out.println("Edad: " + edad + " (>=18: " + (edad >=
            18) + ")");
        System.out.println("Tiene licencia: " + tieneLicencia);
        System.out.println("Est  sobrio: " + estaSobrio);
        System.out.println(" Puede  conducir? " + puedeConducir);

        // Ejemplo pr ctico 2: Descuento por edad
        boolean esNino = edad < 12;
        boolean esAdultoMayor = edad >= 65;

```

```
37          boolean tieneDescuento = esNino || esAdultoMayor;
38          System.out.println("\nDescuento por edad:");
39          System.out.println(" Es  ni o? " + esNino);
40          System.out.println(" Es  adulto mayor? " + esAdultoMayor);
41          System.out.println(" Tiene  descuento? " + tieneDescuento);
42
43          // Ejemplo pr ctico 3: Acceso restringido
44          boolean esEmpleado = true;
45          boolean tienePermiso = false;
46
47          boolean accesoPermitido = esEmpleado && tienePermiso;
48          boolean accesoDenegado = !accesoPermitido;
49
50          System.out.println("\nControl de acceso:");
51          System.out.println(" Es  empleado? " + esEmpleado);
52          System.out.println(" Tiene  permiso? " + tienePermiso);
53          System.out.println(" Acceso  permitido? " + accesoPermitido);
54          System.out.println(" Acceso  denegado? " + accesoDenegado);
55
56          // Operaciones combinadas
57          int numero = 15;
58          boolean condicionCompleja = (numero > 0 && numero < 20) ||
                (numero > 100 && numero < 120);
59          System.out.println("\nCondici n compleja:");
60          System.out.println("N mero: " + numero);
61          System.out.println(" Entre  0-20 o entre 100-120? " +
                condicionCompleja);
62      }
63  }
```

Listing 3.3: Ejemplos de operadores lógicos

3.5. Operadores de Asignación

```
1   public class OperadoresAsignacion {
2       public static void main(String[] args) {
3           // Operador de asignaci n b sico
4           int x = 10;
5           System.out.println("x = " + x);
6
7           // Operadores de asignaci n compuestos
8           x += 5;  // x = x + 5
9           System.out.println("x += 5  -> x = " + x);
```

Operador	Ejemplo	Equivalente	Descripción
=	x = 5	x = 5	Asignación básica
+=	x += 3	x = x + 3	Suma y asigna
-=	x -= 2	x = x - 2	Resta y asigna
*=	x *= 4	x = x * 4	Multiplica y asigna
/=	x /= 2	x = x / 2	Divide y asigna
%=	x%= 3	x = x% 3	Módulo y asigna

Cuadro 3.4: Operadores de asignación en Java

```
10
11          x -= 3;  // x = x - 3
12          System.out.println("x -= 3  -> x = " + x);
13
14          x *= 2;  // x = x * 2
15          System.out.println("x *= 2  -> x = " + x);
16
17          x /= 4;  // x = x / 4
18          System.out.println("x /= 4  -> x = " + x);
19
20          x %= 3;  // x = x % 3
21          System.out.println("x %= 3  -> x = " + x);
22
23          // Ejemplo pr ctico: acumulador
24          System.out.println("\nEjemplo: Acumulador de ventas");
25          double totalVentas = 0.0;
26
27          totalVentas += 150.75;  // Primera venta
28          System.out.println("Venta 1: $150.75 -> Total: $" +
               totalVentas);
29
30          totalVentas += 89.99;   // Segunda venta
31          System.out.println("Venta 2: $89.99  -> Total: $" +
               totalVentas);
32
33          totalVentas += 245.50;  // Tercera venta
34          System.out.println("Venta 3: $245.50 -> Total: $" +
               totalVentas);
35
36          totalVentas *= 0.9;     // Aplicar 10% de descuento
37          System.out.println("Con 10% de descuento: $" + totalVentas);
38
39          // Contador con incremento
40          System.out.println("\nEjemplo: Contador de intentos");
```

```
41          int intentos = 0;
42
43          System.out.println("Intentos: " + intentos);
44          intentos++;  // Primer intento
45          System.out.println("Despu s de intento 1: " + intentos);
46          intentos++;  // Segundo intento
47          System.out.println("Despu s de intento 2: " + intentos);
48          intentos += 3;  // Tres intentos m s
49          System.out.println("Despu s de 3 intentos m s: " +
                intentos);
50
51          // Ejemplo con Strings
52          System.out.println("\nEjemplo con Strings:");
53          String mensaje = "Hola";
54          mensaje += " ";       // Concatenaci n con +=
55          mensaje += "Mundo";
56          mensaje += "!";
57          System.out.println("Mensaje: " + mensaje);
58      }
59  }
```

Listing 3.4: Ejemplos de operadores de asignación

3.6. Precedencia de Operadores

El orden en que se evalúan los operadores es importante. Java sigue reglas de precedencia:

Precedencia	Operadores
Más alta	`()` `[]` `.` (paréntesis, corchetes, punto)
	`++` `--` `!` (incremento, decremento, NOT)
	`*` `/` `%` (multiplicación, división, módulo)
	`+` `-` (suma, resta)
	`<<` `>>` (relacionales)
	`==` `!=` (igualdad)
	`&&` (AND lógico)
Más baja	`||` (OR lógico)
	`=` `+=` `-=` `*=` `/=` `%=` (asignación)

Cuadro 3.5: Precedencia de operadores en Java (de mayor a menor)

```
1  public class PrecedenciaOperadores {
2      public static void main(String[] args) {
```

```
        // Ejemplos de precedencia
        int resultado1 = 10 + 5 * 2;
        int resultado2 = (10 + 5) * 2;

        System.out.println("10 + 5 * 2 = " + resultado1);
        System.out.println("(10 + 5) * 2 = " + resultado2);

        // Ejemplo m s complejo
        int a = 5, b = 10, c = 3;
        boolean expresion1 = a > 0 && b > 0 || c > 0;
        boolean expresion2 = a > 0 && (b > 0 || c > 0);

        System.out.println("\nExpresi n 1: a > 0 && b > 0 || c > 0
            = " + expresion1);
        System.out.println("Expresi n 2: a > 0 && (b > 0 || c > 0)
            = " + expresion2);

        // Operadores unarios
        int x = 5;
        int y = ++x * 3;  // Pre-incremento
        System.out.println("\nPre-incremento:");
        System.out.println("x = 5, y = ++x * 3");
        System.out.println("Resultado: x = " + x + ", y = " + y);

        x = 5;
        y = x++ * 3;  // Post-incremento
        System.out.println("\nPost-incremento:");
        System.out.println("x = 5, y = x++ * 3");
        System.out.println("Resultado: x = " + x + ", y = " + y);

        // Ejemplo pr ctico: f rmula f sica
        System.out.println("\nF rmula f sica: ca da libre");
        double g = 9.8;     // gravedad
        double t = 2.5;     // tiempo en segundos
        double h = 0.5 * g * t * t;  // altura = 1/2 * g * t

        System.out.println("Gravedad: " + g + " m/s ");
        System.out.println("Tiempo: " + t + " segundos");
        System.out.println("Altura: 0.5 * " + g + " * " + t + "   =
            " + h + " metros");
    }
}
```

Listing 3.5: Ejemplos de precedencia de operadores

3.7. Ejemplo Práctico: Calculadora de Ecuación Cuadrática

```
public class EcuacionCuadratica {
    public static void main(String[] args) {
        // Coeficientes de la ecuaci n: ax   + bx + c = 0
        double a = 1.0;
        double b = -5.0;
        double c = 6.0;

        System.out.println("=== SOLUCI N DE ECUACI N CUADR TICA
            ===");
        System.out.println("Ecuaci n: " + a + "x   + " + b + "x + "
            + c + " = 0");
        System.out.println();

        // Calcular discriminante: b   - 4ac
        double discriminante = b * b - 4 * a * c;
        System.out.println("Discriminante: " + b + "   - 4*" + a +
            "*" + c + " = " + discriminante);

        // Verificar el valor del discriminante
        if (discriminante > 0) {
            // Dos soluciones reales distintas
            double x1 = (-b + Math.sqrt(discriminante)) / (2 * a);
            double x2 = (-b - Math.sqrt(discriminante)) / (2 * a);

            System.out.println("Dos soluciones reales distintas:");
            System.out.println("x1 = (" + -b + " +    " +
                discriminante + ") / (2*" + a + ") = " + x1);
            System.out.println("x2 = (" + -b + " -    " +
                discriminante + ") / (2*" + a + ") = " + x2);

        } else if (discriminante == 0) {
            // Una soluci n real doble
            double x = -b / (2 * a);

            System.out.println("Una soluci n real doble:");
            System.out.println("x = " + -b + " / (2*" + a + ") = " +
                x);

        } else {
            // Soluciones complejas
```

```
35              double parteReal = -b / (2 * a);
36              double parteImaginaria = Math.sqrt(-discriminante) / (2
                    * a);
37
38              System.out.println("Dos soluciones complejas
                    conjugadas:");
39              System.out.println("x1 = " + parteReal + " + " +
                    parteImaginaria + "i");
40              System.out.println("x2 = " + parteReal + " - " +
                    parteImaginaria + "i");
41          }
42
43          // Verificar las soluciones sustituyendo
44          System.out.println("\n=== VERIFICACI N ===");
45          if (discriminante >= 0) {
46              System.out.println("Sustituyendo las soluciones en la
                    ecuaci n:");
47
48              if (discriminante > 0) {
49                  double x1 = (-b + Math.sqrt(discriminante)) / (2 *
                        a);
50                  double x2 = (-b - Math.sqrt(discriminante)) / (2 *
                        a);
51
52                  double resultado1 = a * x1 * x1 + b * x1 + c;
53                  double resultado2 = a * x2 * x2 + b * x2 + c;
54
55                  System.out.println("Para x1 = " + x1 + ":");
56                  System.out.println(a + "*(" + x1 + ")   + " + b +
                        "*(" + x1 + ") + " + c + " = " + resultado1);
57
58                  System.out.println("Para x2 = " + x2 + ":");
59                  System.out.println(a + "*(" + x2 + ")   + " + b +
                        "*(" + x2 + ") + " + c + " = " + resultado2);
60              }
61          }
62      }
63  }
```

Listing 3.6: Calculadora de ecuación cuadrática

Ejercicio Propuesto

Ejercicio 3.1: Calculadora de Descuentos

Crea un programa que calcule el precio final después de aplicar descuentos.
El programa debe:

1. Pedir el precio original
2. Aplicar un descuento del 15 % si el precio es mayor a 100
3. Aplicar un descuento adicional del 5 % si el cliente es miembro
4. Calcular el precio final
5. Mostrar: precio original, descuentos aplicados y precio final

Usa operadores aritméticos y lógicos.

Ejercicio Propuesto

Ejercicio 3.2: Validador de Contraseña

Crea un programa que valide si una contraseña cumple con:

1. Longitud mínima de 8 caracteres
2. Contiene al menos un número
3. Contiene al menos una letra mayúscula
4. Contiene al menos un carácter especial (!, @, #, $, etc.)

Usa operadores relacionales y lógicos.

Ejercicio Propuesto

Ejercicio 3.3: Convertidor de Tiempo

Crea un programa que convierta segundos a:

1. Días, horas, minutos y segundos
2. Usa operadores de división y módulo

Ejemplo: 100000 segundos = 1 día, 3 horas, 46 minutos, 40 segundos

3.8. Resumen del Capítulo

- Los operadores aritméticos realizan cálculos matemáticos
- Los operadores relacionales comparan valores y devuelven booleanos
- Los operadores lógicos combinan expresiones booleanas
- Los operadores de asignación almacenan valores en variables
- La precedencia de operadores determina el orden de evaluación
- Usa paréntesis para forzar un orden específico cuando sea necesario

Capítulo 4

Control de Flujo

4.1. Introducción al Control de Flujo

El control de flujo permite que un programa tome decisiones y repita tareas. Es fundamental para crear programas que respondan a diferentes situaciones.

Nota Importante

Sin estructuras de control, nuestros programas solo podrían ejecutar instrucciones en secuencia, una tras otra, sin adaptarse a diferentes condiciones.

4.2. Estructuras Condicionales

4.2.1. La Sentencia if

La sentencia `if` ejecuta un bloque de código solo si una condición es verdadera.

```
if (condicion) {
    // C digo que se ejecuta si la condici n es verdadera
}
```

Listing 4.1: Estructura básica del if

```
public class EjemploIf {
    public static void main(String[] args) {
        int edad = 18;

        System.out.println("Edad: " + edad);

        if (edad >= 18) {
            System.out.println("Eres mayor de edad.");
            System.out.println("Puedes obtener tu licencia de
               conducir.");
        }

```

```
12          System.out.println("Fin del programa.");
13      }
14  }
```

Listing 4.2: Ejemplo simple de if

4.2.2. La Sentencia if-else

La sentencia `if-else` permite ejecutar un bloque u otro dependiendo de una condición.

```
1  if (condicion) {
2      // C digo si la condici n es verdadera
3  } else {
4      // C digo si la condici n es falsa
5  }
```

Listing 4.3: Estructura del if-else

```
1  public class EjemploIfElse {
2      public static void main(String[] args) {
3          int numero = 7;
4
5          System.out.println("N mero: " + numero);
6
7          if (numero % 2 == 0) {
8              System.out.println("El n mero es par.");
9          } else {
10             System.out.println("El n mero es impar.");
11         }
12
13         // Otro ejemplo: calificaci n
14         int calificacion = 85;
15
16         System.out.println("\nCalificaci n: " + calificacion);
17
18         if (calificacion >= 70) {
19             System.out.println("  Aprobado  !");
20         } else {
21             System.out.println("Reprobado.");
22             System.out.println("Necesitas estudiar m s.");
23         }
24     }
25 }
```

Listing 4.4: Ejemplo de if-else

4.2.3. La Sentencia if-else if-else

Para múltiples condiciones, usamos `if-else if-else`.

```
if (condicion1) {
    // C digo si condicion1 es verdadera
} else if (condicion2) {
    // C digo si condicion2 es verdadera
} else if (condicion3) {
    // C digo si condicion3 es verdadera
} else {
    // C digo si ninguna condici n es verdadera
}
```

Listing 4.5: Estructura del if-else if-else

```
public class EjemploIfElseIf {
    public static void main(String[] args) {
        int hora = 14;  // 2 PM

        System.out.println("Hora: " + hora + ":00");

        if (hora < 12) {
            System.out.println("Buenos d as.");
        } else if (hora < 18) {
            System.out.println("Buenas tardes.");
        } else {
            System.out.println("Buenas noches.");
        }

        // Ejemplo: clasificaci n de calificaciones
        System.out.println("\n=== CLASIFICACI N DE CALIFICACIONES
           ===");
        int[] calificaciones = {95, 82, 67, 45, 101, -5};

        for (int calificacion : calificaciones) {
            System.out.print("Calificaci n " + calificacion + ": ");

            if (calificacion < 0 || calificacion > 100) {
                System.out.println("Inv lida");
```

```
            } else if (calificacion >= 90) {
                System.out.println("Excelente (A)");
            } else if (calificacion >= 80) {
                System.out.println("Bueno (B)");
            } else if (calificacion >= 70) {
                System.out.println("Aceptable (C)");
            } else if (calificacion >= 60) {
                System.out.println("Regular (D)");
            } else {
                System.out.println("Reprobado (F)");
            }
        }
    }
}
```

Listing 4.6: Ejemplo de if-else if-else

4.2.4. Ifs Anidados

Podemos colocar una sentencia `if` dentro de otra.

```
public class IfsAnidados {
    public static void main(String[] args) {
        int edad = 25;
        boolean tieneLicencia = true;
        boolean estaSobrio = false;

        System.out.println("=== VERIFICACI N PARA CONDUCIR ===");
        System.out.println("Edad: " + edad);
        System.out.println("Tiene licencia: " + tieneLicencia);
        System.out.println("Est  sobrio: " + estaSobrio);

        if (edad >= 18) {
            if (tieneLicencia) {
                if (estaSobrio) {
                    System.out.println("Puede conducir.");
                } else {
                    System.out.println("No puede conducir: no est
                        sobrio.");
                }
            } else {
                System.out.println("No puede conducir: no tiene
                    licencia.");
            }
```

```
22          } else {
23              System.out.println("No puede conducir: es menor de
                    edad.");
24          }
25
26          // Equivalente con operadores l gicos
27          System.out.println("\n=== VERSI N CON OPERADORES L GICOS
                ===");
28          if (edad >= 18 && tieneLicencia && estaSobrio) {
29              System.out.println("Puede conducir.");
30          } else {
31              System.out.println("No puede conducir.");
32
33              if (edad < 18) {
34                  System.out.println("Motivo: es menor de edad.");
35              } else if (!tieneLicencia) {
36                  System.out.println("Motivo: no tiene licencia.");
37              } else {
38                  System.out.println("Motivo: no est  sobrio.");
39              }
40          }
41      }
42  }
```

Listing 4.7: Ejemplo de ifs anidados

4.3. La Sentencia switch

La sentencia `switch` es una alternativa a múltiples `if-else if` cuando se compara una variable contra valores específicos.

```
1  switch (variable) {
2      case valor1:
3          // C digo si variable == valor1
4          break;
5      case valor2:
6          // C digo si variable == valor2
7          break;
8      case valor3:
9          // C digo si variable == valor3
10         break;
11     default:
12         // C digo si no coincide con ning n case
13 }
```

Listing 4.8: Estructura del switch

```
public class EjemploSwitch {
    public static void main(String[] args) {
        // Ejemplo 1: D as de la semana
        int diaSemana = 3;

        System.out.println("D  a  n mero: " + diaSemana);

        switch (diaSemana) {
            case 1:
                System.out.println("Lunes");
                break;
            case 2:
                System.out.println("Martes");
                break;
            case 3:
                System.out.println("Mi rcoles");
                break;
            case 4:
                System.out.println("Jueves");
                break;
            case 5:
                System.out.println("Viernes");
                break;
            case 6:
                System.out.println("S bado");
                break;
            case 7:
                System.out.println("Domingo");
                break;
            default:
                System.out.println("D  a  inv lido. Debe ser 1-7.");
        }

        // Ejemplo 2: Calculadora simple
        System.out.println("\n=== CALCULADORA SIMPLE ===");
        char operacion = '*';
        double num1 = 10.5;
        double num2 = 5.2;
        double resultado = 0;

        System.out.println("Operaci n: " + num1 + " " + operacion +
```

```
            " " + num2);

        switch (operacion) {
            case '+':
                resultado = num1 + num2;
                System.out.println("Resultado: " + resultado);
                break;
            case '-':
                resultado = num1 - num2;
                System.out.println("Resultado: " + resultado);
                break;
            case '*':
                resultado = num1 * num2;
                System.out.println("Resultado: " + resultado);
                break;
            case '/':
                if (num2 != 0) {
                    resultado = num1 / num2;
                    System.out.println("Resultado: " + resultado);
                } else {
                    System.out.println("Error: Divisi n por cero.");
                }
                break;
            default:
                System.out.println("Operaci n no v lida.");
        }

        // Ejemplo 3: switch sin break (caso ca da)
        System.out.println("\n=== EVALUACI N DE RANGO ===");
        int puntaje = 85;

        System.out.println("Puntaje: " + puntaje);
        System.out.print("Categor a: ");

        switch (puntaje / 10) {
            case 10:
            case 9:
                System.out.println("Excelente");
                break;
            case 8:
                System.out.println("Muy Bueno");
                break;
            case 7:
                System.out.println("Bueno");
                break;
```

```
            case 6:
                System.out.println("Suficiente");
                break;
            default:
                System.out.println("Necesita mejorar");
        }
    }
}
```

Listing 4.9: Ejemplo de switch

¡Advertencia!

¡No olvides los break! Si omites `break` en un caso, el programa continuará ejecutando los casos siguientes (fall-through).

4.4. Bucles (Loops)

Los bucles permiten repetir un bloque de código múltiples veces.

4.4.1. El Bucle for

El bucle `for` se usa cuando sabemos exactamente cuántas veces queremos repetir algo.

```
for (inicializaci n; condici n; actualizaci n) {
    // C digo que se repite
}
```

Listing 4.10: Estructura del for

```
public class EjemploFor {
    public static void main(String[] args) {
        // Ejemplo 1: Contar del 1 al 10
        System.out.println("Contando del 1 al 10:");
        for (int i = 1; i <= 10; i++) {
            System.out.print(i + " ");
        }
        System.out.println();

        // Ejemplo 2: Contar hacia atr s
        System.out.println("\nContando del 10 al 1:");
        for (int i = 10; i >= 1; i--) {
            System.out.print(i + " ");
```

```
        }
        System.out.println();

        // Ejemplo 3: Sumar n meros
        System.out.println("\nSumando n meros del 1 al 100:");
        int suma = 0;
        for (int i = 1; i <= 100; i++) {
            suma += i;
        }
        System.out.println("La suma es: " + suma);

        // Ejemplo 4: Tablas de multiplicar
        System.out.println("\n=== TABLAS DE MULTIPLICAR ===");
        for (int tabla = 1; tabla <= 5; tabla++) {
            System.out.println("\nTabla del " + tabla + ":");
            for (int i = 1; i <= 10; i++) {
                System.out.println(tabla + "    " + i + " = " +
                    (tabla * i));
            }
        }

        // Ejemplo 5: Factorial
        System.out.println("\n=== FACTORIALES ===");
        for (int n = 1; n <= 10; n++) {
            long factorial = 1;
            for (int i = 1; i <= n; i++) {
                factorial *= i;
            }
            System.out.println(n + "! = " + factorial);
        }
    }
}
```

Listing 4.11: Ejemplos de for

4.4.2. El Bucle while

El bucle `while` se usa cuando no sabemos exactamente cuántas veces se repetirá el código, solo sabemos la condición para continuar.

```
while (condicion) {
    // C digo que se repite mientras la condici n sea verdadera
}
```

Listing 4.12: Estructura del while

```java
public class EjemploWhile {
    public static void main(String[] args) {
        // Ejemplo 1: Contador
        System.out.println("Contador con while:");
        int contador = 1;

        while (contador <= 10) {
            System.out.print(contador + " ");
            contador++;  // No olvidar actualizar la variable
        }
        System.out.println();

        // Ejemplo 2: Suma hasta cierto l mite
        System.out.println("\nSuma hasta llegar a 100:");
        int suma = 0;
        int numero = 1;

        while (suma <= 100) {
            suma += numero;
            System.out.println("Sumando " + numero + ", suma total:
                " + suma);
            numero++;
        }
        System.out.println("Se necesitaron " + (numero - 1) + "
            n meros para superar 100.");

        // Ejemplo 3: Adivinar n mero
        System.out.println("\n=== JUEGO DE ADIVINAR ===");
        int numeroSecreto = 42;
        int intento = 0;
        int intentosRealizados = 0;

        java.util.Scanner scanner = new java.util.Scanner(System.in);

        System.out.println("Adivina el n mero (1-100):");

        while (intento != numeroSecreto) {
            System.out.print("Tu intento: ");
            intento = scanner.nextInt();
            intentosRealizados++;

```

```
40              if (intento < numeroSecreto) {
41                  System.out.println("Demasiado bajo. Intenta de
                        nuevo.");
42              } else if (intento > numeroSecreto) {
43                  System.out.println("Demasiado alto. Intenta de
                        nuevo.");
44              } else {
45                  System.out.println(" Correcto ! Adivinaste en " +
                        intentosRealizados + " intentos.");
46              }
47          }
48
49          // Ejemplo 4: Validaci n de entrada
50          System.out.println("\n=== VALIDACI N DE ENTRADA ===");
51          int edad = -1;
52
53          while (edad < 0 || edad > 120) {
54              System.out.print("Ingresa una edad v lida (0-120): ");
55              edad = scanner.nextInt();
56
57              if (edad < 0 || edad > 120) {
58                  System.out.println("Edad inv lida. Intenta de
                        nuevo.");
59              }
60          }
61
62          System.out.println("Edad aceptada: " + edad);
63
64          scanner.close();
65      }
66  }
```

Listing 4.13: Ejemplos de while

4.4.3. El Bucle do-while

El bucle `do-while` es similar al `while`, pero garantiza que el código se ejecute al menos una vez.

```
1  do {
2      // C digo que se repite
3  } while (condicion);
```

Listing 4.14: Estructura del do-while

```
public class EjemploDoWhile {
    public static void main(String[] args) {
        // Ejemplo 1: Men  interactivo
        System.out.println("=== MEN  INTERACTIVO ===");

        java.util.Scanner scanner = new java.util.Scanner(System.in);
        int opcion;

        do {
            System.out.println("\nSelecciona una opci n:");
            System.out.println("1. Ver saludo");
            System.out.println("2. Ver la fecha");
            System.out.println("3. Salir");
            System.out.print("Opci n: ");

            opcion = scanner.nextInt();

            switch (opcion) {
                case 1:
                    System.out.println(" Hola ! Bienvenido al
                       programa.");
                    break;
                case 2:
                    System.out.println("Hoy es " +
                       java.time.LocalDate.now());
                    break;
                case 3:
                    System.out.println("Saliendo del programa...");
                    break;
                default:
                    System.out.println("Opci n inv lida. Intenta
                       de nuevo.");
            }

        } while (opcion != 3);

        // Ejemplo 2: Asegurar entrada positiva
        System.out.println("\n=== ENTRADA POSITIVA ===");
        int numero;

        do {
            System.out.print("Ingresa un n mero positivo: ");
            numero = scanner.nextInt();

```

```
            if (numero <= 0) {
                System.out.println("El n mero debe ser positivo.
                    Intenta de nuevo.");
            }

        } while (numero <= 0);

        System.out.println("N mero aceptado: " + numero);

        // Ejemplo 3: Generar n meros aleatorios
        System.out.println("\n=== N MEROS ALEATORIOS ===");
        java.util.Random random = new java.util.Random();
        int numeroAleatorio;
        int intentos = 0;

        System.out.println("Generando n meros hasta obtener un
            7...");

        do {
            numeroAleatorio = random.nextInt(10) + 1;  // 1-10
            System.out.println("N mero generado: " +
                numeroAleatorio);
            intentos++;
        } while (numeroAleatorio != 7);

        System.out.println(" Se  necesitaron " + intentos + "
            intentos para obtener un 7!");

        scanner.close();
    }
}
```

Listing 4.15: Ejemplos de do-while

4.5. Control de Bucles: break y continue

4.5.1. La Sentencia break

La sentencia `break` termina un bucle o sentencia `switch` inmediatamente.

```
public class EjemploBreak {
    public static void main(String[] args) {
        // Ejemplo 1: Buscar un n mero
        System.out.println("Buscando el n mero 5:");
```

```
5
6           for (int i = 1; i <= 10; i++) {
7               System.out.println("Probando n mero: " + i);
8
9               if (i == 5) {
10                  System.out.println("  Encontrado  ! Terminando
                        b squeda.");
11                  break;
12              }
13          }
14
15          // Ejemplo 2: Suma con l mite
16          System.out.println("\nSumando n meros hasta que la suma sea
                > 50:");
17
18          int suma = 0;
19          for (int i = 1; i <= 100; i++) {
20              suma += i;
21              System.out.println("Sumando " + i + ", suma total: " +
                    suma);
22
23              if (suma > 50) {
24                  System.out.println("L mite alcanzado. Terminando.");
25                  break;
26              }
27          }
28
29          // Ejemplo 3: Bucles anidados con break
30          System.out.println("\n=== BUSQUEDA EN MATRIZ ===");
31
32          int[][] matriz = {
33              {1, 2, 3},
34              {4, 5, 6},
35              {7, 8, 9}
36          };
37
38          int buscar = 5;
39          boolean encontrado = false;
40
41          busqueda:
42          for (int i = 0; i < matriz.length; i++) {
43              for (int j = 0; j < matriz[i].length; j++) {
44                  System.out.println("Buscando en posici n [" + i +
                        "][" + j + "]: " + matriz[i][j]);
45
```

```
                if (matriz[i][j] == buscar) {
                    System.out.println("  Encontrado   " + buscar + "
                        en posici n [" + i + "][" + j + "]!");
                    encontrado = true;
                    break busqueda;  // Break con etiqueta
                }
            }
        }

        if (!encontrado) {
            System.out.println(buscar + " no encontrado en la
                matriz.");
        }
    }
}
```

Listing 4.16: Ejemplos de break

4.5.2. La Sentencia continue

La sentencia `continue` salta a la siguiente iteración del bucle, omitiendo el código restante.

```
public class EjemploContinue {
    public static void main(String[] args) {
        // Ejemplo 1: Mostrar solo n meros pares
        System.out.println("N meros pares del 1 al 20:");

        for (int i = 1; i <= 20; i++) {
            if (i % 2 != 0) {
                continue;  // Saltar n meros impares
            }
            System.out.print(i + " ");
        }
        System.out.println();

        // Ejemplo 2: Procesar solo elementos v lidos
        System.out.println("\nProcesando lista de n meros:");

        int[] numeros = {10, -5, 0, 7, -3, 15, 0, 8};
        int sumaPositivos = 0;
        int cuentaPositivos = 0;

        for (int numero : numeros) {
```

```
22              if (numero <= 0) {
23                  continue;  // Saltar n meros no positivos
24              }
25
26              sumaPositivos += numero;
27              cuentaPositivos++;
28              System.out.println("Procesando n mero positivo: " +
                    numero);
29          }
30
31          System.out.println("Total de n meros positivos: " +
                cuentaPositivos);
32          System.out.println("Suma de n meros positivos: " +
                sumaPositivos);
33
34          // Ejemplo 3: Validaci n en bucle
35          System.out.println("\n=== VALIDACI N DE DATOS ===");
36
37          String[] nombres = {"Ana", "", "Carlos", null, "Elena", "
                ", "Fernando"};
38
39          for (String nombre : nombres) {
40              if (nombre == null || nombre.trim().isEmpty()) {
41                  System.out.println("Nombre inv lido encontrado,
                        omitiendo...");
42                  continue;
43              }
44
45              System.out.println("Procesando nombre: " + nombre);
46              System.out.println("Longitud: " + nombre.length());
47              System.out.println("En may sculas: " +
                    nombre.toUpperCase());
48              System.out.println();
49          }
50
51          // Ejemplo 4: Combinaci n de break y continue
52          System.out.println("=== BUSQUEDA CON LIMITE ===");
53
54          int limite = 1000;
55          int objetivo = 50;
56          int encontrados = 0;
57
58          for (int i = 1; i <= limite; i++) {
59              if (i % 7 != 0) {
60                  continue;  // Solo procesar m ltiplos de 7
```

```
61              }
62
63              if (i > objetivo) {
64                  System.out.println("L mite de valor alcanzado.
                       Terminando.");
65                  break;
66              }
67
68              encontrados++;
69              System.out.println("M ltiplo de 7 encontrado: " + i);
70          }
71
72          System.out.println("Total de m ltiplos de 7 hasta " +
               objetivo + ": " + encontrados);
73      }
74  }
```

Listing 4.17: Ejemplos de continue

4.6. Bucles Anidados

Los bucles anidados son bucles dentro de otros bucles. Son útiles para trabajar con estructuras multidimensionales.

```
1  public class BuclesAnidados {
2      public static void main(String[] args) {
3          // Ejemplo 1: Tabla de multiplicar completa
4          System.out.println("=== TABLA DE MULTIPLICAR COMPLETA ===");
5
6          for (int i = 1; i <= 10; i++) {
7              for (int j = 1; j <= 10; j++) {
8                  System.out.printf("%3d    %2d = %3d  ", i, j, i * j);
9              }
10             System.out.println();
11         }
12
13         // Ejemplo 2: Patr n de asteriscos
14         System.out.println("\n=== PATR N DE ASTERISCOS ===");
15
16         int filas = 5;
17         for (int i = 1; i <= filas; i++) {
18             // Espacios
19             for (int j = filas; j > i; j--) {
20                 System.out.print(" ");
```

```
            }

            // Asteriscos
            for (int j = 1; j <= (2 * i - 1); j++) {
                System.out.print("*");
            }

            System.out.println();
        }

        // Ejemplo 3: Buscar en matriz
        System.out.println("\n=== BUSCAR EN MATRIZ ===");

        int[][] matriz = {
            {12, 45, 23},
            {67, 89, 34},
            {56, 78, 90}
        };

        int buscar = 89;
        boolean encontrado = false;
        int filaEncontrado = -1;
        int columnaEncontrado = -1;

        for (int i = 0; i < matriz.length; i++) {
            for (int j = 0; j < matriz[i].length; j++) {
                System.out.println("Buscando en [" + i + "][" + j +
                    "]: " + matriz[i][j]);

                if (matriz[i][j] == buscar) {
                    encontrado = true;
                    filaEncontrado = i;
                    columnaEncontrado = j;
                    break;  // Solo sale del bucle interno
                }
            }

            if (encontrado) {
                break;  // Sale del bucle externo
            }
        }

        if (encontrado) {
            System.out.println("  Encontrado   " + buscar + " en [" +
                filaEncontrado + "][" + columnaEncontrado + "]!");
```

```
64          } else {
65              System.out.println(buscar + " no encontrado en la
                    matriz.");
66          }
67
68          // Ejemplo 4: Tri ngulo de n meros
69          System.out.println("\n=== TRI NGULO DE N MEROS ===");
70
71          int n = 5;
72          for (int i = 1; i <= n; i++) {
73              // N meros
74              for (int j = 1; j <= i; j++) {
75                  System.out.print(j + " ");
76              }
77
78              // Espacios
79              for (int j = 1; j <= (n - i) * 2; j++) {
80                  System.out.print("  ");
81              }
82
83              // N meros inversos
84              for (int j = i; j >= 1; j--) {
85                  System.out.print(j + " ");
86              }
87
88              System.out.println();
89          }
90      }
91  }
```

Listing 4.18: Ejemplos de bucles anidados

4.7. Ejemplo Práctico: Generador de Tablas de Multiplicar

```
1  public class GeneradorTablasMultiplicar {
2      public static void main(String[] args) {
3          java.util.Scanner scanner = new java.util.Scanner(System.in);
4
5          System.out.println("=== GENERADOR DE TABLAS DE MULTIPLICAR
               ===");
6
```

```
7           boolean continuar = true;
8
9           while (continuar) {
10              mostrarMenu();
11              int opcion = leerEntero(scanner, "Selecciona una
                    opci n: ");
12
13              switch (opcion) {
14                  case 1:
15                      generarTablaIndividual(scanner);
16                      break;
17                  case 2:
18                      generarTablasConsecutivas(scanner);
19                      break;
20                  case 3:
21                      generarTablaCompleta();
22                      break;
23                  case 4:
24                      generarTablaPersonalizada(scanner);
25                      break;
26                  case 5:
27                      System.out.println(" Gracias  por usar el
                            generador!");
28                      continuar = false;
29                      break;
30                  default:
31                      System.out.println("Opci n inv lida. Intenta
                            de nuevo.");
32              }
33
34              if (continuar) {
35                  System.out.println("\nPresiona Enter para
                        continuar...");
36                  scanner.nextLine();
37                  scanner.nextLine();
38              }
39          }
40
41          scanner.close();
42      }
43
44      public static void mostrarMenu() {
45          System.out.println("\n=== MEN  PRINCIPAL ===");
46          System.out.println("1. Generar una tabla individual");
47          System.out.println("2. Generar tablas consecutivas");
```

```
48          System.out.println("3. Generar tabla completa (1-10)");
49          System.out.println("4. Tabla personalizada");
50          System.out.println("5. Salir");
51          System.out.println("======================");
52      }
53
54      public static int leerEntero(java.util.Scanner scanner, String
            mensaje) {
55          int numero = 0;
56          boolean valido = false;
57
58          while (!valido) {
59              try {
60                  System.out.print(mensaje);
61                  numero = scanner.nextInt();
62                  valido = true;
63              } catch (Exception e) {
64                  System.out.println("Error: Debe ingresar un n mero
                        entero.");
65                  scanner.nextLine();
66              }
67          }
68
69          return numero;
70      }
71
72      public static void generarTablaIndividual(java.util.Scanner
            scanner) {
73          System.out.println("\n=== TABLA INDIVIDUAL ===");
74          int numero = leerEntero(scanner, "  Qu   tabla quieres
                generar? (1-20): ");
75
76          if (numero < 1 || numero > 20) {
77              System.out.println("N mero fuera de rango. Usando 10
                    por defecto.");
78              numero = 10;
79          }
80
81          System.out.println("\nTabla del " + numero + ":");
82          System.out.println("====================");
83
84          for (int i = 1; i <= 10; i++) {
85              System.out.printf("%2d    %2d = %3d%n", numero, i,
                    numero * i);
86          }
```

```
87      }
88
89      public static void generarTablasConsecutivas(java.util.Scanner
            scanner) {
90          System.out.println("\n=== TABLAS CONSECUTIVAS ===");
91          int inicio = leerEntero(scanner, " Desde  qu  tabla?
                (1-20): ");
92          int fin = leerEntero(scanner, " Hasta  qu  tabla? (1-20):
                ");
93
94          // Validar rangos
95          if (inicio < 1) inicio = 1;
96          if (fin > 20) fin = 20;
97          if (inicio > fin) {
98              int temp = inicio;
99              inicio = fin;
100             fin = temp;
101         }
102
103         System.out.println("\nGenerando tablas del " + inicio + " al
                " + fin + ":");
104
105         for (int tabla = inicio; tabla <= fin; tabla++) {
106             System.out.println("\nTabla del " + tabla + ":");
107             System.out.println("------------");
108
109             for (int i = 1; i <= 10; i++) {
110                 System.out.printf("%2d    %2d = %3d%n", tabla, i,
                        tabla * i);
111             }
112         }
113     }
114
115     public static void generarTablaCompleta() {
116         System.out.println("\n=== TABLA COMPLETA (1-10) ===");
117
118         System.out.print("    ");
119         for (int i = 1; i <= 10; i++) {
120             System.out.printf("%4d", i);
121         }
122         System.out.println();
123
124         System.out.print("    ");
125         for (int i = 1; i <= 10; i++) {
126             System.out.print("----");
```

```
127         }
128         System.out.println();
129 
130         for (int i = 1; i <= 10; i++) {
131             System.out.printf("%2d |", i);
132 
133             for (int j = 1; j <= 10; j++) {
134                 System.out.printf("%4d", i * j);
135             }
136             System.out.println();
137         }
138     }
139 
140     public static void generarTablaPersonalizada(java.util.Scanner
        scanner) {
141         System.out.println("\n=== TABLA PERSONALIZADA ===");
142 
143         int numero = leerEntero(scanner, "  Qu   tabla? (1-100): ");
144         int desde = leerEntero(scanner, " Desde   qu   n mero?
            (1-100): ");
145         int hasta = leerEntero(scanner, " Hasta   qu   n mero?
            (1-100): ");
146         int incremento = leerEntero(scanner, "  Incremento  ? (1-10):
            ");
147 
148         // Validaciones
149         if (numero < 1) numero = 1;
150         if (numero > 100) numero = 100;
151         if (desde < 1) desde = 1;
152         if (hasta > 100) hasta = 100;
153         if (desde > hasta) {
154             int temp = desde;
155             desde = hasta;
156             hasta = temp;
157         }
158         if (incremento < 1) incremento = 1;
159         if (incremento > 10) incremento = 10;
160 
161         System.out.println("\nTabla del " + numero + " (de " + desde
            + " a " + hasta + " con incremento " + incremento +
            "):");
162         System.out.println("================================================");
163 
164         for (int i = desde; i <= hasta; i += incremento) {
```

```
165                 System.out.printf("%3d     %3d = %6d%n", numero, i,
                        numero * i);
166             }
167         }
168 }
```

Listing 4.19: Generador completo de tablas de multiplicar

Ejercicio Propuesto

Ejercicio 4.1: Calculadora de Factorial

Crea un programa que calcule el factorial de un número usando:

1. Bucle `for`
2. Bucle `while`
3. Bucle `do-while`

El factorial de n (n!) es: n × (n-1) × (n-2) × ... × 1 Ejemplo: 5! = 5 × 4 × 3 × 2 × 1 = 120

Ejercicio Propuesto

Ejercicio 4.2: Juego del Ahorcado Simple

Crea un juego del ahorcado simple que:

1. Tenga una palabra secreta
2. Permita al usuario adivinar letras
3. Muestre el progreso (letras adivinadas, letras incorrectas)
4. Tenga un límite de intentos
5. Use bucles para controlar el juego

Ejercicio Propuesto

Ejercicio 4.3: Generador de Números Primos

Crea un programa que:

1. Genere todos los números primos hasta un límite dado
2. Use bucles anidados para verificar si cada número es primo

3. Optimice el bucle interno verificando solo hasta la raíz cuadrada del número
4. Muestre los números primos encontrados

4.8. Resumen del Capítulo

- Las sentencias `if`, `if-else` y `if-else if-else` permiten tomar decisiones
- La sentencia `switch` es útil para comparar una variable contra múltiples valores
- Los bucles (`for`, `while`, `do-while`) permiten repetir código
- El bucle `for` se usa cuando sabemos cuántas iteraciones necesitamos
- Los bucles `while` y `do-while` se usan cuando la condición de parada depende de algo que ocurre durante la ejecución
- La sentencia `break` termina un bloop o `switch`
- La sentencia `continue` salta a la siguiente iteración de un bucle
- Los bucles anidados son útiles para trabajar con estructuras multidimensionales

Capítulo 5

Entrada y Salida por Consola

5.1. Introducción a la Entrada/Salida (I/O)

La comunicación con el usuario es fundamental para programas interactivos. En este capítulo aprenderemos a leer datos del teclado y mostrar información en la consola.

Nota Importante

En programación lineal, la consola es nuestra principal interfaz con el usuario. Aprender a manejarla correctamente es esencial para crear programas útiles.

5.2. Salida por Consola

Ya hemos usado `System.out.println()`, pero Java ofrece más opciones para formatear la salida.

5.2.1. System.out.print() vs System.out.println()

```
1  public class PrintVsPrintln {
2      public static void main(String[] args) {
3          // System.out.print() - NO a ade salto de l nea
4          System.out.print("Hola ");
5          System.out.print("Mundo");
6          System.out.print("!");
7
8          // System.out.println() - S   a ade salto de l nea
9          System.out.println("\n---");
10         System.out.println("Hola");
11         System.out.println("Mundo");
12         System.out.println("!");
13
14         // Combinaci n de ambos
15         System.out.println("\n=== DATOS PERSONALES ===");
16         System.out.print("Nombre: ");
```

```
17          System.out.println("Ana Garc a");
18          System.out.print("Edad: ");
19          System.out.println(25);
20          System.out.print("Ciudad: ");
21          System.out.println("Madrid");
22      }
23  }
```

Listing 5.1: Diferencia entre print y println

5.2.2. System.out.printf() - Formateo Avanzado

El método `printf()` permite formatear la salida de manera similar a C/C++.

```
1  public class PrintfEjemplo {
2      public static void main(String[] args) {
3          // Formateo b sico
4          System.out.println("=== FORMATEO B SICO ===");
5          System.out.printf("Entero: %d%n", 123);
6          System.out.printf("Decimal: %f%n", 123.456);
7          System.out.printf("Cadena: %s%n", "Hola Mundo");
8          System.out.printf("Car cter: %c%n", 'A');
9          System.out.printf("Booleano: %b%n", true);
10
11         // Especificar ancho y precisi n
12         System.out.println("\n=== ANCHO Y PRECISI N ===");
13         System.out.printf("|%10d|%n", 123);           // 10
               caracteres de ancho
14         System.out.printf("|%-10d|%n", 123);          // Alineaci n
               izquierda
15         System.out.printf("|%10.2f|%n", 123.4567);    // 10 ancho, 2
               decimales
16         System.out.printf("|%10s|%n", "Hola");        // Cadena con
               ancho
17
18         // M ltiples variables
19         System.out.println("\n=== M LTIPLES VARIABLES ===");
20         String nombre = "Carlos";
21         int edad = 30;
22         double salario = 2500.75;
23
24         System.out.printf("Nombre: %s, Edad: %d, Salario: $%.2f%n",
25                           nombre, edad, salario);
26
```

```
27          // Tabla formateada
28          System.out.println("\n=== TABLA DE PRODUCTOS ===");
29          System.out.printf("%-20s %-10s %-10s%n", "Producto",
                "Precio", "Cantidad");
30          System.out.printf("%-20s %-10s %-10s%n",
                "--------------------", "----------", "----------");
31          System.out.printf("%-20s $%-9.2f %-10d%n", "Laptop", 999.99,
                5);
32          System.out.printf("%-20s $%-9.2f %-10d%n", "Mouse", 25.50,
                20);
33          System.out.printf("%-20s $%-9.2f %-10d%n", "Teclado", 45.75,
                15);
34          System.out.printf("%-20s $%-9.2f %-10d%n", "Monitor",
                299.99, 8);
35
36          // Formateo de fechas y horas
37          System.out.println("\n=== FECHAS Y HORAS ===");
38          java.util.Date fecha = new java.util.Date();
39          System.out.printf("Fecha y hora: %tF %tT%n", fecha, fecha);
40          System.out.printf("Solo fecha: %tD%n", fecha);
41          System.out.printf("Solo hora: %tr%n", fecha);
42      }
43  }
```

Listing 5.2: Uso de printf para formateo

5.3. Entrada por Consola con Scanner

La clase `Scanner` nos permite leer datos del teclado de manera sencilla.

5.3.1. Importar y Crear Scanner

```
1  // Necesitamos importar la clase Scanner
2  import java.util.Scanner;
3
4  public class ScannerBasico {
5      public static void main(String[] args) {
6          // Crear un objeto Scanner para leer del teclado
7          Scanner scanner = new Scanner(System.in);
8
9          System.out.println("=== EJEMPLO B SICO DE SCANNER ===");
10
11         // Leer diferentes tipos de datos
```

```
        System.out.print("Ingresa tu nombre: ");
        String nombre = scanner.nextLine();

        System.out.print("Ingresa tu edad: ");
        int edad = scanner.nextInt();

        System.out.print("Ingresa tu altura (en metros): ");
        double altura = scanner.nextDouble();

        System.out.print(" Eres  estudiante? (true/false): ");
        boolean esEstudiante = scanner.nextBoolean();

        // Mostrar los datos ingresados
        System.out.println("\n=== RESUMEN ===");
        System.out.println("Nombre: " + nombre);
        System.out.println("Edad: " + edad);
        System.out.println("Altura: " + altura + " m");
        System.out.println("Estudiante: " + esEstudiante);

        // Importante: cerrar el Scanner cuando ya no se use
        scanner.close();
    }
}
```

Listing 5.3: Importación y creación de Scanner

5.3.2. Métodos de Scanner

Método	Tipo de Dato	Descripción
`nextLine()`	`String`	Lee una línea completa de texto
`next()`	`String`	Lee la siguiente palabra (hasta espacio)
`nextInt()`	`int`	Lee un número entero
`nextDouble()`	`double`	Lee un número decimal
`nextBoolean()`	`boolean`	Lee un valor booleano
`nextFloat()`	`float`	Lee un número decimal de precisión simple
`nextLong()`	`long`	Lee un número entero largo
`nextShort()`	`short`	Lee un número entero corto
`nextByte()`	`byte`	Lee un byte

Cuadro 5.1: Métodos principales de la clase Scanner

```
import java.util.Scanner;

```

```java
public class MetodosScanner {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("=== DIFERENTES M TODOS DE SCANNER ===");

        // Diferencia entre next() y nextLine()
        System.out.println("\n--- Diferencia next() vs nextLine()
            ---");
        System.out.print("Ingresa tu nombre completo: ");

        // next() solo lee hasta el primer espacio
        String nombre1 = scanner.next();
        System.out.println("Usando next(): '" + nombre1 + "'");

        // Limpiar el buffer
        scanner.nextLine();

        System.out.print("Ingresa tu nombre completo otra vez: ");
        String nombre2 = scanner.nextLine();
        System.out.println("Usando nextLine(): '" + nombre2 + "'");

        // Leer m ltiples valores en una l nea
        System.out.println("\n--- M ltiples valores en una l nea
            ---");
        System.out.print("Ingresa tres n meros separados por
            espacios: ");

        int num1 = scanner.nextInt();
        int num2 = scanner.nextInt();
        int num3 = scanner.nextInt();

        System.out.println("N meros ingresados: " + num1 + ", " +
            num2 + ", " + num3);
        System.out.println("Suma: " + (num1 + num2 + num3));
        System.out.println("Promedio: " + (num1 + num2 + num3) /
            3.0);

        // Verificar si hay m s datos disponibles
        System.out.println("\n--- Verificaci n de datos disponibles
            ---");
        System.out.print("Ingresa varias palabras: ");

        int contadorPalabras = 0;
        while (scanner.hasNext()) {
```

```
            String palabra = scanner.next();
            System.out.println("Palabra " + (++contadorPalabras) +
                ": " + palabra);

            // Detener despu s de 5 palabras para el ejemplo
            if (contadorPalabras >= 5) {
                break;
            }
        }

        // Limpiar buffer despu s de next()
        scanner.nextLine();

        scanner.close();
    }
}
```

Listing 5.4: Ejemplos de métodos de Scanner

5.4. Problemas Comunes con la Entrada

5.4.1. El Problema del Buffer (nextLine después de nextInt)

```
import java.util.Scanner;

public class ProblemaBuffer {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("=== PROBLEMA COM N DEL BUFFER ===");

        // Ejemplo del problema
        System.out.print("Ingresa tu edad: ");
        int edad = scanner.nextInt();

        System.out.print("Ingresa tu nombre: ");
        String nombre = scanner.nextLine();  //  PROBLEMA ! Lee el
            salto de l nea

        System.out.println("\nResultado (incorrecto):");
        System.out.println("Edad: " + edad);
        System.out.println("Nombre: '" + nombre + "'");  // Vac o o
            solo salto de l nea
```

```
19
20          System.out.println("\n=== SOLUCIONES ===");
21
22          // Soluci n 1: Usar nextLine() para leer el salto de l nea
                sobrante
23          System.out.print("Ingresa tu edad otra vez: ");
24          edad = scanner.nextInt();
25          scanner.nextLine();  // Leer el salto de l nea sobrante
26
27          System.out.print("Ingresa tu nombre: ");
28          nombre = scanner.nextLine();
29
30          System.out.println("\nResultado (con soluci n 1):");
31          System.out.println("Edad: " + edad);
32          System.out.println("Nombre: '" + nombre + "'");
33
34          // Soluci n 2: Leer todo como String y convertir
35          System.out.print("\nIngresa tu edad (como soluci n 2): ");
36          String edadStr = scanner.nextLine();
37          edad = Integer.parseInt(edadStr);  // Convertir a int
38
39          System.out.print("Ingresa tu nombre: ");
40          nombre = scanner.nextLine();
41
42          System.out.println("\nResultado (con soluci n 2):");
43          System.out.println("Edad: " + edad);
44          System.out.println("Nombre: '" + nombre + "'");
45
46          scanner.close();
47      }
48  }
```

Listing 5.5: El problema común del buffer

5.4.2. Validación de Entrada

```
1  import java.util.Scanner;
2  import java.util.InputMismatchException;
3
4  public class ValidacionEntrada {
5      public static void main(String[] args) {
6          Scanner scanner = new Scanner(System.in);
7
8          System.out.println("=== VALIDACI N DE ENTRADA ===");
```

```
9
10          // Validar n mero entero positivo
11          int edad = 0;
12          boolean entradaValida = false;
13
14          while (!entradaValida) {
15              try {
16                  System.out.print("Ingresa tu edad (n mero
                        positivo): ");
17                  edad = scanner.nextInt();
18
19                  if (edad > 0 && edad <= 120) {
20                      entradaValida = true;
21                  } else {
22                      System.out.println("Error: La edad debe estar
                            entre 1 y 120.");
23                  }
24              } catch (InputMismatchException e) {
25                  System.out.println("Error: Debes ingresar un n mero
                        entero.");
26                  scanner.nextLine();  // Limpiar buffer
27              }
28          }
29
30          System.out.println("Edad aceptada: " + edad);
31
32          // Validar n mero decimal en rango
33          double precio = 0;
34          entradaValida = false;
35
36          while (!entradaValida) {
37              try {
38                  System.out.print("\nIngresa un precio (entre 0.01 y
                        1000): ");
39                  precio = scanner.nextDouble();
40
41                  if (precio >= 0.01 && precio <= 1000) {
42                      entradaValida = true;
43                  } else {
44                      System.out.println("Error: El precio debe estar
                            entre 0.01 y 1000.");
45                  }
46              } catch (InputMismatchException e) {
47                  System.out.println("Error: Debes ingresar un n mero
                        v lido.");
```

```
48                  scanner.nextLine();  // Limpiar buffer
49              }
50          }
51
52          System.out.printf("Precio aceptado: $%.2f%n", precio);
53
54          // Validar opci n de men
55          int opcion = 0;
56          entradaValida = false;
57
58          while (!entradaValida) {
59              try {
60                  System.out.println("\n=== MEN  ===");
61                  System.out.println("1. Opci n A");
62                  System.out.println("2. Opci n B");
63                  System.out.println("3. Opci n C");
64                  System.out.print("Selecciona una opci n (1-3): ");
65
66                  opcion = scanner.nextInt();
67
68                  if (opcion >= 1 && opcion <= 3) {
69                      entradaValida = true;
70                  } else {
71                      System.out.println("Error: Opci n inv lida.
                           Debe ser 1, 2 o 3.");
72                  }
73              } catch (InputMismatchException e) {
74                  System.out.println("Error: Debes ingresar un n mero
                       entero.");
75                  scanner.nextLine();  // Limpiar buffer
76              }
77          }
78
79          System.out.println("Opci n seleccionada: " + opcion);
80
81          scanner.close();
82      }
83  }
```

Listing 5.6: Validación de entrada de datos

5.5. Métodos Útiles para Validación

```java
import java.util.Scanner;

public class MetodosValidacion {

    // M todo para leer un entero con validaci n
    public static int leerEntero(Scanner scanner, String mensaje,
        int min, int max) {
        int numero = 0;
        boolean valido = false;

        while (!valido) {
            try {
                System.out.print(mensaje);
                numero = scanner.nextInt();

                if (numero >= min && numero <= max) {
                    valido = true;
                } else {
                    System.out.printf("Error: El n mero debe estar
                        entre %d y %d.%n", min, max);
                }
            } catch (Exception e) {
                System.out.println("Error: Debes ingresar un n mero
                    entero v lido.");
                scanner.nextLine();  // Limpiar buffer
            }
        }

        scanner.nextLine();  // Limpiar buffer para pr xima lectura
        return numero;
    }

    // M todo para leer un double con validaci n
    public static double leerDouble(Scanner scanner, String mensaje,
        double min, double max) {
        double numero = 0;
        boolean valido = false;

        while (!valido) {
            try {
                System.out.print(mensaje);
                numero = scanner.nextDouble();

                if (numero >= min && numero <= max) {
                    valido = true;
```

```
42                  } else {
43                      System.out.printf("Error: El n mero debe estar
                            entre %.2f y %.2f.%n", min, max);
44                  }
45              } catch (Exception e) {
46                  System.out.println("Error: Debes ingresar un n mero
                        v lido.");
47                  scanner.nextLine();  // Limpiar buffer
48              }
49          }
50
51          scanner.nextLine();  // Limpiar buffer
52          return numero;
53      }
54
55      // M todo para leer un String no vac o
56      public static String leerString(Scanner scanner, String mensaje)
            {
57          String texto = "";
58          boolean valido = false;
59
60          while (!valido) {
61              System.out.print(mensaje);
62              texto = scanner.nextLine().trim();
63
64              if (!texto.isEmpty()) {
65                  valido = true;
66              } else {
67                  System.out.println("Error: No puedes dejar este
                        campo vac o.");
68              }
69          }
70
71          return texto;
72      }
73
74      // M todo para leer una opci n de men
75      public static int leerOpcionMenu(Scanner scanner, String[]
            opciones) {
76          System.out.println("\n===  MEN   ===");
77          for (int i = 0; i < opciones.length; i++) {
78              System.out.println((i + 1) + ". " + opciones[i]);
79          }
80
81          return leerEntero(scanner,
```

```
                            "Selecciona una opci n (1-" +
                                opciones.length + "): ",
                            1, opciones.length);
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("=== SISTEMA DE REGISTRO ===");

        // Usar m todos de validaci n
        String nombre = leerString(scanner, "Ingresa tu nombre
            completo: ");
        int edad = leerEntero(scanner, "Ingresa tu edad (1-120): ",
            1, 120);
        double altura = leerDouble(scanner, "Ingresa tu altura en
            metros (0.5-2.5): ", 0.5, 2.5);

        // Opciones de men
        String[] opcionesMenu = {"Guardar datos", "Editar datos",
            "Eliminar datos", "Salir"};
        int opcion = leerOpcionMenu(scanner, opcionesMenu);

        // Mostrar resultados
        System.out.println("\n=== RESUMEN ===");
        System.out.println("Nombre: " + nombre);
        System.out.println("Edad: " + edad);
        System.out.println("Altura: " + altura + " m");
        System.out.println("Opci n seleccionada: " +
            opcionesMenu[opcion - 1]);

        scanner.close();
    }
}
```

Listing 5.7: Métodos de validación reutilizables

5.6. Ejemplo Práctico: Sistema de Registro de Estudiantes

```
import java.util.Scanner;

```

```java
public class SistemaRegistroEstudiantes {

    // Constantes para l mites
    static final int MAX_ESTUDIANTES = 100;
    static final int MIN_NOTA = 0;
    static final int MAX_NOTA = 100;

    // Arrays para almacenar datos
    static String[] nombres = new String[MAX_ESTUDIANTES];
    static int[] edades = new int[MAX_ESTUDIANTES];
    static double[] notas = new double[MAX_ESTUDIANTES];
    static int totalEstudiantes = 0;

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("=== SISTEMA DE REGISTRO DE ESTUDIANTES
            ===");

        boolean ejecutando = true;

        while (ejecutando) {
            mostrarMenuPrincipal();
            int opcion = leerEntero(scanner, "Selecciona una
                opci n: ", 1, 7);

            switch (opcion) {
                case 1:
                    registrarEstudiante(scanner);
                    break;
                case 2:
                    mostrarEstudiantes();
                    break;
                case 3:
                    buscarEstudiante(scanner);
                    break;
                case 4:
                    actualizarEstudiante(scanner);
                    break;
                case 5:
                    calcularEstadisticas();
                    break;
                case 6:
                    generarReporte();
                    break;
```

```
46                  case 7:
47                      System.out.println("Saliendo del sistema...");
48                      ejecutando = false;
49                      break;
50              }
51
52              if (ejecutando) {
53                  System.out.println("\nPresiona Enter para
                        continuar...");
54                  scanner.nextLine();
55              }
56          }
57
58          scanner.close();
59      }
60
61      public static void mostrarMenuPrincipal() {
62          System.out.println("\n=== MEN  PRINCIPAL ===");
63          System.out.println("1. Registrar nuevo estudiante");
64          System.out.println("2. Mostrar todos los estudiantes");
65          System.out.println("3. Buscar estudiante");
66          System.out.println("4. Actualizar informaci n de
                estudiante");
67          System.out.println("5. Calcular estad sticas");
68          System.out.println("6. Generar reporte");
69          System.out.println("7. Salir");
70          System.out.println("======================");
71      }
72
73      public static int leerEntero(Scanner scanner, String mensaje,
            int min, int max) {
74          int numero = 0;
75          boolean valido = false;
76
77          while (!valido) {
78              try {
79                  System.out.print(mensaje);
80                  numero = scanner.nextInt();
81
82                  if (numero >= min && numero <= max) {
83                      valido = true;
84                  } else {
85                      System.out.printf("Error: Debe ser entre %d y
                            %d.%n", min, max);
86                  }
```

```
87              } catch (Exception e) {
88                  System.out.println("Error: Ingresa un n mero entero
                        v lido.");
89                  scanner.nextLine();
90              }
91          }
92
93          scanner.nextLine();  // Limpiar buffer
94          return numero;
95      }
96
97      public static double leerDouble(Scanner scanner, String mensaje,
            double min, double max) {
98          double numero = 0;
99          boolean valido = false;
100
101         while (!valido) {
102             try {
103                 System.out.print(mensaje);
104                 numero = scanner.nextDouble();
105
106                 if (numero >= min && numero <= max) {
107                     valido = true;
108                 } else {
109                     System.out.printf("Error: Debe ser entre %.1f y
                            %.1f.%n", min, max);
110                 }
111             } catch (Exception e) {
112                 System.out.println("Error: Ingresa un n mero
                        v lido.");
113                 scanner.nextLine();
114             }
115         }
116
117         scanner.nextLine();  // Limpiar buffer
118         return numero;
119     }
120
121     public static String leerString(Scanner scanner, String mensaje)
            {
122         String texto = "";
123         boolean valido = false;
124
125         while (!valido) {
126             System.out.print(mensaje);
```

```
127             texto = scanner.nextLine().trim();
128
129             if (!texto.isEmpty()) {
130                 valido = true;
131             } else {
132                 System.out.println("Error: Este campo no puede estar
                        vac o.");
133             }
134         }
135
136         return texto;
137     }
138
139     public static void registrarEstudiante(Scanner scanner) {
140         if (totalEstudiantes >= MAX_ESTUDIANTES) {
141             System.out.println("Error: Capacidad m xima alcanzada
                    (" + MAX_ESTUDIANTES + " estudiantes).");
142             return;
143         }
144
145         System.out.println("\n=== REGISTRAR NUEVO ESTUDIANTE ===");
146
147         String nombre = leerString(scanner, "Nombre completo: ");
148         int edad = leerEntero(scanner, "Edad (15-80): ", 15, 80);
149         double nota = leerDouble(scanner, "Nota promedio (0-100): ",
                MIN_NOTA, MAX_NOTA);
150
151         // Guardar en arrays
152         nombres[totalEstudiantes] = nombre;
153         edades[totalEstudiantes] = edad;
154         notas[totalEstudiantes] = nota;
155         totalEstudiantes++;
156
157         System.out.println("  Estudiante   registrado exitosamente!");
158         System.out.println("Total de estudiantes: " +
                totalEstudiantes);
159     }
160
161     public static void mostrarEstudiantes() {
162         if (totalEstudiantes == 0) {
163             System.out.println("No hay estudiantes registrados.");
164             return;
165         }
166
167         System.out.println("\n=== LISTA DE ESTUDIANTES ===");
```

```
168         System.out.printf("%-5s %-30s %-10s %-10s %-15s%n",
169                           "ID", "Nombre", "Edad", "Nota", "Estado");
170         System.out.println("-----------------------------------------------------
171
172         for (int i = 0; i < totalEstudiantes; i++) {
173             String estado = notas[i] >= 70 ? "Aprobado" :
                    "Reprobado";
174             System.out.printf("%-5d %-30s %-10d %-10.1f %-15s%n",
175                               i + 1,
176                               nombres[i],
177                               edades[i],
178                               notas[i],
179                               estado);
180         }
181
182         System.out.println("-----------------------------------------------------
183         System.out.println("Total: " + totalEstudiantes + "
                estudiantes");
184     }
185
186     public static void buscarEstudiante(Scanner scanner) {
187         if (totalEstudiantes == 0) {
188             System.out.println("No hay estudiantes registrados.");
189             return;
190         }
191
192         System.out.println("\n=== BUSCAR ESTUDIANTE ===");
193         System.out.println("1. Buscar por nombre");
194         System.out.println("2. Buscar por ID");
195         int opcion = leerEntero(scanner, "Selecciona opci n de
                b squeda: ", 1, 2);
196
197         switch (opcion) {
198             case 1:
199                 buscarPorNombre(scanner);
200                 break;
201             case 2:
202                 buscarPorId(scanner);
203                 break;
204         }
205     }
206
207     public static void buscarPorNombre(Scanner scanner) {
208         System.out.print("Ingresa el nombre o parte del nombre a
                buscar: ");
```

```
209         String busqueda = scanner.nextLine().toLowerCase();
210 
211         boolean encontrado = false;
212 
213         System.out.println("\nResultados de b squeda:");
214         System.out.println("-------------------------------------------------------
215 
216         for (int i = 0; i < totalEstudiantes; i++) {
217             if (nombres[i].toLowerCase().contains(busqueda)) {
218                 mostrarDetalleEstudiante(i);
219                 encontrado = true;
220             }
221         }
222 
223         if (!encontrado) {
224             System.out.println("No se encontraron estudiantes con
                    ese nombre.");
225         }
226     }
227 
228     public static void buscarPorId(Scanner scanner) {
229         int id = leerEntero(scanner, "Ingresa el ID del estudiante:
                ", 1, totalEstudiantes);
230 
231         System.out.println("\nResultado de b squeda:");
232         mostrarDetalleEstudiante(id - 1);
233     }
234 
235     public static void mostrarDetalleEstudiante(int index) {
236         System.out.println("\n--- Detalles del Estudiante ---");
237         System.out.println("ID: " + (index + 1));
238         System.out.println("Nombre: " + nombres[index]);
239         System.out.println("Edad: " + edades[index]);
240         System.out.printf("Nota: %.1f/100%n", notas[index]);
241 
242         String estado = notas[index] >= 70 ? "Aprobado" :
                "Reprobado";
243         System.out.println("Estado: " + estado);
244 
245         if (notas[index] >= 90) {
246             System.out.println("Rendimiento: Excelente");
247         } else if (notas[index] >= 80) {
248             System.out.println("Rendimiento: Bueno");
249         } else if (notas[index] >= 70) {
250             System.out.println("Rendimiento: Aceptable");
```

```
251         } else {
252             System.out.println("Rendimiento: Necesita mejorar");
253         }
254     }
255 
256     public static void actualizarEstudiante(Scanner scanner) {
257         if (totalEstudiantes == 0) {
258             System.out.println("No hay estudiantes registrados.");
259             return;
260         }
261 
262         mostrarEstudiantes();
263         int id = leerEntero(scanner, "\nIngresa el ID del estudiante
                a actualizar: ", 1, totalEstudiantes);
264 
265         int index = id - 1;
266 
267         System.out.println("\n=== ACTUALIZAR ESTUDIANTE ===");
268         mostrarDetalleEstudiante(index);
269 
270         System.out.println("\n  Qu   deseas actualizar?");
271         System.out.println("1. Nombre");
272         System.out.println("2. Edad");
273         System.out.println("3. Nota");
274         System.out.println("4. Todo");
275         int opcion = leerEntero(scanner, "Selecciona una opci n: ",
                1, 4);
276 
277         switch (opcion) {
278             case 1:
279                 nombres[index] = leerString(scanner, "Nuevo nombre:
                        ");
280                 break;
281             case 2:
282                 edades[index] = leerEntero(scanner, "Nueva edad
                        (15-80): ", 15, 80);
283                 break;
284             case 3:
285                 notas[index] = leerDouble(scanner, "Nueva nota
                        (0-100): ", MIN_NOTA, MAX_NOTA);
286                 break;
287             case 4:
288                 nombres[index] = leerString(scanner, "Nuevo nombre:
                        ");
```

```
289                 edades[index] = leerEntero(scanner, "Nueva edad
                        (15-80): ", 15, 80);
290                 notas[index] = leerDouble(scanner, "Nueva nota
                        (0-100): ", MIN_NOTA, MAX_NOTA);
291                 break;
292         }
293 
294         System.out.println("   Informacin    actualizada
                exitosamente!");
295     }
296 
297     public static void calcularEstadisticas() {
298         if (totalEstudiantes == 0) {
299             System.out.println("No hay estudiantes registrados.");
300             return;
301         }
302 
303         System.out.println("\n=== ESTAD STICAS ===");
304 
305         double sumaNotas = 0;
306         double notaMaxima = notas[0];
307         double notaMinima = notas[0];
308         int aprobados = 0;
309         int reprobados = 0;
310 
311         for (int i = 0; i < totalEstudiantes; i++) {
312             sumaNotas += notas[i];
313 
314             if (notas[i] > notaMaxima) {
315                 notaMaxima = notas[i];
316             }
317 
318             if (notas[i] < notaMinima) {
319                 notaMinima = notas[i];
320             }
321 
322             if (notas[i] >= 70) {
323                 aprobados++;
324             } else {
325                 reprobados++;
326             }
327         }
328 
329         double promedio = sumaNotas / totalEstudiantes;
```

```
330         double porcentajeAprobados = (double) aprobados /
                totalEstudiantes * 100;
331
332         System.out.println("Total de estudiantes: " +
                totalEstudiantes);
333         System.out.printf("Promedio de notas: %.2f/100%n", promedio);
334         System.out.printf("Nota m xima: %.1f/100%n", notaMaxima);
335         System.out.printf("Nota m nima: %.1f/100%n", notaMinima);
336         System.out.println("Estudiantes aprobados: " + aprobados);
337         System.out.println("Estudiantes reprobados: " + reprobados);
338         System.out.printf("Porcentaje de aprobaci n: %.1f%%%n",
                porcentajeAprobados);
339     }
340
341     public static void generarReporte() {
342         if (totalEstudiantes == 0) {
343             System.out.println("No hay estudiantes registrados.");
344             return;
345         }
346
347         System.out.println("\n=== REPORTE DETALLADO ===");
348
349         // Ordenar estudiantes por nota (burbuja simple)
350         for (int i = 0; i < totalEstudiantes - 1; i++) {
351             for (int j = 0; j < totalEstudiantes - i - 1; j++) {
352                 if (notas[j] < notas[j + 1]) {
353                     // Intercambiar nombres
354                     String tempNombre = nombres[j];
355                     nombres[j] = nombres[j + 1];
356                     nombres[j + 1] = tempNombre;
357
358                     // Intercambiar edades
359                     int tempEdad = edades[j];
360                     edades[j] = edades[j + 1];
361                     edades[j + 1] = tempEdad;
362
363                     // Intercambiar notas
364                     double tempNota = notas[j];
365                     notas[j] = notas[j + 1];
366                     notas[j + 1] = tempNota;
367                 }
368             }
369         }
370
371         // Mostrar reporte ordenado
```

```
372         System.out.println("\nEstudiantes ordenados por nota (de
               mayor a menor):");
373         System.out.printf("%-5s %-30s %-10s %-10s %-15s%n",
374                           "ID", "Nombre", "Edad", "Nota", "Estado");
375         System.out.println("-------------------------------------------------------
376
377         for (int i = 0; i < totalEstudiantes; i++) {
378             String estado = notas[i] >= 70 ? "Aprobado" :
                   "Reprobado";
379             System.out.printf("%-5d %-30s %-10d %-10.1f %-15s%n",
380                               i + 1,
381                               nombres[i],
382                               edades[i],
383                               notas[i],
384                               estado);
385         }
386
387         // Mejores estudiantes
388         System.out.println("\n--- MEJORES ESTUDIANTES (Nota     90)
               ---");
389         boolean hayMejores = false;
390
391         for (int i = 0; i < totalEstudiantes; i++) {
392             if (notas[i] >= 90) {
393                 System.out.printf("%s - %.1f/100%n", nombres[i],
                       notas[i]);
394                 hayMejores = true;
395             }
396         }
397
398         if (!hayMejores) {
399             System.out.println("No hay estudiantes con nota
                   90.");
400         }
401
402         // Estudiantes que necesitan ayuda
403         System.out.println("\n--- ESTUDIANTES QUE NECESITAN AYUDA
               (Nota < 70) ---");
404         boolean hayAyuda = false;
405
406         for (int i = 0; i < totalEstudiantes; i++) {
407             if (notas[i] < 70) {
408                 System.out.printf("%s - %.1f/100%n", nombres[i],
                       notas[i]);
409                 hayAyuda = true;
```

```
410                 }
411             }
412
413             if (!hayAyuda) {
414                 System.out.println(" Todos  los estudiantes est n
                        aprobados!");
415             }
416         }
417 }
```

Listing 5.8: Sistema completo de registro de estudiantes

Ejercicio Propuesto

Ejercicio 5.1: Calculadora de Operaciones Básicas

Crea una calculadora interactiva que:

1. Muestre un menú con operaciones (suma, resta, multiplicación, división, potencia, raíz cuadrada)
2. Permita al usuario seleccionar una operación
3. Pida los números necesarios
4. Realice la operación y muestre el resultado
5. Valide las entradas (ej: no dividir por cero)
6. Permita realizar múltiples operaciones hasta que el usuario decida salir

Ejercicio Propuesto

Ejercicio 5.2: Sistema de Reservas de Hotel

Crea un sistema de reservas simple para un hotel que:

1. Tenga diferentes tipos de habitaciones (individual, doble, suite) con precios diferentes
2. Permita al usuario hacer una reserva ingresando:
   - Tipo de habitación
   - Número de noches
   - Fecha de entrada
   - Datos personales

3. Calcule el costo total
4. Muestre un resumen de la reserva
5. Valide todas las entradas (fechas válidas, número positivo de noches, etc.)

Ejercicio Propuesto

Ejercicio 5.3: Juego de Adivinanza Mejorado
Mejora el juego de adivinanza del capítulo anterior:

1. Permite al usuario seleccionar el rango de números (ej: 1-50, 1-100, 1-1000)
2. Lleva un registro de los intentos realizados
3. Da pistas inteligentes (ej: "muy caliente", çaliente", "frío", "muy frío")
4. Muestra estadísticas al final (intentos, tiempo aproximado)
5. Permite jugar múltiples rondas
6. Guarda el mejor puntaje

5.7. Resumen del Capítulo

- `System.out.print()` muestra texto sin salto de línea
- `System.out.println()` muestra texto con salto de línea
- `System.out.printf()` permite formatear la salida con especificadores como `%d`, `%f`, `%s`
- La clase `Scanner` permite leer entrada del usuario
- Los métodos `nextInt()`, `nextDouble()`, `nextLine()` leen diferentes tipos de datos
- Es importante validar la entrada del usuario para evitar errores
- El problema del buffer ocurre cuando `nextLine()` sigue a `nextInt()`
- Se pueden crear métodos reutilizables para validación de entrada
- La manipulación adecuada de entrada/salida es esencial para programas interactivos

Capítulo 6

Métodos - Programación Estructurada

6.1. Introducción a los Métodos

Los métodos son bloques de código que realizan una tarea específica y pueden ser reutilizados. En programación lineal (sin POO), usamos **métodos estáticos** para organizar nuestro código.

Nota Importante

En programación lineal, todos nuestros métodos serán `static`. Esto significa que pertenecen a la clase, no a objetos (que no usaremos).

6.2. Ventajas de Usar Métodos

- **Reutilización de código**: Escribir una vez, usar muchas veces
- **Modularidad**: Dividir problemas complejos en partes más simples
- **Mantenibilidad**: Más fácil de entender y modificar
- **Depuración**: Más fácil encontrar y corregir errores
- **Encapsulación**: Ocultar detalles de implementación

6.3. Estructura de un Método

```
// Modificador de acceso - siempre public en nuestro caso
// static - porque no usamos objetos
// Tipo de retorno - void si no retorna nada, o el tipo de dato
// Nombre del m todo - descriptivo, en camelCase
// Par metros - datos que recibe el m todo (puede estar vac o)

```

```
public static tipoRetorno nombreMetodo(tipoParam1 param1, tipoParam2
    param2, ...) {
    // Cuerpo del m todo
    // Instrucciones que realiza el m todo

    // Si el tipo de retorno no es void, debe haber:
    return valor;
}
```

Listing 6.1: Estructura general de un método

6.4. Tipos de Métodos

6.4.1. Métodos sin Retorno (void)

```
public static void saludar() {
    System.out.println(" Hola  desde el m todo saludar!");
}

public static void saludarPersonalizado(String nombre) {
    System.out.println(" Hola , " + nombre + "!");
    System.out.println("Bienvenido/a al sistema.");
}
```

Listing 6.2: Método que no retorna valor

6.4.2. Métodos con Retorno

```
public static int sumar(int a, int b) {
    int resultado = a + b;
    return resultado;  // Retorna un valor entero
}

public static double calcularAreaCirculo(double radio) {
    double area = Math.PI * radio * radio;
    return area;  // Retorna un valor double
}

public static boolean esPar(int numero) {
    if (numero % 2 == 0) {
        return true;
```

```
    } else {
        return false;
    }
    // Tambi n se puede: return numero % 2 == 0;
}
```

Listing 6.3: Método que retorna un valor

6.5. Llamando a Métodos

Para usar un método, simplemente lo "llamamos"por su nombre:

```
public class EjemploMetodos {

    public static void main(String[] args) {
        // Llamada a m todo sin par metros ni retorno
        saludar();

        // Llamada a m todo con par metros
        saludarPersonalizado("Ana");
        saludarPersonalizado("Carlos");

        // Llamada a m todo con retorno - podemos usar el resultado
        int suma = sumar(5, 3);
        System.out.println("5 + 3 = " + suma);

        // Tambi n podemos usar el resultado directamente
        System.out.println("10 + 7 = " + sumar(10, 7));

        // Llamada a m todo con retorno booleano
        if (esPar(8)) {
            System.out.println("8 es par");
        } else {
            System.out.println("8 es impar");
        }
    }

    // Definiciones de m todos aqu ...
}
```

Listing 6.4: Ejemplo de llamada a métodos

6.6. Métodos con Múltiples Parámetros

```java
public static double calcularIMC(double peso, double altura) {
    return peso / (altura * altura);
}

public static void mostrarTablaMultiplicar(int numero, int limite) {
    System.out.println("Tabla del " + numero + ":");
    for (int i = 1; i <= limite; i++) {
        System.out.println(numero + " x " + i + " = " + (numero *
            i));
    }
}

// Uso:
public static void main(String[] args) {
    double miIMC = calcularIMC(70.5, 1.75);
    System.out.printf("Tu IMC es: %.2f%n", miIMC);

    mostrarTablaMultiplicar(5, 10);
}
```

Listing 6.5: Método con varios parámetros

6.7. Métodos que Llaman a Otros Métodos

```java
public static double calcularHipotenusa(double cateto1, double
    cateto2) {
    // Usa Math.pow para elevar al cuadrado y Math.sqrt para ra z
        cuadrada
    return Math.sqrt(Math.pow(cateto1, 2) + Math.pow(cateto2, 2));
}

public static void resolverEcuacionCuadratica(double a, double b,
    double c) {
    // Calcula el discriminante
    double discriminante = calcularDiscriminante(a, b, c);

    if (discriminante > 0) {
        double x1 = (-b + Math.sqrt(discriminante)) / (2 * a);
        double x2 = (-b - Math.sqrt(discriminante)) / (2 * a);
        System.out.println("Dos soluciones reales:");
```

```
14          System.out.println("x1 = " + x1);
15          System.out.println("x2 = " + x2);
16      } else if (discriminante == 0) {
17          double x = -b / (2 * a);
18          System.out.println("Una soluci n real:");
19          System.out.println("x = " + x);
20      } else {
21          System.out.println("No hay soluciones reales");
22      }
23  }
24
25  public static double calcularDiscriminante(double a, double b,
        double c) {
26      return b * b - 4 * a * c;
27  }
```

Listing 6.6: Métodos que usan otros métodos

6.8. Valores por Defecto y Sobrecarga (No Disponible)

¡Advertencia!

En Java, **NO existen parámetros por defecto** como en otros lenguajes. Cada método debe recibir todos los parámetros definidos.
Tampoco usaremos **sobrecarga de métodos** (múltiples métodos con el mismo nombre pero diferentes parámetros) en programación lineal para mantener la simplicidad.

6.9. Ejemplo Completo: Calculadora

```
1  // Calculadora.java
2  // Ejemplo de programa modular con m todos
3
4  public class Calculadora {
5
6      public static void main(String[] args) {
7          mostrarMenu();
8
9          double num1 = 15.5;
```

```
10          double num2 = 4.2;
11
12          System.out.println("Operaciones con " + num1 + " y " + num2
                + ":");
13          System.out.println("Suma: " + sumar(num1, num2));
14          System.out.println("Resta: " + restar(num1, num2));
15          System.out.println("Multiplicaci n: " + multiplicar(num1,
                num2));
16
17          double division = dividir(num1, num2);
18          if (!Double.isNaN(division)) {
19              System.out.println("Divisi n: " + division);
20          }
21
22          System.out.println(" Es  " + num1 + " mayor? " +
                esMayor(num1, num2));
23      }
24
25      public static void mostrarMenu() {
26          System.out.println("=== CALCULADORA B SICA ===");
27          System.out.println("1. Sumar");
28          System.out.println("2. Restar");
29          System.out.println("3. Multiplicar");
30          System.out.println("4. Dividir");
31          System.out.println("5. Salir");
32          System.out.println("========================");
33      }
34
35      public static double sumar(double a, double b) {
36          return a + b;
37      }
38
39      public static double restar(double a, double b) {
40          return a - b;
41      }
42
43      public static double multiplicar(double a, double b) {
44          return a * b;
45      }
46
47      public static double dividir(double a, double b) {
48          if (b != 0) {
49              return a / b;
50          } else {
```

```
51              System.out.println("Error: No se puede dividir por
                    cero");
52              return Double.NaN; // Not a Number
53          }
54      }
55
56      public static boolean esMayor(double a, double b) {
57          return a > b;
58      }
59  }
```

Listing 6.7: Calculadora completa con métodos

6.10. Organización de Métodos en Archivos

Nota Importante

En programación lineal, todos los métodos deben estar en la misma clase. Para programas grandes, podemos organizar métodos relacionados cerca unos de otros, agrupados por funcionalidad.

```
1  public class ProgramaModular {
2
3      // 1. M todo main al inicio
4      public static void main(String[] args) {
5          // L gica principal
6      }
7
8      // 2. M todos de entrada/salida
9      public static void mostrarMenu() { ... }
10     public static double leerNumero() { ... }
11
12     // 3. M todos de c lculo
13     public static double calcularTotal() { ... }
14     public static double calcularPromedio() { ... }
15
16     // 4. M todos de validaci n
17     public static boolean esValido() { ... }
18     public static boolean estaEnRango() { ... }
19
20     // 5. M todos de utilidad
21     public static void limpiarPantalla() { ... }
```

```
22      public static void pausar() { ... }
23  }
```

Listing 6.8: Organización recomendada de métodos

6.11. Errores Comunes con Métodos

¡Advertencia!

Error: "missing return statement"
Solución: Asegúrate que TODOS los caminos posibles en un método con retorno terminen con un `return`.

¡Advertencia!

Error: "method X in class Y cannot be applied to given types"
Solución: Verifica que estás pasando el número correcto de parámetros y del tipo correcto.

6.12. Buenas Prácticas con Métodos

1. **Nombres descriptivos**: `calcularTotal()` en lugar de `ct()`
2. **Una responsabilidad**: Cada método debe hacer solo una cosa
3. **Tamaño reducido**: Si un método tiene más de 30 líneas, considere dividirlo
4. **Comentarios útiles**: Documenta qué hace el método, parámetros y retorno
5. **Validación de parámetros**: Verifica valores antes de usarlos

Ejercicio Propuesto

Ejercicio 6.1: Calculadora de Figuras Geométricas
Crea un programa llamado **`CalculadoraGeometrica.java`** con métodos para:

1. Calcular área de un círculo (recibe radio)
2. Calcular área de un rectángulo (recibe base y altura)
3. Calcular área de un triángulo (recibe base y altura)
4. Calcular perímetro de un círculo (circunferencia)

5. Calcular perímetro de un rectángulo
6. Un método `mostrarMenu()` que muestre las opciones disponibles
7. Un método `main()` que use todos los métodos anteriores

Ejercicio Propuesto

Ejercicio 6.2: Conversor de Unidades

Crea un programa llamado **`ConversorUnidades.java`** con métodos para convertir:

- Celsius a Fahrenheit
- Fahrenheit a Celsius
- Kilómetros a Millas
- Millas a Kilómetros
- Kilogramos a Libras
- Libras a Kilogramos

Además, crea un método `mostrarResultado()` que reciba el valor original, el valor convertido y las unidades, y muestre un mensaje formateado.

Ejercicio Propuesto

Ejercicio 6.3: Validador de Datos

Crea un programa llamado **`Validador.java`** con métodos que validen:

1. Si un número está en un rango específico (recibe número, mínimo, máximo)
2. Si una cadena tiene al menos N caracteres
3. Si una cadena es un email válido (contiene @ y .)
4. Si una cadena es un número entero válido
5. Si una cadena es un número decimal válido

Todos los métodos deben retornar `true` o `false`.

6.13. Proyecto: Sistema de Gestión de Notas

```java
// SistemaNotas.java
// Gesti n de notas de estudiantes usando m todos

public class SistemaNotas {

    public static void main(String[] args) {
        double[] notas = {4.5, 3.8, 4.2, 5.0, 3.5};

        mostrarNotas(notas);
        System.out.println("Promedio: " + calcularPromedio(notas));
        System.out.println("Nota m xima: " + encontrarMaxima(notas);
        System.out.println("Nota m nima: " +
            encontrarMinima(notas));
        System.out.println("Aprobados: " + contarAprobados(notas));

        double[] notasOrdenadas = ordenarNotas(notas);
        System.out.println("Notas ordenadas:");
        mostrarNotas(notasOrdenadas);
    }

    public static void mostrarNotas(double[] notas) {
        System.out.print("Notas: [");
        for (int i = 0; i < notas.length; i++) {
            System.out.print(notas[i]);
            if (i < notas.length - 1) {
                System.out.print(", ");
            }
        }
        System.out.println("]");
    }

    public static double calcularPromedio(double[] notas) {
        double suma = 0;
        for (double nota : notas) {
            suma += nota;
        }
        return suma / notas.length;
    }

    public static double encontrarMaxima(double[] notas) {
        double max = notas[0];
        for (double nota : notas) {
```

```
42              if (nota > max) {
43                  max = nota;
44              }
45          }
46          return max;
47      }
48
49      public static double encontrarMinima(double[] notas) {
50          double min = notas[0];
51          for (double nota : notas) {
52              if (nota < min) {
53                  min = nota;
54              }
55          }
56          return min;
57      }
58
59      public static int contarAprobados(double[] notas) {
60          int aprobados = 0;
61          for (double nota : notas) {
62              if (nota >= 3.0) {
63                  aprobados++;
64              }
65          }
66          return aprobados;
67      }
68
69      public static double[] ordenarNotas(double[] notas) {
70          // M todo de burbuja para ordenar
71          double[] copia = notas.clone();
72          for (int i = 0; i < copia.length - 1; i++) {
73              for (int j = 0; j < copia.length - i - 1; j++) {
74                  if (copia[j] < copia[j + 1]) {
75                      // Intercambiar
76                      double temp = copia[j];
77                      copia[j] = copia[j + 1];
78                      copia[j + 1] = temp;
79                  }
80              }
81          }
82          return copia;
83      }
84  }
```

Listing 6.9: Ejemplo de sistema modular

6.14. Resumen del Capítulo

- Los métodos nos permiten organizar código en bloques reutilizables
- Usamos métodos estáticos en programación lineal
- Un método puede tener parámetros y retornar un valor
- Los métodos `void` no retornan valor
- La modularización hace el código más mantenible y comprensible
- Cada método debe tener una responsabilidad clara
- Es importante nombrar métodos descriptivamente

Con métodos, tu código será más organizado, reutilizable y profesional.

Capítulo 7

Ámbito de Variables y Paso por Valor

7.1. Introducción al Ámbito (Scope)

El ámbito o alcance de una variable determina dónde en el código puede ser accedida y modificada. Comprender el ámbito es esencial para evitar errores y escribir código robusto.

7.2. Tipos de Ámbito

7.2.1. Variables Locales

Las variables locales son declaradas dentro de un método y solo existen mientras ese método se ejecuta.

```
public class VariablesLocales {

    public static void main(String[] args) {
        // Variable local del m todo main
        int numero = 10;
        System.out.println("En main - numero: " + numero);

        // Llamar a un m todo
        mostrarMensaje();

        // No podemos acceder a variables de otros m todos
        // System.out.println(mensaje); // ERROR: mensaje no existe
           aqu
    }

    public static void mostrarMensaje() {
        // Variable local del m todo mostrarMensaje
        String mensaje = "Hola desde el m todo";
        System.out.println(mensaje);

        // No podemos acceder a variables de main
        // System.out.println(numero); // ERROR: numero no existe
```

```
            aqu
    }
}
```

Listing 7.1: Variables locales

7.2.2. Variables de Bloque

Las variables declaradas dentro de bloques (como bucles o condicionales) solo existen dentro de ese bloque.

```
public class VariablesBloque {

    public static void main(String[] args) {
        // Variable del m todo main
        int x = 100;
        System.out.println("Antes del if - x: " + x);

        if (x > 50) {
            // Variable del bloque if
            int y = 200;
            System.out.println("Dentro del if - x: " + x);
            System.out.println("Dentro del if - y: " + y);

            // Podemos modificar x (est  en un  mbito  superior)
            x = 150;
        }

        System.out.println("Despu s del if - x: " + x);
        // System.out.println("y: " + y); // ERROR: y no existe aqu

        // Ejemplo con bucles
        for (int i = 0; i < 3; i++) {
            // Variable del bloque for
            int resultado = i * 10;
            System.out.println("Iteraci n " + i + ": " + resultado);
        }

        // System.out.println("i: " + i); // ERROR: i no existe aqu
        // System.out.println("resultado: " + resultado); // ERROR
    }
}
```

Listing 7.2: Variables de bloque

7.3. Shadowing (Ocultación de Variables)

Ocurre cuando una variable local tiene el mismo nombre que una variable en un ámbito superior.

```
public class ShadowingEjemplo {

    // Variable de clase (no recomendada en programaci n lineal,
       pero para ejemplo)
    static int numero = 100;

    public static void main(String[] args) {
        System.out.println("numero (clase): " + numero);

        // Variable local con mismo nombre (shadowing)
        int numero = 50;
        System.out.println("numero (local en main): " + numero);

        // Acceder a la variable de clase
        System.out.println("numero (clase, usando
           ShadowingEjemplo.numero): " +
                           ShadowingEjemplo.numero);

        // Llamar m todo
        mostrarNumero();

        // Dentro de un bloque
        {
            int numero = 75;  // Shadowing dentro de bloque
            System.out.println("numero (en bloque): " + numero);
        }

        System.out.println("numero (despu s del bloque): " +
           numero);
    }

    public static void mostrarNumero() {
        // Aqu  no hay shadowing, usa la variable de clase
        System.out.println("En m todo - numero: " + numero);
    }
}
```

Listing 7.3: Ejemplo de shadowing

¡Advertencia!

Evita el shadowing cuando sea posible. Usar el mismo nombre para diferentes variables puede causar confusión y errores. Es mejor usar nombres descriptivos y únicos.

7.4. Variables Globales (Estáticas) en Programación Lineal

En programación lineal, podemos usar variables estáticas como "globales"dentro de una clase.

```
public class VariablesGlobales {

    // Variables est ticas (simulan globales)
    static int contador = 0;
    static String nombreAplicacion = "Mi Programa";
    static final double PI = 3.1415926535;

    public static void main(String[] args) {
        System.out.println("=== " + nombreAplicacion + " ===");

        // Usar las variables est ticas
        System.out.println("Contador inicial: " + contador);
        System.out.println("Valor de PI: " + PI);

        // Modificar contador desde diferentes m todos
        incrementarContador();
        incrementarContador();
        mostrarContador();

        incrementarContador(5);
        mostrarContador();

        resetearContador();
        mostrarContador();
    }

    public static void incrementarContador() {
        contador++;  // Accede a la variable est tica
        System.out.println("Contador incrementado a: " + contador);
    }

```

```
32      public static void incrementarContador(int cantidad) {
33          contador += cantidad;
34          System.out.println("Contador incrementado en " + cantidad +
                ": " + contador);
35      }
36
37      public static void resetearContador() {
38          contador = 0;
39          System.out.println("Contador reseteado");
40      }
41
42      public static void mostrarContador() {
43          System.out.println("Contador actual: " + contador);
44      }
45  }
```

Listing 7.4: Variables estáticas como globales

7.5. Paso por Valor

En Java, todos los parámetros se pasan **por valor**. Esto significa que cuando pasas una variable a un método, se pasa una copia de su valor, no la variable original.

```
1  public class PasoPorValor {
2
3      public static void main(String[] args) {
4          // Ejemplo con tipos primitivos
5          int numero = 10;
6          System.out.println("Antes de modificar: " + numero);
7
8          // Intentar modificar el n mero
9          modificarNumero(numero);
10         System.out.println("Despu s de modificar: " + numero);
11
12         // El n mero NO cambia porque se pas  una copia
13
14         // Ejemplo con retorno
15         numero = duplicar(numero);
16         System.out.println("Despu s de duplicar (con retorno): " +
               numero);
17
18         // Ejemplo con String (es inmutable, se comporta como
               primitivo)
19         String mensaje = "Hola";
```

```
20          System.out.println("\nAntes de modificar String: " +
                mensaje);
21          modificarString(mensaje);
22          System.out.println("Despu s de modificar String: " +
                mensaje);
23      }
24
25      public static void modificarNumero(int x) {
26          x = 100;  // Modifica la copia, no el original
27          System.out.println("Dentro del m todo: " + x);
28      }
29
30      public static int duplicar(int x) {
31          return x * 2;  // Retorna el nuevo valor
32      }
33
34      public static void modificarString(String s) {
35          s = "Adi s";  // Crea un nuevo String, no modifica el
                original
36          System.out.println("Dentro del m todo: " + s);
37      }
38  }
```

Listing 7.5: Paso por valor con tipos primitivos

7.6. Paso de Arrays (Referencia por Valor)

Los arrays se comportan de manera especial. Aunque se pasa una copia de la referencia, ambas referencias apuntan al mismo array en memoria.

```
1   public class PasoDeArrays {
2
3       public static void main(String[] args) {
4           // Crear un array
5           int[] numeros = {1, 2, 3, 4, 5};
6
7           System.out.println("Array original:");
8           mostrarArray(numeros);
9
10          // Modificar el array desde un m todo
11          modificarArray(numeros);
12
13          System.out.println("\nArray despu s de modificar:");
14          mostrarArray(numeros);  //  Los  cambios se mantienen!
```

```
15
16          // Crear un nuevo array (no modifica el original)
17          System.out.println("\nIntentando reasignar:");
18          reasignarArray(numeros);
19          mostrarArray(numeros);  // El array original no cambia
20      }
21
22      public static void modificarArray(int[] arr) {
23          System.out.println("\nModificando array dentro del
                m todo:");
24          for (int i = 0; i < arr.length; i++) {
25              arr[i] = arr[i] * 10;  // Modifica los elementos
26              System.out.print(arr[i] + " ");
27          }
28      }
29
30      public static void reasignarArray(int[] arr) {
31          // Esto crea un nuevo array local, no afecta al original
32          arr = new int[]{100, 200, 300};
33          System.out.println("Array dentro del m todo despu s de
                reasignar:");
34          mostrarArray(arr);
35      }
36
37      public static void mostrarArray(int[] arr) {
38          for (int num : arr) {
39              System.out.print(num + " ");
40          }
41          System.out.println();
42      }
43  }
```

Listing 7.6: Paso de arrays

7.7. Duración de Variables (Lifetime)

La duración de una variable es el tiempo durante el cual existe en memoria.

7.8. Errores Comunes con el Ámbito

```
1  public class ErroresAmbito {
2
```

Tipo	Duración	Descripción
Variables locales	Corta	Existen desde su declaración hasta el final del bloque donde fueron declaradas
Variables de parámetro	Corta	Existen durante la ejecución del método
Variables estáticas	Larga	Existen durante toda la ejecución del programa

Cuadro 7.1: Duración de variables en Java

```
3     public static void main(String[] args) {
4         // Error 1: Usar variable antes de declararla
5         // System.out.println(x); // ERROR: no puede resolver
              s mbolo
6         int x = 10;
7
8         // Error 2: Intentar usar variable fuera de su  mbito
9         if (x > 5) {
10            int y = 20;
11            System.out.println("y dentro del if: " + y);
12        }
13        // System.out.println("y fuera del if: " + y); // ERROR
14
15        // Error 3: Redeclarar variable en mismo  mbito
16        int z = 30;
17        // int z = 40; // ERROR: ya definida
18
19        // Pero esto es v lido ( mbitos  diferentes)
20        {
21            int z = 40;  // Shadowing permitido
22            System.out.println("z en bloque interno: " + z);
23        }
24        System.out.println("z en bloque externo: " + z);
25
26        // Error 4: No inicializar variable local
27        int resultado;
28        // System.out.println(resultado); // ERROR: no inicializada
29
30        resultado = calcular();
31        System.out.println("Resultado: " + resultado);  // Ahora s
32    }
33
34    public static int calcular() {
35        // Error 5: M todo con retorno que no siempre retorna
36        int valor = 10;
37
```

```
        if (valor > 5) {
            return valor * 2;
        }
        // ERROR: falta return cuando valor <= 5

        // Soluci n: agregar return al final
        return 0;
    }
}
```

Listing 7.7: Errores comunes de ámbito

7.9. Buenas Prácticas con el Ámbito

1. **Declara variables lo más cerca posible de donde se usan**
2. **Inicializa las variables al declararlas** cuando sea posible
3. **Usa el menor ámbito posible** para mayor seguridad
4. **Evita variables globales innecesarias** (usa estáticas solo cuando sea realmente necesario)
5. **Usa nombres descriptivos** para evitar shadowing accidental
6. **Limpia recursos** cuando ya no los necesites (aunque Java tiene GC)

7.10. Ejemplo Práctico: Contador de Operaciones

```
public class ContadorOperaciones {

    // Variables est ticas para el contador global
    static int totalOperaciones = 0;
    static int operacionesExitosas = 0;
    static int operacionesFallidas = 0;

    public static void main(String[] args) {
        System.out.println("=== CONTADOR DE OPERACIONES ===\n");

        // Realizar varias operaciones
        realizarOperacion("Sumar", 10, 5);
        realizarOperacion("Restar", 20, 8);
```

```
14          realizarOperacion("Multiplicar", 6, 7);
15          realizarOperacion("Dividir", 100, 0);  // Esta fallar
16          realizarOperacion("Dividir", 100, 4);
17
18          // Mostrar estad sticas
19          mostrarEstadisticas();
20
21          // Reiniciar contadores
22          reiniciarContadores();
23          System.out.println("\n=== CONTADORES REINICIADOS ===\n");
24
25          // M s operaciones
26          realizarOperacion("Sumar", 15, 25);
27          realizarOperacion("Restar", 50, 30);
28
29          // Estad sticas finales
30          mostrarEstadisticas();
31      }
32
33      public static void realizarOperacion(String tipo, double a,
            double b) {
34          // Variables locales del m todo
35          boolean exito = true;
36          double resultado = 0;
37
38          // Incrementar contador total
39          totalOperaciones++;
40
41          // Realizar la operaci n
42          switch (tipo.toLowerCase()) {
43              case "sumar":
44                  resultado = sumar(a, b);
45                  break;
46              case "restar":
47                  resultado = restar(a, b);
48                  break;
49              case "multiplicar":
50                  resultado = multiplicar(a, b);
51                  break;
52              case "dividir":
53                  if (b != 0) {
54                      resultado = dividir(a, b);
55                  } else {
56                      System.out.println("Error: Divisi n por cero");
57                      exito = false;
```

```
58                  }
59                  break;
60              default:
61                  System.out.println("Error: Operaci n no v lida");
62                  exito = false;
63          }
64
65          // Actualizar contadores seg n el resultado
66          if (exito) {
67              operacionesExitosas++;
68              System.out.printf("%s: %.2f y %.2f = %.2f%n", tipo, a,
                    b, resultado);
69          } else {
70              operacionesFallidas++;
71          }
72      }
73
74      public static double sumar(double a, double b) {
75          return a + b;
76      }
77
78      public static double restar(double a, double b) {
79          return a - b;
80      }
81
82      public static double multiplicar(double a, double b) {
83          return a * b;
84      }
85
86      public static double dividir(double a, double b) {
87          return a / b;
88      }
89
90      public static void mostrarEstadisticas() {
91          // Variable local para porcentaje
92          double porcentajeExito = 0;
93
94          if (totalOperaciones > 0) {
95              porcentajeExito = (double) operacionesExitosas /
                    totalOperaciones * 100;
96          }
97
98          System.out.println("\n=== ESTAD STICAS ===");
99          System.out.println("Total de operaciones: " +
                totalOperaciones);
```

```
100         System.out.println("Operaciones exitosas: " +
                operacionesExitosas);
101         System.out.println("Operaciones fallidas: " +
                operacionesFallidas);
102         System.out.printf("Porcentaje de  xito : %.1f%%%n",
                porcentajeExito);
103     }
104
105     public static void reiniciarContadores() {
106         // Las variables est ticas pueden ser modificadas desde
                cualquier m todo
107         totalOperaciones = 0;
108         operacionesExitosas = 0;
109         operacionesFallidas = 0;
110     }
111 }
```

Listing 7.8: Sistema con variables de ámbito controlado

Ejercicio Propuesto

Ejercicio 7.1: Simulador de Banco

Crea un simulador de banco que:

1. Tenga una variable estática para el saldo total del banco
2. Permita realizar depósitos y retiros
3. Cada operación debe validar fondos suficientes
4. Lleve un registro de:
   - Total de operaciones
   - Depósitos exitosos/fallidos
   - Retiros exitosos/fallidos
   - Saldo actual
5. Use métodos para cada operación
6. Los métodos deben usar variables locales apropiadas

Ejercicio Propuesto

Ejercicio 7.2: Juego de Memoria

Crea un juego de memoria que:

1. Genere un patrón secreto de números
2. Permita al usuario intentar adivinar el patrón
3. Lleve estadísticas usando variables estáticas:
   - Partidas jugadas
   - Partidas ganadas
   - Mejor puntaje (menos intentos)
   - Promedio de intentos por partida ganada
4. Cada partida debe usar variables locales para el patrón e intentos
5. Muestre las estadísticas al final de cada partida

Ejercicio Propuesto

Ejercicio 7.3: Analizador de Texto

Crea un programa que analice texto y:

1. Cuente caracteres, palabras y líneas
2. Use métodos separados para cada tipo de conteo
3. Cada método debe usar solo variables locales
4. Lleve estadísticas globales de todos los textos analizados
5. Permita reiniciar las estadísticas globales
6. Muestre tanto el análisis del texto actual como las estadísticas globales

7.11. Resumen del Capítulo

- El ámbito determina dónde puede usarse una variable
- Variables locales existen solo dentro de su método o bloque
- Variables estáticas existen durante toda la ejecución del programa

- Java usa paso por valor para todos los parámetros
- Para tipos primitivos, se pasa una copia del valor
- Para arrays y objetos, se pasa una copia de la referencia
- El shadowing ocurre cuando una variable local oculta una variable con el mismo nombre en un ámbito superior
- Es importante inicializar variables locales antes de usarlas
- Usar el menor ámbito posible mejora la seguridad y claridad del código

Capítulo 8

Arrays

8.1. Introducción a los Arrays

Un array es una estructura de datos que almacena una colección de elementos del mismo tipo. Es fundamental para trabajar con conjuntos de datos.

Nota Importante

En Java, los arrays tienen tamaño fijo una vez creados. No pueden crecer o reducirse dinámicamente. Para colecciones dinámicas necesitaríamos otras estructuras, pero en programación lineal nos enfocaremos en arrays.

8.2. Declaración e Inicialización de Arrays

```
1  public class DeclaracionArrays {
2
3      public static void main(String[] args) {
4          // Forma 1: Declarar y luego inicializar
5          int[] numeros1;            // Declaraci n
6          numeros1 = new int[5];     // Inicializaci n con tama o 5
7
8          // Forma 2: Declarar e inicializar en una l nea
9          int[] numeros2 = new int[3];  // Array de 3 enteros (todos 0)
10
11         // Forma 3: Inicializar con valores espec ficos
12         int[] numeros3 = {10, 20, 30, 40, 50};  // Array de 5
               elementos
13
14         // Forma 4: Declarar e inicializar por separado
15         int[] numeros4;
16         numeros4 = new int[]{1, 2, 3, 4, 5};  // Notar el 'new int[]'
17
18         // Arrays de otros tipos
19         double[] precios = {19.99, 29.99, 39.99};
20         boolean[] estados = {true, false, true, true};
```

```
21          char[] letras = {'A', 'B', 'C', 'D'};
22          String[] nombres = {"Ana", "Carlos", "Elena", "David"};
23
24          // Mostrar informaci n de los arrays
25          System.out.println("=== INFORMACI N DE ARRAYS ===");
26          System.out.println("numeros1 tama o: " + numeros1.length);
27          System.out.println("numeros2 tama o: " + numeros2.length);
28          System.out.println("numeros3 tama o: " + numeros3.length);
29          System.out.println("nombres tama o: " + nombres.length);
30
31          // Acceder a elementos individuales
32          System.out.println("\n=== ACCESO A ELEMENTOS ===");
33          System.out.println("numeros3[0]: " + numeros3[0]);  //
               Primer elemento ( ndice  0)
34          System.out.println("numeros3[2]: " + numeros3[2]);  //
               Tercer elemento
35          System.out.println("nombres[1]: " + nombres[1]);    //
               Segundo nombre
36
37          // Cambiar un elemento
38          numeros3[0] = 100;
39          nombres[1] = "Roberto";
40          System.out.println("\nDespu s de modificar:");
41          System.out.println("numeros3[0]: " + numeros3[0]);
42          System.out.println("nombres[1]: " + nombres[1]);
43
44          // Intentar acceder a  ndice  fuera de rango (ERROR)
45          // System.out.println(numeros3[10]);  //
               ArrayIndexOutOfBoundsException
46      }
47  }
```

Listing 8.1: Diferentes formas de crear arrays

8.3. Recorrido de Arrays

8.3.1. Usando Bucle for Tradicional

```
1   public class RecorridoFor {
2
3       public static void main(String[] args) {
4           // Array de ejemplo
5           int[] numeros = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};
```

```
6
7           System.out.println("=== RECORRIDO CON FOR TRADICIONAL
                ===\n");
8
9           // Recorrer de principio a fin
10          System.out.println("Recorrido normal:");
11          for (int i = 0; i < numeros.length; i++) {
12              System.out.println(" ndice   " + i + ": " + numeros[i]);
13          }
14
15          // Recorrer al rev s
16          System.out.println("\nRecorrido al rev s:");
17          for (int i = numeros.length - 1; i >= 0; i--) {
18              System.out.println(" ndice   " + i + ": " + numeros[i]);
19          }
20
21          // Recorrer saltando elementos
22          System.out.println("\nRecorrido de 2 en 2:");
23          for (int i = 0; i < numeros.length; i += 2) {
24              System.out.println(" ndice   " + i + ": " + numeros[i]);
25          }
26
27          // Ejemplo pr ctico: calcular suma y promedio
28          System.out.println("\n=== C LCULOS CON ARRAY ===");
29          double[] temperaturas = {22.5, 23.0, 21.8, 24.2, 25.1, 23.7,
                22.9};
30
31          double suma = 0;
32          double maxima = temperaturas[0];
33          double minima = temperaturas[0];
34
35          for (int i = 0; i < temperaturas.length; i++) {
36              suma += temperaturas[i];
37
38              if (temperaturas[i] > maxima) {
39                  maxima = temperaturas[i];
40              }
41
42              if (temperaturas[i] < minima) {
43                  minima = temperaturas[i];
44              }
45          }
46
47          double promedio = suma / temperaturas.length;
48
```

```
49          System.out.println("Temperaturas:");
50          for (int i = 0; i < temperaturas.length; i++) {
51              System.out.printf("D a %d: %.1f C%n", i + 1,
                    temperaturas[i]);
52          }
53
54          System.out.printf("\nSuma: %.1f C%n", suma);
55          System.out.printf("Promedio: %.1f C%n", promedio);
56          System.out.printf("M xima: %.1f C%n", maxima);
57          System.out.printf("M nima: %.1f C%n", minima);
58      }
59  }
```

Listing 8.2: Recorrido con for tradicional

8.3.2. Usando Bucle for-each (Enhanced for)

```
1   public class RecorridoForEach {
2
3       public static void main(String[] args) {
4           // Array de ejemplo
5           String[] frutas = {"Manzana", "Banana", "Naranja", "Uva",
                "Mango"};
6
7           System.out.println("=== RECORRIDO CON FOR-EACH ===\n");
8
9           // Recorrer con for-each (solo lectura, no podemos modificar
                ndice )
10          System.out.println("Lista de frutas:");
11          for (String fruta : frutas) {
12              System.out.println("- " + fruta);
13          }
14
15          // Ejemplo con n meros
16          int[] numeros = {5, 10, 15, 20, 25, 30};
17
18          System.out.println("\nN meros pares del array:");
19          for (int numero : numeros) {
20              if (numero % 2 == 0) {
21                  System.out.print(numero + " ");
22              }
23          }
24          System.out.println();
25
```

```
26          // Ejemplo: buscar un elemento
27          String[] estudiantes = {"Ana", "Carlos", "Elena", "David",
                "Beatriz"};
28          String buscar = "Elena";
29          boolean encontrado = false;
30
31          System.out.println("\nBuscando a " + buscar + ":");
32          for (String estudiante : estudiantes) {
33              if (estudiante.equals(buscar)) {
34                  encontrado = true;
35                  break;
36              }
37          }
38
39          if (encontrado) {
40              System.out.println(buscar + " est  en la lista.");
41          } else {
42              System.out.println(buscar + " NO est  en la lista.");
43          }
44
45          // Limitaci n: for-each no permite modificar el array
                original
46          System.out.println("\nIntentando duplicar valores (no
                funciona):");
47          for (int numero : numeros) {
48              numero = numero * 2;  // Solo modifica la copia local
49          }
50
51          System.out.println("Array original (sin cambios):");
52          for (int numero : numeros) {
53              System.out.print(numero + " ");
54          }
55          System.out.println();
56
57          // Para modificar, necesitamos for tradicional
58          System.out.println("\nDuplicando valores (con for
                tradicional):");
59          for (int i = 0; i < numeros.length; i++) {
60              numeros[i] = numeros[i] * 2;
61          }
62
63          System.out.println("Array modificado:");
64          for (int numero : numeros) {
65              System.out.print(numero + " ");
66          }
```

```
67          System.out.println();
68      }
69  }
```

Listing 8.3: Recorrido con for-each

8.4. Modificación de Arrays

8.4.1. Cambio de Elementos

```
1  public class ModificacionArrays {
2
3      public static void main(String[] args) {
4          // Array original
5          int[] numeros = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
6
7          System.out.println("Array original:");
8          mostrarArray(numeros);
9
10         // Modificar elementos individuales
11         System.out.println("\n=== MODIFICACIONES INDIVIDUALES ===");
12
13         // Cambiar el primer y  ltimo  elemento
14         numeros[0] = 100;
15         numeros[numeros.length - 1] = 200;
16
17         System.out.println("Despu s de cambiar primero y  ltimo :");
18         mostrarArray(numeros);
19
20         // Cambiar elementos en posiciones espec ficas
21         numeros[2] = numeros[2] * 10;
22         numeros[5] = numeros[5] + 50;
23
24         System.out.println("\nDespu s de m s modificaciones:");
25         mostrarArray(numeros);
26
27         // Modificar todos los elementos
28         System.out.println("\n=== MODIFICAR TODOS LOS ELEMENTOS
               ===");
29
30         // Duplicar todos los valores
31         for (int i = 0; i < numeros.length; i++) {
32             numeros[i] = numeros[i] * 2;
```

```
        }

        System.out.println("Despu s de duplicar todo:");
        mostrarArray(numeros);

        // Aplicar una operaci n condicional
        System.out.println("\n=== APLICAR OPERACI N CONDICIONAL
            ===");
        for (int i = 0; i < numeros.length; i++) {
            if (numeros[i] > 100) {
                numeros[i] = 100;  // Limitar a m ximo 100
            }

            if (numeros[i] % 2 != 0) {
                numeros[i]++;  // Hacer pares los impares
            }
        }

        System.out.println("Despu s de normalizar:");
        mostrarArray(numeros);
    }

    public static void mostrarArray(int[] arr) {
        for (int i = 0; i < arr.length; i++) {
            System.out.print(arr[i] + " ");
        }
        System.out.println();
    }
}
```

Listing 8.4: Modificación de elementos de array

8.4.2. Copiar Arrays

```
public class CopiarArrays {

    public static void main(String[] args) {
        // Array original
        int[] original = {10, 20, 30, 40, 50};

        System.out.println("Array original:");
        mostrarArray("original", original);

        // M todo 1: Copia manual (elemento por elemento)
```

```
11          System.out.println("\n=== M TODO 1: COPIA MANUAL ===");
12          int[] copia1 = new int[original.length];
13
14          for (int i = 0; i < original.length; i++) {
15              copia1[i] = original[i];
16          }
17
18          mostrarArray("copia1", copia1);
19
20          // Modificar la copia (no afecta al original)
21          copia1[0] = 100;
22          System.out.println("\nDespu s de modificar copia1[0] =
                100:");
23          mostrarArray("original", original);
24          mostrarArray("copia1", copia1);
25
26          // M todo 2: Usar System.arraycopy()
27          System.out.println("\n=== M TODO 2: System.arraycopy()
                ===");
28          int[] copia2 = new int[original.length];
29          System.arraycopy(original, 0, copia2, 0, original.length);
30
31          mostrarArray("copia2", copia2);
32
33          // Copiar solo parte del array
34          int[] copiaParcial = new int[3];
35          System.arraycopy(original, 1, copiaParcial, 0, 3);  // Copia
                elementos 1-3
36
37          System.out.println("\nCopia parcial (elementos 1-3):");
38          mostrarArray("copiaParcial", copiaParcial);
39
40          // M todo 3: Usar clone()
41          System.out.println("\n=== M TODO 3: clone() ===");
42          int[] copia3 = original.clone();
43
44          mostrarArray("copia3", copia3);
45
46          // Comparar arrays
47          System.out.println("\n=== COMPARAR ARRAYS ===");
48          int[] arr1 = {1, 2, 3};
49          int[] arr2 = {1, 2, 3};
50          int[] arr3 = arr1;
51
```

```
52          System.out.println("arr1 == arr2: " + (arr1 == arr2));
                    // false (diferentes objetos)
53          System.out.println("arr1 == arr3: " + (arr1 == arr3));
                    // true (mismo objeto)
54          System.out.println("Arrays.equals(arr1, arr2): " +
               java.util.Arrays.equals(arr1, arr2));  // true (mismo
               contenido)
55
56          // Copiar y expandir array
57          System.out.println("\n=== COPIAR Y EXPANDIR ===");
58          int[] peque o = {1, 2, 3};
59          int[] grande = new int[6];  // Array m s grande
60
61          System.arraycopy(peque o, 0, grande, 0, peque o.length);
62          // Llenar el resto con valores por defecto o nuevos valores
63          for (int i = peque o.length; i < grande.length; i++) {
64              grande[i] = (i + 1) * 10;
65          }
66
67          System.out.println("Array expandido:");
68          mostrarArray("grande", grande);
69      }
70
71      public static void mostrarArray(String nombre, int[] arr) {
72          System.out.print(nombre + ": [");
73          for (int i = 0; i < arr.length; i++) {
74              System.out.print(arr[i]);
75              if (i < arr.length - 1) {
76                  System.out.print(", ");
77              }
78          }
79          System.out.println("]");
80      }
81  }
```

Listing 8.5: Diferentes formas de copiar arrays

8.5. Búsqueda en Arrays

8.5.1. Búsqueda Lineal

```
1  public class BusquedaLineal {
2
```

```
3       public static void main(String[] args) {
4           // Array de ejemplo
5           int[] numeros = {45, 23, 67, 89, 12, 34, 56, 78, 90, 33};
6           String[] nombres = {"Ana", "Carlos", "Elena", "David",
                "Beatriz", "Fernando"};
7
8           System.out.println("=== B SQUEDA LINEAL ===\n");
9
10          // B squeda simple
11          System.out.println("Buscando el n mero 67:");
12          int indice = buscarNumero(numeros, 67);
13
14          if (indice != -1) {
15              System.out.println("Encontrado en  ndice : " + indice);
16          } else {
17              System.out.println("No encontrado");
18          }
19
20          // B squeda de String
21          System.out.println("\nBuscando 'Elena':");
22          int indiceNombre = buscarNombre(nombres, "Elena");
23
24          if (indiceNombre != -1) {
25              System.out.println("Encontrado en  ndice : " +
                    indiceNombre);
26          } else {
27              System.out.println("No encontrado");
28          }
29
30          // B squeda de todas las ocurrencias
31          System.out.println("\n=== B SQUEDA DE TODAS LAS OCURRENCIAS
                ===");
32          int[] duplicados = {10, 20, 30, 20, 40, 20, 50};
33          int buscar = 20;
34
35          System.out.println("Buscando todas las ocurrencias de " +
                buscar + ":");
36          int[] indices = buscarTodos(duplicados, buscar);
37
38          if (indices.length > 0) {
39              System.out.print("Encontrado en  ndices : ");
40              for (int i : indices) {
41                  System.out.print(i + " ");
42              }
43              System.out.println();
```

```
            System.out.println("Total de ocurrencias: " +
                indices.length);
        } else {
            System.out.println("No encontrado");
        }

        // B squeda del valor m ximo y m nimo
        System.out.println("\n=== VALOR M XIMO Y M NIMO ===");
        System.out.println("Array: " +
            java.util.Arrays.toString(numeros));

        int maximo = encontrarMaximo(numeros);
        int minimo = encontrarMinimo(numeros);

        System.out.println("Valor m ximo: " + maximo);
        System.out.println("Valor m nimo: " + minimo);

        // B squeda de elementos que cumplen una condici n
        System.out.println("\n=== B SQUEDA CON CONDICI N ===");
        System.out.println("N meros mayores que 50:");

        int[] mayores = buscarMayoresQue(numeros, 50);
        System.out.println("Encontrados: " +
            java.util.Arrays.toString(mayores));
    }

    public static int buscarNumero(int[] arr, int objetivo) {
        for (int i = 0; i < arr.length; i++) {
            if (arr[i] == objetivo) {
                return i;  // Retorna el  ndice  del primer
                    encontrado
            }
        }
        return -1;  // No encontrado
    }

    public static int buscarNombre(String[] arr, String objetivo) {
        for (int i = 0; i < arr.length; i++) {
            if (arr[i].equals(objetivo)) {
                return i;
            }
        }
        return -1;
    }

```

```
85      public static int[] buscarTodos(int[] arr, int objetivo) {
86          // Primero contar cu ntas ocurrencias hay
87          int contador = 0;
88          for (int num : arr) {
89              if (num == objetivo) {
90                  contador++;
91              }
92          }
93
94          // Crear array del tama o adecuado
95          int[] resultados = new int[contador];
96          int indiceResultados = 0;
97
98          // Llenar el array con los  ndices
99          for (int i = 0; i < arr.length; i++) {
100             if (arr[i] == objetivo) {
101                 resultados[indiceResultados] = i;
102                 indiceResultados++;
103             }
104         }
105
106         return resultados;
107     }
108
109     public static int encontrarMaximo(int[] arr) {
110         int max = arr[0];
111         for (int i = 1; i < arr.length; i++) {
112             if (arr[i] > max) {
113                 max = arr[i];
114             }
115         }
116         return max;
117     }
118
119     public static int encontrarMinimo(int[] arr) {
120         int min = arr[0];
121         for (int i = 1; i < arr.length; i++) {
122             if (arr[i] < min) {
123                 min = arr[i];
124             }
125         }
126         return min;
127     }
128
129     public static int[] buscarMayoresQue(int[] arr, int limite) {
```

```
        // Contar cu ntos son mayores
        int contador = 0;
        for (int num : arr) {
            if (num > limite) {
                contador++;
            }
        }

        // Crear array y llenarlo
        int[] resultados = new int[contador];
        int indice = 0;

        for (int num : arr) {
            if (num > limite) {
                resultados[indice] = num;
                indice++;
            }
        }

        return resultados;
    }
}
```

Listing 8.6: Búsqueda lineal en arrays

8.5.2. Búsqueda Binaria (para arrays ordenados)

```
public class BusquedaBinaria {

    public static void main(String[] args) {
        // Array debe estar ordenado para b squeda binaria
        int[] numeros = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};

        System.out.println("Array ordenado: " +
           java.util.Arrays.toString(numeros));
        System.out.println("\n=== B SQUEDA BINARIA ===\n");

        // Buscar diferentes valores
        int[] buscar = {30, 70, 15, 100, 0};

        for (int objetivo : buscar) {
            int indice = busquedaBinaria(numeros, objetivo);

            if (indice != -1) {
```

```
17                System.out.println(objetivo + " encontrado en
                      ndice : " + indice);
18            } else {
19                System.out.println(objetivo + " no encontrado en el
                     array");
20            }
21        }
22
23        // Usar la b squeda binaria de Arrays de Java
24        System.out.println("\n=== USANDO Arrays.binarySearch() ===");
25        int indiceJava = java.util.Arrays.binarySearch(numeros, 60);
26        System.out.println("60 encontrado en  ndice  (Java): " +
             indiceJava);
27
28        // Ejemplo con array no ordenado (NO usar b squeda binaria)
29        System.out.println("\n=== ARRAY NO ORDENADO ===");
30        int[] desordenado = {45, 23, 67, 12, 89, 34};
31
32        // Primero debemos ordenarlo
33        java.util.Arrays.sort(desordenado);
34        System.out.println("Array ordenado: " +
             java.util.Arrays.toString(desordenado));
35
36        // Ahora podemos buscar
37        int indiceDespues = busquedaBinaria(desordenado, 67);
38        System.out.println("67 encontrado en  ndice : " +
             indiceDespues);
39
40        // B squeda en rango espec fico
41        System.out.println("\n=== B SQUEDA EN RANGO ===");
42        int indiceRango = busquedaBinariaEnRango(numeros, 50, 3, 7);
43        System.out.println("Buscando 50 entre  ndices  3-7: " +
             indiceRango);
44    }
45
46    public static int busquedaBinaria(int[] arr, int objetivo) {
47        int izquierda = 0;
48        int derecha = arr.length - 1;
49
50        while (izquierda <= derecha) {
51            int medio = izquierda + (derecha - izquierda) / 2;
52
53            System.out.printf("Buscando: izquierda=%d, medio=%d,
                 derecha=%d%n",
54                              izquierda, medio, derecha);
```

```
55
56              if (arr[medio] == objetivo) {
57                  return medio;  // Encontrado
58              }
59
60              if (arr[medio] < objetivo) {
61                  izquierda = medio + 1;  // Buscar en la mitad derecha
62              } else {
63                  derecha = medio - 1;    // Buscar en la mitad
                        izquierda
64              }
65          }
66
67          return -1;  // No encontrado
68      }
69
70      public static int busquedaBinariaEnRango(int[] arr, int
            objetivo, int inicio, int fin) {
71          int izquierda = inicio;
72          int derecha = fin;
73
74          while (izquierda <= derecha) {
75              int medio = izquierda + (derecha - izquierda) / 2;
76
77              if (arr[medio] == objetivo) {
78                  return medio;
79              }
80
81              if (arr[medio] < objetivo) {
82                  izquierda = medio + 1;
83              } else {
84                  derecha = medio - 1;
85              }
86          }
87
88          return -1;
89      }
90  }
```

Listing 8.7: Búsqueda binaria

8.6. Arrays Multidimensionales

8.6.1. Arrays Bidimensionales (Matrices)

```java
public class Matrices {

    public static void main(String[] args) {
        System.out.println("=== ARRAYS BIDIMENSIONALES (MATRICES) ===\n");

        // Declaraci n e inicializaci n de matrices
        System.out.println("=== FORMAS DE CREAR MATRICES ===");

        // Forma 1: Declarar e inicializar con valores
        int[][] matriz1 = {
            {1, 2, 3},
            {4, 5, 6},
            {7, 8, 9}
        };

        System.out.println("Matriz 1 (3x3):");
        mostrarMatriz(matriz1);

        // Forma 2: Declarar con tama o espec fico
        int[][] matriz2 = new int[2][4];  // 2 filas, 4 columnas

        // Llenar con valores
        int valor = 1;
        for (int i = 0; i < matriz2.length; i++) {
            for (int j = 0; j < matriz2[i].length; j++) {
                matriz2[i][j] = valor++;
            }
        }

        System.out.println("\nMatriz 2 (2x4):");
        mostrarMatriz(matriz2);

        // Forma 3: Matriz irregular (filas de diferente longitud)
        int[][] matriz3 = new int[3][];
        matriz3[0] = new int[2];  // Primera fila con 2 columnas
        matriz3[1] = new int[4];  // Segunda fila con 4 columnas
        matriz3[2] = new int[3];  // Tercera fila con 3 columnas

        // Llenar valores
```

```
        valor = 10;
        for (int i = 0; i < matriz3.length; i++) {
            for (int j = 0; j < matriz3[i].length; j++) {
                matriz3[i][j] = valor++;
            }
        }

        System.out.println("\nMatriz 3 (irregular):");
        mostrarMatriz(matriz3);

        // Operaciones con matrices
        System.out.println("\n=== OPERACIONES CON MATRICES ===");

        // Suma de todos los elementos
        int suma = 0;
        for (int i = 0; i < matriz1.length; i++) {
            for (int j = 0; j < matriz1[i].length; j++) {
                suma += matriz1[i][j];
            }
        }
        System.out.println("Suma de matriz1: " + suma);

        // Encontrar m ximo y m nimo
        int maximo = matriz1[0][0];
        int minimo = matriz1[0][0];

        for (int i = 0; i < matriz1.length; i++) {
            for (int j = 0; j < matriz1[i].length; j++) {
                if (matriz1[i][j] > maximo) {
                    maximo = matriz1[i][j];
                }
                if (matriz1[i][j] < minimo) {
                    minimo = matriz1[i][j];
                }
            }
        }

        System.out.println("M ximo en matriz1: " + maximo);
        System.out.println("M nimo en matriz1: " + minimo);

        // Suma de matrices (deben tener misma dimensi n)
        System.out.println("\n=== SUMA DE MATRICES ===");

        int[][] A = {
            {1, 2, 3},
```

```
85              {4, 5, 6}
86          };
87
88          int[][] B = {
89              {10, 20, 30},
90              {40, 50, 60}
91          };
92
93          System.out.println("Matriz A:");
94          mostrarMatriz(A);
95
96          System.out.println("Matriz B:");
97          mostrarMatriz(B);
98
99          // Sumar A + B
100         if (A.length == B.length && A[0].length == B[0].length) {
101             int[][] C = new int[A.length][A[0].length];
102
103             for (int i = 0; i < A.length; i++) {
104                 for (int j = 0; j < A[i].length; j++) {
105                     C[i][j] = A[i][j] + B[i][j];
106                 }
107             }
108
109             System.out.println("Matriz C (A + B):");
110             mostrarMatriz(C);
111         } else {
112             System.out.println("Las matrices no tienen la misma
                dimensi n.");
113         }
114
115         // Transpuesta de una matriz
116         System.out.println("\n=== MATRIZ TRANSPUESTA ===");
117
118         int[][] original = {
119             {1, 2, 3},
120             {4, 5, 6},
121             {7, 8, 9}
122         };
123
124         System.out.println("Matriz original:");
125         mostrarMatriz(original);
126
127         int[][] transpuesta = new
            int[original[0].length][original.length];
```

```
128
129         for (int i = 0; i < original.length; i++) {
130             for (int j = 0; j < original[i].length; j++) {
131                 transpuesta[j][i] = original[i][j];
132             }
133         }
134
135         System.out.println("Matriz transpuesta:");
136         mostrarMatriz(transpuesta);
137     }
138
139     public static void mostrarMatriz(int[][] matriz) {
140         for (int i = 0; i < matriz.length; i++) {
141             for (int j = 0; j < matriz[i].length; j++) {
142                 System.out.printf("%4d", matriz[i][j]);
143             }
144             System.out.println();
145         }
146     }
147 }
```

Listing 8.8: Trabajo con matrices

8.7. Ejemplo Práctico: Sistema de Gestión de Inventario

```
1  import java.util.Scanner;
2
3  public class SistemaInventario {
4
5      // Constantes
6      static final int MAX_PRODUCTOS = 100;
7      static final int CAMPOS_PRODUCTO = 3;  // ID, Nombre, Cantidad
8
9      // Arrays para almacenar productos
10     static int[] ids = new int[MAX_PRODUCTOS];
11     static String[] nombres = new String[MAX_PRODUCTOS];
12     static int[] cantidades = new int[MAX_PRODUCTOS];
13     static int totalProductos = 0;
14
15     public static void main(String[] args) {
16         Scanner scanner = new Scanner(System.in);
```

```
17
18          // Inicializar con algunos productos de ejemplo
19          inicializarInventario();
20
21          System.out.println("=== SISTEMA DE GESTI N DE INVENTARIO
                ===\n");
22
23          boolean ejecutando = true;
24
25          while (ejecutando) {
26              mostrarMenuPrincipal();
27              int opcion = leerEntero(scanner, "Selecciona una
                    opci n: ", 1, 7);
28
29              switch (opcion) {
30                  case 1:
31                      mostrarInventarioCompleto();
32                      break;
33                  case 2:
34                      agregarProducto(scanner);
35                      break;
36                  case 3:
37                      buscarProducto(scanner);
38                      break;
39                  case 4:
40                      actualizarProducto(scanner);
41                      break;
42                  case 5:
43                      realizarMovimiento(scanner);
44                      break;
45                  case 6:
46                      generarReportes();
47                      break;
48                  case 7:
49                      System.out.println("Saliendo del sistema...");
50                      ejecutando = false;
51                      break;
52              }
53
54              if (ejecutando) {
55                  System.out.println("\nPresiona Enter para
                        continuar...");
56                  scanner.nextLine();
57              }
58          }
```

```
59
60          scanner.close();
61      }
62
63      public static void inicializarInventario() {
64          // Agregar productos de ejemplo
65          agregarProducto(101, "Laptop", 10);
66          agregarProducto(102, "Mouse", 50);
67          agregarProducto(103, "Teclado", 30);
68          agregarProducto(104, "Monitor", 15);
69          agregarProducto(105, "Impresora", 8);
70      }
71
72      public static void agregarProducto(int id, String nombre, int
            cantidad) {
73          if (totalProductos < MAX_PRODUCTOS) {
74              ids[totalProductos] = id;
75              nombres[totalProductos] = nombre;
76              cantidades[totalProductos] = cantidad;
77              totalProductos++;
78          }
79      }
80
81      public static void mostrarMenuPrincipal() {
82          System.out.println("\n=== MEN  PRINCIPAL ===");
83          System.out.println("1. Mostrar inventario completo");
84          System.out.println("2. Agregar nuevo producto");
85          System.out.println("3. Buscar producto");
86          System.out.println("4. Actualizar producto");
87          System.out.println("5. Realizar movimiento
                (entrada/salida)");
88          System.out.println("6. Generar reportes");
89          System.out.println("7. Salir");
90          System.out.println("======================");
91      }
92
93      public static int leerEntero(Scanner scanner, String mensaje,
            int min, int max) {
94          int numero = 0;
95          boolean valido = false;
96
97          while (!valido) {
98              try {
99                  System.out.print(mensaje);
100                 numero = scanner.nextInt();
```

```
101
102                 if (numero >= min && numero <= max) {
103                     valido = true;
104                 } else {
105                     System.out.printf("Error: Debe ser entre %d y
                           %d.%n", min, max);
106                 }
107             } catch (Exception e) {
108                 System.out.println("Error: Ingresa un n mero entero
                       v lido.");
109                 scanner.nextLine();
110             }
111         }
112
113         scanner.nextLine();  // Limpiar buffer
114         return numero;
115     }
116
117     public static void mostrarInventarioCompleto() {
118         if (totalProductos == 0) {
119             System.out.println("El inventario est  vac o.");
120             return;
121         }
122
123         System.out.println("\n=== INVENTARIO COMPLETO ===");
124         System.out.printf("%-10s %-20s %-15s %-15s%n",
125                           "ID", "Nombre", "Cantidad", "Estado");
126         System.out.println("-------------------------------------------------------
127
128         for (int i = 0; i < totalProductos; i++) {
129             String estado = "";
130             if (cantidades[i] == 0) {
131                 estado = "AGOTADO";
132             } else if (cantidades[i] < 5) {
133                 estado = "BAJO STOCK";
134             } else {
135                 estado = "DISPONIBLE";
136             }
137
138             System.out.printf("%-10d %-20s %-15d %-15s%n",
139                               ids[i], nombres[i], cantidades[i],
                                      estado);
140         }
141
142         System.out.println("-------------------------------------------------------
```

```
143         System.out.println("Total de productos: " + totalProductos);
144     }
145 
146     public static void agregarProducto(Scanner scanner) {
147         if (totalProductos >= MAX_PRODUCTOS) {
148             System.out.println("Error: Capacidad m xima
                    alcanzada.");
149             return;
150         }
151 
152         System.out.println("\n=== AGREGAR NUEVO PRODUCTO ===");
153 
154         // Leer datos
155         System.out.print("ID del producto: ");
156         int id = scanner.nextInt();
157         scanner.nextLine();  // Limpiar buffer
158 
159         // Verificar si el ID ya existe
160         if (buscarProductoPorId(id) != -1) {
161             System.out.println("Error: Ya existe un producto con ese
                    ID.");
162             return;
163         }
164 
165         System.out.print("Nombre del producto: ");
166         String nombre = scanner.nextLine();
167 
168         System.out.print("Cantidad inicial: ");
169         int cantidad = scanner.nextInt();
170 
171         // Agregar al array
172         agregarProducto(id, nombre, cantidad);
173         System.out.println("Producto agregado exitosamente.");
174     }
175 
176     public static int buscarProductoPorId(int id) {
177         for (int i = 0; i < totalProductos; i++) {
178             if (ids[i] == id) {
179                 return i;  // Retorna el  ndice
180             }
181         }
182         return -1;  // No encontrado
183     }
184 
185     public static void buscarProducto(Scanner scanner) {
```

```
186         if (totalProductos == 0) {
187             System.out.println("El inventario est  vac o.");
188             return;
189         }
190 
191         System.out.println("\n=== BUSCAR PRODUCTO ===");
192         System.out.println("1. Buscar por ID");
193         System.out.println("2. Buscar por nombre");
194         int opcion = leerEntero(scanner, "Selecciona opci n: ", 1,
                2);
195 
196         switch (opcion) {
197             case 1:
198                 System.out.print("Ingresa el ID a buscar: ");
199                 int id = scanner.nextInt();
200                 int indice = buscarProductoPorId(id);
201 
202                 if (indice != -1) {
203                     mostrarDetalleProducto(indice);
204                 } else {
205                     System.out.println("Producto no encontrado.");
206                 }
207                 break;
208 
209             case 2:
210                 scanner.nextLine();  // Limpiar buffer
211                 System.out.print("Ingresa el nombre o parte del
                        nombre: ");
212                 String nombre = scanner.nextLine().toLowerCase();
213 
214                 boolean encontrado = false;
215                 System.out.println("\nResultados de b squeda:");
216 
217                 for (int i = 0; i < totalProductos; i++) {
218                     if (nombres[i].toLowerCase().contains(nombre)) {
219                         mostrarDetalleProducto(i);
220                         encontrado = true;
221                     }
222                 }
223 
224                 if (!encontrado) {
225                     System.out.println("No se encontraron
                            productos.");
226                 }
227                 break;
```

```
228         }
229     }
230 
231     public static void mostrarDetalleProducto(int indice) {
232         System.out.println("\n--- Detalles del Producto ---");
233         System.out.println("ID: " + ids[indice]);
234         System.out.println("Nombre: " + nombres[indice]);
235         System.out.println("Cantidad: " + cantidades[indice]);
236 
237         String estado = "";
238         if (cantidades[indice] == 0) {
239             estado = "AGOTADO";
240         } else if (cantidades[indice] < 5) {
241             estado = "BAJO STOCK (se recomienda reabastecer)";
242         } else {
243             estado = "DISPONIBLE";
244         }
245 
246         System.out.println("Estado: " + estado);
247     }
248 
249     public static void actualizarProducto(Scanner scanner) {
250         if (totalProductos == 0) {
251             System.out.println("El inventario est  vac o.");
252             return;
253         }
254 
255         System.out.println("\n=== ACTUALIZAR PRODUCTO ===");
256         System.out.print("Ingresa el ID del producto a actualizar:
                ");
257         int id = scanner.nextInt();
258 
259         int indice = buscarProductoPorId(id);
260 
261         if (indice == -1) {
262             System.out.println("Producto no encontrado.");
263             return;
264         }
265 
266         System.out.println("\nProducto actual:");
267         mostrarDetalleProducto(indice);
268 
269         System.out.println("\n  Qu  deseas actualizar?");
270         System.out.println("1. Nombre");
271         System.out.println("2. Cantidad");
```

```
272         System.out.println("3. Ambos");
273         int opcion = leerEntero(scanner, "Selecciona: ", 1, 3);
274 
275         scanner.nextLine();  // Limpiar buffer
276 
277         switch (opcion) {
278             case 1:
279                 System.out.print("Nuevo nombre: ");
280                 nombres[indice] = scanner.nextLine();
281                 break;
282             case 2:
283                 System.out.print("Nueva cantidad: ");
284                 cantidades[indice] = scanner.nextInt();
285                 break;
286             case 3:
287                 System.out.print("Nuevo nombre: ");
288                 nombres[indice] = scanner.nextLine();
289                 System.out.print("Nueva cantidad: ");
290                 cantidades[indice] = scanner.nextInt();
291                 break;
292         }
293 
294         System.out.println("Producto actualizado exitosamente.");
295     }
296 
297     public static void realizarMovimiento(Scanner scanner) {
298         if (totalProductos == 0) {
299             System.out.println("El inventario est  vac o.");
300             return;
301         }
302 
303         System.out.println("\n=== REALIZAR MOVIMIENTO ===");
304         System.out.print("ID del producto: ");
305         int id = scanner.nextInt();
306 
307         int indice = buscarProductoPorId(id);
308 
309         if (indice == -1) {
310             System.out.println("Producto no encontrado.");
311             return;
312         }
313 
314         System.out.println("\nProducto: " + nombres[indice]);
315         System.out.println("Cantidad actual: " + cantidades[indice]);
316 
```

```
317         System.out.println("\nTipo de movimiento:");
318         System.out.println("1. Entrada (a adir stock)");
319         System.out.println("2. Salida (retirar stock)");
320         int tipo = leerEntero(scanner, "Selecciona: ", 1, 2);
321 
322         System.out.print("Cantidad: ");
323         int cantidad = scanner.nextInt();
324 
325         if (tipo == 1) {
326             // Entrada
327             cantidades[indice] += cantidad;
328             System.out.println("Entrada registrada. Nueva cantidad:
                    " + cantidades[indice]);
329         } else {
330             // Salida
331             if (cantidad > cantidades[indice]) {
332                 System.out.println("Error: No hay suficiente
                        stock.");
333             } else {
334                 cantidades[indice] -= cantidad;
335                 System.out.println("Salida registrada. Nueva
                        cantidad: " + cantidades[indice]);
336             }
337         }
338     }
339 
340     public static void generarReportes() {
341         if (totalProductos == 0) {
342             System.out.println("El inventario est  vac o.");
343             return;
344         }
345 
346         System.out.println("\n=== REPORTES DEL INVENTARIO ===\n");
347 
348         // Reporte 1: Productos agotados
349         System.out.println("1. PRODUCTOS AGOTADOS:");
350         boolean hayAgotados = false;
351 
352         for (int i = 0; i < totalProductos; i++) {
353             if (cantidades[i] == 0) {
354                 System.out.println("   - " + nombres[i] + " (ID: " +
                        ids[i] + ")");
355                 hayAgotados = true;
356             }
357         }
```

```
358
359         if (!hayAgotados) {
360             System.out.println("   No hay productos agotados.");
361         }
362
363         // Reporte 2: Productos con bajo stock (< 5)
364         System.out.println("\n2. PRODUCTOS CON BAJO STOCK (< 5
                unidades):");
365         boolean hayBajoStock = false;
366
367         for (int i = 0; i < totalProductos; i++) {
368             if (cantidades[i] > 0 && cantidades[i] < 5) {
369                 System.out.printf("   - %s (ID: %d) - Stock: %d%n",
370                                   nombres[i], ids[i], cantidades[i]);
371                 hayBajoStock = true;
372             }
373         }
374
375         if (!hayBajoStock) {
376             System.out.println("   No hay productos con bajo
                    stock.");
377         }
378
379         // Reporte 3: Estad sticas generales
380         System.out.println("\n3. ESTAD STICAS GENERALES:");
381
382         int totalUnidades = 0;
383         int productosDisponibles = 0;
384         int productosAgotados = 0;
385         int maxStock = cantidades[0];
386         int minStock = cantidades[0];
387         String productoMax = nombres[0];
388         String productoMin = nombres[0];
389
390         for (int i = 0; i < totalProductos; i++) {
391             totalUnidades += cantidades[i];
392
393             if (cantidades[i] > 0) {
394                 productosDisponibles++;
395             } else {
396                 productosAgotados++;
397             }
398
399             if (cantidades[i] > maxStock) {
400                 maxStock = cantidades[i];
```

```
401                 productoMax = nombres[i];
402             }
403
404             if (cantidades[i] < minStock) {
405                 minStock = cantidades[i];
406                 productoMin = nombres[i];
407             }
408         }
409
410         double promedioStock = (double) totalUnidades /
                totalProductos;
411
412         System.out.println("   Total de productos: " +
                totalProductos);
413         System.out.println("   Total de unidades: " + totalUnidades);
414         System.out.println("   Productos disponibles: " +
                productosDisponibles);
415         System.out.println("   Productos agotados: " +
                productosAgotados);
416         System.out.printf("   Promedio de stock por producto: %.1f
                unidades%n", promedioStock);
417         System.out.println("   Producto con m s stock: " +
                productoMax + " (" + maxStock + " unidades)");
418         System.out.println("   Producto con menos stock: " +
                productoMin + " (" + minStock + " unidades)");
419
420         // Reporte 4: Valor total del inventario (simulado)
421         System.out.println("\n4. VALOR ESTIMADO DEL INVENTARIO:");
422
423         // Supongamos precios fijos para el ejemplo
424         double valorTotal = 0;
425         for (int i = 0; i < totalProductos; i++) {
426             double precioUnitario = 0;
427
428             // Precios simulados basados en el nombre
429             if (nombres[i].toLowerCase().contains("laptop")) {
430                 precioUnitario = 999.99;
431             } else if (nombres[i].toLowerCase().contains("mouse")) {
432                 precioUnitario = 25.50;
433             } else if (nombres[i].toLowerCase().contains("teclado"))
                    {
434                 precioUnitario = 45.75;
435             } else if (nombres[i].toLowerCase().contains("monitor"))
                    {
436                 precioUnitario = 299.99;
```

```
437             } else if
                    (nombres[i].toLowerCase().contains("impresora")) {
438                 precioUnitario = 199.99;
439             } else {
440                 precioUnitario = 50.00;  // Precio por defecto
441             }
442
443             valorTotal += cantidades[i] * precioUnitario;
444         }
445
446         System.out.printf("   Valor total estimado: $%.2f%n",
                valorTotal);
447     }
448 }
```

Listing 8.9: Sistema de inventario con arrays

Ejercicio Propuesto

Ejercicio 8.1: Gestor de Calificaciones

Crea un programa que gestione calificaciones de estudiantes:

1. Permite registrar estudiantes (nombre, número de identificación)
2. Permite registrar calificaciones para 5 materias diferentes
3. Calcula el promedio por estudiante
4. Determina si el estudiante aprueba (promedio 70)
5. Genera reportes:
   - Lista de estudiantes aprobados/reprobados
   - Promedio general del grupo
   - Mejor y peor promedio
   - Distribución de calificaciones por materia
6. Usa arrays bidimensionales para las calificaciones

Ejercicio Propuesto

Ejercicio 8.2: Juego del Buscaminas Simple

Crea una versión simple del juego Buscaminas:

1. Crea un tablero de 8x8 usando arrays bidimensionales
2. Coloca minas aleatoriamente en el tablero
3. Calcula números que indican minas adyacentes
4. Permite al usuario revelar casillas
5. Detecta cuando el usuario gana o pierde
6. Muestra el tablero después de cada jugada
7. Lleva registro del tiempo y jugadas

Ejercicio Propuesto

Ejercicio 8.3: Sistema de Reservación de Asientos
Crea un sistema de reservación para un cine:

1. Crea una sala con 10 filas y 8 asientos por fila
2. Muestra el estado de los asientos (disponible/ocupado)
3. Permite reservar múltiples asientos
4. Valida que los asientos estén disponibles
5. Calcula el costo total (diferentes precios por fila)
6. Permite cancelar reservaciones
7. Genera reportes de ocupación y ganancias
8. Guarda las reservaciones en arrays

8.8. Resumen del Capítulo

- Los arrays almacenan colecciones de elementos del mismo tipo
- Tienen tamaño fijo una vez creados
- Se accede a los elementos mediante índices (comenzando en 0)
- Se pueden recorrer con bucles `for` tradicional o `for-each`
- Los arrays se pasan por referencia a los métodos

- Se pueden copiar usando `System.arraycopy()`, `clone()`, o manualmente
- La búsqueda lineal recorre el array elemento por elemento
- La búsqueda binaria requiere un array ordenado pero es más eficiente
- Los arrays multidimensionales (matrices) son arrays de arrays
- Las matrices pueden ser regulares (todas las filas iguales) o irregulares
- Los arrays son fundamentales para trabajar con colecciones de datos en programación lineal

Capítulo 9

Buenas Prácticas en Programación Lineal

9.1. Introducción a las Buenas Prácticas

Las buenas prácticas de programación no solo hacen que tu código funcione, sino que lo hacen **mantenible, legible y eficiente**. En este capítulo aprenderemos principios fundamentales para escribir código de calidad.

Nota Importante

Un buen programador no solo escribe código que funciona, sino código que otros (o él mismo en el futuro) puedan entender y modificar fácilmente.

9.2. Código Legible y Mantenible

9.2.1. Nombres Significativos

```java
public class NombresSignificativos {

    // MALOS NOMBRES
    public static void m() {  //   Qu   hace 'm'?
        int x = 10;           //   Qu   representa 'x'?
        int y = 20;           //   Qu   representa 'y'?
        int z = x + y;        //   Qu   es 'z'?
        System.out.println(z);
    }

    // BUENOS NOMBRES
    public static void calcularYMostrarSuma() {
        int primerNumero = 10;
        int segundoNumero = 20;
        int suma = primerNumero + segundoNumero;
        System.out.println("La suma es: " + suma);
```

```
    }

    // M S EJEMPLOS
    public static void procesar() {  // MAL - muy gen rico
        // C digo...
    }

    public static void calcularPromedioCalificaciones() {  // BIEN -
        espec fico
        // C digo...
    }

    // VARIABLES
    public static void ejemploVariables() {
        // MAL
        int d;  //   'd' de qu ?   das   ?   distancia   ?   datos   ?
        int n;  //   nmero   ?   nombre   ?   nada   ?
        String s;  //   string   ?   sistema   ?   saludo   ?

        // BIEN
        int diasTrabajados;
        int numeroDeIntentos;
        String nombreUsuario;
        double precioConIVA;
        boolean esValido;

        // CONSTANTES
        final double TASA_INTERES = 0.05;  // BIEN - may sculas con
            guiones bajos
        final int MAXIMO_INTENTOS = 3;
        final String NOMBRE_APLICACION = "Mi Programa";
    }
}
```

Listing 9.1: Ejemplos de buenos y malos nombres

9.2.2. Formato y Estructura

```
public class FormatoConsistente {

    // MAL FORMATO
    public static void
        metodoMalFormateado(){System.out.println("Hola");
    for(int i=0;i<10;i++){System.out.println(i);}
```

```
6       if(true){System.out.println("Siempre verdadero");}
7       }
8
9       // BIEN FORMATO
10      public static void metodoBienFormateado() {
11          System.out.println("Hola");
12
13          for (int i = 0; i < 10; i++) {
14              System.out.println(i);
15          }
16
17          if (true) {
18              System.out.println("Siempre verdadero");
19          }
20      }
21
22      // INDENTACI N CONSISTENTE
23      public static void ejemploIndentacion() {
24          // Usa siempre la misma cantidad de espacios (recomendado: 4)
25          int x = 10;
26          int y = 20;
27
28          if (x > y) {
29              System.out.println("x es mayor");
30
31              for (int i = 0; i < 5; i++) {
32                  System.out.println("Iteraci n: " + i);
33
34                  if (i % 2 == 0) {
35                      System.out.println("Par");
36                  }
37              }
38          } else {
39              System.out.println("y es mayor o igual");
40          }
41      }
42
43      // ESPACIADO ADECUADO
44      public static void ejemploEspaciado() {
45          // MAL - muy apretado
46          int a=5,b=10,c=15;
47          double resultado=a*b/c;
48
49          // BIEN - espaciado adecuado
50          int a = 5;
```

```
51          int b = 10;
52          int c = 15;
53          double resultado = a * b / c;
54
55          // Espacios alrededor de operadores
56          int suma = a + b;
57          boolean esMayor = a > b;
58
59          // Espacios despu s de comas
60          System.out.printf("Valores: %d, %d, %d%n", a, b, c);
61
62          // Espacios antes de llaves
63          if (a > 0) {
64              // C digo...
65          }
66
67          for (int i = 0; i < 10; i++) {
68              // C digo...
69          }
70      }
71
72      // L NEAS DE LONGIDAD RAZONABLE
73      public static void ejemploLongitudLineas() {
74          // MAL - l nea muy larga
75          System.out.println("Este es un mensaje muy muy muy muy muy
                muy muy muy muy muy muy muy muy muy muy muy muy muy muy
                muy largo que dificulta la lectura");
76
77          // BIEN - l nea dividida
78          System.out.println("Este es un mensaje razonablemente largo
                " +
79                             "que se divide en m ltiples l neas para
                                   " +
80                             "facilitar la lectura del c digo.");
81
82          // Otro ejemplo
83          double resultado = calcularValorComplejo(argumento1,
                argumento2,
84                                                   argumento3,
                                                        argumento4);
85      }
86
87      // ORDEN L GICO DE M TODOS
88      public static void metodoPrincipal() {
89          // Primero el m todo principal
```

```
90          inicializar();
91          procesarDatos();
92          mostrarResultados();
93          finalizar();
94      }
95
96      public static void inicializar() {
97          // M todos auxiliares despu s
98      }
99
100     public static void procesarDatos() {
101         // Agrupar m todos relacionados
102     }
103
104     public static void mostrarResultados() {
105         // ...
106     }
107
108     public static void finalizar() {
109         // ...
110     }
111 }
```

Listing 9.2: Formato consistente del código

9.3. Estructuración del Código

9.3.1. El Principio de Responsabilidad Única

```
1  public class ResponsabilidadUnica {
2
3      // MAL - un m todo hace demasiadas cosas
4      public static void procesarTodo() {
5          // 1. Leer datos
6          java.util.Scanner scanner = new java.util.Scanner(System.in);
7          System.out.print("Ingresa un n mero: ");
8          int numero = scanner.nextInt();
9
10         // 2. Validar datos
11         if (numero < 0 || numero > 100) {
12             System.out.println("N mero inv lido");
13             return;
14         }
```

```
15
16          // 3. Procesar datos
17          int resultado = 0;
18          for (int i = 1; i <= numero; i++) {
19              resultado += i;
20          }
21
22          // 4. Mostrar resultados
23          System.out.println("Resultado: " + resultado);
24
25          // 5. Guardar resultados (simulado)
26          System.out.println("Resultado guardado en archivo...");
27
28          scanner.close();
29      }
30
31      // BIEN - cada m todo tiene una responsabilidad
32      public static void procesoModular() {
33          int numero = leerNumero();
34
35          if (validarNumero(numero)) {
36              int resultado = calcularSuma(numero);
37              mostrarResultado(resultado);
38              guardarResultado(resultado);
39          } else {
40              System.out.println("N mero inv lido");
41          }
42      }
43
44      public static int leerNumero() {
45          java.util.Scanner scanner = new java.util.Scanner(System.in);
46          System.out.print("Ingresa un n mero: ");
47          int numero = scanner.nextInt();
48          scanner.close();
49          return numero;
50      }
51
52      public static boolean validarNumero(int numero) {
53          return numero >= 0 && numero <= 100;
54      }
55
56      public static int calcularSuma(int numero) {
57          int suma = 0;
58          for (int i = 1; i <= numero; i++) {
59              suma += i;
```

```
60          }
61          return suma;
62      }
63
64      public static void mostrarResultado(int resultado) {
65          System.out.println("Resultado: " + resultado);
66      }
67
68      public static void guardarResultado(int resultado) {
69          // Simulaci n de guardar en archivo
70          System.out.println("Resultado " + resultado + " guardado en
                archivo...");
71      }
72  }
```

Listing 9.3: Principio de responsabilidad única

9.3.2. Cohesión y Acoplamiento

```
1   public class CohesionAcoplamiento {
2
3       // MAL - baja cohesi n (m todos no relacionados)
4       public static class UtilidadesMal {
5           public static void sumarNumeros(int a, int b) {
6               System.out.println(a + b);
7           }
8
9           public static void imprimirFecha() {
10              System.out.println(java.time.LocalDate.now());
11          }
12
13          public static void convertirMoneda(double cantidad) {
14              System.out.println(cantidad * 0.85);
15          }
16      }
17
18      // BIEN - alta cohesi n (m todos relacionados)
19      public static class Calculadora {
20          public static int sumar(int a, int b) {
21              return a + b;
22          }
23
24          public static int restar(int a, int b) {
25              return a - b;
```

```
26          }
27
28          public static int multiplicar(int a, int b) {
29              return a * b;
30          }
31
32          public static double dividir(int a, int b) {
33              if (b != 0) {
34                  return (double) a / b;
35              } else {
36                  return Double.NaN;
37              }
38          }
39      }
40
41      public static class UtilidadesFecha {
42          public static String obtenerFechaActual() {
43              return java.time.LocalDate.now().toString();
44          }
45
46          public static String obtenerHoraActual() {
47              return java.time.LocalTime.now().toString();
48          }
49
50          public static int calcularDiferenciaDias(String fecha1,
                String fecha2) {
51              // Implementaci n simplificada
52              return Math.abs(fecha1.compareTo(fecha2));
53          }
54      }
55
56      // EJEMPLO DE ACOPLAMIENTO
57      public static void procesoAcoplado() {
58          // MAL - alto acoplamiento
59          int[] datos = {1, 2, 3, 4, 5};
60
61          // Muchas variables interconectadas
62          int suma = 0;
63          for (int i = 0; i < datos.length; i++) {
64              suma += datos[i];
65          }
66
67          double promedio = (double) suma / datos.length;
68
69          int mayores = 0;
```

```
70          for (int i = 0; i < datos.length; i++) {
71              if (datos[i] > promedio) {
72                  mayores++;
73              }
74          }
75
76          // Muchos c lculos interconectados
77          System.out.println("Suma: " + suma);
78          System.out.println("Promedio: " + promedio);
79          System.out.println("Mayores que promedio: " + mayores);
80      }
81
82      public static void procesoDesacoplado() {
83          // BIEN - bajo acoplamiento
84          int[] datos = {1, 2, 3, 4, 5};
85
86          int suma = calcularSuma(datos);
87          double promedio = calcularPromedio(datos);
88          int mayores = contarMayoresQuePromedio(datos, promedio);
89
90          mostrarEstadisticas(suma, promedio, mayores);
91      }
92
93      public static int calcularSuma(int[] arr) {
94          int suma = 0;
95          for (int num : arr) {
96              suma += num;
97          }
98          return suma;
99      }
100
101     public static double calcularPromedio(int[] arr) {
102         return (double) calcularSuma(arr) / arr.length;
103     }
104
105     public static int contarMayoresQuePromedio(int[] arr, double
        promedio) {
106         int contador = 0;
107         for (int num : arr) {
108             if (num > promedio) {
109                 contador++;
110             }
111         }
112         return contador;
113     }
```

```
114
115     public static void mostrarEstadisticas(int suma, double
            promedio, int mayores) {
116         System.out.println("Suma: " + suma);
117         System.out.println("Promedio: " + promedio);
118         System.out.println("Mayores que promedio: " + mayores);
119     }
120 }
```

Listing 9.4: Cohesión alta y acoplamiento bajo

9.4. Manejo de Errores y Validación

9.4.1. Validación de Entrada

```
1  import java.util.Scanner;
2  import java.util.InputMismatchException;
3
4  public class ValidacionRobusta {
5
6      public static void main(String[] args) {
7          Scanner scanner = new Scanner(System.in);
8
9          System.out.println("=== SISTEMA DE VALIDACI N ROBUSTA
               ===\n");
10
11         // Ejemplo 1: Validar n mero entero en rango
12         int edad = leerEnteroEnRango(scanner, "Ingresa tu edad: ",
               0, 120);
13         System.out.println("Edad aceptada: " + edad);
14
15         // Ejemplo 2: Validar n mero decimal positivo
16         double precio = leerDoublePositivo(scanner, "Ingresa el
               precio: ");
17         System.out.printf("Precio aceptado: $%.2f%n", precio);
18
19         // Ejemplo 3: Validar String no vac o
20         String nombre = leerStringNoVacio(scanner, "Ingresa tu
               nombre: ");
21         System.out.println("Nombre aceptado: " + nombre);
22
23         // Ejemplo 4: Validar opci n de men
```

```
24          String[] opciones = {"Guardar", "Editar", "Eliminar",
                "Salir"};
25          int opcion = leerOpcionMenu(scanner, opciones);
26          System.out.println("Opci n seleccionada: " +
                opciones[opcion - 1]);
27
28          scanner.close();
29      }
30
31      public static int leerEnteroEnRango(Scanner scanner, String
            mensaje,
32                                          int min, int max) {
33          int numero = 0;
34          boolean valido = false;
35
36          while (!valido) {
37              try {
38                  System.out.print(mensaje);
39                  numero = scanner.nextInt();
40                  scanner.nextLine();  // Limpiar buffer
41
42                  if (numero >= min && numero <= max) {
43                      valido = true;
44                  } else {
45                      System.out.printf("Error: El n mero debe estar
                            entre %d y %d.%n",
46                                        min, max);
47                  }
48              } catch (InputMismatchException e) {
49                  System.out.println("Error: Debes ingresar un n mero
                        entero v lido.");
50                  scanner.nextLine();  // Limpiar buffer
51              }
52          }
53
54          return numero;
55      }
56
57      public static double leerDoublePositivo(Scanner scanner, String
            mensaje) {
58          double numero = 0;
59          boolean valido = false;
60
61          while (!valido) {
62              try {
```

```
63                  System.out.print(mensaje);
64                  numero = scanner.nextDouble();
65                  scanner.nextLine();  // Limpiar buffer
66
67                  if (numero > 0) {
68                      valido = true;
69                  } else {
70                      System.out.println("Error: El n mero debe ser
                            positivo.");
71                  }
72              } catch (InputMismatchException e) {
73                  System.out.println("Error: Debes ingresar un n mero
                        v lido.");
74                  scanner.nextLine();  // Limpiar buffer
75              }
76          }
77
78          return numero;
79      }
80
81      public static String leerStringNoVacio(Scanner scanner, String
            mensaje) {
82          String texto = "";
83          boolean valido = false;
84
85          while (!valido) {
86              System.out.print(mensaje);
87              texto = scanner.nextLine().trim();
88
89              if (!texto.isEmpty()) {
90                  valido = true;
91              } else {
92                  System.out.println("Error: Este campo no puede estar
                        vac o.");
93              }
94          }
95
96          return texto;
97      }
98
99      public static int leerOpcionMenu(Scanner scanner, String[]
            opciones) {
100         // Mostrar men
101         System.out.println("\n===  MEN   ===");
102         for (int i = 0; i < opciones.length; i++) {
```

```
103             System.out.println((i + 1) + ". " + opciones[i]);
104         }
105 
106         // Leer opci n
107         return leerEnteroEnRango(scanner,
108                                  "Selecciona una opci n (1-" +
                                        opciones.length + "): ",
109                                  1, opciones.length);
110     }
111 
112     // M todo para validar email (simple)
113     public static boolean validarEmail(String email) {
114         if (email == null || email.isEmpty()) {
115             return false;
116         }
117 
118         // Verificar que tenga @ y .
119         int arrobaIndex = email.indexOf('@');
120         int puntoIndex = email.lastIndexOf('.');
121 
122         return arrobaIndex > 0 &&
123                puntoIndex > arrobaIndex + 1 &&
124                puntoIndex < email.length() - 1;
125     }
126 
127     // M todo para validar contrase a
128     public static boolean validarContrase a(String contrase a) {
129         if (contrase a == null || contrase a.length() < 8) {
130             return false;
131         }
132 
133         boolean tieneMayuscula = false;
134         boolean tieneMinuscula = false;
135         boolean tieneNumero = false;
136         boolean tieneEspecial = false;
137 
138         for (char c : contrase a.toCharArray()) {
139             if (Character.isUpperCase(c)) {
140                 tieneMayuscula = true;
141             } else if (Character.isLowerCase(c)) {
142                 tieneMinuscula = true;
143             } else if (Character.isDigit(c)) {
144                 tieneNumero = true;
145             } else if (!Character.isLetterOrDigit(c)) {
146                 tieneEspecial = true;
```

```
147             }
148         }
149
150         return tieneMayuscula && tieneMinuscula && tieneNumero &&
                tieneEspecial;
151     }
152 }
```

Listing 9.5: Validación robusta de entrada

9.4.2. Manejo de Casos Especiales

```
1  public class CasosEspeciales {
2
3      public static void main(String[] args) {
4          System.out.println("=== MANEJO DE CASOS ESPECIALES ===\n");
5
6          // Casos de borde comunes
7          System.out.println("1. DIVISI N POR CERO:");
8          double resultado1 = dividirSeguro(10, 0);
9          System.out.println("10 / 0 = " + resultado1);
10
11         double resultado2 = dividirSeguro(10, 2);
12         System.out.println("10 / 2 = " + resultado2);
13
14         System.out.println("\n2. ARRAYS VAC OS:");
15         int[] arrayVacio = {};
16         int[] arrayNormal = {1, 2, 3, 4, 5};
17
18         System.out.println("Promedio array vac o: " +
               calcularPromedioSeguro(arrayVacio));
19         System.out.println("Promedio array normal: " +
               calcularPromedioSeguro(arrayNormal));
20
21         System.out.println("\n3. VALORES NULOS:");
22         String texto1 = null;
23         String texto2 = "Hola Mundo";
24
25         System.out.println("Longitud texto nulo: " +
               obtenerLongitudSegura(texto1));
26         System.out.println("Longitud texto normal: " +
               obtenerLongitudSegura(texto2));
27
28         System.out.println("\n4. RANGOS EXTREMOS:");
```

```
29          System.out.println("Factorial de -5: " +
                factorialSeguro(-5));
30          System.out.println("Factorial de 0: " + factorialSeguro(0));
31          System.out.println("Factorial de 5: " + factorialSeguro(5));
32          System.out.println("Factorial de 20: " +
                factorialSeguro(20));  // Cuidado con desbordamiento
33
34          System.out.println("\n5. CONVERSIONES SEGURAS:");
35          System.out.println("Convertir '123': " +
                convertirAIntSeguro("123"));
36          System.out.println("Convertir 'abc': " +
                convertirAIntSeguro("abc"));
37          System.out.println("Convertir '': " +
                convertirAIntSeguro(""));
38          System.out.println("Convertir null: " +
                convertirAIntSeguro(null));
39      }
40
41      // Manejo seguro de divisi n
42      public static double dividirSeguro(double numerador, double
            denominador) {
43          if (denominador == 0) {
44              System.out.println("  Advertencia: Divisi n por cero
                    detectada.");
45              return Double.NaN;  // Not a Number
46          }
47          return numerador / denominador;
48      }
49
50      // Manejo seguro de promedio con array
51      public static double calcularPromedioSeguro(int[] array) {
52          if (array == null || array.length == 0) {
53              System.out.println("  Advertencia: Array vac o o
                    nulo.");
54              return 0.0;
55          }
56
57          int suma = 0;
58          for (int num : array) {
59              suma += num;
60          }
61
62          return (double) suma / array.length;
63      }
64
```

```
65      // Manejo seguro de longitud de String
66      public static int obtenerLongitudSegura(String texto) {
67          if (texto == null) {
68              System.out.println("  Advertencia: String nulo.");
69              return 0;
70          }
71          return texto.length();
72      }
73
74      // Manejo seguro de factorial
75      public static long factorialSeguro(int n) {
76          if (n < 0) {
77              System.out.println("  Advertencia: Factorial de n mero
                    negativo.");
78              return -1;  // Valor especial para indicar error
79          }
80
81          if (n == 0 || n == 1) {
82              return 1;
83          }
84
85          // Verificar posible desbordamiento
86          if (n > 20) {  // 21! excede el l mite de long
87              System.out.println("  Advertencia: Posible
                    desbordamiento.");
88              return -1;
89          }
90
91          long resultado = 1;
92          for (int i = 2; i <= n; i++) {
93              resultado *= i;
94          }
95
96          return resultado;
97      }
98
99      // Conversi n segura de String a int
100     public static int convertirAIntSeguro(String texto) {
101         if (texto == null || texto.isEmpty()) {
102             System.out.println("  Advertencia: String nulo o
                    vac o.");
103             return 0;  // Valor por defecto
104         }
105
106         try {
```

```
107             return Integer.parseInt(texto);
108         } catch (NumberFormatException e) {
109             System.out.println("  Advertencia: No se puede convertir
                    a n mero.");
110             return 0;  // Valor por defecto
111         }
112     }
113 
114     // M todo para validar  ndice  de array
115     public static boolean esIndiceValido(int[] array, int indice) {
116         return array != null && indice >= 0 && indice < array.length;
117     }
118 
119     // Acceso seguro a elemento de array
120     public static int obtenerElementoSeguro(int[] array, int indice)
            {
121         if (!esIndiceValido(array, indice)) {
122             System.out.println("  Advertencia:  ndice  fuera de
                    rango.");
123             return 0;  // Valor por defecto
124         }
125 
126         return array[indice];
127     }
128 }
```

Listing 9.6: Manejo de casos especiales y bordes

9.5. Optimización y Eficiencia

9.5.1. Evitar Cálculos Redundantes

```
1  public class OptimizacionCalculos {
2  
3      public static void main(String[] args) {
4          System.out.println("=== OPTIMIZACI N DE C LCULOS ===\n");
5  
6          // Ejemplo 1: C lculos dentro de bucles
7          int[] numeros = new int[10000];
8  
9          // MAL - c lculo redundante en cada iteraci n
10         long inicioMal = System.currentTimeMillis();
11         for (int i = 0; i < numeros.length; i++) {
```

```
12              numeros[i] = i * (numeros.length - 1);  //
                    numeros.length se calcula 10000 veces
13          }
14          long finMal = System.currentTimeMillis();
15
16          // BIEN - c lculo fuera del bucle
17          long inicioBien = System.currentTimeMillis();
18          int longitud = numeros.length;  // Calcular una vez
19          for (int i = 0; i < longitud; i++) {
20              numeros[i] = i * (longitud - 1);  // Usar variable ya
                    calculada
21          }
22          long finBien = System.currentTimeMillis();
23
24          System.out.println("Tiempo MAL: " + (finMal - inicioMal) + "
                ms");
25          System.out.println("Tiempo BIEN: " + (finBien - inicioBien)
                + " ms");
26
27          // Ejemplo 2: Operaciones costosas repetidas
28          String texto = "Este es un texto de ejemplo para procesar";
29
30          // MAL
31          System.out.println("\nProcesamiento MAL:");
32          for (int i = 0; i < texto.length(); i++) {
33              char c = texto.toLowerCase().charAt(i);  //
                    toLowerCase() en cada iteraci n
34              System.out.print(c);
35          }
36          System.out.println();
37
38          // BIEN
39          System.out.println("\nProcesamiento BIEN:");
40          String textoMinusculas = texto.toLowerCase();  // Una sola
                vez
41          for (int i = 0; i < textoMinusculas.length(); i++) {
42              char c = textoMinusculas.charAt(i);
43              System.out.print(c);
44          }
45          System.out.println();
46
47          // Ejemplo 3: Condiciones optimizadas
48          int x = 10;
49          int y = 20;
50
```

```
51          // MAL - condiciones innecesariamente complejas
52          if (x > 0 && y > 0 && x < 100 && y < 100 && x + y > 50) {
53              System.out.println("\nCondici n compleja cumplida");
54          }
55
56          // BIEN - condiciones simplificadas y ordenadas
57          // Primero las condiciones m s probables de fallar
58          if (x <= 0 || y <= 0 || x >= 100 || y >= 100) {
59              // Condici n falla temprano
60          } else if (x + y > 50) {
61              System.out.println("\nCondici n optimizada cumplida");
62          }
63
64          // Ejemplo 4: Uso eficiente de Strings
65          System.out.println("\n=== CONCATENACI N DE STRINGS ===");
66
67          // MAL - concatenaci n en bucle (crea muchos objetos)
68          String resultadoMal = "";
69          for (int i = 0; i < 1000; i++) {
70              resultadoMal += "n mero " + i + ", ";  // Crea nuevo
                    String cada vez
71          }
72
73          // BIEN - StringBuilder para concatenaci n en bucle
74          StringBuilder sb = new StringBuilder();
75          for (int i = 0; i < 1000; i++) {
76              sb.append("n mero ").append(i).append(", ");
77          }
78          String resultadoBien = sb.toString();
79
80          System.out.println("Longitud resultado: " +
                resultadoBien.length());
81      }
82
83      // Ejemplo 5: Algoritmos eficientes
84      public static void ejemploAlgoritmos() {
85          System.out.println("\n=== ALGORITMOS EFICIENTES ===");
86
87          // B squeda en array desordenado vs ordenado
88          int[] arrayDesordenado = {45, 23, 67, 12, 89, 34, 78, 56,
                90, 33};
89          int[] arrayOrdenado = {12, 23, 33, 34, 45, 56, 67, 78, 89,
                90};
90          int buscar = 67;
91
```

```
92          // B squeda lineal (para arrays desordenados)
93          long inicioLineal = System.nanoTime();
94          int indiceLineal = busquedaLineal(arrayDesordenado, buscar);
95          long finLineal = System.nanoTime();
96
97          // B squeda binaria (para arrays ordenados)
98          long inicioBinaria = System.nanoTime();
99          int indiceBinaria = busquedaBinaria(arrayOrdenado, buscar);
100         long finBinaria = System.nanoTime();
101
102         System.out.println("B squeda lineal: " + (finLineal -
                inicioLineal) + " ns");
103         System.out.println("B squeda binaria: " + (finBinaria -
                inicioBinaria) + " ns");
104         System.out.println("Binaria es " +
105                            (double)(finLineal -
                                inicioLineal)/(finBinaria -
                                inicioBinaria) +
106                            " veces m s r pida");
107     }
108
109     public static int busquedaLineal(int[] arr, int objetivo) {
110         for (int i = 0; i < arr.length; i++) {
111             if (arr[i] == objetivo) {
112                 return i;
113             }
114         }
115         return -1;
116     }
117
118     public static int busquedaBinaria(int[] arr, int objetivo) {
119         int izquierda = 0;
120         int derecha = arr.length - 1;
121
122         while (izquierda <= derecha) {
123             int medio = izquierda + (derecha - izquierda) / 2;
124
125             if (arr[medio] == objetivo) {
126                 return medio;
127             }
128
129             if (arr[medio] < objetivo) {
130                 izquierda = medio + 1;
131             } else {
132                 derecha = medio - 1;
```

```
            }
        }

        return -1;
    }
}
```

Listing 9.7: Evitar cálculos redundantes

9.6. Documentación y Comentarios

```
/**
 * Clase: CalculadoraCientifica
 *
 * Descripci n: Proporciona operaciones matem ticas avanzadas
 * para c lculos cient ficos. Todas las operaciones son est ticas
 * y no requieren creaci n de objetos.
 *
 * Autor: [Tu Nombre]
 * Versi n: 1.0
 * Fecha: 2024
 */
public class CalculadoraCientifica {

    // Constantes matem ticas
    /** Valor de PI con 10 decimales de precisi n */
    public static final double PI = 3.1415926535;

    /** N mero de Euler (e) con 10 decimales de precisi n */
    public static final double E = 2.7182818284;

    /** Velocidad de la luz en m/s */
    public static final double VELOCIDAD_LUZ = 299792458;

    /**
     * Calcula el factorial de un n mero entero no negativo.
     *
     * @param n El n mero entero para calcular factorial
     * @return El factorial de n
     * @throws IllegalArgumentException si n es negativo
     */
    public static long factorial(int n) {
        // Validar entrada
```

```
33          if (n < 0) {
34              throw new IllegalArgumentException(
35                  "El factorial no est  definido para n meros
                        negativos. " +
36                  "Recibido: " + n
37              );
38          }
39
40          // Casos base
41          if (n == 0 || n == 1) {
42              return 1;
43          }
44
45          // Verificar posible desbordamiento
46          if (n > 20) {
47              System.out.println("Advertencia: 21! excede el l mite
                    de long.");
48              return -1; // Indicador de error
49          }
50
51          // C lculo iterativo
52          long resultado = 1;
53          for (int i = 2; i <= n; i++) {
54              resultado *= i;
55          }
56
57          return resultado;
58      }
59
60      /**
61       * Calcula la potencia de un n mero.
62       *
63       * @param base La base de la potencia
64       * @param exponente El exponente (entero)
65       * @return base elevado a exponente
66       */
67      public static double potencia(double base, int exponente) {
68          // Caso especial: cualquier n mero elevado a 0 es 1
69          if (exponente == 0) {
70              return 1;
71          }
72
73          // Manejar exponentes negativos
74          if (exponente < 0) {
75              return 1 / potencia(base, -exponente);
```

```
76          }
77
78          // Exponente positivo: multiplicaci n iterativa
79          double resultado = 1;
80          for (int i = 0; i < exponente; i++) {
81              resultado *= base;
82          }
83
84          return resultado;
85      }
86
87      /**
88       * Calcula el  rea  de un c rculo dado su radio.
89       *
90       * @param radio El radio del c rculo (debe ser positivo)
91       * @return El  rea  del c rculo
92       * @throws IllegalArgumentException si radio es negativo
93       */
94      public static double areaCirculo(double radio) {
95          if (radio < 0) {
96              throw new IllegalArgumentException(
97                  "El radio no puede ser negativo. Recibido: " + radio
98              );
99          }
100
101         return PI * radio * radio;
102     }
103
104     /**
105      * Calcula el volumen de una esfera dado su radio.
106      * F rmula: V = (4/3) *    * r
107      *
108      * @param radio El radio de la esfera (debe ser positivo)
109      * @return El volumen de la esfera
110      */
111     public static double volumenEsfera(double radio) {
112         if (radio < 0) {
113             throw new IllegalArgumentException(
114                 "El radio no puede ser negativo. Recibido: " + radio
115             );
116         }
117
118         return (4.0 / 3.0) * PI * radio * radio * radio;
119     }
120
```

```
121     /**
122      * Convierte grados Celsius a Fahrenheit.
123      * F rmula: F = (C    9/5) + 32
124      *
125      * @param celsius Temperatura en grados Celsius
126      * @return Temperatura equivalente en grados Fahrenheit
127      */
128     public static double celsiusAFahrenheit(double celsius) {
129         return (celsius * 9.0 / 5.0) + 32;
130     }
131 
132     /**
133      * Convierte grados Fahrenheit a Celsius.
134      * F rmula: C = (F - 32)    5/9
135      *
136      * @param fahrenheit Temperatura en grados Fahrenheit
137      * @return Temperatura equivalente en grados Celsius
138      */
139     public static double fahrenheitACelsius(double fahrenheit) {
140         return (fahrenheit - 32) * 5.0 / 9.0;
141     }
142 
143     /**
144      * Calcula el  ndice  de Masa Corporal (IMC).
145      * F rmula: IMC = peso / (altura )
146      *
147      * @param peso Peso en kilogramos (debe ser positivo)
148      * @param altura Altura en metros (debe ser positiva)
149      * @return El IMC calculado
150      * @throws IllegalArgumentException si peso o altura no son
         positivos
151      */
152     public static double calcularIMC(double peso, double altura) {
153         if (peso <= 0) {
154             throw new IllegalArgumentException(
155                 "El peso debe ser positivo. Recibido: " + peso
156             );
157         }
158 
159         if (altura <= 0) {
160             throw new IllegalArgumentException(
161                 "La altura debe ser positiva. Recibido: " + altura
162             );
163         }
164 
```

```
165         return peso / (altura * altura);
166     }
167 
168     /**
169      * Clasifica el IMC seg n los est ndares de la OMS.
170      *
171      * @param imc El valor del IMC a clasificar
172      * @return Una cadena con la clasificaci n
173      */
174     public static String clasificarIMC(double imc) {
175         // Comentarios explicativos para cada rango
176         if (imc < 18.5) {
177             return "Bajo peso";          // IMC < 18.5
178         } else if (imc < 25) {
179             return "Normal";             // 18.5     IMC < 25
180         } else if (imc < 30) {
181             return "Sobrepeso";          // 25     IMC < 30
182         } else if (imc < 35) {
183             return "Obesidad grado I";   // 30     IMC < 35
184         } else if (imc < 40) {
185             return "Obesidad grado II";  // 35     IMC < 40
186         } else {
187             return "Obesidad grado III"; // IMC     40
188         }
189     }
190 
191     /**
192      * M todo principal para probar la calculadora.
193      *
194      * @param args Argumentos de l nea de comandos (no usados)
195      */
196     public static void main(String[] args) {
197         System.out.println("=== CALCULADORA CIENT FICA ===\n");
198 
199         // Probar factorial
200         System.out.println("Factorial de 5: " + factorial(5));
201         System.out.println("Factorial de 0: " + factorial(0));
202 
203         // Probar potencia
204         System.out.println("\n2 elevado a 3: " + potencia(2, 3));
205         System.out.println("2 elevado a -3: " + potencia(2, -3));
206 
207         // Probar conversi n de temperatura
208         System.out.println("\n25 C a Fahrenheit: " +
            celsiusAFahrenheit(25));
```

```
209         System.out.println("77 F  a Celsius: " +
                fahrenheitACelsius(77));
210
211         // Probar IMC
212         double peso = 70.5;   // kg
213         double altura = 1.75; // m
214         double imc = calcularIMC(peso, altura);
215
216         System.out.println("\nIMC para " + peso + " kg y " + altura
                + " m:");
217         System.out.printf("IMC: %.2f%n", imc);
218         System.out.println("Clasificaci n: " + clasificarIMC(imc));
219
220         // Probar  reas  y vol menes
221         double radio = 5.0;
222         System.out.println("\nPara radio = " + radio + ":");
223         System.out.println(" rea  del c rculo: " +
                areaCirculo(radio));
224         System.out.println("Volumen de la esfera: " +
                volumenEsfera(radio));
225     }
226 }
```

Listing 9.8: Documentación adecuada del código

9.7. Ejemplo Práctico: Sistema de Gestión de Biblioteca

```
1  import java.util.Scanner;
2
3  /**
4   * Sistema de Gesti n de Biblioteca
5   *
6   * Este sistema permite gestionar libros, usuarios y pr stamos
7   * en una biblioteca. Implementa buenas pr cticas de programaci n
8   * como validaci n, modularidad y manejo de errores.
9   *
10  * Caracter sticas:
11  * - Gesti n de libros (agregar, buscar, actualizar)
12  * - Gesti n de usuarios (registrar, buscar)
13  * - Gesti n de pr stamos (prestar, devolver)
14  * - Reportes y estad sticas
```

```
 *
 * Nota: Usa arrays est ticos para almacenamiento en memoria.
 */
public class SistemaBiblioteca {

    // ========== CONSTANTES ==========

    /** Capacidad m xima de libros en la biblioteca */
    private static final int MAX_LIBROS = 100;

    /** Capacidad m xima de usuarios registrados */
    private static final int MAX_USUARIOS = 50;

    /** Capacidad m xima de pr stamos simult neos */
    private static final int MAX_PRESTAMOS = 200;

    /** Duraci n est ndar de pr stamo en d as */
    private static final int DIAS_PRESTAMO = 14;

    // ========== ESTRUCTURAS DE DATOS ==========

    /** Array de IDs de libros */
    private static int[] idsLibros = new int[MAX_LIBROS];

    /** Array de t tulos de libros */
    private static String[] titulosLibros = new String[MAX_LIBROS];

    /** Array de autores de libros */
    private static String[] autoresLibros = new String[MAX_LIBROS];

    /** Array de disponibilidad de libros (true = disponible) */
    private static boolean[] disponiblesLibros = new
        boolean[MAX_LIBROS];

    /** Contador de libros registrados */
    private static int totalLibros = 0;

    /** Array de IDs de usuarios */
    private static int[] idsUsuarios = new int[MAX_USUARIOS];

    /** Array de nombres de usuarios */
    private static String[] nombresUsuarios = new
        String[MAX_USUARIOS];

    /** Array de correos de usuarios */
```

```
58      private static String[] correosUsuarios = new
           String[MAX_USUARIOS];
59
60      /** Array de tel fonos de usuarios */
61      private static String[] telefonosUsuarios = new
           String[MAX_USUARIOS];
62
63      /** Contador de usuarios registrados */
64      private static int totalUsuarios = 0;
65
66      /** Array de IDs de pr stamos */
67      private static int[] idsPrestamos = new int[MAX_PRESTAMOS];
68
69      /** Array de IDs de libros prestados */
70      private static int[] librosPrestados = new int[MAX_PRESTAMOS];
71
72      /** Array de IDs de usuarios que pidieron pr stamo */
73      private static int[] usuariosPrestamos = new int[MAX_PRESTAMOS];
74
75      /** Array de fechas de pr stamo (formato YYYY-MM-DD) */
76      private static String[] fechasPrestamo = new
           String[MAX_PRESTAMOS];
77
78      /** Array de fechas de devoluci n (formato YYYY-MM-DD) */
79      private static String[] fechasDevolucion = new
           String[MAX_PRESTAMOS];
80
81      /** Array de estado de pr stamos (true = activo) */
82      private static boolean[] activosPrestamos = new
           boolean[MAX_PRESTAMOS];
83
84      /** Contador de pr stamos registrados */
85      private static int totalPrestamos = 0;
86
87      // ========== M TODO PRINCIPAL ==========
88
89      /**
90       * Punto de entrada principal del sistema.
91       *
92       * @param args Argumentos de l nea de comandos (no utilizados)
93       */
94      public static void main(String[] args) {
95          Scanner scanner = new Scanner(System.in);
96
97          // Inicializar con datos de ejemplo
```

```
98          inicializarDatosEjemplo();
99
100         System.out.println("=== SISTEMA DE GESTI N DE BIBLIOTECA
               ===\n");
101         System.out.println("Bienvenido al sistema de gesti n de
               biblioteca");
102         System.out.println("Versi n 1.0 - Programaci n Lineal en
               Java\n");
103
104         boolean sistemaActivo = true;
105
106         while (sistemaActivo) {
107             mostrarMenuPrincipal();
108             int opcion = leerEnteroValidado(scanner, "Seleccione una
                   opci n: ", 1, 6);
109
110             switch (opcion) {
111                 case 1:
112                     gestionarLibros(scanner);
113                     break;
114                 case 2:
115                     gestionarUsuarios(scanner);
116                     break;
117                 case 3:
118                     gestionarPrestamos(scanner);
119                     break;
120                 case 4:
121                     generarReportes();
122                     break;
123                 case 5:
124                     mostrarAyuda();
125                     break;
126                 case 6:
127                     System.out.println("\nGracias por usar el
                           sistema.  Hasta  pronto!");
128                     sistemaActivo = false;
129                     break;
130             }
131
132             if (sistemaActivo) {
133                 System.out.println("\nPresione Enter para
                       continuar...");
134                 scanner.nextLine();
135             }
136         }
```

```
137
138         scanner.close();
139     }
140
141     // ========== INICIALIZACI N ==========
142
143     /**
144      * Inicializa el sistema con datos de ejemplo para pruebas.
145      */
146     private static void inicializarDatosEjemplo() {
147         // Agregar libros de ejemplo
148         agregarLibro(1001, "Cien a os de soledad", "Gabriel Garc a
                M rquez");
149         agregarLibro(1002, "Don Quijote de la Mancha", "Miguel de
                Cervantes");
150         agregarLibro(1003, "1984", "George Orwell");
151         agregarLibro(1004, "El principito", "Antoine de
                Saint-Exup ry");
152         agregarLibro(1005, "Orgullo y prejuicio", "Jane Austen");
153
154         // Marcar algunos como prestados
155         disponiblesLibros[2] = false;  // 1984 est  prestado
156
157         // Agregar usuarios de ejemplo
158         registrarUsuario(2001, "Ana Garc a",
                "ana.garcia@email.com", "555-0101");
159         registrarUsuario(2002, "Carlos M ndez",
                "carlos.mendez@email.com", "555-0102");
160         registrarUsuario(2003, "Laura Fern ndez",
                "laura.fernandez@email.com", "555-0103");
161
162         // Agregar pr stamo de ejemplo
163         registrarPrestamo(3001, 1003, 2002, "2024-01-15",
                "2024-01-29");
164     }
165
166     // ========== MEN S Y NAVEGACI N ==========
167
168     /**
169      * Muestra el men  principal del sistema.
170      */
171     private static void mostrarMenuPrincipal() {
172         System.out.println("\n=== MEN  PRINCIPAL ===");
173         System.out.println("1. Gesti n de Libros");
174         System.out.println("2. Gesti n de Usuarios");
```

```
175         System.out.println("3. Gesti n de Pr stamos");
176         System.out.println("4. Reportes y Estad sticas");
177         System.out.println("5. Ayuda");
178         System.out.println("6. Salir");
179         System.out.println("======================");
180     }
181 
182     /**
183      * Muestra el men  de gesti n de libros.
184      */
185     private static void mostrarMenuLibros() {
186         System.out.println("\n=== GESTI N DE LIBROS ===");
187         System.out.println("1. Mostrar todos los libros");
188         System.out.println("2. Agregar nuevo libro");
189         System.out.println("3. Buscar libro");
190         System.out.println("4. Actualizar libro");
191         System.out.println("5. Eliminar libro");
192         System.out.println("6. Volver al men  principal");
193         System.out.println("============================");
194     }
195 
196     /**
197      * Muestra el men  de gesti n de usuarios.
198      */
199     private static void mostrarMenuUsuarios() {
200         System.out.println("\n=== GESTI N DE USUARIOS ===");
201         System.out.println("1. Mostrar todos los usuarios");
202         System.out.println("2. Registrar nuevo usuario");
203         System.out.println("3. Buscar usuario");
204         System.out.println("4. Actualizar usuario");
205         System.out.println("5. Eliminar usuario");
206         System.out.println("6. Volver al men  principal");
207         System.out.println("============================");
208     }
209 
210     /**
211      * Muestra el men  de gesti n de pr stamos.
212      */
213     private static void mostrarMenuPrestamos() {
214         System.out.println("\n=== GESTI N DE PR STAMOS ===");
215         System.out.println("1. Mostrar pr stamos activos");
216         System.out.println("2. Registrar nuevo pr stamo");
217         System.out.println("3. Registrar devoluci n");
218         System.out.println("4. Buscar pr stamo");
219         System.out.println("5. Ver historial de pr stamos");
```

```
220         System.out.println("6. Volver al men  principal");
221         System.out.println("============================");
222     }
223 
224     /**
225      * Muestra informaci n de ayuda sobre el sistema.
226      */
227     private static void mostrarAyuda() {
228         System.out.println("\n=== AYUDA DEL SISTEMA ===");
229         System.out.println("\nEste sistema permite gestionar una
                biblioteca con las siguientes funcionalidades:");
230         System.out.println("\n1. GESTI N DE LIBROS:");
231         System.out.println("   - Agregar nuevos libros al
                cat logo");
232         System.out.println("   - Buscar libros por t tulo o autor");
233         System.out.println("   - Actualizar informaci n de libros");
234         System.out.println("   - Controlar disponibilidad");
235         System.out.println("\n2. GESTI N DE USUARIOS:");
236         System.out.println("   - Registrar nuevos usuarios");
237         System.out.println("   - Buscar usuarios por nombre");
238         System.out.println("   - Actualizar informaci n de
                contacto");
239         System.out.println("\n3. GESTI N DE PR STAMOS:");
240         System.out.println("   - Registrar nuevos pr stamos");
241         System.out.println("   - Registrar devoluciones");
242         System.out.println("   - Controlar pr stamos activos");
243         System.out.println("   - Ver historial de pr stamos");
244         System.out.println("\n4. REPORTES:");
245         System.out.println("   - Estad sticas de uso");
246         System.out.println("   - Libros m s populares");
247         System.out.println("   - Usuarios m s activos");
248         System.out.println("\nPara navegar, ingresa el n mero
                correspondiente a la opci n deseada.");
249     }
250 
251     // ========== GESTI N DE LIBROS ==========
252 
253     /**
254      * Maneja el submen  de gesti n de libros.
255      *
256      * @param scanner Objeto Scanner para leer entrada
257      */
258     private static void gestionarLibros(Scanner scanner) {
259         boolean enMenuLibros = true;
260 
```

```
261         while (enMenuLibros) {
262             mostrarMenuLibros();
263             int opcion = leerEnteroValidado(scanner, "Seleccione
                    opci n: ", 1, 6);
264 
265             switch (opcion) {
266                 case 1:
267                     mostrarTodosLibros();
268                     break;
269                 case 2:
270                     agregarNuevoLibro(scanner);
271                     break;
272                 case 3:
273                     buscarLibro(scanner);
274                     break;
275                 case 4:
276                     actualizarLibro(scanner);
277                     break;
278                 case 5:
279                     eliminarLibro(scanner);
280                     break;
281                 case 6:
282                     enMenuLibros = false;
283                     break;
284             }
285         }
286     }
287 
288     /**
289      * Muestra todos los libros en el cat logo.
290      */
291     private static void mostrarTodosLibros() {
292         if (totalLibros == 0) {
293             System.out.println("\nNo hay libros registrados en el
                    cat logo.");
294             return;
295         }
296 
297         System.out.println("\n=== CAT LOGO COMPLETO DE LIBROS ===");
298         System.out.printf("%-10s %-30s %-25s %-15s%n",
299                           "ID", "T tulo", "Autor", "Disponible");
300         System.out.println("-----------------------------------------------------
301 
302         for (int i = 0; i < totalLibros; i++) {
303             String disponible = disponiblesLibros[i] ? "S  " : "No";
```

```
304             System.out.printf("%-10d %-30s %-25s %-15s%n",
305                               idsLibros[i],
306                               titulosLibros[i],
307                               autoresLibros[i],
308                               disponible);
309         }
310
311         System.out.println("------------------------------------------------------------
312         System.out.println("Total de libros: " + totalLibros);
313         System.out.println("Libros disponibles: " +
                contarLibrosDisponibles());
314     }
315
316     /**
317      * Agrega un nuevo libro al cat logo.
318      *
319      * @param scanner Objeto Scanner para leer entrada
320      */
321     private static void agregarNuevoLibro(Scanner scanner) {
322         if (totalLibros >= MAX_LIBROS) {
323             System.out.println("\nError: Capacidad m xima de libros
                    alcanzada.");
324             return;
325         }
326
327         System.out.println("\n=== AGREGAR NUEVO LIBRO ===");
328
329         // Leer ID
330         int id = leerEnteroValidado(scanner, "ID del libro: ", 1000,
                9999);
331
332         // Verificar que el ID no exista
333         if (buscarLibroPorId(id) != -1) {
334             System.out.println("Error: Ya existe un libro con ese
                    ID.");
335             return;
336         }
337
338         scanner.nextLine(); // Limpiar buffer
339
340         // Leer t tulo
341         String titulo = leerStringNoVacio(scanner, "T tulo del
                libro: ");
342
343         // Leer autor
```

```
344         String autor = leerStringNoVacio(scanner, "Autor del libro:
                ");
345
346         // Agregar libro
347         agregarLibro(id, titulo, autor);
348
349         System.out.println("\n Libro agregado exitosamente!");
350     }
351
352     /**
353      * Busca libros en el cat logo.
354      *
355      * @param scanner Objeto Scanner para leer entrada
356      */
357     private static void buscarLibro(Scanner scanner) {
358         if (totalLibros == 0) {
359             System.out.println("\nNo hay libros registrados en el
                    cat logo.");
360             return;
361         }
362
363         System.out.println("\n=== BUSCAR LIBRO ===");
364         System.out.println("1. Buscar por t tulo");
365         System.out.println("2. Buscar por autor");
366         System.out.println("3. Buscar por ID");
367
368         int opcion = leerEnteroValidado(scanner, "Seleccione opci n
                de b squeda: ", 1, 3);
369         scanner.nextLine(); // Limpiar buffer
370
371         switch (opcion) {
372             case 1:
373                 System.out.print("Ingrese t tulo o parte del
                        t tulo: ");
374                 String titulo = scanner.nextLine();
375                 buscarLibrosPorTitulo(titulo);
376                 break;
377
378             case 2:
379                 System.out.print("Ingrese autor o parte del autor:
                        ");
380                 String autor = scanner.nextLine();
381                 buscarLibrosPorAutor(autor);
382                 break;
383
```

```
384             case 3:
385                 System.out.print("Ingrese ID del libro: ");
386                 int id = scanner.nextInt();
387                 buscarLibroPorIdYMostrar(id);
388                 break;
389         }
390     }
391 
392     // ========== M TODOS DE VALIDACI N REUTILIZABLES ==========
393 
394     /**
395      * Lee un entero validado dentro de un rango espec fico.
396      *
397      * @param scanner Objeto Scanner para leer entrada
398      * @param mensaje Mensaje a mostrar al usuario
399      * @param min Valor m nimo permitido
400      * @param max Valor m ximo permitido
401      * @return El entero validado
402      */
403     private static int leerEnteroValidado(Scanner scanner, String
        mensaje, int min, int max) {
404         int numero = 0;
405         boolean valido = false;
406 
407         while (!valido) {
408             try {
409                 System.out.print(mensaje);
410                 numero = scanner.nextInt();
411 
412                 if (numero >= min && numero <= max) {
413                     valido = true;
414                 } else {
415                     System.out.printf("Error: Debe ser un n mero
                           entre %d y %d.%n", min, max);
416                 }
417             } catch (Exception e) {
418                 System.out.println("Error: Debe ingresar un n mero
                       entero v lido.");
419                 scanner.nextLine(); // Limpiar buffer
420             }
421         }
422 
423         return numero;
424     }
425 
```

```
426     /**
427      * Lee una cadena que no puede estar vac a.
428      *
429      * @param scanner Objeto Scanner para leer entrada
430      * @param mensaje Mensaje a mostrar al usuario
431      * @return La cadena no vac a
432      */
433     private static String leerStringNoVacio(Scanner scanner, String
            mensaje) {
434         String texto = "";
435         boolean valido = false;
436 
437         while (!valido) {
438             System.out.print(mensaje);
439             texto = scanner.nextLine().trim();
440 
441             if (!texto.isEmpty()) {
442                 valido = true;
443             } else {
444                 System.out.println("Error: Este campo no puede estar
                        vac o.");
445             }
446         }
447 
448         return texto;
449     }
450 
451     // ========== M TODOS DE GESTI N B SICOS ==========
452 
453     /**
454      * Agrega un libro a los arrays.
455      *
456      * @param id ID del libro
457      * @param titulo T tulo del libro
458      * @param autor Autor del libro
459      */
460     private static void agregarLibro(int id, String titulo, String
            autor) {
461         idsLibros[totalLibros] = id;
462         titulosLibros[totalLibros] = titulo;
463         autoresLibros[totalLibros] = autor;
464         disponiblesLibros[totalLibros] = true;
465         totalLibros++;
466     }
467 
```

```
468     /**
469      * Busca un libro por su ID.
470      *
471      * @param id ID del libro a buscar
472      * @return  ndice  del libro en el array, o -1 si no se encuentra
473      */
474     private static int buscarLibroPorId(int id) {
475         for (int i = 0; i < totalLibros; i++) {
476             if (idsLibros[i] == id) {
477                 return i;
478             }
479         }
480         return -1;
481     }
482 
483     /**
484      * Cuenta los libros disponibles en el cat logo.
485      *
486      * @return N mero de libros disponibles
487      */
488     private static int contarLibrosDisponibles() {
489         int contador = 0;
490         for (int i = 0; i < totalLibros; i++) {
491             if (disponiblesLibros[i]) {
492                 contador++;
493             }
494         }
495         return contador;
496     }
497 
498     /**
499      * Busca libros por t tulo o parte del t tulo.
500      *
501      * @param tituloBuscado T tulo o parte del t tulo a buscar
502      */
503     private static void buscarLibrosPorTitulo(String tituloBuscado) {
504         boolean encontrado = false;
505         String tituloLower = tituloBuscado.toLowerCase();
506 
507         System.out.println("\nResultados de b squeda por t tulo:");
508         System.out.println("----------------------------------");
509 
510         for (int i = 0; i < totalLibros; i++) {
511             if
                (titulosLibros[i].toLowerCase().contains(tituloLower))
```

```
                {
512                 mostrarDetalleLibro(i);
513                 encontrado = true;
514             }
515         }
516
517         if (!encontrado) {
518             System.out.println("No se encontraron libros con ese
                    t tulo.");
519         }
520     }
521
522     /**
523      * Muestra el detalle completo de un libro.
524      *
525      * @param indice  ndice  del libro en el array
526      */
527     private static void mostrarDetalleLibro(int indice) {
528         System.out.println("\n--- Detalles del Libro ---");
529         System.out.println("ID: " + idsLibros[indice]);
530         System.out.println("T tulo: " + titulosLibros[indice]);
531         System.out.println("Autor: " + autoresLibros[indice]);
532         System.out.println("Disponible: " +
                (disponiblesLibros[indice] ? "S  " : "No"));
533
534         // Mostrar informaci n de pr stamo si est  prestado
535         if (!disponiblesLibros[indice]) {
536             int indicePrestamo =
                    buscarPrestamoActivoPorLibro(idsLibros[indice]);
537             if (indicePrestamo != -1) {
538                 System.out.println("Prestado a: " +
                        obtenerNombreUsuario(usuariosPrestamos[indicePrestamo]));
539                 System.out.println("Fecha de pr stamo: " +
                        fechasPrestamo[indicePrestamo]);
540                 System.out.println("Fecha de devoluci n estimada: "
                        + fechasDevolucion[indicePrestamo]);
541             }
542         }
543     }
544
545     /**
546      * Busca un pr stamo activo por ID de libro.
547      *
548      * @param idLibro ID del libro a buscar
```

```
549          * @return  ndice  del pr stamo en el array, o -1 si no se
                 encuentra
550          */
551         private static int buscarPrestamoActivoPorLibro(int idLibro) {
552             for (int i = 0; i < totalPrestamos; i++) {
553                 if (activosPrestamos[i] && librosPrestados[i] ==
                        idLibro) {
554                     return i;
555                 }
556             }
557             return -1;
558         }
559 
560         /**
561          * Obtiene el nombre de un usuario por su ID.
562          *
563          * @param idUsuario ID del usuario
564          * @return Nombre del usuario, o "Desconocido" si no se encuentra
565          */
566         private static String obtenerNombreUsuario(int idUsuario) {
567             for (int i = 0; i < totalUsuarios; i++) {
568                 if (idsUsuarios[i] == idUsuario) {
569                     return nombresUsuarios[i];
570                 }
571             }
572             return "Desconocido";
573         }
574 
575         /**
576          * Busca libros por autor o parte del autor.
577          *
578          * @param autorBuscado Autor o parte del autor a buscar
579          */
580         private static void buscarLibrosPorAutor(String autorBuscado) {
581             boolean encontrado = false;
582             String autorLower = autorBuscado.toLowerCase();
583 
584             System.out.println("\nResultados de b squeda por autor:");
585             System.out.println("----------------------------------");
586 
587             for (int i = 0; i < totalLibros; i++) {
588                 if (autoresLibros[i].toLowerCase().contains(autorLower))
                        {
589                     mostrarDetalleLibro(i);
590                     encontrado = true;
```

```
591                 }
592             }
593
594             if (!encontrado) {
595                 System.out.println("No se encontraron libros de ese
                        autor.");
596             }
597         }
598
599         /**
600          * Busca un libro por ID y muestra su detalle.
601          *
602          * @param id ID del libro a buscar
603          */
604         private static void buscarLibroPorIdYMostrar(int id) {
605             int indice = buscarLibroPorId(id);
606
607             if (indice != -1) {
608                 mostrarDetalleLibro(indice);
609             } else {
610                 System.out.println("\nNo se encontr  ning n libro con
                        ID " + id);
611             }
612         }
613
614         /**
615          * Actualiza la informaci n de un libro.
616          *
617          * @param scanner Objeto Scanner para leer entrada
618          */
619         private static void actualizarLibro(Scanner scanner) {
620             if (totalLibros == 0) {
621                 System.out.println("\nNo hay libros registrados en el
                        cat logo.");
622                 return;
623             }
624
625             System.out.println("\n=== ACTUALIZAR LIBRO ===");
626             System.out.print("Ingrese el ID del libro a actualizar: ");
627             int id = scanner.nextInt();
628
629             int indice = buscarLibroPorId(id);
630
631             if (indice == -1) {
```

```
632             System.out.println("Error: No se encontr  ning n libro
                    con ID " + id);
633             return;
634         }
635
636         scanner.nextLine(); // Limpiar buffer
637
638         System.out.println("\nLibro actual:");
639         mostrarDetalleLibro(indice);
640
641         System.out.println("\n  Qu   desea actualizar?");
642         System.out.println("1. T tulo");
643         System.out.println("2. Autor");
644         System.out.println("3. Estado de disponibilidad");
645         System.out.println("4. Cancelar");
646
647         int opcion = leerEnteroValidado(scanner, "Seleccione
                opci n: ", 1, 4);
648
649         switch (opcion) {
650             case 1:
651                 System.out.print("Nuevo t tulo: ");
652                 String nuevoTitulo = leerStringNoVacio(scanner, "");
653                 titulosLibros[indice] = nuevoTitulo;
654                 System.out.println("T tulo actualizado
                        exitosamente.");
655                 break;
656
657             case 2:
658                 System.out.print("Nuevo autor: ");
659                 String nuevoAutor = leerStringNoVacio(scanner, "");
660                 autoresLibros[indice] = nuevoAutor;
661                 System.out.println("Autor actualizado
                        exitosamente.");
662                 break;
663
664             case 3:
665                 System.out.print("  Est   disponible? (s/n): ");
666                 String respuesta = scanner.nextLine().toLowerCase();
667                 disponiblesLibros[indice] = respuesta.equals("s");
668                 System.out.println("Estado de disponibilidad
                        actualizado.");
669                 break;
670
671             case 4:
```

```
672                 System.out.println("Actualizaci n cancelada.");
673                 break;
674             }
675         }
676 
677         /**
678          * Elimina un libro del cat logo.
679          *
680          * @param scanner Objeto Scanner para leer entrada
681          */
682         private static void eliminarLibro(Scanner scanner) {
683             if (totalLibros == 0) {
684                 System.out.println("\nNo hay libros registrados en el
                        cat logo.");
685                 return;
686             }
687 
688             System.out.println("\n=== ELIMINAR LIBRO ===");
689             System.out.print("Ingrese el ID del libro a eliminar: ");
690             int id = scanner.nextInt();
691 
692             int indice = buscarLibroPorId(id);
693 
694             if (indice == -1) {
695                 System.out.println("Error: No se encontr  ning n libro
                        con ID " + id);
696                 return;
697             }
698 
699             // Verificar si el libro est  prestado
700             if (!disponiblesLibros[indice]) {
701                 System.out.println("Error: No se puede eliminar un libro
                        que est  prestado.");
702                 return;
703             }
704 
705             // Confirmar eliminaci n
706             scanner.nextLine(); // Limpiar buffer
707             System.out.print("  Est   seguro de eliminar el libro '" +
                    titulosLibros[indice] + "'? (s/n): ");
708             String confirmacion = scanner.nextLine().toLowerCase();
709 
710             if (confirmacion.equals("s")) {
711                 // Eliminar libro moviendo los elementos posteriores
712                 for (int i = indice; i < totalLibros - 1; i++) {
```

```
713                 idsLibros[i] = idsLibros[i + 1];
714                 titulosLibros[i] = titulosLibros[i + 1];
715                 autoresLibros[i] = autoresLibros[i + 1];
716                 disponiblesLibros[i] = disponiblesLibros[i + 1];
717             }
718 
719             totalLibros--;
720             System.out.println("Libro eliminado exitosamente.");
721         } else {
722             System.out.println("Eliminaci n cancelada.");
723         }
724     }
725 
726     // ========== GESTI N DE USUARIOS ==========
727 
728     /**
729      * Maneja el submen  de gesti n de usuarios.
730      *
731      * @param scanner Objeto Scanner para leer entrada
732      */
733     private static void gestionarUsuarios(Scanner scanner) {
734         boolean enMenuUsuarios = true;
735 
736         while (enMenuUsuarios) {
737             mostrarMenuUsuarios();
738             int opcion = leerEnteroValidado(scanner, "Seleccione
                    opci n: ", 1, 6);
739 
740             switch (opcion) {
741                 case 1:
742                     mostrarTodosUsuarios();
743                     break;
744                 case 2:
745                     registrarNuevoUsuario(scanner);
746                     break;
747                 case 3:
748                     buscarUsuario(scanner);
749                     break;
750                 case 4:
751                     actualizarUsuario(scanner);
752                     break;
753                 case 5:
754                     eliminarUsuario(scanner);
755                     break;
756                 case 6:
```

```
                    enMenuUsuarios = false;
                    break;
                }
            }
        }

        /**
         * Muestra todos los usuarios registrados.
         */
        private static void mostrarTodosUsuarios() {
            if (totalUsuarios == 0) {
                System.out.println("\nNo hay usuarios registrados.");
                return;
            }

            System.out.println("\n=== LISTA DE USUARIOS ===");
            System.out.printf("%-10s %-25s %-30s %-15s%n",
                              "ID", "Nombre", "Correo Electr nico",
                                 "Tel fono");
            System.out.println("-----------------------------------------------------------

            for (int i = 0; i < totalUsuarios; i++) {
                System.out.printf("%-10d %-25s %-30s %-15s%n",
                                  idsUsuarios[i],
                                  nombresUsuarios[i],
                                  correosUsuarios[i],
                                  telefonosUsuarios[i]);
            }

            System.out.println("-----------------------------------------------------------
            System.out.println("Total de usuarios: " + totalUsuarios);
        }

        /**
         * Registra un nuevo usuario en el sistema.
         *
         * @param scanner Objeto Scanner para leer entrada
         */
        private static void registrarNuevoUsuario(Scanner scanner) {
            if (totalUsuarios >= MAX_USUARIOS) {
                System.out.println("\nError: Capacidad m xima de
                   usuarios alcanzada.");
                return;
            }

```

```
800         System.out.println("\n=== REGISTRAR NUEVO USUARIO ===");
801
802         // Leer ID
803         int id = leerEnteroValidado(scanner, "ID del usuario: ",
                2000, 2999);
804
805         // Verificar que el ID no exista
806         if (buscarUsuarioPorId(id) != -1) {
807             System.out.println("Error: Ya existe un usuario con ese
                    ID.");
808             return;
809         }
810
811         scanner.nextLine(); // Limpiar buffer
812
813         // Leer nombre
814         String nombre = leerStringNoVacio(scanner, "Nombre completo:
                ");
815
816         // Leer correo
817         String correo = leerStringNoVacio(scanner, "Correo
                electr nico: ");
818
819         // Validar formato b sico de correo
820         if (!correo.contains("@")) {
821             System.out.println("Advertencia: El correo no parece
                    tener un formato v lido.");
822         }
823
824         // Leer tel fono
825         System.out.print("Tel fono (opcional, presione Enter para
                omitir): ");
826         String telefono = scanner.nextLine();
827
828         // Si est  vac o, asignar valor por defecto
829         if (telefono.trim().isEmpty()) {
830             telefono = "No proporcionado";
831         }
832
833         // Registrar usuario
834         registrarUsuario(id, nombre, correo, telefono);
835
836         System.out.println("\n Usuario registrado exitosamente!");
837     }
838
```

```
839     /**
840      * Registra un usuario en los arrays.
841      *
842      * @param id ID del usuario
843      * @param nombre Nombre del usuario
844      * @param correo Correo del usuario
845      * @param telefono Tel fono del usuario
846      */
847     private static void registrarUsuario(int id, String nombre,
        String correo, String telefono) {
848         idsUsuarios[totalUsuarios] = id;
849         nombresUsuarios[totalUsuarios] = nombre;
850         correosUsuarios[totalUsuarios] = correo;
851         telefonosUsuarios[totalUsuarios] = telefono;
852         totalUsuarios++;
853     }
854 
855     /**
856      * Busca un usuario por su ID.
857      *
858      * @param id ID del usuario a buscar
859      * @return  ndice  del usuario en el array, o -1 si no se
            encuentra
860      */
861     private static int buscarUsuarioPorId(int id) {
862         for (int i = 0; i < totalUsuarios; i++) {
863             if (idsUsuarios[i] == id) {
864                 return i;
865             }
866         }
867         return -1;
868     }
869 
870     // ========== GESTI N DE PR STAMOS ==========
871 
872     /**
873      * Maneja el submen  de gesti n de pr stamos.
874      *
875      * @param scanner Objeto Scanner para leer entrada
876      */
877     private static void gestionarPrestamos(Scanner scanner) {
878         boolean enMenuPrestamos = true;
879 
880         while (enMenuPrestamos) {
881             mostrarMenuPrestamos();
```

```
882             int opcion = leerEnteroValidado(scanner, "Seleccione
                    opci n: ", 1, 6);
883
884             switch (opcion) {
885                 case 1:
886                     mostrarPrestamosActivos();
887                     break;
888                 case 2:
889                     registrarNuevoPrestamo(scanner);
890                     break;
891                 case 3:
892                     registrarDevolucion(scanner);
893                     break;
894                 case 4:
895                     buscarPrestamo(scanner);
896                     break;
897                 case 5:
898                     mostrarHistorialPrestamos();
899                     break;
900                 case 6:
901                     enMenuPrestamos = false;
902                     break;
903             }
904         }
905     }
906
907     /**
908      * Muestra todos los pr stamos activos.
909      */
910     private static void mostrarPrestamosActivos() {
911         if (totalPrestamos == 0) {
912             System.out.println("\nNo hay pr stamos registrados.");
913             return;
914         }
915
916         System.out.println("\n=== PR STAMOS ACTIVOS ===");
917         System.out.printf("%-10s %-25s %-30s %-15s %-15s%n",
918                           "ID", "Libro", "Usuario", "Fecha
                                  Pr stamo", "Fecha Devoluci n");
919         System.out.println("-----------------------------------------------------------
920
921         int activos = 0;
922         for (int i = 0; i < totalPrestamos; i++) {
923             if (activosPrestamos[i]) {
```

```
924                 String nombreLibro =
                        obtenerTituloLibro(librosPrestados[i]);
925                 String nombreUsuario =
                        obtenerNombreUsuario(usuariosPrestamos[i]);
926
927                 System.out.printf("%-10d %-25s %-30s %-15s %-15s%n",
928                                   idsPrestamos[i],
929                                   nombreLibro,
930                                   nombreUsuario,
931                                   fechasPrestamo[i],
932                                   fechasDevolucion[i]);
933                 activos++;
934             }
935         }
936
937         System.out.println("-----------------------------------------------------------
938         System.out.println("Total de pr stamos activos: " +
                activos);
939     }
940
941     /**
942      * Obtiene el t tulo de un libro por su ID.
943      *
944      * @param idLibro ID del libro
945      * @return T tulo del libro, o "Desconocido" si no se encuentra
946      */
947     private static String obtenerTituloLibro(int idLibro) {
948         for (int i = 0; i < totalLibros; i++) {
949             if (idsLibros[i] == idLibro) {
950                 return titulosLibros[i];
951             }
952         }
953         return "Desconocido";
954     }
955
956     /**
957      * Registra un nuevo pr stamo.
958      *
959      * @param scanner Objeto Scanner para leer entrada
960      */
961     private static void registrarNuevoPrestamo(Scanner scanner) {
962         if (totalPrestamos >= MAX_PRESTAMOS) {
963             System.out.println("\nError: Capacidad m xima de
                    pr stamos alcanzada.");
964             return;
```

```
965         }
966
967         System.out.println("\n=== REGISTRAR NUEVO PR STAMO ===");
968
969         // Verificar que haya libros disponibles
970         if (contarLibrosDisponibles() == 0) {
971             System.out.println("Error: No hay libros disponibles
                    para pr stamo.");
972             return;
973         }
974
975         // Verificar que haya usuarios registrados
976         if (totalUsuarios == 0) {
977             System.out.println("Error: No hay usuarios
                    registrados.");
978             return;
979         }
980
981         // Leer ID de pr stamo
982         int idPrestamo = leerEnteroValidado(scanner, "ID del
                pr stamo: ", 3000, 3999);
983
984         // Verificar que el ID no exista
985         if (buscarPrestamoPorId(idPrestamo) != -1) {
986             System.out.println("Error: Ya existe un pr stamo con
                    ese ID.");
987             return;
988         }
989
990         // Mostrar libros disponibles
991         System.out.println("\nLibros disponibles:");
992         for (int i = 0; i < totalLibros; i++) {
993             if (disponiblesLibros[i]) {
994                 System.out.println(idsLibros[i] + " - " +
                        titulosLibros[i] + " (" + autoresLibros[i] +
                        ")");
995             }
996         }
997
998         // Leer ID del libro
999         System.out.print("\nID del libro a prestar: ");
1000        int idLibro = scanner.nextInt();
1001
1002        int indiceLibro = buscarLibroPorId(idLibro);
1003
```

```
1004            if (indiceLibro == -1) {
1005                System.out.println("Error: No se encontr  ning n libro
                        con ID " + idLibro);
1006                return;
1007            }
1008
1009            if (!disponiblesLibros[indiceLibro]) {
1010                System.out.println("Error: El libro no est
                        disponible.");
1011                return;
1012            }
1013
1014            // Mostrar usuarios registrados
1015            System.out.println("\nUsuarios registrados:");
1016            for (int i = 0; i < totalUsuarios; i++) {
1017                System.out.println(idsUsuarios[i] + " - " +
                        nombresUsuarios[i]);
1018            }
1019
1020            // Leer ID del usuario
1021            System.out.print("\nID del usuario: ");
1022            int idUsuario = scanner.nextInt();
1023
1024            if (buscarUsuarioPorId(idUsuario) == -1) {
1025                System.out.println("Error: No se encontr  ning n
                        usuario con ID " + idUsuario);
1026                return;
1027            }
1028
1029            scanner.nextLine(); // Limpiar buffer
1030
1031            // Leer fecha de pr stamo (hoy por defecto)
1032            System.out.print("Fecha de pr stamo (YYYY-MM-DD, Enter para
                    hoy): ");
1033            String fechaPrestamo = scanner.nextLine();
1034
1035            if (fechaPrestamo.trim().isEmpty()) {
1036                fechaPrestamo = java.time.LocalDate.now().toString();
1037            }
1038
1039            // Calcular fecha de devoluci n (14 d as despu s)
1040            String fechaDevolucion =
                    calcularFechaDevolucion(fechaPrestamo);
1041
1042            // Registrar pr stamo
```

```
1043            registrarPrestamo(idPrestamo, idLibro, idUsuario,
                    fechaPrestamo, fechaDevolucion);
1044
1045            // Actualizar disponibilidad del libro
1046            disponiblesLibros[indiceLibro] = false;
1047
1048            System.out.println("\n Pr stamo registrado exitosamente!");
1049            System.out.println("Fecha de devoluci n estimada: " +
                    fechaDevolucion);
1050        }
1051
1052        /**
1053         * Calcula la fecha de devoluci n sumando d as a una fecha.
1054         *
1055         * @param fechaPrestamo Fecha de pr stamo en formato YYYY-MM-DD
1056         * @return Fecha de devoluci n en formato YYYY-MM-DD
1057         */
1058        private static String calcularFechaDevolucion(String
                fechaPrestamo) {
1059            try {
1060                java.time.LocalDate fecha =
                        java.time.LocalDate.parse(fechaPrestamo);
1061                java.time.LocalDate fechaDevolucion =
                        fecha.plusDays(DIAS_PRESTAMO);
1062                return fechaDevolucion.toString();
1063            } catch (Exception e) {
1064                // Si hay error en el formato, devolver fecha por defecto
1065                return
                        java.time.LocalDate.now().plusDays(DIAS_PRESTAMO).toString();
1066            }
1067        }
1068
1069        /**
1070         * Registra un pr stamo en los arrays.
1071         *
1072         * @param idPrestamo ID del pr stamo
1073         * @param idLibro ID del libro prestado
1074         * @param idUsuario ID del usuario que pide el pr stamo
1075         * @param fechaPrestamo Fecha de pr stamo
1076         * @param fechaDevolucion Fecha estimada de devoluci n
1077         */
1078        private static void registrarPrestamo(int idPrestamo, int
                idLibro, int idUsuario,
1079                                              String fechaPrestamo,
                                                      String fechaDevolucion)
```

```
                                                {
1080            idsPrestamos[totalPrestamos] = idPrestamo;
1081            librosPrestados[totalPrestamos] = idLibro;
1082            usuariosPrestamos[totalPrestamos] = idUsuario;
1083            fechasPrestamo[totalPrestamos] = fechaPrestamo;
1084            fechasDevolucion[totalPrestamos] = fechaDevolucion;
1085            activosPrestamos[totalPrestamos] = true;
1086            totalPrestamos++;
1087        }
1088
1089        /**
1090         * Busca un pr stamo por su ID.
1091         *
1092         * @param id ID del pr stamo a buscar
1093         * @return  ndice  del pr stamo en el array, o -1 si no se
                encuentra
1094         */
1095        private static int buscarPrestamoPorId(int id) {
1096            for (int i = 0; i < totalPrestamos; i++) {
1097                if (idsPrestamos[i] == id) {
1098                    return i;
1099                }
1100            }
1101            return -1;
1102        }
1103
1104        // ========== GENERACI N DE REPORTES ==========
1105
1106        /**
1107         * Genera reportes y estad sticas del sistema.
1108         */
1109        private static void generarReportes() {
1110            System.out.println("\n=== REPORTES Y ESTAD STICAS ===\n");
1111
1112            // Reporte 1: Resumen general
1113            System.out.println("1. RESUMEN GENERAL:");
1114            System.out.println("   Libros registrados: " + totalLibros);
1115            System.out.println("   Libros disponibles: " +
                   contarLibrosDisponibles());
1116            System.out.println("   Libros prestados: " + (totalLibros -
                   contarLibrosDisponibles()));
1117            System.out.println("   Usuarios registrados: " +
                   totalUsuarios);
1118            System.out.println("   Pr stamos registrados: " +
                   totalPrestamos);
```

```
1119        System.out.println("   Pr stamos activos: " +
                contarPrestamosActivos());
1120
1121        // Reporte 2: Libros m s populares (los m s prestados)
1122        System.out.println("\n2. LIBROS M S PRESTADOS:");
1123
1124        if (totalPrestamos == 0) {
1125            System.out.println("   No hay pr stamos registrados.");
1126        } else {
1127            // Contar pr stamos por libro
1128            int[] conteoPorLibro = new int[totalLibros];
1129
1130            for (int i = 0; i < totalPrestamos; i++) {
1131                int indiceLibro =
                        buscarLibroPorId(librosPrestados[i]);
1132                if (indiceLibro != -1) {
1133                    conteoPorLibro[indiceLibro]++;
1134                }
1135            }
1136
1137            // Encontrar los 3 libros m s prestados
1138            for (int top = 1; top <= 3 && top <= totalLibros; top++)
                    {
1139                int maxIndex = -1;
1140                int maxCount = -1;
1141
1142                for (int i = 0; i < totalLibros; i++) {
1143                    if (conteoPorLibro[i] > maxCount) {
1144                        maxCount = conteoPorLibro[i];
1145                        maxIndex = i;
1146                    }
1147                }
1148
1149                if (maxIndex != -1 && maxCount > 0) {
1150                    System.out.printf("   %d. %s (%s) - %d
                            pr stamos%n",
1151                                      top,
1152                                      titulosLibros[maxIndex],
1153                                      autoresLibros[maxIndex],
1154                                      maxCount);
1155                    conteoPorLibro[maxIndex] = -1; // Marcar como ya
                            mostrado
1156                }
1157            }
1158        }
```

```
1159
1160        // Reporte 3: Usuarios m s activos
1161        System.out.println("\n3. USUARIOS M S ACTIVOS:");
1162
1163        if (totalPrestamos == 0) {
1164            System.out.println("   No hay pr stamos registrados.");
1165        } else {
1166            // Contar pr stamos por usuario
1167            int[] conteoPorUsuario = new int[totalUsuarios];
1168
1169            for (int i = 0; i < totalPrestamos; i++) {
1170                int indiceUsuario =
                        buscarUsuarioPorId(usuariosPrestamos[i]);
1171                if (indiceUsuario != -1) {
1172                    conteoPorUsuario[indiceUsuario]++;
1173                }
1174            }
1175
1176            // Encontrar los 3 usuarios m s activos
1177            for (int top = 1; top <= 3 && top <= totalUsuarios;
                    top++) {
1178                int maxIndex = -1;
1179                int maxCount = -1;
1180
1181                for (int i = 0; i < totalUsuarios; i++) {
1182                    if (conteoPorUsuario[i] > maxCount) {
1183                        maxCount = conteoPorUsuario[i];
1184                        maxIndex = i;
1185                    }
1186                }
1187
1188                if (maxIndex != -1 && maxCount > 0) {
1189                    System.out.printf("   %d. %s - %d pr stamos%n",
1190                                      top,
1191                                      nombresUsuarios[maxIndex],
1192                                      maxCount);
1193                    conteoPorUsuario[maxIndex] = -1; // Marcar como
                            ya mostrado
1194                }
1195            }
1196        }
1197
1198        // Reporte 4: Pr stamos vencidos o por vencer
1199        System.out.println("\n4. PR STAMOS POR VENCER (en los
            pr ximos 3 d as):");
```

```
1200
1201        String hoy = java.time.LocalDate.now().toString();
1202        int prestamosPorVencer = 0;
1203
1204        for (int i = 0; i < totalPrestamos; i++) {
1205            if (activosPrestamos[i]) {
1206                try {
1207                    java.time.LocalDate fechaDevolucion =
                           java.time.LocalDate.parse(fechasDevolucion[i]);
1208                    java.time.LocalDate fechaHoy =
                           java.time.LocalDate.parse(hoy);
1209
1210                    long diasRestantes =
                           java.time.temporal.ChronoUnit.DAYS.between(fechaHoy,
                           fechaDevolucion);
1211
1212                    if (diasRestantes >= 0 && diasRestantes <= 3) {
1213                        String nombreLibro =
                               obtenerTituloLibro(librosPrestados[i]);
1214                        String nombreUsuario =
                               obtenerNombreUsuario(usuariosPrestamos[i]);
1215
1216                        System.out.printf("   - %s prestado a %s
                               (vence en %d d as)%n",
1217                                          nombreLibro, nombreUsuario,
                                              diasRestantes);
1218                        prestamosPorVencer++;
1219                    }
1220                } catch (Exception e) {
1221                    // Ignorar fechas con formato incorrecto
1222                }
1223            }
1224        }
1225
1226        if (prestamosPorVencer == 0) {
1227            System.out.println("   No hay pr stamos por vencer en
                   los pr ximos 3 d as.");
1228        }
1229
1230        // Reporte 5: Tasa de utilizaci n
1231        System.out.println("\n5. TASA DE UTILIZACI N:");
1232
1233        if (totalPrestamos > 0) {
1234            double tasaUtilizacion = (double)
                   contarPrestamosActivos() / totalLibros * 100;
```

```
1235            System.out.printf("   Tasa de utilizaci n actual:
                    %.1f%%%n", tasaUtilizacion);
1236
1237            if (tasaUtilizacion > 80) {
1238                System.out.println("    Alta  demanda! Considera
                        adquirir m s libros.");
1239            } else if (tasaUtilizacion < 20) {
1240                System.out.println("   Baja utilizaci n. Considera
                        promover los libros disponibles.");
1241            }
1242        }
1243    }
1244
1245    /**
1246     * Cuenta los pr stamos activos.
1247     *
1248     * @return N mero de pr stamos activos
1249     */
1250    private static int contarPrestamosActivos() {
1251        int contador = 0;
1252        for (int i = 0; i < totalPrestamos; i++) {
1253            if (activosPrestamos[i]) {
1254                contador++;
1255            }
1256        }
1257        return contador;
1258    }
1259
1260    /**
1261     * Registra la devoluci n de un pr stamo.
1262     *
1263     * @param scanner Objeto Scanner para leer entrada
1264     */
1265    private static void registrarDevolucion(Scanner scanner) {
1266        System.out.println("\n=== REGISTRAR DEVOLUCI N ===");
1267
1268        if (contarPrestamosActivos() == 0) {
1269            System.out.println("No hay pr stamos activos para
                    devolver.");
1270            return;
1271        }
1272
1273        // Mostrar pr stamos activos
1274        mostrarPrestamosActivos();
1275
```

```
1276            System.out.print("\nID del pr stamo a devolver: ");
1277            int idPrestamo = scanner.nextInt();
1278
1279            int indicePrestamo = buscarPrestamoPorId(idPrestamo);
1280
1281            if (indicePrestamo == -1) {
1282                System.out.println("Error: No se encontr  ning n
                       pr stamo con ID " + idPrestamo);
1283                return;
1284            }
1285
1286            if (!activosPrestamos[indicePrestamo]) {
1287                System.out.println("Error: Este pr stamo ya est
                       devuelto.");
1288                return;
1289            }
1290
1291            // Marcar pr stamo como devuelto
1292            activosPrestamos[indicePrestamo] = false;
1293
1294            // Marcar libro como disponible
1295            int idLibro = librosPrestados[indicePrestamo];
1296            int indiceLibro = buscarLibroPorId(idLibro);
1297
1298            if (indiceLibro != -1) {
1299                disponiblesLibros[indiceLibro] = true;
1300            }
1301
1302            System.out.println("\n Devoluci n registrada
                   exitosamente!");
1303            System.out.println("Libro '" + obtenerTituloLibro(idLibro) +
                   "' ahora est  disponible.");
1304        }
1305
1306        /**
1307         * Busca un pr stamo en el sistema.
1308         *
1309         * @param scanner Objeto Scanner para leer entrada
1310         */
1311        private static void buscarPrestamo(Scanner scanner) {
1312            System.out.println("\n=== BUSCAR PR STAMO ===");
1313
1314            if (totalPrestamos == 0) {
1315                System.out.println("No hay pr stamos registrados.");
1316                return;
```

```
1317         }
1318
1319         System.out.println("1. Buscar por ID de pr stamo");
1320         System.out.println("2. Buscar por ID de libro");
1321         System.out.println("3. Buscar por ID de usuario");
1322
1323         int opcion = leerEnteroValidado(scanner, "Seleccione
                 opci n: ", 1, 3);
1324
1325         switch (opcion) {
1326             case 1:
1327                 System.out.print("ID del pr stamo: ");
1328                 int idPrestamo = scanner.nextInt();
1329                 buscarPrestamoPorIdYMostrar(idPrestamo);
1330                 break;
1331
1332             case 2:
1333                 System.out.print("ID del libro: ");
1334                 int idLibro = scanner.nextInt();
1335                 buscarPrestamosPorLibro(idLibro);
1336                 break;
1337
1338             case 3:
1339                 System.out.print("ID del usuario: ");
1340                 int idUsuario = scanner.nextInt();
1341                 buscarPrestamosPorUsuario(idUsuario);
1342                 break;
1343         }
1344     }
1345
1346     /**
1347      * Busca un pr stamo por ID y muestra su detalle.
1348      *
1349      * @param id ID del pr stamo a buscar
1350      */
1351     private static void buscarPrestamoPorIdYMostrar(int id) {
1352         int indice = buscarPrestamoPorId(id);
1353
1354         if (indice != -1) {
1355             mostrarDetallePrestamo(indice);
1356         } else {
1357             System.out.println("\nNo se encontr  ning n pr stamo
                     con ID " + id);
1358         }
1359     }
```

```
1360
1361      /**
1362       * Muestra el detalle completo de un pr stamo.
1363       *
1364       * @param indice  ndice  del pr stamo en el array
1365       */
1366      private static void mostrarDetallePrestamo(int indice) {
1367          System.out.println("\n--- Detalles del Pr stamo ---");
1368          System.out.println("ID del pr stamo: " +
                 idsPrestamos[indice]);
1369          System.out.println("Libro: " +
                 obtenerTituloLibro(librosPrestados[indice]) +
1370                             " (ID: " + librosPrestados[indice] + ")");
1371          System.out.println("Usuario: " +
                 obtenerNombreUsuario(usuariosPrestamos[indice]) +
1372                             " (ID: " + usuariosPrestamos[indice] +
                                    ")");
1373          System.out.println("Fecha de pr stamo: " +
                 fechasPrestamo[indice]);
1374          System.out.println("Fecha de devoluci n estimada: " +
                 fechasDevolucion[indice]);
1375          System.out.println("Estado: " + (activosPrestamos[indice] ?
                 "Activo" : "Devuelto"));
1376      }
1377
1378      /**
1379       * Muestra el historial completo de pr stamos.
1380       */
1381      private static void mostrarHistorialPrestamos() {
1382          if (totalPrestamos == 0) {
1383              System.out.println("\nNo hay pr stamos registrados.");
1384              return;
1385          }
1386
1387          System.out.println("\n=== HISTORIAL COMPLETO DE PR STAMOS
                 ===");
1388          System.out.printf("%-10s %-25s %-25s %-15s %-15s %-10s%n",
1389                            "ID", "Libro", "Usuario", "Fecha
                                    Pr stamo", "Fecha Devoluci n",
                                    "Estado");
1390          System.out.println("-----------------------------------------------------------
1391
1392          for (int i = 0; i < totalPrestamos; i++) {
1393              String nombreLibro =
                     obtenerTituloLibro(librosPrestados[i]);
```

```
1394                String nombreUsuario =
                        obtenerNombreUsuario(usuariosPrestamos[i]);
1395                String estado = activosPrestamos[i] ? "Activo" :
                        "Devuelto";
1396
1397                System.out.printf("%-10d %-25s %-25s %-15s %-15s
                        %-10s%n",
1398                                  idsPrestamos[i],
1399                                  nombreLibro,
1400                                  nombreUsuario,
1401                                  fechasPrestamo[i],
1402                                  fechasDevolucion[i],
1403                                  estado);
1404            }
1405
1406            System.out.println("----------------------------------------------------------
1407            System.out.println("Total de pr stamos en historial: " +
                    totalPrestamos);
1408        }
1409    }
```

Listing 9.9: Sistema completo con buenas prácticas

Ejercicio Propuesto

Ejercicio 9.1: Refactorización de Código

Toma uno de tus programas anteriores y:

1. Renombra todas las variables para que sean más descriptivas
2. Divide métodos largos en métodos más pequeños
3. Añade comentarios Javadoc a todos los métodos públicos
4. Mejora el formato y la indentación
5. Añada validación de entrada donde sea necesario
6. Maneja casos especiales y errores

Ejercicio Propuesto

Ejercicio 9.2: Análisis de Complejidad

Analiza la complejidad temporal de los métodos en tu sistema de biblioteca:

1. Identifica qué métodos tienen complejidad O(n), O(n^2), etc.
2. Propone mejoras para métodos que puedan ser optimizados
3. Implementa al menos una optimización significativa
4. Mide el tiempo antes y después de la optimización

Ejercicio Propuesto

Ejercicio 9.3: Sistema de Logs

Añade un sistema de logs al sistema de biblioteca:

1. Registra todas las operaciones importantes (agregar, modificar, eliminar)
2. Guarda la fecha, hora, tipo de operación y detalles
3. Permite ver los logs por fecha o tipo de operación
4. Implementa diferentes niveles de log (INFO, WARNING, ERROR)
5. Los logs deben almacenarse en un array con tamaño máximo

9.8. Resumen del Capítulo

- Los nombres significativos hacen el código más legible y mantenible
- El formato consistente mejora la claridad del código
- El principio de responsabilidad única hace los métodos más reutilizables
- Alta cohesión y bajo acoplamiento mejoran la modularidad
- La validación de entrada previene errores y mejora la robustez
- El manejo de casos especiales hace el código más resiliente
- Evitar cálculos redundantes mejora el rendimiento
- Los algoritmos eficientes (como búsqueda binaria) son cruciales para grandes conjuntos de datos
- La documentación adecuada (comentarios Javadoc) ayuda a otros a entender tu código

- Las buenas prácticas no son solo reglas, sino hábitos que mejoran con la práctica

Capítulo 10

Proyectos Prácticos por Consola

10.1. Introducción a los Proyectos

En este capítulo final, aplicaremos todo lo aprendido en proyectos completos y realistas. Cada proyecto está diseñado para consolidar los conceptos y demostrar cómo se combinan en aplicaciones útiles.

Nota Importante

Estos proyectos son más complejos que los ejercicios anteriores. Tómate tu tiempo para entender cada parte y no dudes en experimentar y modificar el código.

10.2. Proyecto 1: Sistema de Gestión de Tareas

Un sistema completo para gestionar tareas personales con prioridades, fechas y categorías.

```
import java.util.Scanner;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.time.format.DateTimeParseException;

/**
 * Sistema de Gesti n de Tareas Personales
 *
 * Caracter sticas:
 * - Crear, editar, eliminar y listar tareas
 * - Asignar prioridades (Alta, Media, Baja)
 * - Asignar categor as
 * - Establecer fechas de vencimiento
 * - Marcar tareas como completadas
 * - Buscar y filtrar tareas
 * - Recordatorios de tareas pr ximas a vencer
 * - Estad sticas de productividad
 */
```

```
19 public class SistemaTareas {
20
21     // ========== CONSTANTES ==========
22     private static final int MAX_TAREAS = 100;
23     private static final String[] PRIORIDADES = {"Alta", "Media",
           "Baja"};
24     private static final String[] CATEGORIAS = {"Trabajo",
           "Estudio", "Personal", "Hogar", "Otros"};
25     private static final DateTimeFormatter FORMATO_FECHA =
           DateTimeFormatter.ofPattern("dd/MM/yyyy");
26
27     // ========== ESTRUCTURAS DE DATOS ==========
28     private static int[] ids = new int[MAX_TAREAS];
29     private static String[] descripciones = new String[MAX_TAREAS];
30     private static String[] prioridades = new String[MAX_TAREAS];
31     private static String[] categorias = new String[MAX_TAREAS];
32     private static String[] fechasVencimiento = new
           String[MAX_TAREAS];
33     private static boolean[] completadas = new boolean[MAX_TAREAS];
34     private static String[] fechasCompletacion = new
           String[MAX_TAREAS];
35     private static int totalTareas = 0;
36
37     // ========== M TODO PRINCIPAL ==========
38     public static void main(String[] args) {
39         Scanner scanner = new Scanner(System.in);
40
41         // Inicializar con tareas de ejemplo
42         inicializarTareasEjemplo();
43
44         System.out.println("=== SISTEMA DE GESTI N DE TAREAS
               PERSONALES ===\n");
45         System.out.println("  Bienvenido  ! Organiza tus tareas de
               manera eficiente.\n");
46
47         boolean sistemaActivo = true;
48
49         while (sistemaActivo) {
50             mostrarMenuPrincipal();
51             int opcion = leerEnteroValidado(scanner, "Seleccione una
                   opci n: ", 1, 9);
52
53             switch (opcion) {
54                 case 1:
55                     mostrarTodasTareas();
```

```
56                      break;
57                  case 2:
58                      agregarNuevaTarea(scanner);
59                      break;
60                  case 3:
61                      editarTarea(scanner);
62                      break;
63                  case 4:
64                      eliminarTarea(scanner);
65                      break;
66                  case 5:
67                      gestionarCompletacion(scanner);
68                      break;
69                  case 6:
70                      buscarYFiltrarTareas(scanner);
71                      break;
72                  case 7:
73                      mostrarRecordatorios();
74                      break;
75                  case 8:
76                      mostrarEstadisticas();
77                      break;
78                  case 9:
79                      System.out.println("\n Gracias por usar el
                            sistema! Hasta pronto!");
80                      sistemaActivo = false;
81                      break;
82              }
83
84              if (sistemaActivo) {
85                  System.out.println("\nPresione Enter para
                        continuar...");
86                  scanner.nextLine();
87              }
88          }
89
90          scanner.close();
91      }
92
93      // ========== INICIALIZACI N ==========
94      private static void inicializarTareasEjemplo() {
95          agregarTarea("Preparar presentaci n para la reuni n",
                "Alta", "Trabajo", "15/12/2024");
96          agregarTarea("Estudiar para examen de matem ticas", "Alta",
                "Estudio", "20/12/2024");
```

```
97          agregarTarea("Comprar v veres para la semana", "Media",
               "Hogar", "10/12/2024");
98          agregarTarea("Ir al gimnasio", "Baja", "Personal",
               "05/12/2024");
99          agregarTarea("Llamar al m dico para cita", "Media",
               "Personal", "12/12/2024");
100
101         // Marcar algunas como completadas
102         completadas[2] = true;
103         fechasCompletacion[2] =
               LocalDate.now().format(FORMATO_FECHA);
104     }
105
106     // ========== MEN S ==========
107     private static void mostrarMenuPrincipal() {
108         System.out.println("\n=== MEN  PRINCIPAL ===");
109         System.out.println("1. Mostrar todas las tareas");
110         System.out.println("2. Agregar nueva tarea");
111         System.out.println("3. Editar tarea existente");
112         System.out.println("4. Eliminar tarea");
113         System.out.println("5. Marcar/Desmarcar como completada");
114         System.out.println("6. Buscar y filtrar tareas");
115         System.out.println("7. Recordatorios");
116         System.out.println("8. Estad sticas");
117         System.out.println("9. Salir");
118         System.out.println("======================");
119     }
120
121     // ========== VALIDACI N ==========
122     private static int leerEnteroValidado(Scanner scanner, String
           mensaje, int min, int max) {
123         int numero = 0;
124         boolean valido = false;
125
126         while (!valido) {
127             try {
128                 System.out.print(mensaje);
129                 numero = scanner.nextInt();
130                 scanner.nextLine(); // Limpiar buffer
131
132                 if (numero >= min && numero <= max) {
133                     valido = true;
134                 } else {
135                     System.out.printf("Error: Debe ser un n mero
                           entre %d y %d.%n", min, max);
```

```
136                 }
137             } catch (Exception e) {
138                 System.out.println("Error: Debe ingresar un n mero
                        entero v lido.");
139                 scanner.nextLine(); // Limpiar buffer
140             }
141         }
142 
143         return numero;
144     }
145 
146     private static String leerStringNoVacio(Scanner scanner, String
        mensaje) {
147         String texto = "";
148         boolean valido = false;
149 
150         while (!valido) {
151             System.out.print(mensaje);
152             texto = scanner.nextLine().trim();
153 
154             if (!texto.isEmpty()) {
155                 valido = true;
156             } else {
157                 System.out.println("Error: Este campo no puede estar
                        vac o.");
158             }
159         }
160 
161         return texto;
162     }
163 
164     private static String leerFechaValidada(Scanner scanner, String
        mensaje) {
165         String fecha = "";
166         boolean valido = false;
167 
168         while (!valido) {
169             System.out.print(mensaje + " (DD/MM/YYYY o Enter para
                    hoy): ");
170             fecha = scanner.nextLine().trim();
171 
172             if (fecha.isEmpty()) {
173                 fecha = LocalDate.now().format(FORMATO_FECHA);
174                 valido = true;
175             } else {
```

```
176                 try {
177                     LocalDate.parse(fecha, FORMATO_FECHA);
178                     valido = true;
179                 } catch (DateTimeParseException e) {
180                     System.out.println("Error: Formato de fecha
                           inv lido. Use DD/MM/YYYY.");
181                 }
182             }
183         }
184 
185         return fecha;
186     }
187 
188     // ========== OPERACIONES B SICAS ==========
189     private static void agregarTarea(String descripcion, String
           prioridad, String categoria, String fechaVencimiento) {
190         if (totalTareas >= MAX_TAREAS) {
191             System.out.println("Error: Capacidad m xima de tareas
                   alcanzada.");
192             return;
193         }
194 
195         ids[totalTareas] = totalTareas + 1;
196         descripciones[totalTareas] = descripcion;
197         prioridades[totalTareas] = prioridad;
198         categorias[totalTareas] = categoria;
199         fechasVencimiento[totalTareas] = fechaVencimiento;
200         completadas[totalTareas] = false;
201         fechasCompletacion[totalTareas] = "";
202 
203         totalTareas++;
204     }
205 
206     private static void agregarNuevaTarea(Scanner scanner) {
207         System.out.println("\n=== AGREGAR NUEVA TAREA ===");
208 
209         // Leer descripci n
210         String descripcion = leerStringNoVacio(scanner,
               "Descripci n de la tarea: ");
211 
212         // Seleccionar prioridad
213         System.out.println("\nSeleccione la prioridad:");
214         for (int i = 0; i < PRIORIDADES.length; i++) {
215             System.out.println((i + 1) + ". " + PRIORIDADES[i]);
216         }
```

```
217         int opcionPrioridad = leerEnteroValidado(scanner, "Opci n:
                ", 1, PRIORIDADES.length);
218         String prioridad = PRIORIDADES[opcionPrioridad - 1];
219 
220         // Seleccionar categor a
221         System.out.println("\nSeleccione la categor a:");
222         for (int i = 0; i < CATEGORIAS.length; i++) {
223             System.out.println((i + 1) + ". " + CATEGORIAS[i]);
224         }
225         int opcionCategoria = leerEnteroValidado(scanner, "Opci n:
                ", 1, CATEGORIAS.length);
226         String categoria = CATEGORIAS[opcionCategoria - 1];
227 
228         // Leer fecha de vencimiento
229         String fechaVencimiento = leerFechaValidada(scanner, "Fecha
                de vencimiento");
230 
231         // Agregar tarea
232         agregarTarea(descripcion, prioridad, categoria,
                fechaVencimiento);
233 
234         System.out.println("\n Tarea agregada exitosamente con ID:
                " + totalTareas + "!");
235     }
236 
237     // ========== VISUALIZACI N ==========
238     private static void mostrarTodasTareas() {
239         if (totalTareas == 0) {
240             System.out.println("\nNo hay tareas registradas.");
241             return;
242         }
243 
244         System.out.println("\n=== LISTA COMPLETA DE TAREAS ===");
245         System.out.printf("%-5s %-40s %-10s %-12s %-15s %-12s%n",
246                           "ID", "Descripci n", "Prioridad",
                                  "Categor a", "Vencimiento", "Estado");
247         System.out.println("-----------------------------------------------------------
248 
249         int tareasPendientes = 0;
250         int tareasCompletadas = 0;
251 
252         for (int i = 0; i < totalTareas; i++) {
253             String estado = completadas[i] ? "    Completada" : "
                    Pendiente";
254             String fechaVencimiento = fechasVencimiento[i];
```

```
255
256             // Resaltar tareas vencidas
257             try {
258                 LocalDate vencimiento =
                        LocalDate.parse(fechasVencimiento[i],
                        FORMATO_FECHA);
259                 LocalDate hoy = LocalDate.now();
260
261                 if (vencimiento.isBefore(hoy) && !completadas[i]) {
262                     estado = "    Vencida";
263                     fechaVencimiento = fechaVencimiento + " (!)";
264                 }
265             } catch (Exception e) {
266                 // Ignorar errores de fecha
267             }
268
269             System.out.printf("%-5d %-40s %-10s %-12s %-15s %-12s%n",
270                             ids[i],
271                             truncarTexto(descripciones[i], 38),
272                             prioridades[i],
273                             categorias[i],
274                             fechaVencimiento,
275                             estado);
276
277             if (completadas[i]) {
278                 tareasCompletadas++;
279             } else {
280                 tareasPendientes++;
281             }
282         }
283
284         System.out.println("-----------------------------------------------------
285         System.out.printf("Total: %d tareas | Pendientes: %d |
                Completadas: %d%n",
286                         totalTareas, tareasPendientes,
                            tareasCompletadas);
287     }
288
289     private static String truncarTexto(String texto, int longitud) {
290         if (texto.length() <= longitud) {
291             return texto;
292         } else {
293             return texto.substring(0, longitud - 3) + "...";
294         }
295     }
```

```
296
297     // ========== EDICI N Y ELIMINACI N ==========
298     private static void editarTarea(Scanner scanner) {
299         if (totalTareas == 0) {
300             System.out.println("\nNo hay tareas para editar.");
301             return;
302         }
303
304         System.out.println("\n=== EDITAR TAREA ===");
305         mostrarTodasTareas();
306
307         System.out.print("\nID de la tarea a editar (0 para
                cancelar): ");
308         int id = scanner.nextInt();
309         scanner.nextLine(); // Limpiar buffer
310
311         if (id == 0) {
312             System.out.println("Edici n cancelada.");
313             return;
314         }
315
316         int indice = buscarTareaPorId(id);
317
318         if (indice == -1) {
319             System.out.println("Error: No se encontr  ninguna tarea
                con ID " + id);
320             return;
321         }
322
323         System.out.println("\nTarea actual:");
324         mostrarDetalleTarea(indice);
325
326         System.out.println("\n Qu  desea editar?");
327         System.out.println("1. Descripci n");
328         System.out.println("2. Prioridad");
329         System.out.println("3. Categor a");
330         System.out.println("4. Fecha de vencimiento");
331         System.out.println("5. Todo");
332         System.out.println("6. Cancelar");
333
334         int opcion = leerEnteroValidado(scanner, "Opci n: ", 1, 6);
335
336         switch (opcion) {
337             case 1:
338                 String nuevaDescripcion = leerStringNoVacio(scanner,
```

```
                    "Nueva descripci n: ");
339                 descripciones[indice] = nuevaDescripcion;
340                 System.out.println("Descripci n actualizada.");
341                 break;
342
343             case 2:
344                 System.out.println("\nSeleccione la nueva
                        prioridad:");
345                 for (int i = 0; i < PRIORIDADES.length; i++) {
346                     System.out.println((i + 1) + ". " +
                            PRIORIDADES[i]);
347                 }
348                 int opcionPrioridad = leerEnteroValidado(scanner,
                        "Opci n: ", 1, PRIORIDADES.length);
349                 prioridades[indice] = PRIORIDADES[opcionPrioridad -
                        1];
350                 System.out.println("Prioridad actualizada.");
351                 break;
352
353             case 3:
354                 System.out.println("\nSeleccione la nueva
                        categor a:");
355                 for (int i = 0; i < CATEGORIAS.length; i++) {
356                     System.out.println((i + 1) + ". " +
                            CATEGORIAS[i]);
357                 }
358                 int opcionCategoria = leerEnteroValidado(scanner,
                        "Opci n: ", 1, CATEGORIAS.length);
359                 categorias[indice] = CATEGORIAS[opcionCategoria - 1];
360                 System.out.println("Categor a actualizada.");
361                 break;
362
363             case 4:
364                 String nuevaFecha = leerFechaValidada(scanner,
                        "Nueva fecha de vencimiento");
365                 fechasVencimiento[indice] = nuevaFecha;
366                 System.out.println("Fecha de vencimiento
                        actualizada.");
367                 break;
368
369             case 5:
370                 // Editar todo
371                 String desc = leerStringNoVacio(scanner, "Nueva
                        descripci n: ");
372                 descripciones[indice] = desc;
```

```
373
374                 System.out.println("\nSeleccione la nueva
                        prioridad:");
375                 for (int i = 0; i < PRIORIDADES.length; i++) {
376                     System.out.println((i + 1) + ". " +
                            PRIORIDADES[i]);
377                 }
378                 opcionPrioridad = leerEnteroValidado(scanner,
                        "Opci n: ", 1, PRIORIDADES.length);
379                 prioridades[indice] = PRIORIDADES[opcionPrioridad -
                        1];
380
381                 System.out.println("\nSeleccione la nueva
                        categor a:");
382                 for (int i = 0; i < CATEGORIAS.length; i++) {
383                     System.out.println((i + 1) + ". " +
                            CATEGORIAS[i]);
384                 }
385                 opcionCategoria = leerEnteroValidado(scanner,
                        "Opci n: ", 1, CATEGORIAS.length);
386                 categorias[indice] = CATEGORIAS[opcionCategoria - 1];
387
388                 nuevaFecha = leerFechaValidada(scanner, "Nueva fecha
                        de vencimiento");
389                 fechasVencimiento[indice] = nuevaFecha;
390
391                 System.out.println("Tarea completamente
                        actualizada.");
392                 break;
393
394             case 6:
395                 System.out.println("Edici n cancelada.");
396                 break;
397         }
398     }
399
400     private static int buscarTareaPorId(int id) {
401         for (int i = 0; i < totalTareas; i++) {
402             if (ids[i] == id) {
403                 return i;
404             }
405         }
406         return -1;
407     }
408
```

```
409     private static void mostrarDetalleTarea(int indice) {
410         System.out.println("\n--- Detalles de la Tarea ---");
411         System.out.println("ID: " + ids[indice]);
412         System.out.println("Descripci n: " + descripciones[indice]);
413         System.out.println("Prioridad: " + prioridades[indice]);
414         System.out.println("Categor a: " + categorias[indice]);
415         System.out.println("Fecha de vencimiento: " +
                fechasVencimiento[indice]);
416         System.out.println("Estado: " + (completadas[indice] ?
                "Completada" : "Pendiente"));
417
418         if (completadas[indice] &&
                !fechasCompletacion[indice].isEmpty()) {
419             System.out.println("Fecha de completaci n: " +
                    fechasCompletacion[indice]);
420         }
421     }
422
423     // ========== COMPLETACI N DE TAREAS ==========
424     private static void gestionarCompletacion(Scanner scanner) {
425         if (totalTareas == 0) {
426             System.out.println("\nNo hay tareas para gestionar.");
427             return;
428         }
429
430         System.out.println("\n=== GESTIONAR COMPLETACI N ===");
431         mostrarTodasTareas();
432
433         System.out.print("\nID de la tarea a marcar/desmarcar como
                completada (0 para cancelar): ");
434         int id = scanner.nextInt();
435         scanner.nextLine(); // Limpiar buffer
436
437         if (id == 0) {
438             System.out.println("Operaci n cancelada.");
439             return;
440         }
441
442         int indice = buscarTareaPorId(id);
443
444         if (indice == -1) {
445             System.out.println("Error: No se encontr  ninguna tarea
                    con ID " + id);
446             return;
447         }
```

```
448
449         if (completadas[indice]) {
450             // Desmarcar como completada
451             completadas[indice] = false;
452             fechasCompletacion[indice] = "";
453             System.out.println("Tarea marcada como PENDIENTE.");
454         } else {
455             // Marcar como completada
456             completadas[indice] = true;
457             fechasCompletacion[indice] =
                    LocalDate.now().format(FORMATO_FECHA);
458             System.out.println(" Tarea  marcada como COMPLETADA!
                     Buen  trabajo!");
459         }
460     }
461
462     // ========== B SQUEDA Y FILTROS ==========
463     private static void buscarYFiltrarTareas(Scanner scanner) {
464         System.out.println("\n=== BUSCAR Y FILTRAR TAREAS ===");
465         System.out.println("1. Buscar por texto en descripci n");
466         System.out.println("2. Filtrar por prioridad");
467         System.out.println("3. Filtrar por categor a");
468         System.out.println("4. Filtrar por estado");
469         System.out.println("5. Filtrar por fecha de vencimiento");
470         System.out.println("6. Mostrar tareas vencidas");
471         System.out.println("7. Mostrar tareas para hoy");
472         System.out.println("8. Volver al men  principal");
473
474         int opcion = leerEnteroValidado(scanner, "Opci n: ", 1, 8);
475
476         switch (opcion) {
477             case 1:
478                 buscarPorTexto(scanner);
479                 break;
480             case 2:
481                 filtrarPorPrioridad(scanner);
482                 break;
483             case 3:
484                 filtrarPorCategoria(scanner);
485                 break;
486             case 4:
487                 filtrarPorEstado(scanner);
488                 break;
489             case 5:
490                 filtrarPorFecha(scanner);
```

```
491                 break;
492             case 6:
493                 mostrarTareasVencidas();
494                 break;
495             case 7:
496                 mostrarTareasParaHoy();
497                 break;
498             case 8:
499                 // Volver al men  principal
500                 break;
501         }
502     }
503 
504     private static void buscarPorTexto(Scanner scanner) {
505         System.out.print("\nTexto a buscar en descripciones: ");
506         String texto = scanner.nextLine().toLowerCase();
507 
508         System.out.println("\n=== RESULTADOS DE B SQUEDA ===");
509         int encontradas = 0;
510 
511         for (int i = 0; i < totalTareas; i++) {
512             if (descripciones[i].toLowerCase().contains(texto)) {
513                 mostrarDetalleTarea(i);
514                 encontradas++;
515             }
516         }
517 
518         System.out.println("\nTotal encontradas: " + encontradas);
519     }
520 
521     private static void filtrarPorPrioridad(Scanner scanner) {
522         System.out.println("\nSeleccione la prioridad para
               filtrar:");
523         for (int i = 0; i < PRIORIDADES.length; i++) {
524             System.out.println((i + 1) + ". " + PRIORIDADES[i]);
525         }
526 
527         int opcion = leerEnteroValidado(scanner, "Opci n: ", 1,
               PRIORIDADES.length);
528         String prioridadFiltrar = PRIORIDADES[opcion - 1];
529 
530         System.out.println("\n=== TAREAS CON PRIORIDAD " +
               prioridadFiltrar.toUpperCase() + " ===");
531         int encontradas = 0;
532 
```

```
533         for (int i = 0; i < totalTareas; i++) {
534             if (prioridades[i].equals(prioridadFiltrar)) {
535                 mostrarResumenTarea(i);
536                 encontradas++;
537             }
538         }
539 
540         System.out.println("\nTotal encontradas: " + encontradas);
541     }
542 
543     private static void mostrarResumenTarea(int indice) {
544         String estado = completadas[indice] ? "   " : "   ";
545         System.out.printf("%s ID %d: %s (Vence: %s)%n",
546                           estado, ids[indice], descripciones[indice],
                                  fechasVencimiento[indice]);
547     }
548 
549     // ========== RECORDATORIOS ==========
550     private static void mostrarRecordatorios() {
551         System.out.println("\n=== RECORDATORIOS ===");
552 
553         LocalDate hoy = LocalDate.now();
554         int recordatorios = 0;
555 
556         System.out.println("\  n      TAREAS QUE VENCEN HOY:");
557         for (int i = 0; i < totalTareas; i++) {
558             if (!completadas[i]) {
559                 try {
560                     LocalDate vencimiento =
                            LocalDate.parse(fechasVencimiento[i],
                            FORMATO_FECHA);
561 
562                     if (vencimiento.isEqual(hoy)) {
563                         System.out.printf("       %s (ID: %d)%n",
                                    descripciones[i], ids[i]);
564                         recordatorios++;
565                     }
566                 } catch (Exception e) {
567                     // Ignorar fechas con formato incorrecto
568                 }
569             }
570         }
571 
572         if (recordatorios == 0) {
573             System.out.println("   No hay tareas que venzan hoy.");
```

```
574         }
575
576         recordatorios = 0;
577         System.out.println("\ n     TAREAS QUE VENCEN MA ANA:");
578         LocalDate manana = hoy.plusDays(1);
579
580         for (int i = 0; i < totalTareas; i++) {
581             if (!completadas[i]) {
582                 try {
583                     LocalDate vencimiento =
                            LocalDate.parse(fechasVencimiento[i],
                            FORMATO_FECHA);
584
585                     if (vencimiento.isEqual(manana)) {
586                         System.out.printf("       %s (ID: %d)%n",
                                descripciones[i], ids[i]);
587                         recordatorios++;
588                     }
589                 } catch (Exception e) {
590                     // Ignorar fechas con formato incorrecto
591                 }
592             }
593         }
594
595         if (recordatorios == 0) {
596             System.out.println("   No hay tareas que venzan
                    ma ana.");
597         }
598
599         recordatorios = 0;
600         System.out.println("\ n     TAREAS VENCIDAS (URGENTE!):");
601
602         for (int i = 0; i < totalTareas; i++) {
603             if (!completadas[i]) {
604                 try {
605                     LocalDate vencimiento =
                            LocalDate.parse(fechasVencimiento[i],
                            FORMATO_FECHA);
606
607                     if (vencimiento.isBefore(hoy)) {
608                         System.out.printf("       %s (ID: %d) -
                                Vencida el %s%n",
609                                         descripciones[i], ids[i],
                                            fechasVencimiento[i]);
610                         recordatorios++;
```

```
611                     }
612                 } catch (Exception e) {
613                     // Ignorar fechas con formato incorrecto
614                 }
615             }
616         }
617 
618         if (recordatorios == 0) {
619             System.out.println("     Excelente  ! No hay tareas
                    vencidas.");
620         } else {
621             System.out.println("\  n        Tienes   " + recordatorios
                    + " tarea(s) vencida(s)!");
622         }
623     }
624 
625     // ========== ESTAD STICAS ==========
626     private static void mostrarEstadisticas() {
627         System.out.println("\n=== ESTAD STICAS DE PRODUCTIVIDAD
                ===");
628 
629         if (totalTareas == 0) {
630             System.out.println("No hay tareas para analizar.");
631             return;
632         }
633 
634         // Contadores
635         int completadasCount = 0;
636         int pendientesCount = 0;
637         int vencidasCount = 0;
638         int[] conteoPrioridad = new int[PRIORIDADES.length];
639         int[] conteoCategoria = new int[CATEGORIAS.length];
640 
641         LocalDate hoy = LocalDate.now();
642 
643         for (int i = 0; i < totalTareas; i++) {
644             // Contar por estado
645             if (completadas[i]) {
646                 completadasCount++;
647             } else {
648                 pendientesCount++;
649 
650                 // Verificar si est  vencida
651                 try {
652                     LocalDate vencimiento =
```

```
                        LocalDate.parse(fechasVencimiento[i],
                        FORMATO_FECHA);
653                     if (vencimiento.isBefore(hoy)) {
654                         vencidasCount++;
655                     }
656                 } catch (Exception e) {
657                     // Ignorar fechas con formato incorrecto
658                 }
659             }
660
661             // Contar por prioridad
662             for (int j = 0; j < PRIORIDADES.length; j++) {
663                 if (prioridades[i].equals(PRIORIDADES[j])) {
664                     conteoPrioridad[j]++;
665                     break;
666                 }
667             }
668
669             // Contar por categor a
670             for (int j = 0; j < CATEGORIAS.length; j++) {
671                 if (categorias[i].equals(CATEGORIAS[j])) {
672                     conteoCategoria[j]++;
673                     break;
674                 }
675             }
676         }
677
678         // Calcular porcentajes
679         double porcentajeCompletadas = (double) completadasCount /
               totalTareas * 100;
680         double porcentajePendientes = (double) pendientesCount /
               totalTareas * 100;
681         double porcentajeVencidas = pendientesCount > 0 ? (double)
               vencidasCount / pendientesCount * 100 : 0;
682
683         // Mostrar estad sticas
684         System.out.println("\ n     RESUMEN GENERAL:");
685         System.out.println("   Total de tareas: " + totalTareas);
686         System.out.printf("   Completadas: %d (%.1f%%)%n",
               completadasCount, porcentajeCompletadas);
687         System.out.printf("   Pendientes: %d (%.1f%%)%n",
               pendientesCount, porcentajePendientes);
688
689         if (vencidasCount > 0) {
690             System.out.printf("           Vencidas: %d (%.1f%% de
```

```
                las pendientes)%n", vencidasCount,
                porcentajeVencidas);
691         }
692
693         System.out.println("\ n     DISTRIBUCI N POR PRIORIDAD:");
694         for (int i = 0; i < PRIORIDADES.length; i++) {
695             double porcentaje = (double) conteoPrioridad[i] /
                    totalTareas * 100;
696             System.out.printf("   %s: %d tareas (%.1f%%)%n",
                    PRIORIDADES[i], conteoPrioridad[i], porcentaje);
697         }
698
699         System.out.println("\ n     DISTRIBUCI N POR CATEGOR A:");
700         for (int i = 0; i < CATEGORIAS.length; i++) {
701             if (conteoCategoria[i] > 0) {
702                 double porcentaje = (double) conteoCategoria[i] /
                        totalTareas * 100;
703                 System.out.printf("   %s: %d tareas (%.1f%%)%n",
                        CATEGORIAS[i], conteoCategoria[i], porcentaje);
704             }
705         }
706
707         // Recomendaciones
708         System.out.println("\ n     RECOMENDACIONES:");
709
710         if (vencidasCount > 0) {
711             System.out.println("           Prioriza las tareas
                    vencidas inmediatamente!");
712         }
713
714         if (conteoPrioridad[0] > 0) { // Prioridad Alta
715             System.out.println("       Tienes " + conteoPrioridad[0]
                    + " tarea(s) de alta prioridad.");
716         }
717
718         if (porcentajeCompletadas >= 80) {
719             System.out.println("          Excelente   tasa de
                    completaci n! Sigue as  .");
720         } else if (porcentajeCompletadas >= 50) {
721             System.out.println("        Buen progreso. Contin a
                    completando tareas.");
722         } else {
723             System.out.println("        Enf cate en completar m s
                    tareas.   T   puedes!");
724         }
```

```
725
726         // Encontrar la categor a con m s tareas pendientes
727         int maxPendientesCategoria = -1;
728         int maxPendientesCount = -1;
729
730         for (int i = 0; i < CATEGORIAS.length; i++) {
731             int pendientesEnCategoria = 0;
732
733             for (int j = 0; j < totalTareas; j++) {
734                 if (categorias[j].equals(CATEGORIAS[i]) &&
                        !completadas[j]) {
735                     pendientesEnCategoria++;
736                 }
737             }
738
739             if (pendientesEnCategoria > maxPendientesCount) {
740                 maxPendientesCount = pendientesEnCategoria;
741                 maxPendientesCategoria = i;
742             }
743         }
744
745         if (maxPendientesCount > 0) {
746             System.out.println("        La categor a '" +
                    CATEGORIAS[maxPendientesCategoria] +
747                             "' tiene m s tareas pendientes (" +
                                maxPendientesCount + ").");
748         }
749     }
750
751     // ========== M TODOS DE FILTRADO ADICIONALES ==========
752     private static void mostrarTareasVencidas() {
753         System.out.println("\n=== TAREAS VENCIDAS ===");
754
755         LocalDate hoy = LocalDate.now();
756         int vencidas = 0;
757
758         for (int i = 0; i < totalTareas; i++) {
759             if (!completadas[i]) {
760                 try {
761                     LocalDate vencimiento =
                            LocalDate.parse(fechasVencimiento[i],
                            FORMATO_FECHA);
762
763                     if (vencimiento.isBefore(hoy)) {
764                         mostrarDetalleTarea(i);
```

```
765                         vencidas++;
766                     }
767                 } catch (Exception e) {
768                     // Ignorar fechas con formato incorrecto
769                 }
770             }
771         }
772 
773         if (vencidas == 0) {
774             System.out.println(" No  hay tareas vencidas!  Buen
                trabajo!");
775         } else {
776             System.out.println("\nTotal de tareas vencidas: " +
                vencidas);
777         }
778     }
779 
780     private static void mostrarTareasParaHoy() {
781         System.out.println("\n=== TAREAS PARA HOY ===");
782 
783         LocalDate hoy = LocalDate.now();
784         String hoyStr = hoy.format(FORMATO_FECHA);
785         int paraHoy = 0;
786 
787         for (int i = 0; i < totalTareas; i++) {
788             if (!completadas[i] &&
                fechasVencimiento[i].equals(hoyStr)) {
789                 mostrarResumenTarea(i);
790                 paraHoy++;
791             }
792         }
793 
794         if (paraHoy == 0) {
795             System.out.println("No hay tareas programadas para
                hoy.");
796         } else {
797             System.out.println("\nTotal de tareas para hoy: " +
                paraHoy);
798         }
799     }
800 }
```

Listing 10.1: Sistema de Gestión de Tareas

10.3. Proyecto 2: Juego de Rol por Consola

Un juego de aventuras por consola con combate, exploración y sistema de progresión.

```
import java.util.Scanner;
import java.util.Random;

/**
 * Juego de Rol por Consola
 *
 * Caracter sticas:
 * - Creaci n de personaje con atributos
 * - Sistema de combate por turnos
 * - Exploraci n de mazmorras
 * - Sistema de niveles y experiencia
 * - Inventario de objetos
 * - Tienda para comprar equipo
 * - Jefes y eventos especiales
 * - Guardado y carga de progreso
 */
public class JuegoRolConsola {

    // ========== CONSTANTES DEL JUEGO ==========
    private static final int MAX_VIDA_BASE = 100;
    private static final int MAX_MANA_BASE = 50;
    private static final int EXPERIENCIA_NIVEL_BASE = 100;
    private static final int MAX_INVENTARIO = 10;
    private static final int MAX_ENEMIGOS = 5;

    // ========== ESTRUCTURAS DE DATOS DEL JUGADOR ==========
    private static String nombreJugador = "";
    private static String claseJugador = "";
    private static int nivel = 1;
    private static int experiencia = 0;
    private static int experienciaSiguienteNivel =
        EXPERIENCIA_NIVEL_BASE;

    private static int vidaMaxima = MAX_VIDA_BASE;
    private static int vidaActual = MAX_VIDA_BASE;
    private static int manaMaximo = MAX_MANA_BASE;
    private static int manaActual = MAX_MANA_BASE;

    private static int ataque = 10;
    private static int defensa = 5;
    private static int magia = 8;
```

```
41      private static int velocidad = 7;
42
43      private static int oro = 50;
44      private static int pocionesVida = 3;
45      private static int pocionesMana = 2;
46
47      // Inventario
48      private static String[] inventario = new String[MAX_INVENTARIO];
49      private static int cantidadInventario = 0;
50
51      // ========== ESTRUCTURAS DE DATOS DEL ENEMIGO ==========
52      private static String nombreEnemigo = "";
53      private static int vidaEnemigo = 0;
54      private static int ataqueEnemigo = 0;
55      private static int defensaEnemigo = 0;
56      private static int experienciaOtorgada = 0;
57      private static int oroOtorgado = 0;
58
59      // ========== ESTADO DEL JUEGO ==========
60      private static boolean juegoActivo = true;
61      private static boolean enCombate = false;
62      private static Random random = new Random();
63      private static Scanner scanner = new Scanner(System.in);
64
65      // ========== M TODO PRINCIPAL ==========
66      public static void main(String[] args) {
67          System.out.println("
68          System.out.println("              AVENTURAS EN LA MAZMORRA
               OSCURA           ");
69          System.out.println("                 Un juego de rol por
               consola            ");
70          System.out.println("
71
72          // Men  principal
73          boolean menuPrincipal = true;
74
75          while (menuPrincipal) {
76              System.out.println("=== MEN  PRINCIPAL ===");
77              System.out.println("1. Nuevo juego");
78              System.out.println("2. Cargar juego (no implementado)");
79              System.out.println("3. Instrucciones");
80              System.out.println("4. Salir");
81              System.out.println("======================");
82
83              System.out.print("Selecciona una opci n: ");
```

```
84              String opcion = scanner.nextLine();
85
86              switch (opcion) {
87                  case "1":
88                      crearPersonaje();
89                      menuPrincipal = false;
90                      break;
91                  case "2":
92                      System.out.println("\  n      Funcionalidad no
                           implementada en esta versi n.");
93                      System.out.println("Presiona Enter para
                           continuar...");
94                      scanner.nextLine();
95                      break;
96                  case "3":
97                      mostrarInstrucciones();
98                      break;
99                  case "4":
100                     System.out.println("\n Gracias por jugar!
                            Hasta  la pr xima aventura!");
101                     return;
102                 default:
103                     System.out.println("\nOpci n  inv lida. Intenta
                           de nuevo.");
104             }
105         }
106
107         // Bucle principal del juego
108         while (juegoActivo) {
109             mostrarMenuPrincipal();
110         }
111
112         scanner.close();
113     }
114
115     // ========== CREACI N DE PERSONAJE ==========
116     private static void crearPersonaje() {
117         System.out.println("\                                                n
118         System.out.println("                    CREACI N DE
               PERSONAJE                ");
119         System.out.println("
120
121         // Nombre del personaje
122         System.out.print("  Cul   es el nombre de tu h roe? ");
123         nombreJugador = scanner.nextLine();
```

```
124
125         // Selecci n de clase
126         System.out.println("\nSelecciona tu clase:");
127         System.out.println("1. Guerrero - Fuerza y defensa
                excelentes");
128         System.out.println("2. Mago - Poder m gico y conocimiento
                arcano");
129         System.out.println("3. Arquero - Precisi n y velocidad");
130         System.out.println("4. Explorador - Vers til y con suerte");
131
132         boolean claseValida = false;
133
134         while (!claseValida) {
135             System.out.print("Opci n (1-4): ");
136             String opcion = scanner.nextLine();
137
138             switch (opcion) {
139                 case "1":
140                     claseJugador = "Guerrero";
141                     vidaMaxima = 120;
142                     ataque = 15;
143                     defensa = 12;
144                     magia = 4;
145                     velocidad = 6;
146                     claseValida = true;
147                     break;
148                 case "2":
149                     claseJugador = "Mago";
150                     vidaMaxima = 80;
151                     manaMaximo = 80;
152                     ataque = 6;
153                     defensa = 4;
154                     magia = 18;
155                     velocidad = 8;
156                     claseValida = true;
157                     break;
158                 case "3":
159                     claseJugador = "Arquero";
160                     vidaMaxima = 90;
161                     ataque = 12;
162                     defensa = 7;
163                     magia = 6;
164                     velocidad = 15;
165                     claseValida = true;
166                     break;
```

```
                case "4":
                    claseJugador = "Explorador";
                    vidaMaxima = 100;
                    ataque = 10;
                    defensa = 8;
                    magia = 8;
                    velocidad = 12;
                    oro = 100; // Bonus de oro
                    claseValida = true;
                    break;
                default:
                    System.out.println("Opci n inv lida. Intenta
                        de nuevo.");
            }
        }

        vidaActual = vidaMaxima;
        manaActual = manaMaximo;

        // A adir objetos iniciales al inventario
        inventario[0] = "Espada de entrenamiento";
        inventario[1] = "Armadura de cuero";
        cantidadInventario = 2;

        if (claseJugador.equals("Mago")) {
            inventario[0] = "Varita b sica";
            inventario[1] = "T nica de aprendiz";
        } else if (claseJugador.equals("Arquero")) {
            inventario[0] = "Arco corto";
            inventario[1] = "Armadura ligera";
        }

        System.out.println("\ n     Personaje   creado con   xito !");
        System.out.println("Nombre: " + nombreJugador);
        System.out.println("Clase: " + claseJugador);
        System.out.println("Nivel: " + nivel);

        System.out.println("\nPresiona Enter para comenzar tu
            aventura...");
        scanner.nextLine();
    }

    // ========== MEN  PRINCIPAL DEL JUEGO ==========
    private static void mostrarMenuPrincipal() {
        System.out.println("\                                    n
```

```
210         System.out.println("                    MAZMORRA OSCURA");
211         System.out.println("
212
213         mostrarEstadoJugador();
214
215         System.out.println("\n Qu  deseas hacer?");
216         System.out.println("1. Explorar la mazmorra");
217         System.out.println("2. Ver inventario");
218         System.out.println("3. Visitar la tienda");
219         System.out.println("4. Descansar (restaurar vida/mana)");
220         System.out.println("5. Ver estad sticas detalladas");
221         System.out.println("6. Guardar y salir");
222         System.out.println("
223
224         System.out.print("Opci n: ");
225         String opcion = scanner.nextLine();
226
227         switch (opcion) {
228             case "1":
229                 explorarMazmorra();
230                 break;
231             case "2":
232                 mostrarInventario();
233                 break;
234             case "3":
235                 visitarTienda();
236                 break;
237             case "4":
238                 descansar();
239                 break;
240             case "5":
241                 mostrarEstadisticasDetalladas();
242                 break;
243             case "6":
244                 System.out.println("\  n      El juego se cerrar
                        sin guardar.");
245                 System.out.print("  Ests   seguro? (s/n): ");
246                 String confirmacion =
                        scanner.nextLine().toLowerCase();
247                 if (confirmacion.equals("s")) {
248                     juegoActivo = false;
249                     System.out.println("\n Gracias por jugar!
                            Hasta  la pr xima aventura!");
250                 }
251                 break;
```

```
252             default:
253                 System.out.println("\nOpci n  inv lida.");
254         }
255     }
256 
257     // ========== ESTADO DEL JUGADOR ==========
258     private static void mostrarEstadoJugador() {
259         System.out.println("\                                                    n
260         System.out.println("H ROE: " + nombreJugador + " - Nivel "
                + nivel + " " + claseJugador);
261         System.out.println("
262 
263         // Barra de vida
264         int porcentajeVida = (int) ((double) vidaActual / vidaMaxima
                * 20);
265         System.out.print("Vida:  [");
266         for (int i = 0; i < 20; i++) {
267             System.out.print(i < porcentajeVida ? "   " : "   ");
268         }
269         System.out.printf("] %d/%d%n", vidaActual, vidaMaxima);
270 
271         // Barra de man  (solo para magos)
272         if (manaMaximo > 0) {
273             int porcentajeMana = (int) ((double) manaActual /
                    manaMaximo * 20);
274             System.out.print("Man :  [");
275             for (int i = 0; i < 20; i++) {
276                 System.out.print(i < porcentajeMana ? "   " : "   ");
277             }
278             System.out.printf("] %d/%d%n", manaActual, manaMaximo);
279         }
280 
281         // Experiencia
282         int porcentajeExp = (int) ((double) experiencia /
                experienciaSiguienteNivel * 20);
283         System.out.print("EXP:   [");
284         for (int i = 0; i < 20; i++) {
285             System.out.print(i < porcentajeExp ? "   " : "   ");
286         }
287         System.out.printf("] %d/%d%n", experiencia,
                experienciaSiguienteNivel);
288 
289         System.out.println("Oro:   " + oro + " monedas");
290         System.out.println("
291     }
```

```
292
293     // ========== EXPLORACI N ==========
294     private static void explorarMazmorra() {
295         System.out.println("\nTe adentras en las profundidades de la
                mazmorra...");
296
297         // Evento aleatorio
298         int evento = random.nextInt(100);
299
300         if (evento < 40) {
301             // Encontrar enemigo (40% de probabilidad)
302             generarEnemigo();
303             iniciarCombate();
304         } else if (evento < 65) {
305             // Encontrar tesoro (25% de probabilidad)
306             encontrarTesoro();
307         } else if (evento < 80) {
308             // Encontrar poci n (15% de probabilidad)
309             encontrarPocion();
310         } else if (evento < 90) {
311             // Trampa (10% de probabilidad)
312             encontrarTrampa();
313         } else {
314             // Sala vac a (10% de probabilidad)
315             System.out.println("La sala est  vac a. Solo escuchas
                    el eco de tus pasos...");
316         }
317
318         if (vidaActual <= 0) {
319             gameOver();
320         }
321     }
322
323     private static void generarEnemigo() {
324         String[] nombresEnemigos = {
325             "Goblin", "Esqueleto", "Orco", "Ara a gigante", "Lobo
                    salvaje",
326             "Bandido", "Zombie", "Fantasma", "Troll", "Drag n
                    juvenil"
327         };
328
329         String[] tiposEnemigos = {"D bil", "Normal", "Fuerte",
                " lite ", "Jefe"};
330
331         int indiceNombre = random.nextInt(nombresEnemigos.length);
```

```
332         int indiceTipo = Math.min(nivel / 3, tiposEnemigos.length -
                1);
333
334         nombreEnemigo = nombresEnemigos[indiceNombre] + " " +
                tiposEnemigos[indiceTipo];
335
336         // Estad sticas basadas en nivel y tipo
337         int baseVida = 20 + (nivel * 5) + (indiceTipo * 15);
338         int baseAtaque = 5 + (nivel * 2) + (indiceTipo * 5);
339         int baseDefensa = 2 + (nivel * 1) + (indiceTipo * 3);
340
341         vidaEnemigo = baseVida + random.nextInt(10);
342         ataqueEnemigo = baseAtaque + random.nextInt(5);
343         defensaEnemigo = baseDefensa + random.nextInt(3);
344
345         experienciaOtorgada = (10 + (nivel * 5) + (indiceTipo * 10));
346         oroOtorgado = (5 + (nivel * 3) + (indiceTipo * 5));
347
348         System.out.println("\  n        Has  encontrado un " +
                nombreEnemigo + "!");
349         System.out.println("Vida del enemigo: " + vidaEnemigo);
350     }
351
352     // ========== SISTEMA DE COMBATE ==========
353     private static void iniciarCombate() {
354         enCombate = true;
355
356         System.out.println("\                                                n
357         System.out.println("                      COMBATE  !");
358         System.out.println("
359
360         boolean turnoJugador = random.nextBoolean(); // Qui n ataca
                primero
361
362         while (enCombate && vidaActual > 0 && vidaEnemigo > 0) {
363             if (turnoJugador) {
364                 mostrarOpcionesCombate();
365             } else {
366                 turnoEnemigo();
367             }
368
369             turnoJugador = !turnoJugador;
370
371             // Mostrar estado despu s de cada turno
372             System.out.println("\
```

```
373             System.out.printf("%s: %d/%d HP", nombreJugador,
                    vidaActual, vidaMaxima);
374             if (manaMaximo > 0) {
375                 System.out.printf(" | %d/%d MP", manaActual,
                        manaMaximo);
376             }
377             System.out.printf("%n%s: %d HP%n", nombreEnemigo,
                    vidaEnemigo);
378             System.out.println("
379         }
380
381         if (vidaEnemigo <= 0) {
382             victoriaCombate();
383         } else if (vidaActual <= 0) {
384             System.out.println("\ n     Has sido derrotado por " +
                    nombreEnemigo + "...");
385         }
386
387         enCombate = false;
388     }
389
390     private static void mostrarOpcionesCombate() {
391         System.out.println("\nEs tu turno.   Qu   deseas hacer?");
392         System.out.println("1. Atacar");
393         System.out.println("2. Usar hechizo (costo: 10 MP)");
394         System.out.println("3. Usar poci n de vida (" +
                pocionesVida + " disponibles)");
395         System.out.println("4. Usar poci n de man  (" +
                pocionesMana + " disponibles)");
396         System.out.println("5. Intentar huir");
397
398         System.out.print("Opci n: ");
399         String opcion = scanner.nextLine();
400
401         switch (opcion) {
402             case "1":
403                 atacarEnemigo();
404                 break;
405             case "2":
406                 if (manaActual >= 10) {
407                     usarHechizo();
408                 } else {
409                     System.out.println(" No  tienes suficiente
                            man !");
410                     mostrarOpcionesCombate(); // Volver a mostrar
```

```
                        opciones
411                 }
412                 break;
413             case "3":
414                 if (pocionesVida > 0) {
415                     usarPocionVida();
416                 } else {
417                     System.out.println(" No  tienes pociones de
                        vida!");
418                     mostrarOpcionesCombate();
419                 }
420                 break;
421             case "4":
422                 if (pocionesMana > 0) {
423                     usarPocionMana();
424                 } else {
425                     System.out.println(" No  tienes pociones de
                        man !");
426                     mostrarOpcionesCombate();
427                 }
428                 break;
429             case "5":
430                 intentarHuir();
431                 break;
432             default:
433                 System.out.println("Opci n inv lida.");
434                 mostrarOpcionesCombate();
435         }
436     }
437
438     private static void atacarEnemigo() {
439         int da oBase = ataque + random.nextInt(5);
440         int da oFinal = Math.max(1, da oBase - (defensaEnemigo /
                2));
441
442         // Cr tico (10% de probabilidad)
443         boolean critico = random.nextInt(100) < 10;
444         if (critico) {
445             da oFinal *= 2;
446             System.out.println("      GOLPE   CR TICO!");
447         }
448
449         vidaEnemigo -= da oFinal;
450
451         System.out.printf("Atacas a %s y le causas %d puntos de
```

```
            da o.%n",
452                         nombreEnemigo, da oFinal);
453
454         if (vidaEnemigo <= 0) {
455             System.out.println(" Has  derrotado al enemigo!");
456         }
457     }
458
459     private static void usarHechizo() {
460         if (manaActual < 10) {
461             System.out.println("No tienes suficiente man  para usar
                un hechizo.");
462             return;
463         }
464
465         manaActual -= 10;
466
467         // Diferentes hechizos seg n la clase
468         int da o = 0;
469         String nombreHechizo = "";
470
471         if (claseJugador.equals("Mago")) {
472             da o = magia + random.nextInt(10);
473             nombreHechizo = "Bola de Fuego";
474         } else if (claseJugador.equals("Guerrero")) {
475             da o = (ataque / 2) + random.nextInt(5);
476             nombreHechizo = "Golpe Poderoso";
477         } else if (claseJugador.equals("Arquero")) {
478             da o = (velocidad / 2) + random.nextInt(5);
479             nombreHechizo = "Flecha M gica";
480         } else {
481             da o = (magia / 2) + random.nextInt(5);
482             nombreHechizo = "Ataque Especial";
483         }
484
485         int da oFinal = Math.max(1, da o - (defensaEnemigo / 3));
486         vidaEnemigo -= da oFinal;
487
488         System.out.printf("     Lanzas '%s' y causas %d puntos de
            da o m gico.%n",
489                         nombreHechizo, da oFinal);
490     }
491
492     private static void turnoEnemigo() {
493         System.out.println("\nTurno del enemigo...");
```

```
494
495         // El enemigo puede atacar o usar habilidad especial
496         int accion = random.nextInt(100);
497
498         if (accion < 80) { // 80% de probabilidad de ataque normal
499             int daoBase = ataqueEnemigo + random.nextInt(5);
500             int daoFinal = Math.max(1, daoBase - (defensa / 2));
501
502             vidaActual -= daoFinal;
503
504             System.out.printf("%s te ataca y causa %d puntos de
                    daño.%n",
505                               nombreEnemigo, daoFinal);
506         } else { // 20% de probabilidad de ataque especial
507             int daoEspecial = (ataqueEnemigo * 3 / 2) +
                    random.nextInt(8);
508             int daoFinal = Math.max(1, daoEspecial - (defensa /
                    3));
509
510             vidaActual -= daoFinal;
511
512             System.out.printf("    %s usa un ataque especial y causa
                    %d puntos de daño!%n",
513                               nombreEnemigo, daoFinal);
514         }
515
516         if (vidaActual <= 0) {
517             System.out.println(" Has  sido derrotado!");
518         }
519     }
520
521     private static void victoriaCombate() {
522         System.out.println("\                                                n
523         System.out.println("                          VICTORIA  !
                                      ");
524         System.out.println("
525
526         System.out.println("\nHas derrotado a " + nombreEnemigo);
527         System.out.println("Recompensas obtenidas:");
528         System.out.println("      Experiencia: +" +
                experienciaOtorgada);
529         System.out.println("      Oro: +" + oroOtorgado + "
                monedas");
530
531         experiencia += experienciaOtorgada;
```

```
532         oro += oroOtorgado;
533 
534         // Posible bot n adicional
535         if (random.nextInt(100) < 30) { // 30% de probabilidad
536             String[] posiblesBotines = {
537                 "Poci n de vida", "Poci n de man ",
538                 "Piedra preciosa", "Pergamino antiguo"
539             };
540 
541             String botin =
                    posiblesBotines[random.nextInt(posiblesBotines.length)];
542             System.out.println("      Bot n: " + botin);
543 
544             // A adir al inventario si hay espacio
545             if (cantidadInventario < MAX_INVENTARIO) {
546                 inventario[cantidadInventario] = botin;
547                 cantidadInventario++;
548             } else {
549                 System.out.println("    (Tu inventario est  lleno,
                        lo dejas atr s)");
550             }
551         }
552 
553         // Verificar subida de nivel
554         while (experiencia >= experienciaSiguienteNivel) {
555             subirNivel();
556         }
557 
558         System.out.println("\nPresiona Enter para continuar...");
559         scanner.nextLine();
560     }
561 
562     // ========== SISTEMA DE NIVELES ==========
563     private static void subirNivel() {
564         nivel++;
565         experiencia -= experienciaSiguienteNivel;
566         experienciaSiguienteNivel = (int) (EXPERIENCIA_NIVEL_BASE *
                Math.pow(1.5, nivel - 1));
567 
568         System.out.println("\ n      Has  subido al nivel " + nivel
                + "!");
569 
570         // Mejorar estad sticas seg n la clase
571         vidaMaxima += 20 + random.nextInt(10);
572         vidaActual = vidaMaxima;
```

```
573
574         if (manaMaximo > 0) {
575             manaMaximo += 10 + random.nextInt(5);
576             manaActual = manaMaximo;
577         }
578
579         switch (claseJugador) {
580             case "Guerrero":
581                 ataque += 3 + random.nextInt(2);
582                 defensa += 2 + random.nextInt(2);
583                 break;
584             case "Mago":
585                 magia += 4 + random.nextInt(2);
586                 manaMaximo += 5; // Bonus extra de man
587                 manaActual = manaMaximo;
588                 break;
589             case "Arquero":
590                 ataque += 2 + random.nextInt(2);
591                 velocidad += 3 + random.nextInt(2);
592                 break;
593             case "Explorador":
594                 ataque += 2 + random.nextInt(2);
595                 defensa += 1 + random.nextInt(2);
596                 velocidad += 2 + random.nextInt(2);
597                 magia += 1 + random.nextInt(2);
598                 break;
599         }
600
601         System.out.println("Tus estad sticas han mejorado:");
602         System.out.println("      Vida m xima: " + vidaMaxima);
603         if (manaMaximo > 0) {
604             System.out.println("      Man  m ximo: " + manaMaximo);
605         }
606         System.out.println("      Ataque: " + ataque);
607         System.out.println("      Defensa: " + defensa);
608         System.out.println("      Magia: " + magia);
609         System.out.println("      Velocidad: " + velocidad);
610
611         // Otorgar puntos de habilidad (simplificado)
612         System.out.println("\n Has ganado 1 punto de habilidad para
               distribuir!");
613         System.out.println(" En  qu  estad stica deseas
               invertirlo?");
614         System.out.println("1. Ataque (+2)");
615         System.out.println("2. Defensa (+2)");
```

```
616         if (claseJugador.equals("Mago") || magia > 5) {
617             System.out.println("3. Magia (+2)");
618         }
619         System.out.println("4. Velocidad (+2)");
620 
621         boolean eleccionValida = false;
622         while (!eleccionValida) {
623             System.out.print("Opci n: ");
624             String opcion = scanner.nextLine();
625 
626             switch (opcion) {
627                 case "1":
628                     ataque += 2;
629                     System.out.println(" Ataque  aumentado en 2
                        puntos!");
630                     eleccionValida = true;
631                     break;
632                 case "2":
633                     defensa += 2;
634                     System.out.println(" Defensa  aumentada en 2
                        puntos!");
635                     eleccionValida = true;
636                     break;
637                 case "3":
638                     if (claseJugador.equals("Mago") || magia > 5) {
639                         magia += 2;
640                         System.out.println(" Magia  aumentada en 2
                            puntos!");
641                         eleccionValida = true;
642                     } else {
643                         System.out.println("Opci n no disponible
                            para tu clase.");
644                     }
645                     break;
646                 case "4":
647                     velocidad += 2;
648                     System.out.println(" Velocidad  aumentada en 2
                        puntos!");
649                     eleccionValida = true;
650                     break;
651                 default:
652                     System.out.println("Opci n inv lida.");
653             }
654         }
655     }
```

```
656
657     // ========== INVENTARIO ==========
658     private static void mostrarInventario() {
659         System.out.println("\                                                    n
660         System.out.println("                    INVENTARIO");
661         System.out.println("
662
663         System.out.println("Oro: " + oro + " monedas");
664         System.out.println("Pociones de vida: " + pocionesVida);
665         System.out.println("Pociones de man : " + pocionesMana);
666
667         System.out.println("\nEquipo:");
668         if (cantidadInventario == 0) {
669             System.out.println("  (Vac o)");
670         } else {
671             for (int i = 0; i < cantidadInventario; i++) {
672                 System.out.println("  " + (i + 1) + ". " +
                        inventario[i]);
673             }
674         }
675
676         System.out.println("\                                                    n
677         System.out.println("Presiona Enter para continuar...");
678         scanner.nextLine();
679     }
680
681     // ========== TIENDA ==========
682     private static void visitarTienda() {
683         System.out.println("\                                                      n
684         System.out.println("                    TIENDA DEL PUEBLO
                            ");
685         System.out.println("
686
687         System.out.println("\n Bienvenido a la tienda!");
688         System.out.println("Tu oro: " + oro + " monedas\n");
689
690         boolean enTienda = true;
691
692         while (enTienda) {
693             System.out.println("
694             System.out.println("  Qu   deseas comprar?");
695             System.out.println("1. Poci n de vida - 20 monedas
                    (restaura 50 HP)");
696             System.out.println("2. Poci n de man  - 30 monedas
                    (restaura 30 MP)");
```

```
697             System.out.println("3. Espada de hierro - 100 monedas
                    (+5 ataque)");
698             System.out.println("4. Armadura de placas - 120 monedas
                    (+7 defensa)");
699             System.out.println("5. Amuleto m gico - 150 monedas (+5
                    magia)");
700             System.out.println("6. Botas de velocidad - 80 monedas
                    (+5 velocidad)");
701             System.out.println("7. Salir de la tienda");
702             System.out.println("
703
704             System.out.print("Opci n: ");
705             String opcion = scanner.nextLine();
706
707             switch (opcion) {
708                 case "1":
709                     if (oro >= 20) {
710                         oro -= 20;
711                         pocionesVida++;
712                         System.out.println(" Has  comprado 1 poci n
                                de vida!");
713                     } else {
714                         System.out.println(" No  tienes suficiente
                                oro!");
715                     }
716                     break;
717                 case "2":
718                     if (oro >= 30) {
719                         oro -= 30;
720                         pocionesMana++;
721                         System.out.println(" Has  comprado 1 poci n
                                de man !");
722                     } else {
723                         System.out.println(" No  tienes suficiente
                                oro!");
724                     }
725                     break;
726                 case "3":
727                     if (oro >= 100) {
728                         oro -= 100;
729                         ataque += 5;
730                         System.out.println(" Has  comprado una
                                espada de hierro! (+5 ataque)");
731                     } else {
732                         System.out.println(" No  tienes suficiente
```

```
                                oro!");
733                         }
734                         break;
735                     case "4":
736                         if (oro >= 120) {
737                             oro -= 120;
738                             defensa += 7;
739                             System.out.println(" Has  comprado armadura
                                de placas! (+7 defensa)");
740                         } else {
741                             System.out.println(" No  tienes suficiente
                                oro!");
742                         }
743                         break;
744                     case "5":
745                         if (oro >= 150) {
746                             oro -= 150;
747                             magia += 5;
748                             System.out.println(" Has  comprado un
                                amuleto m gico! (+5 magia)");
749                         } else {
750                             System.out.println(" No  tienes suficiente
                                oro!");
751                         }
752                         break;
753                     case "6":
754                         if (oro >= 80) {
755                             oro -= 80;
756                             velocidad += 5;
757                             System.out.println(" Has  comprado botas de
                                velocidad! (+5 velocidad)");
758                         } else {
759                             System.out.println(" No  tienes suficiente
                                oro!");
760                         }
761                         break;
762                     case "7":
763                         enTienda = false;
764                         System.out.println(" Gracias  por tu visita!");
765                         break;
766                     default:
767                         System.out.println("Opci n inv lida.");
768             }
769
770             if (enTienda) {
```

```
771                 System.out.println("Tu oro actual: " + oro + "
                        monedas");
772                 System.out.println("\nPresiona Enter para
                        continuar...");
773                 scanner.nextLine();
774             }
775         }
776     }
777
778     // ========== EVENTOS ESPECIALES ==========
779     private static void encontrarTesoro() {
780         System.out.println("\ n     Has   encontrado un cofre del
                tesoro!");
781
782         int tipoTesoro = random.nextInt(100);
783
784         if (tipoTesoro < 50) { // 50% - Tesoro peque o
785             int oroEncontrado = 10 + random.nextInt(20);
786             oro += oroEncontrado;
787             System.out.println("Contiene " + oroEncontrado + "
                    monedas de oro.");
788         } else if (tipoTesoro < 80) { // 30% - Tesoro mediano
789             int oroEncontrado = 30 + random.nextInt(40);
790             oro += oroEncontrado;
791             System.out.println("Contiene " + oroEncontrado + "
                    monedas de oro!");
792         } else if (tipoTesoro < 95) { // 15% - Tesoro grande
793             int oroEncontrado = 70 + random.nextInt(60);
794             oro += oroEncontrado;
795             System.out.println("      Contiene   " + oroEncontrado + "
                    monedas de oro!    ");
796         } else { // 5% - Tesoro  pico
797             int oroEncontrado = 150 + random.nextInt(100);
798             oro += oroEncontrado;
799
800             // Objeto especial
801             String[] objetosEspeciales = {
802                 "Espada legendaria", "Armadura ancestral",
803                 "Poci n de la eterna juventud", "Gema del poder"
804             };
805
806             String objetoEspecial =
                    objetosEspeciales[random.nextInt(objetosEspeciales.length)];
807
808             System.out.println("       TESORO    PICO   ENCONTRADO!
```

```
                    ");
809             System.out.println("Contiene " + oroEncontrado + "
                    monedas de oro y " + objetoEspecial + "!");
810
811             if (cantidadInventario < MAX_INVENTARIO) {
812                 inventario[cantidadInventario] = objetoEspecial;
813                 cantidadInventario++;
814                 System.out.println("Has guardado " + objetoEspecial
                        + " en tu inventario.");
815             } else {
816                 System.out.println("Tu inventario est  lleno,
                        tienes que dejar " + objetoEspecial + "
                        atr s...");
817             }
818         }
819
820         System.out.println("\nPresiona Enter para continuar...");
821         scanner.nextLine();
822     }
823
824     private static void encontrarPocion() {
825         System.out.println("\ n     Encuentras un frasco brillante
                en el suelo...");
826
827         int tipoPocion = random.nextInt(100);
828
829         if (tipoPocion < 60) { // 60% - Poci n de vida
830             pocionesVida++;
831             System.out.println(" Es  una poci n de vida! Ahora
                    tienes " + pocionesVida + ".");
832         } else if (tipoPocion < 90) { // 30% - Poci n de man
833             pocionesMana++;
834             System.out.println(" Es  una poci n de man ! Ahora
                    tienes " + pocionesMana + ".");
835         } else { // 10% - Poci n misteriosa
836             System.out.println(" Es  una poci n misteriosa!");
837
838             int efecto = random.nextInt(100);
839             if (efecto < 40) { // 40% - Efecto positivo
840                 System.out.println("Bebes la poci n y sientes una
                        oleada de energ a.");
841                 vidaActual = Math.min(vidaMaxima, vidaActual + 30);
842                 if (manaMaximo > 0) {
843                     manaActual = Math.min(manaMaximo, manaActual +
                            20);
```

```
844                 }
845                 System.out.println("  Restaurado   30 HP y 20 MP!");
846             } else if (efecto < 70) { // 30% - Efecto negativo
847                 System.out.println("Bebes la poci n y te sientes
                        mareado...");
848                 vidaActual = Math.max(1, vidaActual - 15);
849                 System.out.println("  Pierdes   15 HP!");
850             } else { // 30% - Efecto aleatorio
851                 System.out.println("Bebes la poci n y sientes
                        cambios extra os...");
852                 int stat = random.nextInt(4);
853                 switch (stat) {
854                     case 0:
855                         ataque += 3;
856                         System.out.println(" Tu  ataque aumenta en 3
                            puntos temporalmente!");
857                         break;
858                     case 1:
859                         defensa += 3;
860                         System.out.println(" Tu  defensa aumenta en
                            3 puntos temporalmente!");
861                         break;
862                     case 2:
863                         velocidad += 3;
864                         System.out.println(" Tu  velocidad aumenta
                            en 3 puntos temporalmente!");
865                         break;
866                     case 3:
867                         magia += 3;
868                         System.out.println(" Tu  magia aumenta en 3
                            puntos temporalmente!");
869                         break;
870                 }
871             }
872         }
873
874         System.out.println("\nPresiona Enter para continuar...");
875         scanner.nextLine();
876     }
877
878     private static void encontrarTrampa() {
879         System.out.println("\  n        Has  activado una trampa!");
880
881         int tipoTrampa = random.nextInt(100);
882
```

```
883         if (tipoTrampa < 40) { // 40% - Trampa de dardos
884             int da o = 10 + random.nextInt(15);
885             vidaActual = Math.max(1, vidaActual - da o);
886             System.out.println(" Dardos  envenenados te golpean!
                    Pierdes " + da o + " HP.");
887         } else if (tipoTrampa < 70) { // 30% - Trampa de hoyo
888             int da o = 15 + random.nextInt(20);
889             vidaActual = Math.max(1, vidaActual - da o);
890             System.out.println(" Caes  en un hoyo con p as! Pierdes
                    " + da o + " HP.");
891         } else if (tipoTrampa < 90) { // 20% - Trampa de gas
892             System.out.println(" Gas  venenoso te envuelve!");
893             vidaActual = Math.max(1, vidaActual - 8);
894             if (manaMaximo > 0) {
895                 manaActual = Math.max(0, manaActual - 10);
896                 System.out.println("Pierdes 8 HP y 10 MP.");
897             } else {
898                 System.out.println("Pierdes 8 HP.");
899             }
900         } else { // 10% - Trampa maldici n
901             System.out.println(" Una  maldici n antigua te
                    afecta!");
902 
903             // Reducir una estad stica aleatoria
904             int stat = random.nextInt(4);
905             switch (stat) {
906                 case 0:
907                     ataque = Math.max(1, ataque - 2);
908                     System.out.println("Tu ataque se reduce en 2
                            puntos.");
909                     break;
910                 case 1:
911                     defensa = Math.max(1, defensa - 2);
912                     System.out.println("Tu defensa se reduce en 2
                            puntos.");
913                     break;
914                 case 2:
915                     velocidad = Math.max(1, velocidad - 2);
916                     System.out.println("Tu velocidad se reduce en 2
                            puntos.");
917                     break;
918                 case 3:
919                     magia = Math.max(1, magia - 2);
920                     System.out.println("Tu magia se reduce en 2
                            puntos.");
```

```
921                     break;
922                 }
923             }
924 
925             System.out.println("\nPresiona Enter para continuar...");
926             scanner.nextLine();
927         }
928 
929         // ========== ACCIONES DEL JUGADOR ==========
930         private static void usarPocionVida() {
931             int curacion = 50;
932             vidaActual = Math.min(vidaMaxima, vidaActual + curacion);
933             pocionesVida--;
934 
935             System.out.println("     Usas una poci n de vida y
                restauras " + curacion + " HP.");
936             System.out.println("Te quedan " + pocionesVida + " pociones
                de vida.");
937         }
938 
939         private static void usarPocionMana() {
940             int curacion = 30;
941             manaActual = Math.min(manaMaximo, manaActual + curacion);
942             pocionesMana--;
943 
944             System.out.println("     Usas una poci n de man  y
                restauras " + curacion + " MP.");
945             System.out.println("Te quedan " + pocionesMana + " pociones
                de man .");
946         }
947 
948         private static void intentarHuir() {
949             int probabilidadHuir = velocidad * 3; // M s velocidad =
                m s f cil huir
950 
951             if (random.nextInt(100) < probabilidadHuir) {
952                 System.out.println("      Logras  huir del combate!");
953                 enCombate = false;
954             } else {
955                 System.out.println("         No  logras huir! El enemigo
                    te ataca por la espalda.");
956 
957                 // El enemigo ataca con ventaja
958                 int da o = (ataqueEnemigo * 3 / 2) + random.nextInt(5);
959                 vidaActual = Math.max(1, vidaActual - da o);
```

```
960
961             System.out.println("Recibes " + daño + " puntos de
                    daño por intentar huir.");
962         }
963     }
964
965     private static void descansar() {
966         System.out.println("\nTe tomas un momento para
                descansar...");
967
968         int curacionVida = vidaMaxima / 4;
969         int curacionMana = manaMaximo / 4;
970
971         vidaActual = Math.min(vidaMaxima, vidaActual + curacionVida);
972
973         if (manaMaximo > 0) {
974             manaActual = Math.min(manaMaximo, manaActual +
                    curacionMana);
975             System.out.printf("Descansas y restauras %d HP y %d
                    MP.%n", curacionVida, curacionMana);
976         } else {
977             System.out.println("Descansas y restauras " +
                    curacionVida + " HP.");
978         }
979
980         // Posible evento durante el descanso
981         if (random.nextInt(100) < 20) { // 20% de probabilidad
982             System.out.println("\nMientras descansas, algo
                    sucede...");
983
984             int evento = random.nextInt(100);
985             if (evento < 50) { // 50% - Evento positivo
986                 System.out.println("Encuentras algunas hierbas
                        medicinales.");
987                 vidaActual = Math.min(vidaMaxima, vidaActual + 10);
988                 System.out.println("Restauras 10 HP adicionales.");
989             } else { // 50% - Evento negativo
990                 System.out.println(" Un  animal salvaje te ataca
                        mientras descansas!");
991                 int daño = 5 + random.nextInt(10);
992                 vidaActual = Math.max(1, vidaActual - daño);
993                 System.out.println("Pierdes " + daño + " HP.");
994             }
995         }
996
```

```
997         System.out.println("\nPresiona Enter para continuar...");
998         scanner.nextLine();
999     }
1000
1001    private static void mostrarEstadisticasDetalladas() {
1002        System.out.println("\                                                    n
1003        System.out.println("           ESTAD STICAS DETALLADAS");
1004        System.out.println("
1005
1006        System.out.println("Nombre: " + nombreJugador);
1007        System.out.println("Clase: " + claseJugador);
1008        System.out.println("Nivel: " + nivel);
1009        System.out.println("Experiencia: " + experiencia + "/" +
               experienciaSiguienteNivel);
1010        System.out.println("Oro: " + oro + " monedas");
1011
1012        System.out.println("\n--- ESTAD STICAS PRINCIPALES ---");
1013        System.out.println("Vida: " + vidaActual + "/" + vidaMaxima);
1014        System.out.println("Man  : " + manaActual + "/" +
               manaMaximo);
1015        System.out.println("Ataque: " + ataque + " (da o  f sico)");
1016        System.out.println("Defensa: " + defensa + " (reduce da o
               f sico)");
1017        System.out.println("Magia: " + magia + " (poder m gico)");
1018        System.out.println("Velocidad: " + velocidad + " (turnos y
               huida)");
1019
1020        System.out.println("\n--- INVENTARIO ---");
1021        System.out.println("Pociones de vida: " + pocionesVida);
1022        System.out.println("Pociones de man  : " + pocionesMana);
1023        System.out.println("Espacio en inventario: " +
               cantidadInventario + "/" + MAX_INVENTARIO);
1024
1025        if (cantidadInventario > 0) {
1026            System.out.println("Objetos:");
1027            for (int i = 0; i < cantidadInventario; i++) {
1028                System.out.println("      " + inventario[i]);
1029            }
1030        }
1031
1032        System.out.println("\n--- LOGROS ---");
1033        System.out.println("Nivel m ximo alcanzado: " + nivel);
1034        System.out.println("Oro m ximo obtenido: " + oro + "
               monedas");
1035
```

```
1036            System.out.println("\                                                    n
1037            System.out.println("Presiona Enter para continuar...");
1038            scanner.nextLine();
1039        }
1040
1041        private static void mostrarInstrucciones() {
1042            System.out.println("\                                                    n
1043            System.out.println("                    INSTRUCCIONES");
1044            System.out.println("
1045
1046            System.out.println("\ n     C MO JUGAR:");
1047            System.out.println("1. Explora la mazmorra para encontrar
                   enemigos, tesoros y eventos.");
1048            System.out.println("2. Derrota enemigos para ganar
                   experiencia y oro.");
1049            System.out.println("3. Sube de nivel para mejorar tus
                   estad sticas.");
1050            System.out.println("4. Visita la tienda para comprar equipo
                   y pociones.");
1051            System.out.println("5. Descansa para restaurar vida y
                   man .");
1052            System.out.println("\nEl objetivo es sobrevivir y alcanzar
                   el nivel m s alto posible.");
1053
1054            System.out.println("\  n      SISTEMA DE COMBATE:");
1055            System.out.println("- Los combates son por turnos.");
1056            System.out.println("- Ataca para causar da o f sico.");
1057            System.out.println("- Usa hechizos para da o m gico
                   (cuesta man ).");
1058            System.out.println("- Usa pociones para curarte durante el
                   combate.");
1059            System.out.println("- Intenta huir si est s en
                   desventaja.");
1060
1061            System.out.println("\ n     CLASES:");
1062            System.out.println("- Guerrero: Alta vida, ataque y
                   defensa.");
1063            System.out.println("- Mago: Poder m gico y man , pero poca
                   vida.");
1064            System.out.println("- Arquero: Velocidad y precisi n.");
1065            System.out.println("- Explorador: Equilibrado con bonus de
                   oro inicial.");
1066
1067            System.out.println("\ n     ESTAD STICAS:");
1068            System.out.println("- Vida (HP): Si llega a 0, mueres.");
```

```
1069            System.out.println("- Man  (MP): Necesario para hechizos
                    (solo magos).");
1070            System.out.println("- Ataque: Da o f sico causado.");
1071            System.out.println("- Defensa: Reduce el da o recibido.");
1072            System.out.println("- Magia: Poder de hechizos.");
1073            System.out.println("- Velocidad: Probabilidad de atacar
                    primero y huir.");
1074
1075            System.out.println("\                                    n
1076            System.out.println("Presiona Enter para volver al men
                    principal...");
1077            scanner.nextLine();
1078        }
1079
1080        private static void gameOver() {
1081            System.out.println("\                                      n
1082            System.out.println("                    GAME OVER
                                        ");
1083            System.out.println("
1084
1085            System.out.println("\ n     " + nombreJugador + " ha ca do
                    en combate...");
1086            System.out.println("\                                    n
1087            System.out.println("            RESUMEN DE TU AVENTURA");
1088            System.out.println("
1089            System.out.println("Nivel alcanzado: " + nivel);
1090            System.out.println("Oro acumulado: " + oro + " monedas");
1091            System.out.println("Experiencia total: " + experiencia);
1092
1093            if (nivel >= 10) {
1094                System.out.println("\ n       IMPRESIONANTE  ! Llegaste al
                        nivel " + nivel + ".");
1095            } else if (nivel >= 5) {
1096                System.out.println("\ n      Buen  trabajo! Llegaste al
                        nivel " + nivel + ".");
1097            } else {
1098                System.out.println("\ n      Sigue  intent ndolo! La
                        pr xima llegar s  m  s lejos.");
1099            }
1100
1101            System.out.println("\                                    n
1102            System.out.println("1. Jugar de nuevo");
1103            System.out.println("2. Salir del juego");
1104            System.out.println("
1105
```

```
1106            System.out.print("Opci n: ");
1107            String opcion = scanner.nextLine();
1108
1109            if (opcion.equals("1")) {
1110                // Reiniciar el juego
1111                reiniciarJuego();
1112            } else {
1113                juegoActivo = false;
1114                System.out.println("\n Gracias por jugar!  Hasta  la
                       pr xima aventura!");
1115            }
1116        }
1117
1118        private static void reiniciarJuego() {
1119            // Reiniciar todas las variables
1120            nombreJugador = "";
1121            claseJugador = "";
1122            nivel = 1;
1123            experiencia = 0;
1124            experienciaSiguienteNivel = EXPERIENCIA_NIVEL_BASE;
1125
1126            vidaMaxima = MAX_VIDA_BASE;
1127            vidaActual = MAX_VIDA_BASE;
1128            manaMaximo = MAX_MANA_BASE;
1129            manaActual = MAX_MANA_BASE;
1130
1131            ataque = 10;
1132            defensa = 5;
1133            magia = 8;
1134            velocidad = 7;
1135
1136            oro = 50;
1137            pocionesVida = 3;
1138            pocionesMana = 2;
1139
1140            // Limpiar inventario
1141            for (int i = 0; i < MAX_INVENTARIO; i++) {
1142                inventario[i] = null;
1143            }
1144            cantidadInventario = 0;
1145
1146            // Volver a crear personaje
1147            crearPersonaje();
1148        }
1149    }
```

Listing 10.2: Juego de Rol por Consola

Ejercicio Propuesto

Proyecto Final: Sistema de Gestión Bancaria

Crea un sistema bancario completo que incluya:

1. Sistema de autenticación de usuarios
2. Diferentes tipos de cuentas (ahorro, corriente, inversión)
3. Operaciones: depósitos, retiros, transferencias
4. Historial de transacciones
5. Sistema de préstamos con tasas de interés
6. Generación de estados de cuenta
7. Sistema de seguridad (PIN, límites de transacción)
8. Reportes para administradores
9. Persistencia básica (guardar en arrays)

Ejercicio Propuesto

Proyecto Avanzado: Simulador de Ecosistema

Crea un simulador de ecosistema que incluya:

1. Diferentes especies de animales con comportamientos únicos
2. Sistema de alimentación y cadena alimenticia
3. Ciclos de vida (nacimiento, crecimiento, reproducción, muerte)
4. Factores ambientales (clima, recursos, desastres naturales)
5. Simulación por turnos con estadísticas
6. Gráficos ASCII para visualizar el ecosistema
7. Capacidad de intervenir en la simulación
8. Guardar y cargar estados de simulación

Ejercicio Propuesto

Proyecto de Integración: Sistema de Reservas de Vuelos

Crea un sistema de reservas de vuelos completo:

1. Base de datos de vuelos (origen, destino, horarios, precios)
2. Sistema de reservas con selección de asientos
3. Gestión de pasajeros y sus datos
4. Sistema de pago simulado
5. Generación de itinerarios y pases de abordar
6. Sistema de check-in
7. Gestión de equipaje
8. Reportes de ocupación y ganancias
9. Sistema de cancelaciones y cambios

10.4. Consejos para Proyectos Futuros

1. **Planifica antes de programar**: Dibuja diagramas, escribe pseudocódigo
2. **Divide y vencerás**: Separa el proyecto en módulos manejables
3. **Prototipo primero**: Crea una versión simple que funcione, luego añade características
4. **Prueba constantemente**: No esperes a terminar para probar
5. **Documenta tu código**: Los comentarios ayudan cuando retomas un proyecto
6. **Control de versiones**: Aunque simple, guarda copias de tu trabajo
7. **Pide retroalimentación**: Otros pueden ver errores que tú no ves
8. **No te rindas**: Los proyectos complejos requieren tiempo y paciencia

10.5. Resumen del Curso

A lo largo de este libro, hemos cubierto:

- **Fundamentos de Java**: Instalación, sintaxis básica, estructura de programas
- **Variables y tipos de datos**: Tipos primitivos, Strings, conversiones
- **Operadores y expresiones**: Aritméticos, relacionales, lógicos, precedencia
- **Control de flujo**: Condicionales, bucles, break, continue
- **Entrada/Salida**: Scanner, formateo, validación de datos
- **Métodos**: Modularización, parámetros, retorno de valores
- **Ámbito de variables**: Locales, estáticas, paso por valor
- **Arrays**: Unidimensionales, multidimensionales, búsqueda, ordenamiento
- **Buenas prácticas**: Código legible, validación, documentación, eficiencia
- **Proyectos completos**: Sistemas reales aplicando todos los conceptos

Nota Importante

Has completado un curso completo de programación lineal en Java. ¡Felicidades! Este es solo el comienzo de tu viaje en la programación. Continúa practicando, construyendo proyectos y aprendiendo.

10.6. Recursos para Continuar Aprendiendo

- **Practicar problemas**: LeetCode, HackerRank, Codewars
- **Proyectos open source**: Contribuye a proyectos pequeños en GitHub
- **Libros avanzados**: .Effective Java", Clean Code"
- **Frameworks y librerías**: Aprende Spring Boot para backend
- **Bases de datos**: MySQL, PostgreSQL, MongoDB
- **Patrones de diseño**: Aplica patrones comunes a tus proyectos
- **Testing**: JUnit para pruebas unitarias
- **Control de versiones**: Git y GitHub

¡Éxito en tu viaje de programación!

."El único modo de hacer un gran trabajo es amar lo que haces."
- Steve Jobs