Tema 2: Estructuras de Control en Java

1. Estructuras de Selección

Las estructuras de selección permiten que un programa tome decisiones y ejecute diferentes bloques de código según las condiciones especificadas.

1.1 Estructura if

La estructura `if` ejecuta un bloque de código solo si la condición es verdadera.

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

        // if simple
        if (edad >= 18) {
            System.out.println("Eres mayor de edad");
        }

        // if con múltiples condiciones
        double nota = 7.5;
        boolean asistencia = true;

        if (nota >= 5.0 && asistencia) {
            System.out.println("Has aprobado la asignatura");
            System.out.println("Tu nota es: " + nota);
        }

        // if anidados
        int hora = 14;

        if (hora >= 6) {
            if (hora < 12) {
                System.out.println("Buenos días");
            }
        }
    }
}
```

1.2 Estructura if-else

La estructura `if-else` permite ejecutar un bloque de código si la condición es verdadera y otro diferente si es falsa.

```
import java.util.Scanner;

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

        // if-else simple
        System.out.print("Introduce tu edad: ");
        int edad = scanner.nextInt();
```

```
        if (edad >= 18) {
            System.out.println("Puedes votar");
            System.out.println("Puedes conducir");
        } else {
            System.out.println("Eres menor de edad");
            System.out.println("No puedes votar ni conducir");
        }

        // if-else-if encadenados
        System.out.print("Introduce tu nota (0-10): ");
        double nota = scanner.nextDouble();

        if (nota < 0 || nota > 10) {
            System.out.println("Nota inválida");
        } else if (nota < 5) {
            System.out.println("Suspenso");
        } else if (nota < 6) {
            System.out.println("Aprobado");
        } else if (nota < 7) {
            System.out.println("Bien");
        } else if (nota < 9) {
            System.out.println("Notable");
        } else if (nota <= 10) {
            System.out.println("Sobresaliente");
        }

        // Ejemplo práctico: Calculadora de IMC
        System.out.print("Introduce tu peso (kg): ");
        double peso = scanner.nextDouble();
        System.out.print("Introduce tu altura (m): ");
        double altura = scanner.nextDouble();

        double imc = peso / (altura * altura);
        System.out.printf("Tu IMC es: %.2f - ", imc);

        if (imc < 18.5) {
            System.out.println("Bajo peso");
        } else if (imc < 25) {
            System.out.println("Peso normal");
        } else if (imc < 30) {
            System.out.println("Sobrepeso");
        } else {
            System.out.println("Obesidad");
        }

        scanner.close();
    }
}
```

1.3 Operador Condicional (Ternario)

El operador condicional `? :` es una forma compacta de escribir una expresión if-else simple.

```
public class OperadorTernario {
    public static void main(String[] args) {
        // Sintaxis: condición ? valorSiVerdadero : valorSiFalso

        // Ejemplo básico
        int edad = 20;
```

```
        String mensaje = (edad >= 18) ? "Mayor de edad" : "Menor de
edad";
        System.out.println(mensaje);

        // Asignación de valores
        int a = 10, b = 20;
        int mayor = (a > b) ? a : b;
        System.out.println("El mayor es: " + mayor);

        // En impresiones directas
        double nota = 6.5;
        System.out.println("Estado: " + ((nota >= 5) ? "Aprobado" :
"Suspenso"));

        // Operadores ternarios anidados (usar con precaución)
        int numero = 0;
        String tipo = (numero > 0) ? "Positivo" :
                      (numero < 0) ? "Negativo" : "Cero";
        System.out.println("El número es: " + tipo);

        // Uso en métodos
        boolean esPar = verificarPar(8);
        System.out.println("8 es " + (esPar ? "par" : "impar"));

        // Ejemplo práctico: Descuento
        double precio = 100.0;
        boolean esVIP = true;
        double precioFinal = precio * (esVIP ? 0.8 : 1.0); // 20%
descuento si es VIP
        System.out.println("Precio final: " + precioFinal + "€");
    }

    public static boolean verificarPar(int num) {
        return (num % 2 == 0) ? true : false;
    }
}
```

1.4 Estructura switch

La estructura `switch` evalúa una expresión y ejecuta el código correspondiente al caso coincidente.

```
import java.util.Scanner;

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

        // Switch básico con int
        System.out.print("Introduce un día (1-7): ");
        int dia = scanner.nextInt();

        switch (dia) {
            case 1:
                System.out.println("Lunes");
                break;
            case 2:
                System.out.println("Martes");
                break;
            case 3:
```

```
            System.out.println("Miércoles");
            break;
        case 4:
            System.out.println("Jueves");
            break;
        case 5:
            System.out.println("Viernes");
            break;
        case 6:
            System.out.println("Sábado");
            break;
        case 7:
            System.out.println("Domingo");
            break;
        default:
            System.out.println("Día inválido");
            break;
}

// Switch con múltiples casos
System.out.print("Introduce un mes (1-12): ");
int mes = scanner.nextInt();

switch (mes) {
    case 1: case 3: case 5: case 7: case 8: case 10: case 12:
        System.out.println("Este mes tiene 31 días");
        break;
    case 4: case 6: case 9: case 11:
        System.out.println("Este mes tiene 30 días");
        break;
    case 2:
        System.out.println("Este mes tiene 28 o 29 días");
        break;
    default:
        System.out.println("Mes inválido");
}

// Switch con String
System.out.print("Introduce una operación (+, -, *, /): ");
String operacion = scanner.next();
double num1 = 10, num2 = 5;
double resultado = 0;

switch (operacion) {
    case "+":
        resultado = num1 + num2;
        break;
    case "-":
        resultado = num1 - num2;
        break;
    case "*":
        resultado = num1 * num2;
        break;
    case "/":
        if (num2 != 0) {
            resultado = num1 / num2;
        } else {
            System.out.println("División por cero!");
            break;
        }
        break;
```

```
            default:
                System.out.println("Operación no válida");
                scanner.close();
                return;
        }

        System.out.println("Resultado: " + resultado);

        // Switch con char
        System.out.print("Introduce una vocal: ");
        char vocal = scanner.next().charAt(0);

        switch (Character.toLowerCase(vocal)) {
            case 'a':
                System.out.println("Primera vocal");
                break;
            case 'e':
                System.out.println("Segunda vocal");
                break;
            case 'i':
                System.out.println("Tercera vocal");
                break;
            case 'o':
                System.out.println("Cuarta vocal");
                break;
            case 'u':
                System.out.println("Quinta vocal");
                break;
            default:
                System.out.println("No es una vocal");
        }

        scanner.close();
    }
}
```

Switch Expressions (Java 14+)

```
public class SwitchExpression {
    public static void main(String[] args) {
        // Nueva sintaxis de switch (Java 14+)
        int dia = 3;

        String nombreDia = switch (dia) {
            case 1 -> "Lunes";
            case 2 -> "Martes";
            case 3 -> "Miércoles";
            case 4 -> "Jueves";
            case 5 -> "Viernes";
            case 6, 7 -> "Fin de semana";
            default -> "Día inválido";
        };

        System.out.println("Día: " + nombreDia);

        // Con bloques de código
        int mes = 2;
        int año = 2024;

        int dias = switch (mes) {
            case 1, 3, 5, 7, 8, 10, 12 -> 31;
```

```
            case 4, 6, 9, 11 -> 30;
            case 2 -> {
                if (año % 4 == 0 && (año % 100 != 0 || año % 400 ==
0)) {
                    yield 29; // Año bisiesto
                } else {
                    yield 28;
                }
            }
            default -> 0;
        };

        System.out.println("Días en el mes " + mes + ": " + dias);
    }
}
```

2. Estructuras de Repetición

Las estructuras de repetición (bucles) permiten ejecutar un bloque de código múltiples veces.

2.1 Bucle while

El bucle `while` ejecuta el código mientras la condición sea verdadera. La condición se evalúa antes de cada iteración.

```
import java.util.Scanner;

public class BucleWhile {
    public static void main(String[] args) {
        // while básico - contador
        int contador = 1;
        System.out.println("Contando del 1 al 5:");

        while (contador <= 5) {
            System.out.println("Contador: " + contador);
            contador++;
        }

        // while con condición compleja
        Scanner scanner = new Scanner(System.in);
        int suma = 0;
        int numero;

        System.out.println("Introduce números (0 para terminar):");
        numero = scanner.nextInt();

        while (numero != 0) {
            suma += numero;
            System.out.println("Suma actual: " + suma);
            numero = scanner.nextInt();
        }

        System.out.println("Suma total: " + suma);

        // Validación de entrada con while
        int edad = -1;
```

```
        while (edad < 0 || edad > 120) {
            System.out.print("Introduce una edad válida (0-120): ");
            edad = scanner.nextInt();

            if (edad < 0 || edad > 120) {
                System.out.println("Edad inválida. Inténtalo de
nuevo.");
            }
        }

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

        // Ejemplo práctico: Adivinar número
        int numeroSecreto = 42;
        int intento;
        int intentos = 0;

        System.out.println("¡Adivina el número (1-100)!");

        intento = scanner.nextInt();
        intentos++;

        while (intento != numeroSecreto) {
            if (intento < numeroSecreto) {
                System.out.println("Demasiado bajo. Intenta de
nuevo:");
            } else {
                System.out.println("Demasiado alto. Intenta de
nuevo:");
            }
            intento = scanner.nextInt();
            intentos++;
        }

        System.out.println("¡Correcto! Lo adivinaste en " + intentos +
" intentos");

        scanner.close();
    }
}
```

2.2 Bucle do-while

El bucle `do-while` ejecuta el código al menos una vez y luego repite mientras la condición sea verdadera.

```
import java.util.Scanner;

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

        // do-while básico
        int i = 1;
        System.out.println("Números del 1 al 5:");

        do {
            System.out.println(i);
            i++;
        } while (i <= 5);
```

```
        // Menú con do-while
        int opcion;

        do {
            System.out.println("\n=== MENÚ PRINCIPAL ===");
            System.out.println("1. Saludar");
            System.out.println("2. Mostrar fecha");
            System.out.println("3. Contar hasta 10");
            System.out.println("0. Salir");
            System.out.print("Elige una opción: ");

            opcion = scanner.nextInt();

            switch (opcion) {
                case 1:
                    System.out.println("¡Hola! ¿Cómo estás?");
                    break;
                case 2:
                    System.out.println("Hoy es un buen día para
programar");
                    break;
                case 3:
                    for (int j = 1; j <= 10; j++) {
                        System.out.print(j + " ");
                    }
                    System.out.println();
                    break;
                case 0:
                    System.out.println("¡Hasta luego!");
                    break;
                default:
                    System.out.println("Opción no válida");
            }
        } while (opcion != 0);

        // Validación con do-while
        String password;
        final String PASSWORD_CORRECTA = "java2024";
        int intentos = 0;
        final int MAX_INTENTOS = 3;

        do {
            System.out.print("Introduce la contraseña: ");
            password = scanner.next();
            intentos++;

            if (!password.equals(PASSWORD_CORRECTA) && intentos <
MAX_INTENTOS) {
                System.out.println("Contraseña incorrecta. Te quedan "
+
                                   (MAX_INTENTOS - intentos) + "
intentos");
            }
        } while (!password.equals(PASSWORD_CORRECTA) && intentos <
MAX_INTENTOS);

        if (password.equals(PASSWORD_CORRECTA)) {
            System.out.println("Acceso concedido");
        } else {
```

```
            System.out.println("Acceso denegado. Máximo de intentos
alcanzado");
        }

        // Ejemplo práctico: Calculadora con repetición
        char continuar;

        do {
            System.out.print("\nPrimer número: ");
            double num1 = scanner.nextDouble();
            System.out.print("Segundo número: ");
            double num2 = scanner.nextDouble();

            System.out.println("Suma: " + (num1 + num2));

            System.out.print("¿Deseas hacer otro cálculo? (s/n): ");
            continuar = scanner.next().charAt(0);
        } while (continuar == 's' || continuar == 'S');

        scanner.close();
    }
}
```

2.3 Bucle for

El bucle `for` es ideal cuando se conoce el número de iteraciones. Combina inicialización, condición y actualización.

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

        // for descendente
        System.out.println("\nCuenta atrás:");
        for (int i = 10; i >= 0; i--) {
            System.out.println(i);
        }

        // for con incremento diferente
        System.out.println("\nNúmeros pares del 0 al 20:");
        for (int i = 0; i <= 20; i += 2) {
            System.out.print(i + " ");
        }
        System.out.println();

        // for con múltiples variables
        System.out.println("\nDos contadores:");
        for (int i = 0, j = 10; i <= 10; i++, j--) {
            System.out.println("i = " + i + ", j = " + j);
        }

        // for anidados - Tabla de multiplicar
        System.out.println("\nTabla de multiplicar del 1 al 5:");
        for (int i = 1; i <= 5; i++) {
            System.out.println("\nTabla del " + i + ":");
```

```
            for (int j = 1; j <= 10; j++) {
                System.out.printf("%d x %d = %d\n", i, j, i * j);
            }
        }

        // for con arrays
        int[] numeros = {10, 20, 30, 40, 50};

        System.out.println("\nRecorriendo array con for
tradicional:");
        for (int i = 0; i < numeros.length; i++) {
            System.out.println("Índice " + i + ": " + numeros[i]);
        }

        // for-each (enhanced for loop)
        System.out.println("\nRecorriendo array con for-each:");
        for (int numero : numeros) {
            System.out.println("Valor: " + numero);
        }

        // Ejemplo práctico: Factorial
        int n = 5;
        long factorial = 1;

        for (int i = 1; i <= n; i++) {
            factorial *= i;
        }

        System.out.println("\nFactorial de " + n + " = " + factorial);

        // Ejemplo: Números primos
        System.out.println("\nNúmeros primos del 2 al 50:");
        for (int num = 2; num <= 50; num++) {
            boolean esPrimo = true;

            for (int divisor = 2; divisor <= Math.sqrt(num);
divisor++) {
                if (num % divisor == 0) {
                    esPrimo = false;
                    break;
                }
            }

            if (esPrimo) {
                System.out.print(num + " ");
            }
        }
        System.out.println();
    }
}
```

For-each con Colecciones

```
import java.util.ArrayList;
import java.util.List;

public class ForEach {
    public static void main(String[] args) {
        // Con arrays
        String[] dias = {"Lunes", "Martes", "Miércoles", "Jueves",
"Viernes"};
```

```
        System.out.println("Días laborables:");
        for (String dia : dias) {
            System.out.println("- " + dia);
        }

        // Con ArrayList
        List<String> frutas = new ArrayList<>();
        frutas.add("Manzana");
        frutas.add("Naranja");
        frutas.add("Plátano");
        frutas.add("Fresa");

        System.out.println("\nFrutas disponibles:");
        for (String fruta : frutas) {
            System.out.println("• " + fruta);
        }

        // Con array bidimensional
        int[][] matriz = {
            {1, 2, 3},
            {4, 5, 6},
            {7, 8, 9}
        };

        System.out.println("\nMatriz:");
        for (int[] fila : matriz) {
            for (int elemento : fila) {
                System.out.print(elemento + " ");
            }
            System.out.println();
        }
    }
}
```

3. Sentencias de Salto

3.1 Sentencia break

La sentencia `break` termina la ejecución del bucle o switch más interno.

```
import java.util.Scanner;

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

        // break en bucle for
        System.out.println("Buscando el primer múltiplo de 7:");
        for (int i = 1; i <= 100; i++) {
            if (i % 7 == 0) {
                System.out.println("Encontrado: " + i);
                break; // Sale del bucle
            }
        }

        // break en while
        System.out.println("\nIntroduce números (999 para salir):");
        while (true) { // Bucle infinito
```

```
            int numero = scanner.nextInt();
            if (numero == 999) {
                System.out.println("Saliendo del bucle...");
                break;
            }
            System.out.println("Has introducido: " + numero);
        }

        // break en bucles anidados
        System.out.println("\nBuscando en matriz:");
        int[][] matriz = {
            {1, 2, 3},
            {4, 5, 6},
            {7, 8, 9}
        };

        int buscar = 5;
        boolean encontrado = false;

        for (int i = 0; i < matriz.length; i++) {
            for (int j = 0; j < matriz[i].length; j++) {
                System.out.println("Verificando posición [" + i + "]["
+ j + "]");
                if (matriz[i][j] == buscar) {
                    System.out.println("¡Encontrado " + buscar + " en
[" + i + "][" + j + "]!");
                    encontrado = true;
                    break; // Solo sale del bucle interno
                }
            }
            if (encontrado) {
                break; // Sale del bucle externo
            }
        }

        // break con etiquetas (labels)
        System.out.println("\nBreak con etiquetas:");

        busqueda: // Etiqueta
        for (int i = 0; i < 3; i++) {
            System.out.println("Bucle externo: " + i);
            for (int j = 0; j < 3; j++) {
                System.out.println("  Bucle interno: " + j);
                if (i == 1 && j == 1) {
                    System.out.println("  ¡Saliendo de ambos
bucles!");
                    break busqueda; // Sale de ambos bucles
                }
            }
        }

        // Ejemplo práctico: Validación de entrada
        System.out.println("\nValidación de edad:");
        while (true) {
            System.out.print("Introduce tu edad (0-120): ");
            int edad = scanner.nextInt();

            if (edad >= 0 && edad <= 120) {
                System.out.println("Edad válida: " + edad);
                break;
            }
```

```
            System.out.println("Edad inválida. Inténtalo de nuevo.");
        }

        scanner.close();
    }
}
```

3.2 Sentencia continue

La sentencia `continue` salta a la siguiente iteración del bucle.

```
public class SentenciaContinue {
    public static void main(String[] args) {
        // continue en for - Números impares
        System.out.println("Números impares del 1 al 10:");
        for (int i = 1; i <= 10; i++) {
            if (i % 2 == 0) {
                continue; // Salta los números pares
            }
            System.out.print(i + " ");
        }
        System.out.println();

        // continue en while
        System.out.println("\nNúmeros no divisibles por 3:");
        int contador = 0;
        while (contador < 15) {
            contador++;
            if (contador % 3 == 0) {
                continue; // Salta los múltiplos de 3
            }
            System.out.print(contador + " ");
        }
        System.out.println();

        // continue con múltiples condiciones
        System.out.println("\nProcesando valores:");
        for (int i = -5; i <= 10; i++) {
            if (i == 0) {
                System.out.println("Saltando división por cero");
                continue;
            }
            if (i < 0) {
                System.out.println("Saltando número negativo: " + i);
                continue;
            }
            double resultado = 100.0 / i;
            System.out.printf("100 / %d = %.2f\n", i, resultado);
        }

        // continue en bucles anidados
        System.out.println("\nTabla de multiplicar (saltando
diagonal):");
        for (int i = 1; i <= 5; i++) {
            for (int j = 1; j <= 5; j++) {
                if (i == j) {
                    System.out.print("   X");
                    continue; // Salta cuando i == j
                }
                System.out.printf("%4d", i * j);
```

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

        // continue con etiquetas
        System.out.println("\nContinue con etiquetas:");

        externo:
        for (int i = 0; i < 3; i++) {
            System.out.println("Bucle externo: " + i);
            for (int j = 0; j < 3; j++) {
                if (j == 1) {
                    System.out.println("  Saltando al siguiente ciclo
externo");
                    continue externo; // Salta al siguiente ciclo del
bucle externo
                }
                System.out.println("  Bucle interno: " + j);
            }
        }

        // Ejemplo práctico: Suma de números positivos
        int[] numeros = {10, -5, 20, -3, 15, 0, -8, 25};
        int suma = 0;

        System.out.println("\nSumando solo números positivos:");
        for (int num : numeros) {
            if (num <= 0) {
                System.out.println("Saltando: " + num);
                continue;
            }
            suma += num;
            System.out.println("Sumando: " + num + " (Total: " + suma
+ ")");
        }
        System.out.println("Suma final: " + suma);
    }
}
```

4. Manejo de Excepciones

Las excepciones son eventos que ocurren durante la ejecución y alteran el flujo normal del programa.

4.1 Captura de Excepciones (try-catch)

```
import java.util.Scanner;
import java.util.InputMismatchException;

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

        // try-catch básico
        try {
            int resultado = 10 / 0;
            System.out.println("Resultado: " + resultado);
        } catch (ArithmeticException e) {
            System.out.println("Error: División por cero");
```

```
            System.out.println("Mensaje: " + e.getMessage());
        }

        // try-catch con múltiples excepciones
        try {
            System.out.print("Introduce un número: ");
            int numero = scanner.nextInt();

            int[] array = {1, 2, 3};
            System.out.print("Introduce un índice (0-2): ");
            int indice = scanner.nextInt();

            int resultado = array[indice] / numero;
            System.out.println("Resultado: " + resultado);

        } catch (InputMismatchException e) {
            System.out.println("Error: Debes introducir un número
entero");
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Error: Índice fuera de rango");
        } catch (ArithmeticException e) {
            System.out.println("Error: División por cero");
        }

        // try-catch-finally
        try {
            System.out.println("\nAbriendo archivo...");
            // Simulación de operación con archivo
            String texto = null;
            System.out.println(texto.length()); // Provocará
NullPointerException

        } catch (NullPointerException e) {
            System.out.println("Error: Referencia nula");
        } finally {
            System.out.println("Cerrando archivo... (finally siempre
se ejecuta)");
        }

        // Captura de múltiples excepciones en un solo catch (Java 7+)
        try {
            System.out.print("\nIntroduce un número para dividir 100:
");
            scanner.nextLine(); // Limpiar buffer
            String entrada = scanner.nextLine();
            int divisor = Integer.parseInt(entrada);
            int resultado = 100 / divisor;
            System.out.println("Resultado: " + resultado);

        } catch (NumberFormatException | ArithmeticException e) {
            System.out.println("Error: Entrada inválida o división por
cero");
            System.out.println("Tipo de error: " +
e.getClass().getSimpleName());
        }

        // Captura genérica con Exception
        try {
            // Código que puede lanzar cualquier excepción
            realizarOperacionPeligrosa();
```

```
        } catch (Exception e) {
            System.out.println("Error general: " + e.getMessage());
            e.printStackTrace(); // Imprime la traza completa del
error
        }

        scanner.close();
    }

    private static void realizarOperacionPeligrosa() {
        throw new RuntimeException("Algo salió mal");
    }
}
```

4.2 Delegación de Excepciones (throws)

```
import java.io.*;
import java.util.Scanner;

public class DelegacionExcepciones {

    // Método que declara que puede lanzar excepciones
    public static int dividir(int a, int b) throws ArithmeticException
{
        if (b == 0) {
            throw new ArithmeticException("División por cero no
permitida");
        }
        return a / b;
    }

    // Método que propaga la excepción
    public static void leerArchivo(String nombreArchivo) throws
FileNotFoundException, IOException {
        FileReader archivo = new FileReader(nombreArchivo);
        BufferedReader lector = new BufferedReader(archivo);
        String linea = lector.readLine();
        System.out.println("Primera línea: " + linea);
        lector.close();
    }

    // Método que puede lanzar múltiples excepciones
    public static double calcularRaizCuadrada(double numero) throws
IllegalArgumentException {
        if (numero < 0) {
            throw new IllegalArgumentException("No se puede calcular
la raíz de un número negativo");
        }
        return Math.sqrt(numero);
    }

    // Cadena de delegación
    public static void metodoA() throws Exception {
        metodoB();
    }

    public static void metodoB() throws Exception {
        metodoC();
    }
```

```
    public static void metodoC() throws Exception {
        throw new Exception("Excepción originada en método C");
    }

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

        // Uso de método con throws
        try {
            System.out.print("Introduce dos números para dividir: ");
            int num1 = scanner.nextInt();
            int num2 = scanner.nextInt();

            int resultado = dividir(num1, num2);
            System.out.println("Resultado: " + resultado);

        } catch (ArithmeticException e) {
            System.out.println("Error en división: " +
e.getMessage());
        }

        // Manejo de IOException
        try {
            leerArchivo("archivo.txt");
        } catch (FileNotFoundException e) {
            System.out.println("Error: Archivo no encontrado");
        } catch (IOException e) {
            System.out.println("Error de E/S: " + e.getMessage());
        }

        // Manejo de IllegalArgumentException
        try {
            System.out.print("\nIntroduce un número para calcular su
raíz cuadrada: ");
            double numero = scanner.nextDouble();
            double raiz = calcularRaizCuadrada(numero);
            System.out.printf("La raíz cuadrada de %.2f es %.2f\n",
numero, raiz);

        } catch (IllegalArgumentException e) {
            System.out.println("Error: " + e.getMessage());
        }

        // Cadena de delegación
        try {
            metodoA();
        } catch (Exception e) {
            System.out.println("Excepción capturada en main: " +
e.getMessage());
        }

        scanner.close();
    }
}
```

4.3 Excepciones de Usuario (Personalizadas)

```
// Definición de excepciones personalizadas
class EdadInvalidaException extends Exception {
    public EdadInvalidaException(String mensaje) {
```

```
        super(mensaje);
    }
}

class SaldoInsuficienteException extends Exception {
    private double saldoActual;
    private double cantidadSolicitada;

    public SaldoInsuficienteException(double saldoActual, double
cantidadSolicitada) {
        super("Saldo insuficiente. Saldo actual: " + saldoActual +
              ", Cantidad solicitada: " + cantidadSolicitada);
        this.saldoActual = saldoActual;
        this.cantidadSolicitada = cantidadSolicitada;
    }

    public double getSaldoActual() {
        return saldoActual;
    }

    public double getCantidadSolicitada() {
        return cantidadSolicitada;
    }
}

class PasswordDebilidadException extends Exception {
    private String razon;

    public PasswordDebilidadException(String razon) {
        super("Contraseña débil: " + razon);
        this.razon = razon;
    }

    public String getRazon() {
        return razon;
    }
}

// Clase que usa las excepciones personalizadas
class CuentaBancaria {
    private double saldo;
    private String titular;

    public CuentaBancaria(String titular, double saldoInicial) {
        this.titular = titular;
        this.saldo = saldoInicial;
    }

    public void retirar(double cantidad) throws
SaldoInsuficienteException {
        if (cantidad > saldo) {
            throw new SaldoInsuficienteException(saldo, cantidad);
        }
        saldo -= cantidad;
        System.out.println("Retiro exitoso. Nuevo saldo: " + saldo);
    }

    public double getSaldo() {
        return saldo;
    }
}
```

```
public class ExcepcionesPersonalizadas {

    // Método que valida edad
    public static void validarEdad(int edad) throws
EdadInvalidaException {
        if (edad < 0) {
            throw new EdadInvalidaException("La edad no puede ser
negativa");
        }
        if (edad > 150) {
            throw new EdadInvalidaException("La edad no puede ser
mayor a 150");
        }
        System.out.println("Edad válida: " + edad);
    }

    // Método que valida contraseña
    public static void validarPassword(String password) throws
PasswordDebilidadException {
        if (password.length() < 8) {
            throw new PasswordDebilidadException("Debe tener al menos
8 caracteres");
        }
        if (!password.matches(".*[0-9].*")) {
            throw new PasswordDebilidadException("Debe contener al
menos un número");
        }
        if (!password.matches(".*[A-Z].*")) {
            throw new PasswordDebilidadException("Debe contener al
menos una mayúscula");
        }
        if (!password.matches(".*[!@#$%^&*()].*")) {
            throw new PasswordDebilidadException("Debe contener al
menos un carácter especial");
        }
        System.out.println("Contraseña fuerte");
    }

    public static void main(String[] args) {
        // Uso de EdadInvalidaException
        try {
            validarEdad(25);
            validarEdad(-5);
        } catch (EdadInvalidaException e) {
            System.out.println("Error de validación: " +
e.getMessage());
        }

        // Uso de SaldoInsuficienteException
        CuentaBancaria cuenta = new CuentaBancaria("Juan Pérez",
1000);

        try {
            cuenta.retirar(500);
            cuenta.retirar(600);
        } catch (SaldoInsuficienteException e) {
            System.out.println(e.getMessage());
            System.out.println("Déficit: " +
                (e.getCantidadSolicitada() - e.getSaldoActual()));
        }
```

```
        // Uso de PasswordDebilidadException
        String[] passwords = {"abc", "abcdefgh", "Abcdefgh1",
"Abcdefgh1!"};

        for (String pwd : passwords) {
            System.out.println("\nValidando: " + pwd);
            try {
                validarPassword(pwd);
            } catch (PasswordDebilidadException e) {
                System.out.println("Contraseña rechazada: " +
e.getRazon());
            }
        }
    }
}
```

4.4 Lanzamiento y Redefinición de Excepciones

```
import java.io.*;
import java.util.logging.Logger;

class ServicioException extends Exception {
    public ServicioException(String mensaje, Throwable causa) {
        super(mensaje, causa);
    }
}

class BaseDeDatosException extends Exception {
    public BaseDeDatosException(String mensaje) {
        super(mensaje);
    }
}

public class LanzamientoRedefinicion {
    private static final Logger logger =
Logger.getLogger(LanzamientoRedefinicion.class.getName());

    // Lanzamiento simple
    public static void verificarNumero(int numero) {
        if (numero < 0) {
            throw new IllegalArgumentException("El número no puede ser
negativo: " + numero);
        }
        System.out.println("Número válido: " + numero);
    }

    // Re-lanzamiento de excepción
    public static void procesarArchivo(String nombre) throws
IOException {
        try {
            FileReader archivo = new FileReader(nombre);
            // Procesar archivo...
            archivo.close();
        } catch (IOException e) {
            logger.severe("Error al procesar archivo: " + nombre);
            throw e; // Re-lanzar la excepción
        }
    }
```

```
    // Encapsulación de excepciones
    public static void operacionBaseDatos() throws ServicioException {
        try {
            // Simular operación de base de datos
            throw new SQLException("Error de conexión a la base de
datos");
        } catch (SQLException e) {
            // Encapsular la excepción original en una personalizada
            throw new ServicioException("Error en el servicio de
datos", e);
        }
    }

    // Redefinición con más información
    public static double calcularDescuento(double precio, double
porcentaje)
            throws IllegalArgumentException {
        try {
            if (precio < 0) {
                throw new IllegalArgumentException("Precio negativo");
            }
            if (porcentaje < 0 || porcentaje > 100) {
                throw new IllegalArgumentException("Porcentaje
inválido");
            }
            return precio * (porcentaje / 100);

        } catch (IllegalArgumentException e) {
            // Redefinir con más contexto
            throw new IllegalArgumentException(
                String.format("Error calculando descuento:
precio=%.2f, porcentaje=%.2f. %s",
                        precio, porcentaje, e.getMessage())
            );
        }
    }

    // Conversión de excepciones checked a unchecked
    public static String leerConfiguracion(String archivo) {
        try {
            BufferedReader reader = new BufferedReader(new
FileReader(archivo));
            String config = reader.readLine();
            reader.close();
            return config;
        } catch (IOException e) {
            // Convertir checked exception a unchecked
            throw new RuntimeException("No se pudo leer la
configuración", e);
        }
    }

    // Try-with-resources (Java 7+)
    public static void leerArchivoConRecursos(String nombre) {
        try (BufferedReader reader = new BufferedReader(new
FileReader(nombre))) {
            String linea;
            while ((linea = reader.readLine()) != null) {
                System.out.println(linea);
            }
            // No es necesario cerrar explícitamente
```

```
        } catch (IOException e) {
            System.err.println("Error leyendo archivo: " +
e.getMessage());
        }
    }

    public static void main(String[] args) {
        // Ejemplo de lanzamiento simple
        try {
            verificarNumero(10);
            verificarNumero(-5);
        } catch (IllegalArgumentException e) {
            System.out.println("Excepción capturada: " +
e.getMessage());
        }

        // Ejemplo de re-lanzamiento
        try {
            procesarArchivo("archivo_inexistente.txt");
        } catch (IOException e) {
            System.out.println("Error en main: " + e.getMessage());
        }

        // Ejemplo de encapsulación
        try {
            operacionBaseDatos();
        } catch (ServicioException e) {
            System.out.println("Error de servicio: " +
e.getMessage());
            System.out.println("Causa original: " +
e.getCause().getMessage());
        }

        // Ejemplo de redefinición con contexto
        try {
            double descuento = calcularDescuento(100, 150);
        } catch (IllegalArgumentException e) {
            System.out.println("Error: " + e.getMessage());
        }

        // Ejemplo de conversión a unchecked
        try {
            String config =
leerConfiguracion("config_inexistente.ini");
        } catch (RuntimeException e) {
            System.out.println("Error runtime: " + e.getMessage());
        }
    }
}

// Para simular SQLException
class SQLException extends Exception {
    public SQLException(String mensaje) {
        super(mensaje);
    }
}
```

Ejemplo Integrador: Sistema de Gestión de Estudiantes

```
import java.util.Scanner;
import java.util.ArrayList;
import java.util.List;

// Excepciones personalizadas
class NotaInvalidaException extends Exception {
    public NotaInvalidaException(double nota) {
        super("Nota inválida: " + nota + ". Debe estar entre 0 y 10");
    }
}

class EstudianteNoEncontradoException extends Exception {
    public EstudianteNoEncontradoException(String id) {
        super("Estudiante con ID " + id + " no encontrado");
    }
}

// Clase Estudiante
class Estudiante {
    private String id;
    private String nombre;
    private List<Double> notas;

    public Estudiante(String id, String nombre) {
        this.id = id;
        this.nombre = nombre;
        this.notas = new ArrayList<>();
    }

    public void agregarNota(double nota) throws NotaInvalidaException
{
        if (nota < 0 || nota > 10) {
            throw new NotaInvalidaException(nota);
        }
        notas.add(nota);
    }

    public double calcularPromedio() {
        if (notas.isEmpty()) return 0;

        double suma = 0;
        for (double nota : notas) {
            suma += nota;
        }
        return suma / notas.size();
    }

    public String getId() { return id; }
    public String getNombre() { return nombre; }
    public List<Double> getNotas() { return notas; }
}

// Sistema principal
public class SistemaGestionEstudiantes {
    private static List<Estudiante> estudiantes = new ArrayList<>();
    private static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        int opcion;

        do {
```

```
            mostrarMenu();
            opcion = leerOpcion();

            switch (opcion) {
                case 1:
                    registrarEstudiante();
                    break;
                case 2:
                    agregarNotas();
                    break;
                case 3:
                    mostrarEstudiantes();
                    break;
                case 4:
                    buscarEstudiante();
                    break;
                case 5:
                    generarReporte();
                    break;
                case 0:
                    System.out.println("¡Hasta luego!");
                    break;
                default:
                    System.out.println("Opción inválida");
            }
        } while (opcion != 0);

        scanner.close();
    }

    private static void mostrarMenu() {
        System.out.println("\n=== SISTEMA DE GESTIÓN DE ESTUDIANTES
===");
        System.out.println("1. Registrar estudiante");
        System.out.println("2. Agregar notas");
        System.out.println("3. Mostrar todos los estudiantes");
        System.out.println("4. Buscar estudiante");
        System.out.println("5. Generar reporte de
aprobados/suspendidos");
        System.out.println("0. Salir");
        System.out.print("Elige una opción: ");
    }

    private static int leerOpcion() {
        while (true) {
            try {
                return scanner.nextInt();
            } catch (Exception e) {
                System.out.print("Por favor, introduce un número: ");
                scanner.nextLine(); // Limpiar buffer
            }
        }
    }

    private static void registrarEstudiante() {
        scanner.nextLine(); // Limpiar buffer

        System.out.print("ID del estudiante: ");
        String id = scanner.nextLine();

        // Verificar si ya existe
```

```
        for (Estudiante e : estudiantes) {
            if (e.getId().equals(id)) {
                System.out.println("Error: Ya existe un estudiante con
ese ID");
                return;
            }
        }

        System.out.print("Nombre del estudiante: ");
        String nombre = scanner.nextLine();

        estudiantes.add(new Estudiante(id, nombre));
        System.out.println("Estudiante registrado exitosamente");
    }

    private static void agregarNotas() {
        scanner.nextLine(); // Limpiar buffer

        System.out.print("ID del estudiante: ");
        String id = scanner.nextLine();

        try {
            Estudiante estudiante = buscarEstudiantePorId(id);

            System.out.print("¿Cuántas notas deseas agregar? ");
            int cantidad = scanner.nextInt();

            for (int i = 1; i <= cantidad; i++) {
                while (true) {
                    try {
                        System.out.print("Nota " + i + ": ");
                        double nota = scanner.nextDouble();
                        estudiante.agregarNota(nota);
                        break;
                    } catch (NotaInvalidaException e) {
                        System.out.println(e.getMessage());
                    }
                }
            }

            System.out.println("Notas agregadas exitosamente");

        } catch (EstudianteNoEncontradoException e) {
            System.out.println(e.getMessage());
        }
    }

    private static void mostrarEstudiantes() {
        if (estudiantes.isEmpty()) {
            System.out.println("No hay estudiantes registrados");
            return;
        }

        System.out.println("\n=== LISTA DE ESTUDIANTES ===");
        for (Estudiante e : estudiantes) {
            System.out.println("\nID: " + e.getId());
            System.out.println("Nombre: " + e.getNombre());

            if (e.getNotas().isEmpty()) {
                System.out.println("Notas: Sin notas registradas");
            } else {
```

```
                System.out.print("Notas: ");
                for (double nota : e.getNotas()) {
                    System.out.print(nota + " ");
                }
                System.out.printf("\nPromedio: %.2f\n",
e.calcularPromedio());
            }
        }
    }

    private static void buscarEstudiante() {
        scanner.nextLine(); // Limpiar buffer

        System.out.print("ID del estudiante: ");
        String id = scanner.nextLine();

        try {
            Estudiante estudiante = buscarEstudiantePorId(id);

            System.out.println("\n=== INFORMACIÓN DEL ESTUDIANTE
===");
            System.out.println("ID: " + estudiante.getId());
            System.out.println("Nombre: " + estudiante.getNombre());

            if (estudiante.getNotas().isEmpty()) {
                System.out.println("Sin notas registradas");
            } else {
                double promedio = estudiante.calcularPromedio();
                System.out.printf("Promedio: %.2f\n", promedio);

                if (promedio >= 5.0) {
                    System.out.println("Estado: APROBADO");
                } else {
                    System.out.println("Estado: SUSPENDIDO");
                }
            }

        } catch (EstudianteNoEncontradoException e) {
            System.out.println(e.getMessage());
        }
    }

    private static void generarReporte() {
        if (estudiantes.isEmpty()) {
            System.out.println("No hay estudiantes registrados");
            return;
        }

        int aprobados = 0;
        int suspendidos = 0;
        int sinNotas = 0;

        System.out.println("\n=== REPORTE DE ESTUDIANTES ===");

        for (Estudiante e : estudiantes) {
            if (e.getNotas().isEmpty()) {
                sinNotas++;
                continue;
            }

            double promedio = e.calcularPromedio();
```

```
            if (promedio >= 5.0) {
                aprobados++;
                System.out.printf("✓ %s - %.2f (APROBADO)\n",
e.getNombre(), promedio);
            } else {
                suspendidos++;
                System.out.printf("✗ %s - %.2f (SUSPENDIDO)\n",
e.getNombre(), promedio);
            }
        }

        System.out.println("\n=== RESUMEN ===");
        System.out.println("Total estudiantes: " +
estudiantes.size());
        System.out.println("Aprobados: " + aprobados);
        System.out.println("Suspendidos: " + suspendidos);
        System.out.println("Sin evaluar: " + sinNotas);

        if (aprobados + suspendidos > 0) {
            double porcentajeAprobados = (aprobados * 100.0) /
(aprobados + suspendidos);
            System.out.printf("Porcentaje de aprobación: %.1f%%\n",
porcentajeAprobados);
        }
    }

    private static Estudiante buscarEstudiantePorId(String id)
            throws EstudianteNoEncontradoException {
        for (Estudiante e : estudiantes) {
            if (e.getId().equals(id)) {
                return e;
            }
        }
        throw new EstudianteNoEncontradoException(id);
    }
}
```

Ejercicios Propuestos

Ejercicio 1: Calculadora Avanzada

Crea una calculadora que maneje operaciones básicas y avanzadas (potencia, raíz). Debe incluir un menú con switch, validación de entrada con excepciones personalizadas, y la opción de continuar o salir usando do-while.

Ejercicio 2: Validador de Contraseñas

Desarrolla un sistema que valide contraseñas según diferentes niveles de seguridad. Usa excepciones personalizadas para cada tipo de fallo, estructuras de control para verificar requisitos, y permite reintentos con un límite usando bucles.

Ejercicio 3: Juego de Adivinanza

Implementa un juego donde el usuario debe adivinar un número. Incluye niveles de dificultad (switch), límite de intentos (for), pistas después de cada intento (if-else), y manejo de excepciones para entradas inválidas.

Ejercicio 4: Sistema de Inventario

Crea un sistema de inventario con opciones para agregar, eliminar y buscar productos. Usa un menú con do-while, validación con excepciones personalizadas, y permite operaciones masivas con break y continue.

Ejercicio 5: Analizador de Texto

Desarrolla un programa que analice un texto: cuenta palabras, busca palabras específicas, calcula estadísticas. Usa for-each para recorrer palabras, excepciones para manejar archivos, y estructuras de control para el análisis.

Resumen de Buenas Prácticas

1. **Estructuras de selección**: Usa if-else para condiciones simples, switch para múltiples opciones fijas
2. **Bucles**: Elige while para condiciones, do-while cuando necesites al menos una iteración, for cuando conozcas las iteraciones
3. **Break y Continue**: Úsalos con moderación para mantener el código legible
4. **Excepciones**: Captura excepciones específicas antes que las generales
5. **Try-with-resources**: Úsalo para gestionar recursos automáticamente
6. **Excepciones personalizadas**: Crea las tuyas cuando necesites información específica del dominio
7. **Finally**: Úsalo para código de limpieza que debe ejecutarse siempre
8. **Validación**: Siempre valida entradas del usuario con manejo apropiado de excepciones

Este tema establece las bases del control de flujo en Java, esencial para crear programas interactivos y robustos.