

# Sesión Sincrónica Nº 5

Fundamentos de Programación (PRY2201)


Profesor: Miguel Puebla

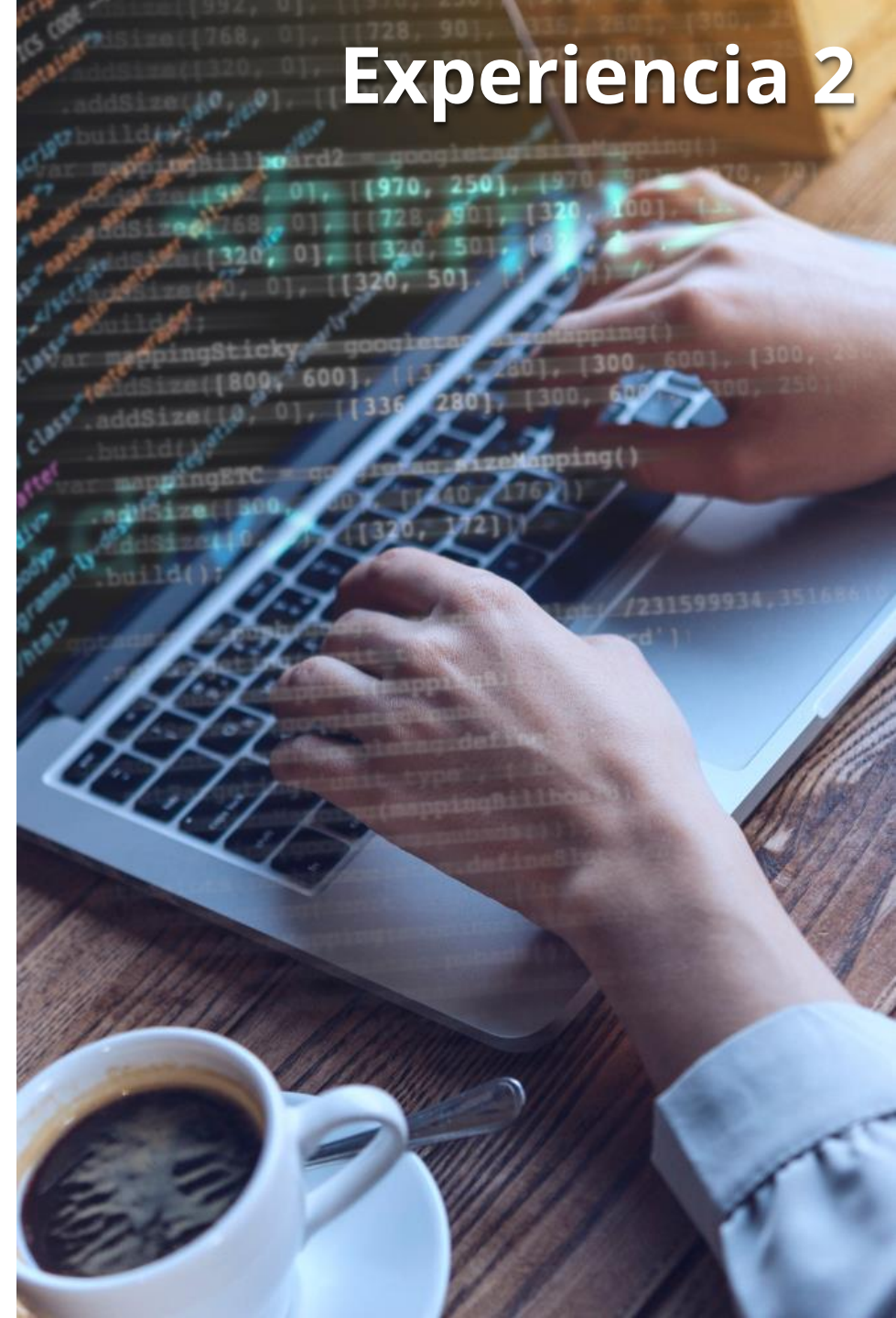
# Experiencia 2

## Experiencia 2: Explorando mi Software de Programación en Java

**RA2.** Aplica elementos básicos de Java para el desarrollo de programas que solucionen problemáticas planteadas.

### Semana 5: Variables en Java

-  **IL5.** Emplea Variables en Java en la resolución de problemáticas planteadas, asegurando el correcto almacenamiento y manipulación de datos..





# Semana 5



## Conocimientos Generales

Optimización del Código	Declaración de una variable	Inicialización de una variable
Tipos de Datos Básicos	Concatenación de Cadenas	Validación de Resultado



# Optimización del Código

Es el proceso de mejorar la eficiencia y el rendimiento del código, reduciendo su tiempo de ejecución y consumo de recursos como memoria y procesador.

## Código sin optimizar

```
import java.util.Scanner;

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

        System.out.print("Ingrese el capital inicial: ");
        double capital = input.nextDouble();

        System.out.print("Ingrese la tasa de interés anual (en porcentaje): ");
        double tasaInteres = input.nextDouble();

        System.out.print("Ingrese el número de años: ");
        int años = input.nextInt();

        System.out.print("Ingrese la cantidad de veces que se capitaliza por año: ");
        int capitalizacion = input.nextInt();

        double tasaDecimal = tasaInteres / 100;
        double monto = capital * Math.pow(1 + (tasaDecimal / capitalizacion), capitalizacion * años);

        System.out.printf("El monto total después de %d años es: %.2f%n", años, monto);

        input.close();
    }
}
```

# Optimización del Código

## Código optimizado

```
import java.util.Scanner;

public class InteresCompuestoOptimizado {

    public static double calcularInteresCompuesto(double capital, double tasaInteres, int anios, int capitalizacion) {
        final double tasaDecimal = tasaInteres / 100; // Convierte la tasa de porcentaje a decimal
        return capital * Math.pow(1 + (tasaDecimal / capitalizacion), capitalizacion * anios);
    }

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

        // Entradas de datos
        System.out.print("Ingrese el capital inicial: ");
        double capital = input.nextDouble();

        System.out.print("Ingrese la tasa de interés anual (en porcentaje): ");
        double tasaInteres = input.nextDouble();

        System.out.print("Ingrese el número de años: ");
        int anios = input.nextInt();

        System.out.print("Ingrese la cantidad de veces que se capitaliza por año: ");
        int capitalizacion = input.nextInt();

        // Cálculo del interés compuesto con método separado
        double monto = calcularInteresCompuesto(capital, tasaInteres, anios, capitalizacion);

        System.out.printf("El monto total después de %d años es: %.2f%n", anios, monto);

        input.close();
    }
}
```



# Optimización del Código



## Optimización aplicada:

1. **Uso de funciones:** El cálculo del interés compuesto se coloca en una función aparte, `calcularInteresCompuesto`, lo que mejora la legibilidad y modularidad del código.
2. **Eliminación de redundancias:** Se evita recalcular la tasa de interés en varias partes. Ahora se hace solo una vez en el método `calcularInteresCompuesto`.
3. **Uso de `final` para constantes:** La tasa de interés convertida a decimal se marca como `final`, asegurando que no se modifique accidentalmente.
4. **Separación de responsabilidades:** El método `main` se encarga solo de la entrada/salida, mientras que el cálculo está encapsulado en una función, facilitando el mantenimiento y la ampliación.

## Comparativa

Característica	Ejemplo sin optimizar	Ejemplo optimizado
Modularidad	Todo el código está en el <code>main</code>	Uso de función separada para el cálculo
Reutilización de cálculos	Recalcula tasas en diferentes lugares	Tasa de interés calculada una vez
Manejo de constantes	Variables innecesariamente recalculadas	Uso de <code>final</code> para tasas y cálculos
Legibilidad	Código menos legible	Mejor legibilidad y separación de lógica
Mantenimiento	Difícil de mantener y ampliar	Fácil de mantener y añadir funcionalidades

# Declaración de una variable

Es el acto de definir una variable en un programa, especificando su nombre y tipo de dato, reservando así un espacio en memoria.

Cuadro Comparativo General de Tipos de Variables

Tipo de Variable	Definición	Ventajas	Desventajas
Globales	Variables declaradas fuera de cualquier función o método y accesibles en todo el programa.	<ul style="list-style-type: none"><li>- Accesibles desde cualquier parte del código.</li><li>- Facilitan compartir datos entre funciones.</li></ul>	<ul style="list-style-type: none"><li>- Pueden ser modificadas accidentalmente desde cualquier parte.</li><li>- Ocupan memoria durante toda la ejecución.</li></ul>
Locales	Variables declaradas dentro de una función o método y solo accesibles dentro de dicho bloque.	<ul style="list-style-type: none"><li>- Más seguras, ya que solo existen en su ámbito.</li><li>- Liberan memoria al salir del bloque donde se declaran.</li></ul>	<ul style="list-style-type: none"><li>- No pueden ser utilizadas fuera del ámbito en que fueron creadas.</li><li>- Pueden ser redundantes si se repiten.</li></ul>
Constantes	Variables cuyo valor no cambia después de ser inicializadas.	<ul style="list-style-type: none"><li>- Aumentan la seguridad al evitar cambios accidentales.</li><li>- Facilitan la lectura al indicar valores fijos.</li></ul>	<ul style="list-style-type: none"><li>- No pueden modificarse una vez definidas, lo que puede ser limitante en casos donde se requiere flexibilidad.</li></ul>
Estáticas	Variables asociadas a la clase en lugar de a instancias individuales de la clase.	<ul style="list-style-type: none"><li>- Compartidas entre todas las instancias de una clase.</li><li>- Útiles para mantener datos comunes o contadores.</li></ul>	<ul style="list-style-type: none"><li>- Su modificación afecta a todas las instancias de la clase.</li><li>- Pueden causar errores si no se manejan bien.</li></ul>



# Declaración de una variable

Es el acto de definir una variable en un programa, especificando su nombre y tipo de dato, reservando así un espacio en memoria.

Cuadro Comparativo General de Tipos de Variables

Tipo de Variable	Definición	Ventajas	Desventajas
Globales	Variables declaradas fuera de cualquier función o método y accesibles en todo el programa.	<ul style="list-style-type: none"><li>- Accesibles desde cualquier parte del código.</li><li>- Facilitan compartir datos entre funciones.</li></ul>	<ul style="list-style-type: none"><li>- Pueden ser modificadas accidentalmente desde cualquier parte.</li><li>- Ocupan memoria durante toda la ejecución.</li></ul>
Locales	Variables declaradas dentro de una función o método y solo accesibles dentro de dicho bloque.	<ul style="list-style-type: none"><li>- Más seguras, ya que solo existen en su ámbito.</li><li>- Liberan memoria al salir del bloque donde se declaran.</li></ul>	<ul style="list-style-type: none"><li>- No pueden ser utilizadas fuera del ámbito en que fueron creadas.</li><li>- Pueden ser redundantes si se repiten.</li></ul>
Constantes	Variables cuyo valor no cambia después de ser inicializadas.	<ul style="list-style-type: none"><li>- Aumentan la seguridad al evitar cambios accidentales.</li><li>- Facilitan la lectura al indicar valores fijos.</li></ul>	<ul style="list-style-type: none"><li>- No pueden modificarse una vez definidas, lo que puede ser limitante en casos donde se requiere flexibilidad.</li></ul>
Estáticas	Variables asociadas a la clase en lugar de a instancias individuales de la clase.	<ul style="list-style-type: none"><li>- Compartidas entre todas las instancias de una clase.</li><li>- Útiles para mantener datos comunes o contadores.</li></ul>	<ul style="list-style-type: none"><li>- Su modificación afecta a todas las instancias de la clase.</li><li>- Pueden causar errores si no se manejan bien.</li></ul>





# Declaración de una variable

Cuadro Comparativo Específico del Ejercicio

Variable	Tipo	Ámbito	Ventajas	Desventajas
Scanner input	Global	Clase	<ul style="list-style-type: none"><li>- Permite manejar la entrada del usuario desde cualquier función.</li><li>- Evita pasar el objeto <code>Scanner</code> como parámetro entre funciones.</li></ul>	<ul style="list-style-type: none"><li>- Está disponible durante toda la ejecución, ocupando memoria.</li><li>- Puede ser utilizada erróneamente en otros contextos.</li></ul>
double precioInicial	Global	Clase	<ul style="list-style-type: none"><li>- Compartida entre funciones.</li><li>- Evita tener que pasar el valor como parámetro constantemente.</li></ul>	<ul style="list-style-type: none"><li>- Puede ser modificada accidentalmente desde cualquier función.</li><li>- Puede hacer el programa menos modular.</li></ul>
double tasaInteres	Global	Clase	<ul style="list-style-type: none"><li>- Facilita el uso de la tasa de interés entre diferentes partes del programa.</li><li>- Menor repetición de código.</li></ul>	<ul style="list-style-type: none"><li>- Sus cambios son visibles y posibles desde cualquier parte del programa.</li><li>- Ocurre lo mismo que con <code>precioInicial</code>.</li></ul>
int aniosPlazo	Global	Clase	<ul style="list-style-type: none"><li>- Reduce la cantidad de variables locales.</li><li>- Facilita el uso y modificación del plazo entre funciones.</li></ul>	<ul style="list-style-type: none"><li>- Exposición a cambios accidentales.</li><li>- El ámbito global puede dificultar la detección de errores.</li></ul>



# Declaración de una variable

Cuadro Comparativo Específico del Ejercicio

Variable	Tipo	Ámbito	Ventajas	Desventajas
double precioFinal	Local	main	- Solo existe en el ámbito de la función <code>main</code> , lo que la hace más segura y clara.	- No es reutilizable fuera de la función. - Requiere ser retornada si se desea utilizar en otras funciones.
final double FACTOR	Constante Local	calcularPrecioFinal	- Garantiza que el factor de crecimiento no será modificado después de su inicialización. - Mejora la seguridad.	- Solo se puede usar en la función <code>calcularPrecioFinal</code> . - No es flexible para otros cálculos dinámicos.
double precio	Local	obtenerPrecioInicial	- Limita su uso solo a la función de validación, mejorando la claridad y seguridad.	- No es reutilizable fuera de la función. - Debe asignarse a una variable global o retornar su valor.



# Inicialización de una variable

La inicialización de variables en un ejercicio que calcula múltiples descuentos de una compra es clave para evitar errores lógicos y garantizar que cada variable comienza con un valor conocido antes de ser utilizada. Esto es especialmente importante al usar bucles y funciones, ya que las variables mal inicializadas pueden generar resultados incorrectos.

### Cuadro Resumen: Importancia de Inicializar Variables y Validar Entradas

Concepto	Descripción	Ventajas	Desventajas de No Hacerlo
<b>Inicialización de Variables</b>	Consiste en asignar un valor inicial a una variable al momento de su declaración, garantizando que comience con un valor definido antes de ser utilizada en el programa.	<ul style="list-style-type: none"> <li>- Evita errores lógicos como resultados incorrectos o impredecibles.</li> <li>- Asegura que las variables tengan un valor válido antes de ser utilizadas.</li> </ul>	<ul style="list-style-type: none"> <li>- Puede causar errores de ejecución o comportamientos no definidos.</li> <li>- Dificulta la depuración y localización de errores en el código.</li> </ul>
<b>Validación de Entradas</b>	Proceso de verificar que los datos proporcionados por el usuario sean correctos y del tipo esperado antes de ser procesados.	<ul style="list-style-type: none"> <li>- Previene que datos no válidos (como texto en lugar de números) afecten el funcionamiento del programa.</li> <li>- Mejora la experiencia del usuario.</li> </ul>	<ul style="list-style-type: none"> <li>- La falta de validación puede llevar a errores de tipo, resultados inesperados o fallos en el programa.</li> </ul>

# Tipos de Datos Básicos

Son las categorías fundamentales de datos en programación, como enteros, flotantes, caracteres y booleanos, que determinan el tipo de valor almacenado.

Tipo de Dato	Tamaño	Descripción	Ejemplo de Uso
int	4 bytes	Almacena números enteros desde -2,147,483,648 a 2,147,483,647	Edad, cantidad de productos
double	8 bytes	Almacena números decimales de doble precisión	Precio, altura, peso
char	2 bytes	Almacena un único carácter Unicode	Inicial de un nombre, símbolos
boolean	1 bit (depende)	Almacena valores <code>true</code> o <code>false</code>	Estado lógico, banderas
String	Depende del contenido	Cadena de caracteres (no es un tipo de dato primitivo, pero se utiliza con frecuencia)	Nombre, dirección, mensajes
long	8 bytes	Almacena números enteros más grandes que <code>int</code> (-9,223,372,036,854,775,808 a 9,223,372,036,854,775,807)	Identificadores largos, grandes cálculos numéricos
float	4 bytes	Almacena números decimales de menor precisión que <code>double</code>	Temperaturas, coordenadas GPS
byte	1 byte	Almacena números enteros entre -128 y 127	Memoria optimizada en grandes colecciones
short	2 bytes	Almacena números enteros entre -32,768 y 32,767	Uso de enteros pequeños



# Concatenación de Cadenas



Es el proceso de unir dos o más cadenas de texto en programación, creando una nueva cadena combinada a partir de las originales.

```
public class FechasImportantes {

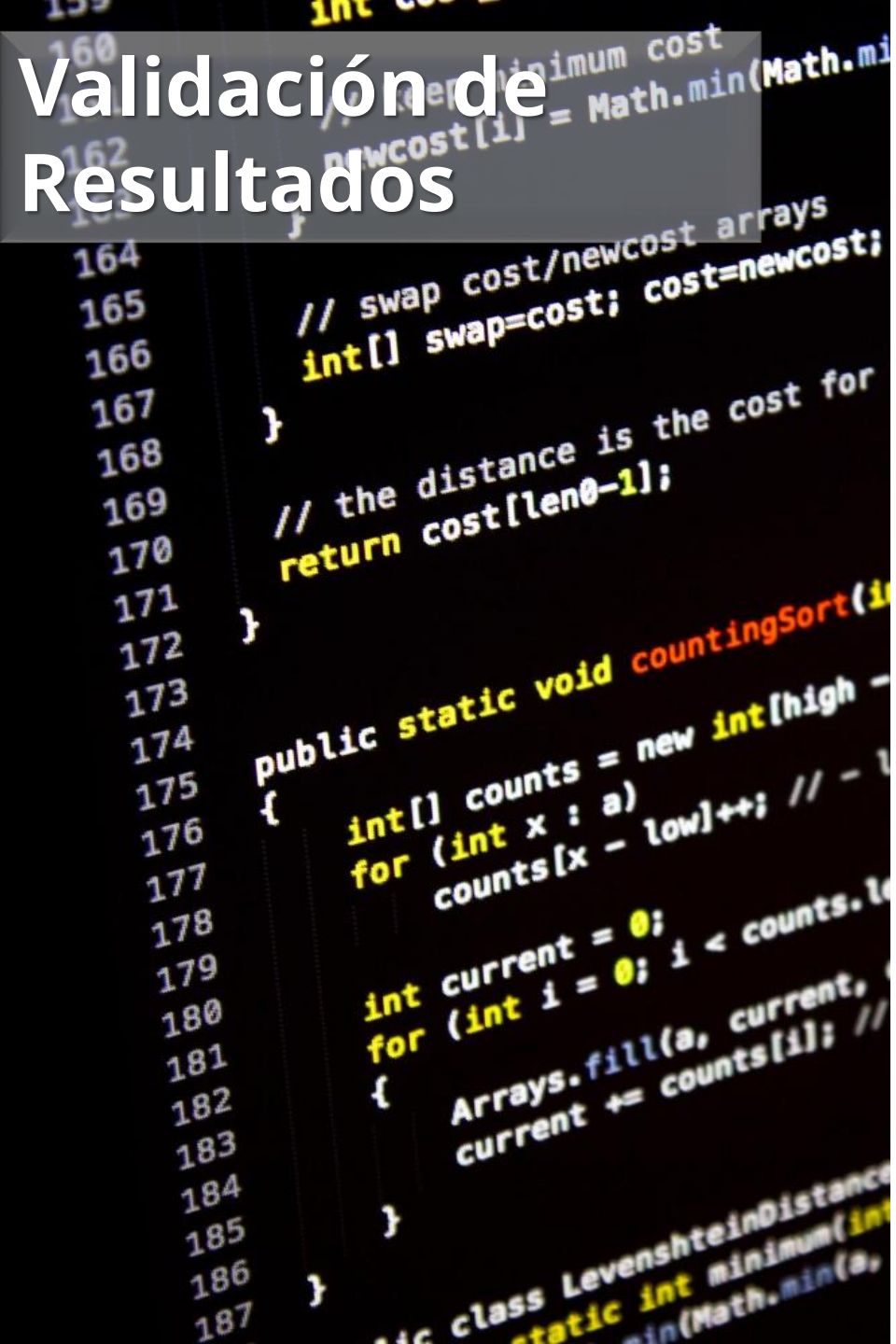
    public static void main(String[] args) {
        // Variables locales que almacenan fechas importantes como cadenas de texto
        String anoNuevo = "1 de enero";
        String diaIndependencia = "18 de septiembre";
        String navidad = "25 de diciembre";

        // Concatenar las fechas importantes en una sola cadena de texto
        String fechas = "Fechas importantes del año:\n"
            + "Año Nuevo: " + anoNuevo + "\n"
            + "Día de la Independencia: " + diaIndependencia + "\n"
            + "Navidad: " + navidad;

        // Mostrar las fechas concatenadas en la consola
        System.out.println(fechas);
    }
}
```

## Importancia de la Concatenación de Cadenas

- **Facilita la Construcción de Mensajes:** Permite combinar varias partes de texto y variables en un solo mensaje.
- **Mejora la Legibilidad del Código:** Hace que el código sea más claro y fácil de entender cuando se presentan varios datos relacionados.
- **Flexibilidad:** Permite formar mensajes dinámicamente al combinar variables y literales de texto.



# Validación de Resultados

Es el proceso de verificar que los resultados de un programa o función sean correctos y cumplan con los requisitos especificados, asegurando precisión.

Aspecto	Descripción
Importancia	Asegura que el código funcione correctamente y detecta errores, mejorando la calidad del software.

Tipo	Descripción
Validación de Entrada	Verifica la validez y formato de los datos ingresados.
Pruebas Unitarias	Testea componentes individuales para asegurar su correcto funcionamiento.
Pruebas de Integración	Verifica la interacción correcta entre diferentes componentes.
Pruebas de Sistema	Valida el sistema completo para asegurar que todos los componentes funcionen juntos.
Pruebas de Aceptación del Usuario	Asegura que el software cumple con las expectativas del usuario final.
Verificación de Resultados	Confirma que el resultado producido es el esperado y correcto.



# Ejercicios



# Lecturas



- Capítulo 4: Making the Most of Variables and Their Values  
Burd, B. (2022). Java for Dummies. New Jersey: John Wiley & Sons.  
[https://webezproxy.duoc.cl/login?url=http://biblioteca.duoc.cl/bdigital/elibros/a50163-Java\\_fordummies/68](https://webezproxy.duoc.cl/login?url=http://biblioteca.duoc.cl/bdigital/elibros/a50163-Java_fordummies/68) Páginas 57 a 68
- Capítulo 1: Introducción a Java - Variables  
Vegas Gertrudix, J. M. (2022). Java 17: Fundamentos prácticos de programación.  
Bogotá: Ediciones de la U.  
[https://webezproxy.duoc.cl/login?url=http://biblioteca.duoc.cl/bdigital/elibros/a50229-Java\\_17/33/](https://webezproxy.duoc.cl/login?url=http://biblioteca.duoc.cl/bdigital/elibros/a50229-Java_17/33/) Páginas 37 a 40

**¡Muchas Gracias!**



