



# Preparación para Certificación Java 17

**Objetivo:** Aprobar el examen OCP (Oracle Certified Professional) Java 17

---



## Tabla de Contenidos

- [🎯 Estrategias de Examen](#)
  - [📖 Estructura del Libro de Estudio](#)
  - [🔧 Configuración del Entorno](#)
  - [📖 Capítulo 1: Construcciones Básicas](#)
  - [🔗 Enlaces Útiles](#)
- 



## Estrategias de Examen



### Consejos Importantes

- **Enfócate en lo importante** que están preguntando
- **Asume** que ciertas porciones de código ya se encuentran:
  - Imports
  - Getters y Setters
- **Variable final** : No se puede reasignar
- **Oddities (Rarezas)**: Pueden permitir obtener altos resultados



### Tipos de Preguntas Comunes

1. **Preguntas con información extra**
  - Ejemplo: `XmlParseException` - aunque no conozcas XML, la pregunta trata sobre excepciones
2. **Preguntas con preguntas embebidas**
  - Contestar bien dos subpreguntas antes de la pregunta principal
  - Son respuestas no relacionadas muchas veces
3. **Preguntas con APIs no familiares**
4. **Preguntas con conceptos erróneos**
  - Ejemplo: interfaz que herede de una clase
  - Uso de palabra reservada `struct`
5. **Preguntas fuera de contexto**
  - Haz tu mejor esfuerzo por contestarlas



### Estrategia de Respuesta

- Si puedes determinar que el código **compila** y qué línea causa el error → pregunta fácil
- Si todas las respuestas son **valores impresos** y no hay opción "no compila" → es un regalo
- Ir **eliminando respuestas** mientras lees la pregunta
- **Marcar las más difíciles** para responderlas después



### Criterio de Aprobación

- **80% o más** en exámenes de capítulo → continuar
  - **Menos del 80%** → revisar temas y comparar con apéndice
-

## Estructura del Libro de Estudio

### Capítulos (15 totales)

Capítulo	Tema	Descripción
1	Construcciones Básicas	Bloques, tipos de datos
2	Operadores	Operadores aritméticos y lógicos
3	Haciendo Decisiones	Control de flujo
4	API Principales	APIs fundamentales de Java
5	Métodos	Diseño y escritura de métodos
6	Diseño de Clases	Arquitectura de clases
7	Interfaces y Más	Interfaces, enums, sealed clases, records, nested classes
8	Lambdas	Interfaces funcionales
9	Colecciones y Genéricos	Collections framework
10	Streams y Optional	 <b>Leer múltiples veces</b>
11	Excepciones y Localización	Manejo de errores
12	Módulos	Compilación de módulos
13	Concurrencia	Administración segura de hilos
14	I/O	Administración de archivos
15	JDBC	Acceso a bases de datos

## Configuración del Entorno

### Herramientas JDK

```
javac    # Compilador: transforma .java → .class
java     # Intérprete: ejecuta .class
jar      # Empaqueta archivos necesarios (.class)
javadoc  # Generador de documentación
```

### Compilación con Paquetes

```
# Compilar múltiples paquetes
javac -d classes packagea/*.java packageb/*.java packagec/*.java

# Ejecutar programa
java -cp classes packageb.ClassB
```

```
java -classpath classes packageb.ClassB
java --class-path classes packageb.ClassB
```

## Compilación con JAR Files









```
# Ejecutar con JAR en classpath
java -cp ".;C:\temp\someOtherLocation;c:\temp\myJar.jar" myPackage.MyClass

# Usar wildcard para todos los JARs
java -cp "C:\path\to\jars\*" packageb.ClassB
```



## Capítulo 1: Construcciones Básicas

### Objetivos OCP Cubiertos

-  Manejo de fechas, tiempo, texto y valores booleanos
-  Uso de primitivos y clases envolventes (incluyendo API Math)
-  Uso de paréntesis, promoción de tipos y casting
-  Evaluación de valores aritméticos y booleanos
-  Comprensión del contexto de variables
-  Uso de variables locales con inferencia de tipos
-  Aplicación de encapsulación
-  Creación de objetos inmutables



## Conceptos Fundamentales



### Clases y Objetos

```
// Clase: bloque básico de construcción en Java
public class Animal {
    private String name; // Campo/variable de instancia

    // Constructor
    public Animal() {
        name = "Duke";
    }

    // Método público
    public void setName(String newName) {
        name = newName;
    }
}
```

### Definiciones importantes:





- **Clase:** Bloque básico de construcción
- **Objeto:** Instancia de una clase en tiempo de ejecución
- **Referencia:** Variable que apunta a un objeto
- **Método:** Función que puede ser llamada

- **Campo:** Variable que mantiene el estado

## 📖 Método main()

```
public static void main(String[] args) {  
    // Punto de entrada del programa  
}
```

### Reglas del método main:

-  `public` - modificador de acceso
-  `static` - pertenece a la clase, no necesita objeto
-  `void` - no retorna valor
-  `String[] args` - parámetros válidos:
  - `String[] args`
  - `String options[]`
  - `String... friends`

### Modificadores opcionales permitidos:



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

## 📦 Paquetes e Imports

### 📖 Declaración de Paquetes

```
package com.wiley.java.my.name;  
  
// Import específico  
import java.util.Date;  
  
// Import con wildcard  
import java.util.*;  
  
// Import conflictivo - usar nombre completo  
java.util.Date date;  
java.sql.Date sqlDate;
```

### ⚠️ Reglas de Import

```
//  CORRECTO  
import java.nio.file.*;  
  
//  INCORRECTO  
import java.nio.*;           // Wildcard solo coincide con nombres de clase  
import java.nio.*.*;        // Solo un wildcard al final  
import java.nio.file.Paths.*; // No se pueden importar métodos
```

### 🔧 Compilación con Paquetes

```
# Crear estructura de directorios
mkdir -p classes/packagea classes/packageb classes/packagec

# Compilar
javac -d classes packagea/*.java packageb/*.java packagec/*.java

# Ejecutar
java -cp classes packageb.ClassB
```





## Creación de Objetos

### Constructores

```
public class Chicken {
    int numEggs = 12; // Inicialización en línea
    String name;

    public Chicken() {
        name = "Duke"; // Constructor inicializa atributos
    }
}
```

#### Reglas de constructores:

-  Debe coincidir con el nombre de la clase
-  No retorna tipo (no puede tener `return` )
-  Inicializa atributos
-  Si no hay constructor, el compilador crea uno por defecto

### Bloques de Inicialización

```
public class Bird {
    { System.out.println("Snowy"); } // Inicializador de instancia

    public static void main(String[] args) {
        { System.out.println("Feathers"); } // Dentro del método
    }
}
```

#### Orden de inicialización:

1. Atributos e inicializadores de bloques (orden del archivo)
2. Constructor (después de todos los inicializadores)

## Referencias vs Tipos Primitivos

### Diferencias Clave

Aspecto	Primitivos	Referencias
---------	------------	-------------

<b>Nomenclatura</b>	int, boolean, char	String, Integer, Boolean
<b>Métodos</b>	❌ No tienen métodos	✅ Pueden llamar métodos
<b>Valor null</b>	❌ No pueden ser null	✅ Pueden ser null
<b>Memoria</b>	Almacenan valor directamente	Apuntan a objeto en memoria

### 💡 Ejemplos

```
// Primitivos
int primitive = 42;
int bad = primitive.length(); // ❌ NO COMPILA

// Referencias
String reference = "hello";
int len = reference.length(); // ✅ COMPILA

// Null
int value = null; // ❌ NO COMPILA
String name = null; // ✅ COMPILA

// Wrapper classes para null
Integer wrapper = null; // ✅ COMPILA
```

### 📦 Clases Wrapper

```
// Conversión de String a primitivo
int primitive = Integer.parseInt("123");

// Conversión de String a wrapper
Integer wrapper = Integer.valueOf("123");
```

## 📖 Variables y Tipos

### 📍 Variables Locales

```
public void method() {
    int y = 10; // ✅ Inicializada
    int x; // ❌ No inicializada
    int reply = x + y; // ❌ NO COMPILA - usa x no inicializada
}
```

### 🔒 Variables Finales

```
final int y = 10;
y = 20; // ❌ NO COMPILA - no se puede reasignar

final int[] numbers = new int[10];
```

```
numbers[0] = 10;    // ✅ COMPILA - modifica contenido
numbers = null;     // ❌ NO COMPILA - cambia referencia
```

## 🔗 Inferencia de Tipos con `var`

```
public void whatTypeAmI() {
    var name = "Hello"; // ✅ String
    var size = 7;       // ✅ int
}

// ❌ NO COMPILA - solo para variables locales
class VarKeyword {
    var tricky = "Hello"; // ❌ NO COMPILA
}

// ❌ NO COMPILA - debe inicializarse en la misma línea
public void invalid() {
    var question; // ❌ NO COMPILA
    question = 1;
}

// ❌ NO COMPILA - no se puede mezclar tipos
int a, var b = 3; // ❌ NO COMPILA

// ❌ NO COMPILA - null no tiene tipo específico
var n = null; // ❌ NO COMPILA

// ✅ COMPILA - reasignación de null después de declaración
String nombre = null;
var stringEscondido = nombre; // ✅ COMPILA
```

## ⚠️ Restricciones de `var`

```
// ❌ NO COMPILA - no se puede usar en parámetros
public int addition(var a, var b) {
    return a + b;
}

// ✅ COMPILA - Java es case sensitive
public class Var {
    public void var() {
        var var = "var"; // ✅ COMPILA
    }
}
```

---

## 🏠 Variables de Instancia y de Clase

### 🏠 Variables de Instancia

```
public class Persona {
    String nombre; // Variable de instancia - cada objeto tiene su propio valor

    public Persona(String nombre) {
        this.nombre = nombre;
    }
}
```

## Variables de Clase (Static)

```
public class Zoo {
    static int totalPersonas; // Variable de clase - compartida por todos los objetos

    public static void main(String[] args) {
        Zoo.totalPersonas = 100; // Acceso sin crear objeto
    }
}
```

## Valores por Defecto

Tipo	Valor por Defecto
int, long, short, byte	0
double, float	0.0
boolean	false
char	'\u0000' (NUL)
Referencias	null

## Enlaces Útiles

### Recursos de Estudio

- **CodeRanch:** [coderanch.com](https://coderanch.com) - Foro de Java
- **Actualizaciones:** [www.selikoff.net/ocp17](http://www.selikoff.net/ocp17)
- **Material adicional:** [www.wiley.com/go/Sybextestprep](http://www.wiley.com/go/Sybextestprep)

### Consejos de Estudio

1. **Construir un plan de estudios** ajustado a tus horarios
2. **Consistencia diaria** - aunque sea un poco en el almuerzo
3. **Preguntarse por qué** no comprendes algo
4. **Estudiar esas áreas** aún más
5. **Determinar si el código compila** y qué línea causa el error

## Notas de Fechas



- **30-06-2025:** "Cuando no sepas cómo partir, parte como sea, con la idea que sea! CSMC"
- **04/07:** Inicialización de variables
- **11-07-2025:** Variables de instancia y de clase

---

*Documentación creada por: alberto san martin mas IA cursor :)*

*Proyecto: Java17SinPackages*