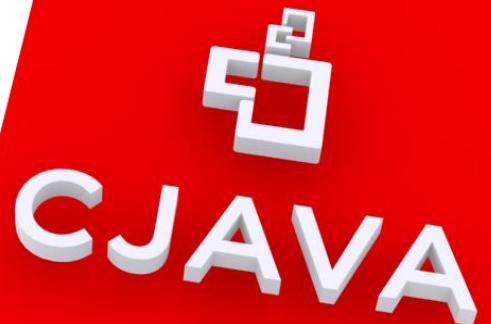


*Edwin Maraví*  
*emaravi@cjavaperu.com*





0101010101010101  
001010101010101010  
0101010101010101  
0101010101010101

01010101010101  
001010101010101010  
0101010101010101

01010101010101  
001010101010101010  
0101010101010101

# Misión

Nuestro equipo trabaja para integrar la tecnología Java en la sociedad como solución a todas sus necesidades.





**CJAVA**  
siempre para apoyarte

010101010101010101010101010101  
001010101010101010101010101010101  
010101010101010101010101010100101  
01010101010101010101  
1010101010101010  
0101010101010101010101010101010101  
101010101010101010101001010010101010101  
01010101010101

## Visión

Poder aportar al desarrollo del País usando tecnología Java.



# Servicios Académicos

**Programer** (80 horas - Certificación Java 11)

[Certificado: Java Programer]

**Developer** (80 horas - Spring FrameWork y Angular)

[Certificado: Java Developer]

**Expert** (80 horas – Microservicios y DevOps)

[Certificado: Java Expert]

**Architect** (80 horas)

[Certificado: Java Arquitect]

**Carrera** (12 meses)

[Diploma: Carrera Java]

Programmer



# Quienes Somos

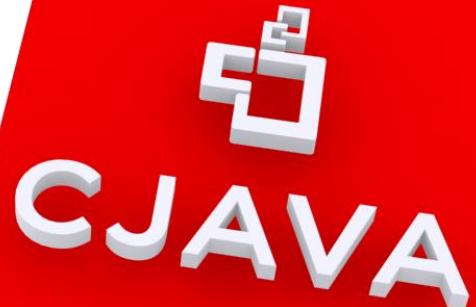
Somos una organización orientada a **desarrollar, capacitar e investigar tecnología JAVA** a través de un prestigioso staff de profesionales a nivel nacional.



# Me presento... Edwin Maraví



- Ing. De Sistemas de la UPSMP.
- Maestría en Docencia Universitaria UCV
- Certificado Oracle Linux y Oracle Java SCJP 8.0
- Scrum Master Certifier en Scrum Study.
- **Director** de CJava
- **Gerente** de DECSEF Perú.
- Consultor TI / Arquitecto en Cloud Solutions.
- Trabajé para: MTC, MINCETUR, MINTRA, MINJUS, CONCYTEC, RENIEC.
- Consultor en: BN, Banco de Comercio, BCP, CONTASIS, Grupo Intercor.
- Docente: PUCP, UNIFE, UPSMP, ISIL, CIBERTEC- DAT, UPLA, UNCP.
- Expositor: CONEIS, COREIS, y otras convenciones del País.



## Contáctenos

- Av. Arenales 395 oficina 405  
Santa Beatriz - Lima 01 - Perú
- Teléfono: 433-6948
- RPC / WhatsApp: 932 656 459
- Email: [info@cjavaperu.com](mailto:info@cjavaperu.com)

## Síguenos

- [/cjava.peru.1](https://www.facebook.com/cjava.peru.1)
- [@cjava\\_peru](https://twitter.com/cjava_peru)
- [/cjavaperu](https://www.linkedin.com/company/cjavaperu/)

# Introducción.





# Metas del curso

- Este curso aborda las principales API que se usan para diseñar aplicaciones orientadas a objetos con Java. También aborda la escritura de programas de base de datos con JDBC.
- Utilice este curso para ampliar sus conocimientos del lenguaje Java y prepararse para el examen de programador Oracle Certified Professional, Java SE 11
- Tener conceptos claros de lo necesario para hacer desarrollo de software.



# Objetivos del curso

Al finalizar este curso, debería estar capacitado para lo siguiente:

- Crear aplicaciones con tecnología Java en las que se usen las funciones orientadas a objetos del lenguaje Java, como la encapsulación, la herencia y el polimorfismo. Reconocer el valor del uso de patrones.
- Reconocer y usar Java Records
- Ejecutar una aplicación Java desde la línea de comandos.
- Reconocer la programación basada en módulos.
- Crear aplicaciones que usen el collections y generics.
- Reconocer la programación funcional usando Lambdas.
- Implantar técnicas de manejo de errores mediante el manejo de excepciones.
- Reconocer Java.time como parte útil del desarrollo de aplicaciones.
- Implantar la funcionalidad de entrada/salida (E/S) de lectura y escritura de datos y archivos de texto y comprender los flujos de E/S avanzados.



# Objetivos del curso

- (continuación)
- Manipular archivos, directorios y sistemas de archivos mediante la especificación JDK NIO.2
- Realizar varias operaciones en tablas de bases de datos, incluida la creación, la lectura, la actualización y la supresión mediante la API JDBC
- Procesar cadenas mediante una serie de expresiones regulares.
- Crear aplicaciones multithread de alto rendimiento que eviten los interbloqueos.
- Localizar aplicaciones Java.



# Asistentes

Entre el público al que va dirigido se incluyen aquellos que:

- Hayan terminado el curso *Conceptos fundamentales de Java SE*, o bien que tengan experiencia con el lenguaje Java y que sean capaces de crear, compilar y ejecutar programas.
- Tengan experiencia con al menos un lenguaje de programación.
- Comprendan los principios orientados a objetos.
- Tengan experiencia con los conceptos básicos y conocimientos básicos de SQL.



# Requisitos

Para completar este curso satisfactoriamente, debe saber cómo:

- Compilar y ejecutar aplicaciones Java
- Crear clases Java
- Crear instancias de objetos con la palabra clave new
- Declarar variables de referencia y primitivas de Java
- Declarar métodos Java con valores de retorno y parámetros
- Usar construcciones condicionales como sentencias if y switch
- Usar construcciones en bucle, como bucles for, while y do
- Declarar e instanciar matrices Java
- Usar la especificación de la API Java Platform, Standard Edition (Javadocs)



# Presentaciones a la clase

Preséntese brevemente:

- Nombre
- Cargo o puesto
- Compañía
- Experiencia con programación Java y aplicaciones Java
- Motivos para asistir



# Entorno del curso

## Classroom PC

### Core Apps

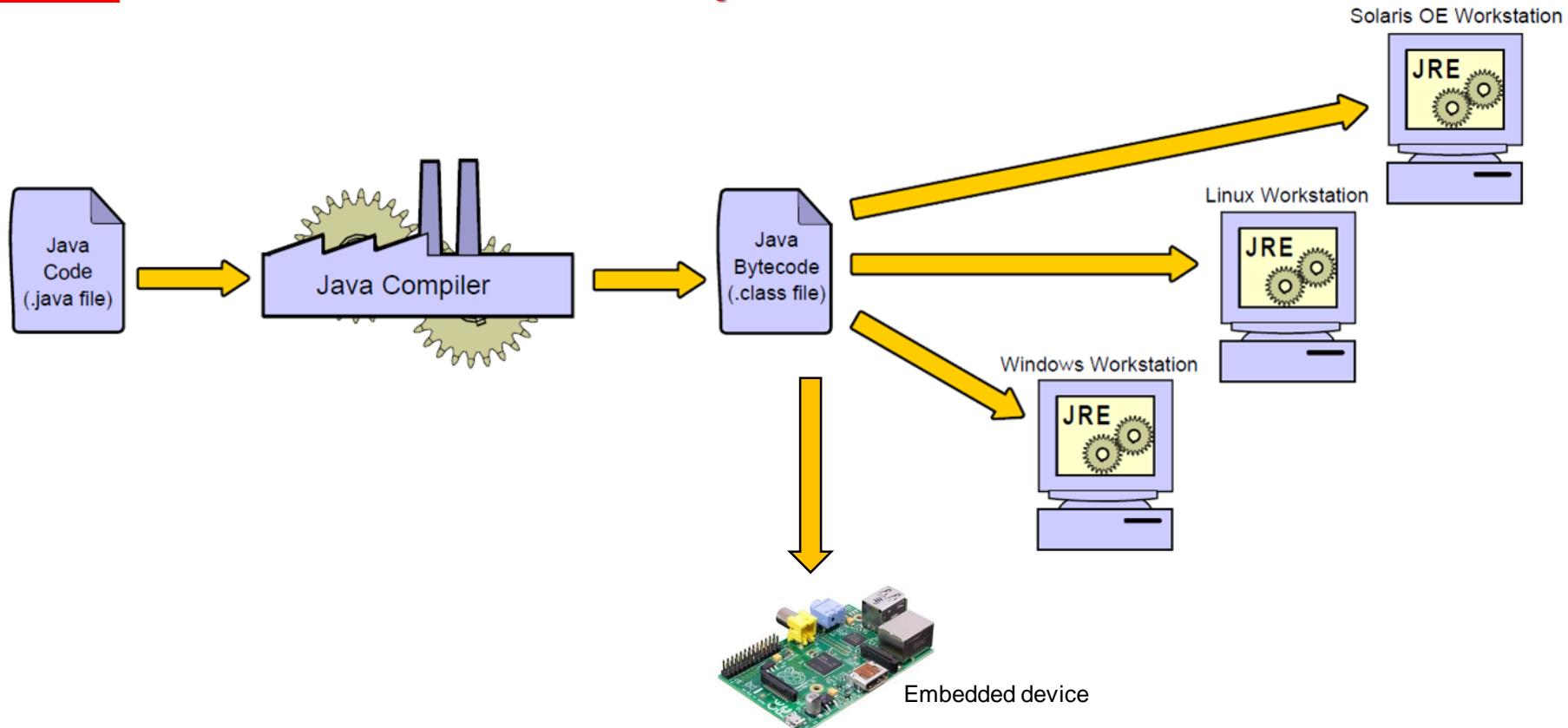
- JDK 8
- JDK 11
- JDK 14
- NetBeans o
- IntelliJ

### Additional Tools

- Google Chrome
- MySQL

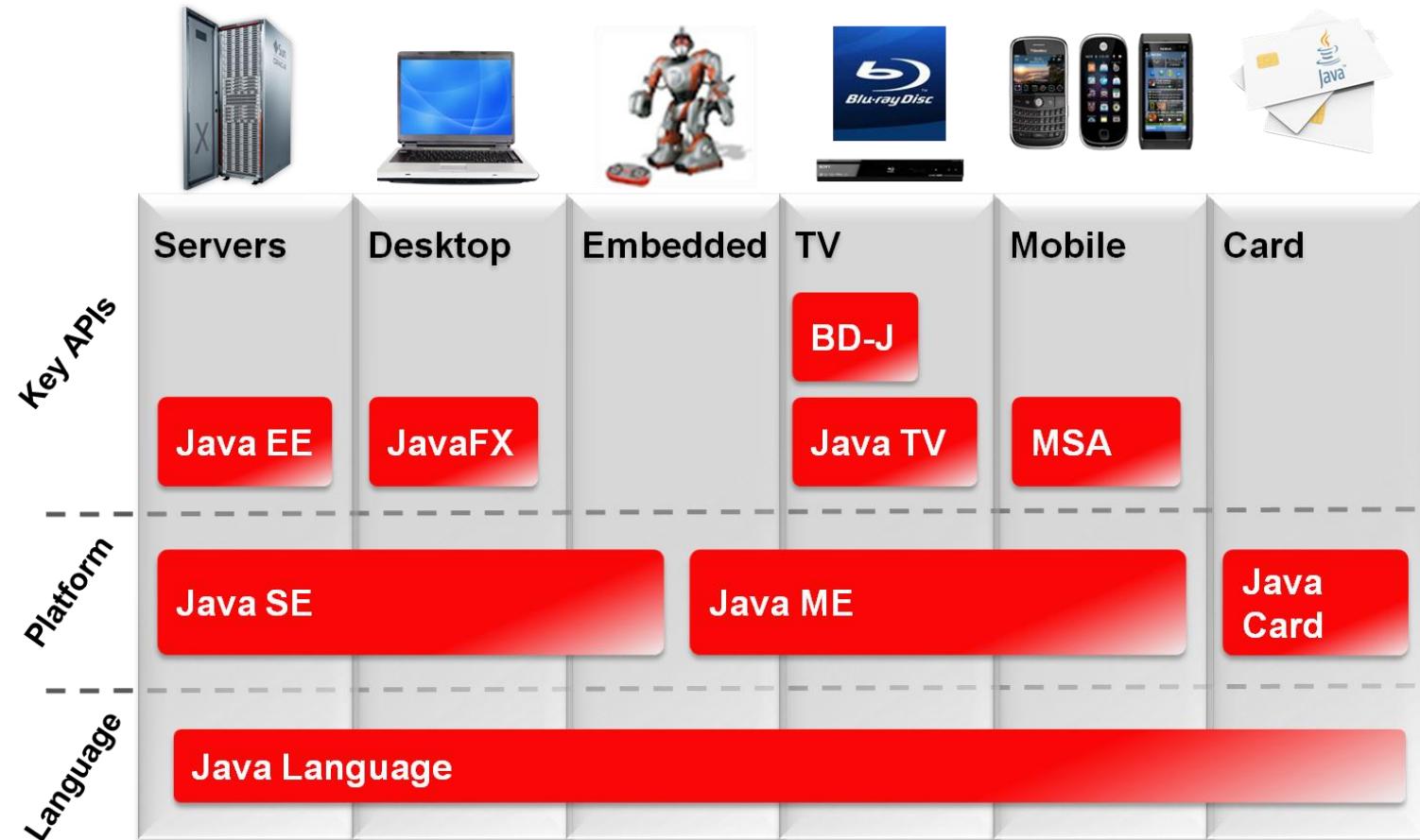


# Los programas Java son independientes de la plataforma





# Grupos de productos de tecnología Java





# Descarga e instalación del JDK

The screenshot shows the Oracle Java SE Downloads page. At the top, there's a navigation bar with links for Products and Services, Downloads, Store, Support, Training, Partners, About, and Oracle Technology Network. Below the navigation bar, the URL is http://www.oracle.com/technetwork/java/javase/downloads/index.html. The main content area has tabs for Overview, Downloads, Documentation, Community, Technologies, and Training. The Downloads tab is selected. It features a "Java SE Downloads" section with five tabs: Latest Release, Next Release (Early Access), Embedded Use, Real-Time, and Previous Releases. Below these tabs are three download buttons: "Java Platform (JDK)", "JDK + NetBeans Bundle", and "JDK + Java EE Bundle". To the left is a sidebar with links for Java SE, Java EE, Java ME, Java Support, Java Advanced & Suite, Java Embedded, JavaFX, Java DB, Web Tier, Java Card, Java TV, Community, and Java Magazine. On the right, there are two sections: "Java SDKs and Tools" (with links for Java SE, Java EE and Glassfish, Java ME, Java FX, Java Card, and NetBeans IDE) and "Java Resources" (with links for New to Java?, APIs, Code Samples & Apps, Developer Training, Documentation, Java BluePrints, Java.com, Java.net, Student Developers, and Tutorials). The status bar at the bottom indicates "Internet | Protected Mode: On" and "100%".

# Java en entornos del servidor

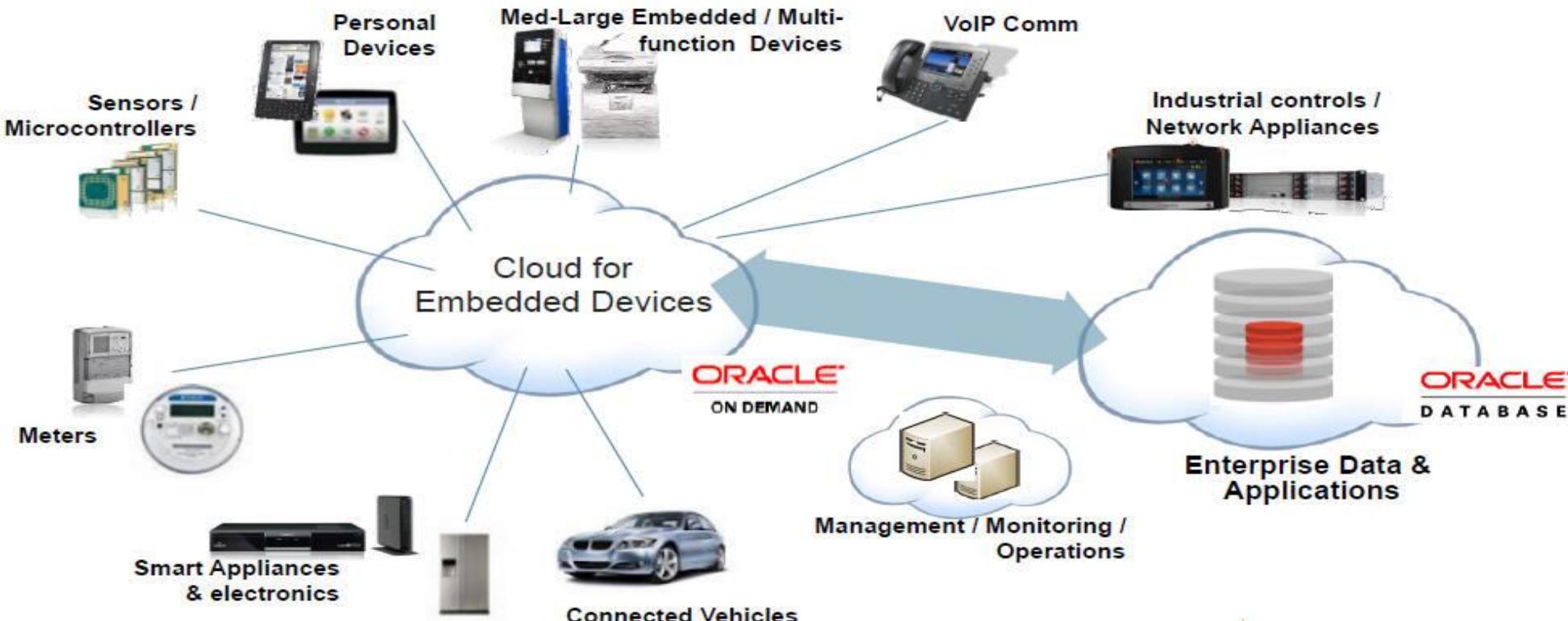


Java se suele usar en entornos de empresa:

- Oracle Fusion Middleware
  - Servidores de aplicaciones Java
    - GlassFish
    - WebLogic
- Servidores de base de datos
  - MySQL
  - Oracle Database

# The Internet of Things

Devices on the “edge” represent a huge growth opportunity.





# Comunidad Java

OpenJDK



Mobile &  
Embedded



Spring Framework

Claya, siempre para apoyarte.

# Java Community Process(JCP)

- JCP se usa para desarrollar nuevos estándares Java:
- <http://jcp.org>
- Descarga gratuita de todas las solicitudes de especificación Java (JSR)
- Acceso anticipado a las especificaciones
- Posibilidad de recibir comentarios y revisiones por parte de otros miembros
- Participación libre

# OpenJDK

OpenJDK es la implantación de código fuente de Java:

- <http://openjdk.java.net/>
- Proyecto de código abierto con licencia GPL
- Implantación de referencia JDK
- Donde se desarrollan nuevas funciones
- Permite contribuciones a la comunidad
- Base de Oracle JDK



# Soporte de Oracle Java SE

Java está disponible de forma gratuita. Sin embargo, Oracle proporciona soluciones Java de pago:

- Java SE Support Program ofrece actualizaciones para versiones Java con un fin de vida determinado.
- Oracle Java SE Advanced y Oracle Java SE Suite:
  - JRockit Mission Control
  - Detección de falta de memoria
  - Low Latency GC (Suite)
  - JRockit Virtual Edition (Suite)



# Recursos adicionales

Topic	Website
Education and Training	<a href="http://education.oracle.com">http://education.oracle.com</a>
Product Documentation	<a href="http://www.oracle.com/technology/documentation">http://www.oracle.com/technology/documentation</a>
Product Downloads	<a href="http://www.oracle.com/technology/software">http://www.oracle.com/technology/software</a>
Product Articles	<a href="http://www.oracle.com/technology/pub/articles">http://www.oracle.com/technology/pub/articles</a>
Product Support	<a href="http://www.oracle.com/support">http://www.oracle.com/support</a>
Product Forums	<a href="http://forums.oracle.com">http://forums.oracle.com</a>
Product Tutorials	<a href="http://www.oracle.com/technetwork/tutorials/index.html">http://www.oracle.com/technetwork/tutorials/index.html</a>
Sample Code	<a href="https://www.samplecode.oracle.com">https://www.samplecode.oracle.com</a>
Oracle Technology Network for Java Developers	<a href="http://www.oracle.com/technetwork/java/index.html">http://www.oracle.com/technetwork/java/index.html</a>
Oracle Learning Library	<a href="http://www.oracle.com/goto/oll">http://www.oracle.com/goto/oll</a>



# Sintaxis Java y revisión de clases.



# Objetivos

Al finalizar esta lección, debería estar capacitado para lo siguiente:

- Crear clases Java simples
  - Crear variables primitivas
  - Manipular cadenas
  - Usar las sentencias de bifurcación if-else y switch
  - Iterar con bucles
  - Crear matrices
- Usar campos, constructores y métodos Java
- Usar las sentencias package e import



# Revisión del lenguaje Java

En esta lección se repasan los conceptos fundamentales de Java y de la programación. Se supone que los alumnos conocen los siguientes conceptos:

- La estructura básica de una clase Java
- Los bloques y comentarios de un programa
- Variables
- Las construcciones de bifurcación básicas if-else y switch
- La iteración con bucles for y while



# Estructura de la clase

```
package <package_name>;  
  
import <other_packages>;  
  
public class ClassName {  
  
    <variables(also known as fields)>;  
  
    <constructor method(s)>;  
  
    <other methods>;  
}
```



# Clase simple

Una clase Java simple con un método main:

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



# Bloques de código

- Todas las declaraciones de clase se incluyen en un bloque de código.
- Las declaraciones de métodos se incluyen en bloques de código.
- El ámbito de los campos y los métodos Java es el bloque (o la clase).
- Los bloques de códigos se definen entre corchetes:

```
{ }
```

– Ejemplo:

```
public class SayHello { ←  
    public static void main(String[] args) { ←  
        System.out.println("Hello world");  
    } ←  
}
```



# Java Naming Conventions

```
1  public class CreditCard {  
2      public final int VISA = 5001;  
3      public String accountName;  
4      public String cardNumber;  
5      public Date expDate;  
6  
7      public double getCharges(){  
8          // ...  
9      }  
10  
11     public void disputeCharge(String chargeId, float amount){  
12         // ...  
13     }  
14 }
```

Class names are nouns in upper camel case.

Constants should be declared in all uppercase letters

Variable names are short but meaningful in lower camel case.

Methods should be verbs, in lower camel case.



# How to Compile and Run

Java class files must be compiled before running them.  
To compile a Java source file, use the Java compiler (javac).

```
javac -cp <path to other classes> -d <compiler output path> <path to source>.java
```

- You can use the CLASSPATH environment variable to the directory above the location of the package hierarchy.
- After compiling the source .java file, a .class file is generated.
- To run the Java application, run it using the Java interpreter (java):

```
java -cp <path to other classes> <package name>.<classname>
```



# How to Compile and Run: Example

- Assume that the class shown in the notes is in the directory test in the path /home/oracle:

```
$ javac HelloWorld.java
```

- To run the application, you use the interpreter and the class name:

```
$ java HelloWorld
Hello World
```

- The advantage of an IDE like NetBeans is that management of the class path, compilation, and running the Java application are handled through the tool.



# Tipos de datos primitivos

Integer	Floating Point	Character	True False
byte short int long	float double	char	boolean
1, 2, 3, 42 07 0xff	3.0F .3337F 4.022E23	'a' '\u0061' '\n'	true false
0	0.0	'\u0000'	false

Agregue las letras "L" o "F" en mayúsculas o minúsculas al número para especificar un número largo o uno flotante.



# Literales numéricas desde Java SE 7

En Java SE 7 (y versiones posteriores), puede aparecer cualquier número de caracteres subrayados (\_) entre dígitos en un campo numérico. Esto puede mejorar la lectura del código.

```
long creditCardNumber = 1234_5678_9012_3456L;
long socialSecurityNumber = 999_99_9999L;
long hexBytes = 0xFF_EC_DE_5E;
long hexWords = 0xCAFE_BABE;
long maxLong = 0x7fff_ffff_ffff_ffffL;
byte nybbles = 0b0010_0101;
long bytes = 0b11010010_01101001_10010100_10010010;
```



# Literales binarios de Java SE 7

En Java SE 7 (y versiones posteriores), los literales binarios también se pueden expresar con el sistema binario agregando los prefijos 0b o 0B al número:

```
// An 8-bit 'byte' value:  
byte aByte = 0b0010_0001;  
  
// A 16-bit 'short' value:  
short aShort = (short)0b1010_0001_0100_0101;  
  
// Some 32-bit 'int' values:  
int anInt1 = 0b1010_0001_0100_0101_1010_0001_0100_0101;  
int anInt2 = 0b101;  
int anInt3 = 0B101; // The B can be upper or lower case.
```



# Operadores

- Operador de asignación simple
  - = Operador de asignación simple
- Operadores aritméticos
  - + Operador de suma (también se usa para la concatenación de cadenas)
  - Operador de resta
  - \* Operador de multiplicación
  - / Operador de división
  - % Operador de resto
- Operadores unarios
  - + Operador más unario; indica positivo
  - Operador menos unario; niega una expresión
  - ++ Operador de aumento; aumenta un valor en 1
  - Operador de disminución; disminuye un valor en 1
  - ! Operador de complemento lógico; invierte el valor de un booleano



# Cadenas

```
1 public class Strings {  
2  
3     public static void main(String args[]){  
4  
5         char letter = 'a';  
6  
7         String string1 = "Hello";  
8         String string2 = "World";  
9         String string3 = "";  
10        String dontDoThis = new String ("Bad Practice");  
11  
12        string3 = string1 + string2; // Concatenate strings  
13  
14        System.out.println("Output: " + string3 + " " + letter);  
15  
16    }  
17 }
```

Los literales de cadena se crean automáticamente como objetos String.



# Operaciones de cadenas

```
1 public class StringOperations {  
2     public static void main(String arg[]) {  
3         String string2 = "World";  
4         String string3 = "";  
5  
6         string3 = "Hello".concat(string2);  
7         System.out.println("string3: " + string3);  
8  
9         // Get length  
10        System.out.println("Length: " + string1.length());  
11  
12        // Get SubString  
13        System.out.println("Sub: " + string3.substring(0, 5));  
14  
15        // Uppercase  
16        System.out.println("Upper: " + string3.toUpperCase());  
17    }  
18}
```

Los literales de cadena se crean automáticamente como objetos String.



# if else

```
1 public class IfElse {  
2  
3     public static void main(String args[]){  
4         long a = 1;  
5         long b = 2;  
6  
7         if (a == b){  
8             System.out.println("True");  
9         } else {  
10            System.out.println("False");  
11        }  
12    }  
13}  
14 }
```



# Operadores lógicos

- Operadores de igualdad y relacionales
  - == Igual que
  - != Distinto de
  - > Mayor que
  - >= Mayor o igual que
  - < Menor que
  - <= Menor o igual que
- Operadores condicionales
  - && AND condicional
  - || OR condicional
  - ? : Ternario (versión abreviada de la sentencia if-then-else)
- Operador de comparación de tipos
  - instanceof Compara un objeto con un tipo especificado



# Matrices y bucle for - each

```
1 public class ArrayOperations {  
2     public static void main(String args[]) {  
3         String[] names = new String[3];  
4         names[0] = "Blue Shirt";  
5         names[1] = "Red Shirt";  
6         names[2] = "Black Shirt";  
7         int[] numbers = {100, 200, 300};  
8         for (String name:names) {  
9             System.out.println("Name: " + name);  
10        }  
11        for (int number:numbers) {  
12            System.out.println("Number: " + number);  
13        }  
14    }  
15}
```

Las matrices son objetos. Los objetos de matriz tienen una longitud de campo final.



# Bucle for

```
1 public class ForLoop {  
2  
3     public static void main(String args[]) {  
4  
5         for (int i = 0; i < 9; i++) {  
6             System.out.println("i: " + i);  
7         }  
8     }  
9 }  
10 }
```



# Bucle While

```
1 public class WhileLoop {  
2  
3     public static void main(String args[]) {  
4  
5         int i = 0;  
6         int[] numbers = {100, 200, 300};  
7  
8         while (i < numbers.length) {  
9             System.out.println("Number: " + numbers[i]);  
10            i++;  
11        }  
12    }  
13 }
```



# Sentencia switch de cadena

```
1 public class SwitchStringStatement {  
2     public static void main(String args[]) {  
3  
4         String color = "Blue";  
5         String shirt = " Shirt";  
6  
7         switch (color){  
8             case "Blue":  
9                 shirt = "Blue" + shirt;  
10                break;  
11            case "Red":  
12                shirt = "Red" + shirt;  
13                break;  
14            default:  
15                shirt = "White" + shirt;  
16        }  
17  
18        System.out.println("Shirt type: " + shirt);  
19    }  
20 }
```



# Convenciones de nomenclatura Java

```
1 public class CreditCard {  
2     public final int VISA = 5001;  
3     public String accountName;  
4     public String cardNumber;  
5     public Date expDate;  
6  
7     public double getCharges () {  
8         // ...  
9     }  
10  
11    public void disputeCharge (String chargeId, float  
amount) {  
12        // ...  
13    }  
14 }
```

Los nombres de clases son nombres en formato CamelCase en mayúsculas.

Las constantes se deben declarar con todas las letras en mayúsculas.

Los nombres de variables son breves, pero significativos y tienen formato CamelCase en minúsculas.

Los métodos deben ser verbos en formato CamelCase en minúsculas.



# Una clase Java simple: Employee

Una clase Java se suele usar para representar un concepto.

```
1 package com.example.domain;
2 public class Employee {      declaración de clase
3     public int empId;
4     public String name;
5     public String ssn;          }
6     public double salary;
7
8     public Employee () {      un constructor
9     }
10
11    public int getEmpId () {  un método
12        return empId;
13    }
14 }
```



# Métodos de la clase Employee

Cuando una clase tiene campos de datos, una práctica habitual consiste en proporcionar métodos para almacenar datos (métodos setter) y recuperar datos (métodos getter) de los campos.

```
1 package com.example.domain;
2 public class Employee {
3     public int empId;
4     // other fields...
5     public void setEmpId(int empId) {
6         this.empId = empId;
7     }
8     public int getEmpId() {
9         return empId;
10    }
11    // getter/setter methods for other fields...
12 }
```

A menudo un par de métodos para definir y obtener el valor del campo actual.



# Creación de una instancia de un objeto

Para crear una instancia (objeto) de la clase Employee, utilice la palabra clave

```
/* In some other class, or a main method */
Employee emp = new Employee();
emp.empId = 101;    // legal if the field is public,
                   // but not good OO practice
emp.setEmpId(101); // use a method instead
emp.setName("John Smith");
emp.setSsn("011-22-3467");
emp.setSalary(120345.27);
```

Invocando la instancia  
de un objeto.

- En este fragmento de código Java, crea una instancia de la clase Employee y asigna la referencia al nuevo objeto a una variable denominada emp.
- A continuación, asigna valores al objeto Employee.



# Constructores

```
public class Employee {  
    public Employee() {  
    }  
}
```



A simple no-argument (no-arg) constructor.

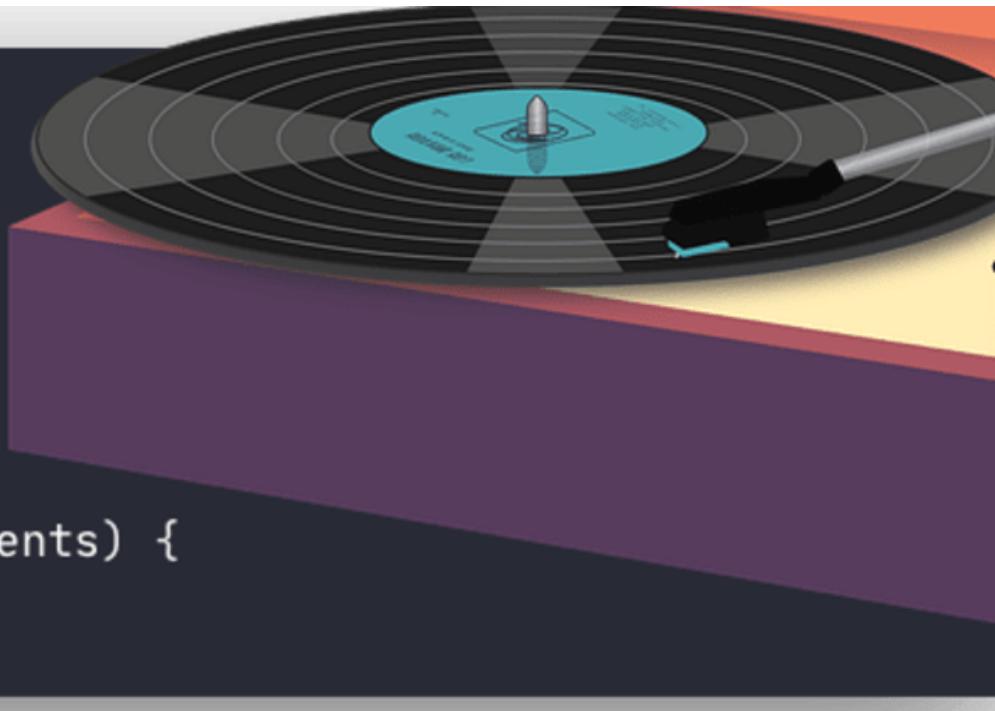
```
Employee emp = new Employee();
```

- A constructor is used to create an instance of a class.
- Constructors can take parameters.
- A constructor that takes no arguments is called a *no-arg* constructor.



# Hablemos de Java 14

```
record Person(  
    String firstName,  
    String lastName,  
    String address,  
    LocalDate birthday,  
    List<String> achievements) {  
}
```





# Sentencia package

La palabra clave package se usa en Java para agrupar clases. Un paquete se implanta como carpeta y, al igual que una carpeta, proporciona un *espacio de nombre* a una clase.

vista de namespace

com.example.domain

Employee

Manager

vista de carpeta

+com

|\_+example

|\_+domain

|\_+Employee.java

|\_+Manager.java

Los paquetes se deben declarar siempre.



# Sentencias import

La palabra clave import se usa para identificar a las clases a las que desea hacer referencia en la clase.

- La sentencia import ofrece un método práctico para identificar clases a las que desea hacer referencia en la clase.

```
import java.util.Date;
```

- Puede importar una única clase o un paquete completo:

```
import java.util.*;
```

- Puede incluir varias sentencias import:

```
import java.util.Date;  
import java.util.Calendar;
```

- Se aconseja usar todo el paquete y el nombre de clase en lugar del carácter comodín \* para evitar conflictos de nombres de clase.



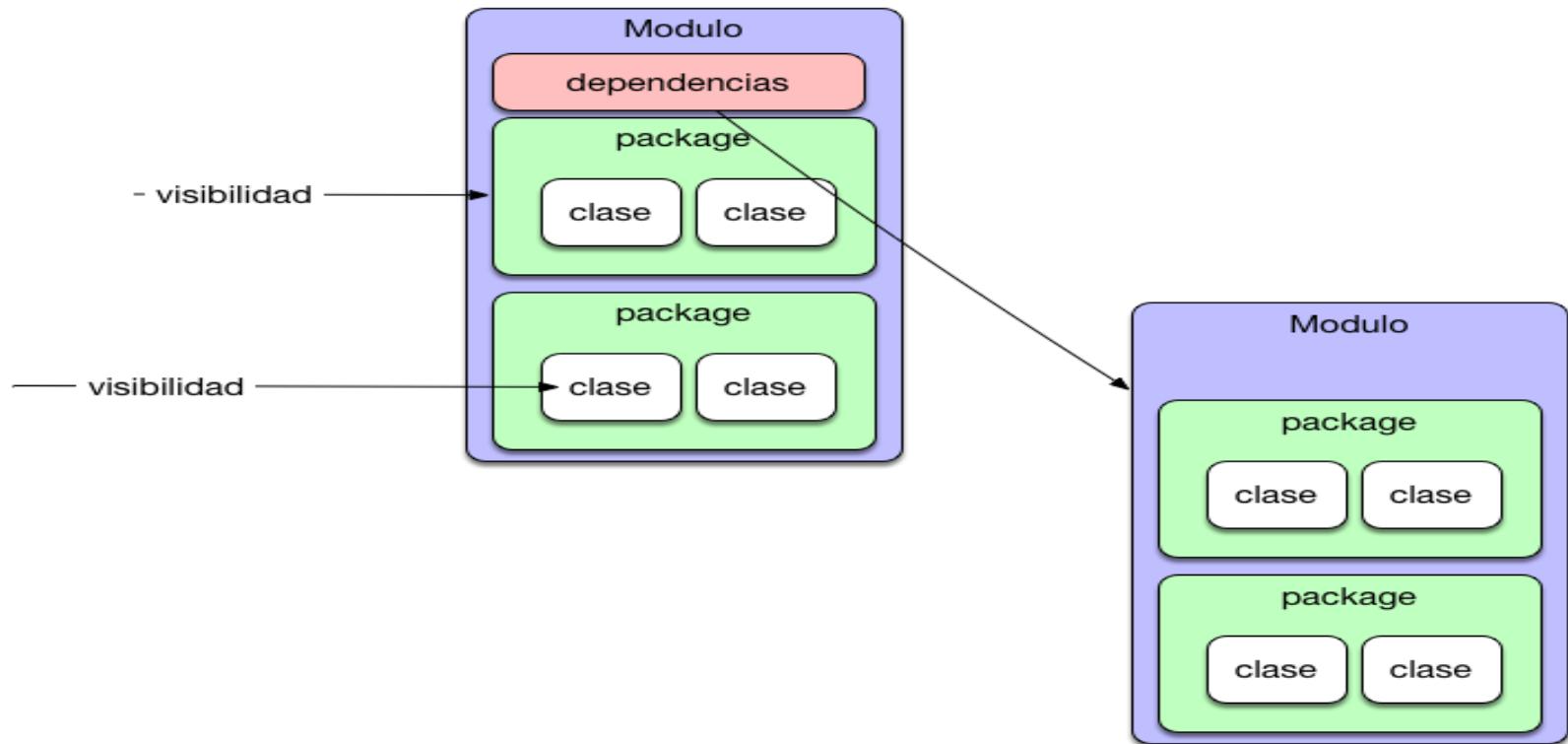
# Más información sobre import

- Las sentencias import van después de la declaración del paquete y antes de la declaración de la clase.
- No es necesaria una sentencia import.
- Por defecto, su clase siempre importa java.lang.\*
- No es necesario que importe clases que estén en el mismo paquete:

```
package com.example.domain;  
import com.example.domain.Manager; // unused import
```



# Desde Java 9 LOS MODULOS





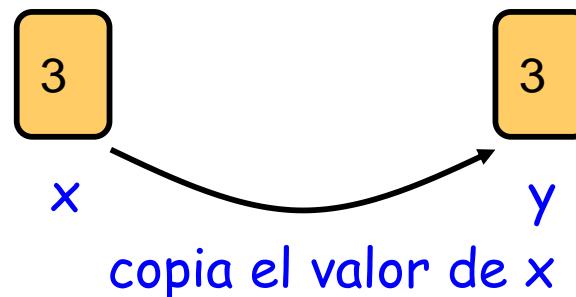
# Java se transfiere por valor

El lenguaje Java (a diferencia de C++) usa la transferencia por valor para pasar todos los parámetros.

- Para visualizar esto con primitivos, tenga en cuenta lo siguiente:

```
int x = 3;  
int y = x;
```

- El valor de x se copia y transfiere a y:



- Si se modifica x (por ejemplo, `x = 5;`), no se cambia el valor de y.

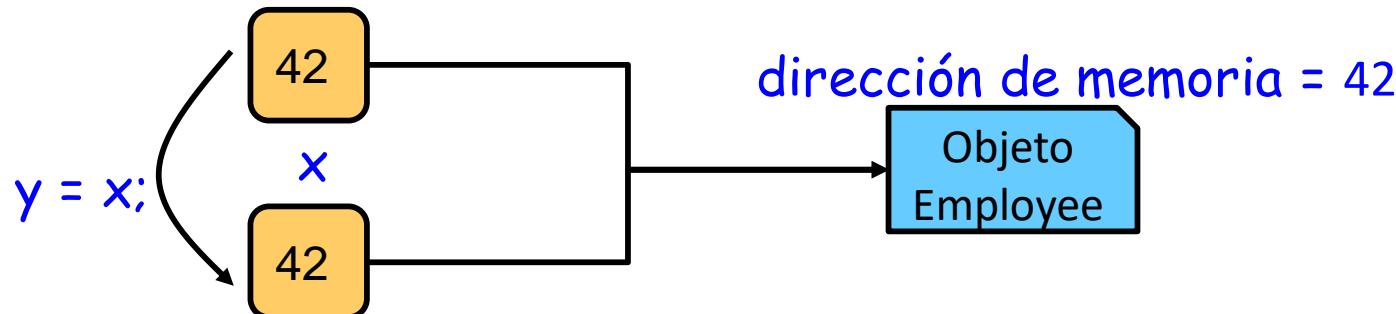


# Transferencia por valor para referencias de objetos

En el caso de objetos Java, el *valor* del lado derecho de una asignación es una referencia a la memoria que almacena un objeto Java.

```
Employee x = new Employee();  
Employee y = x;
```

- La referencia es alguna dirección de la memoria.



- Tras la asignación, el valor de `y` es el mismo que el valor de `x`: una referencia al mismo objeto `Employee`.

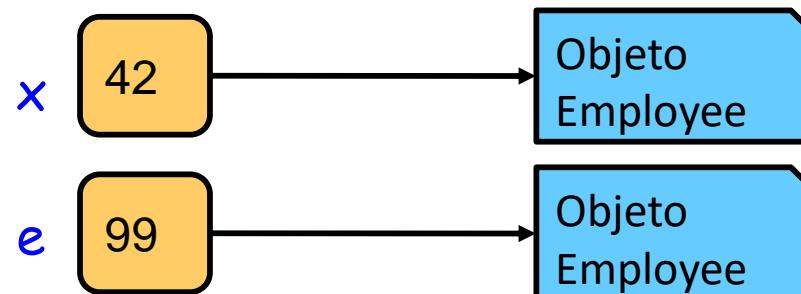
# Objetos transferidos como parámetros

- Siempre que se cree un nuevo objeto, se crea una nueva referencia. Considere los siguientes fragmentos de código:

```
Employee x = new Employee();  
foo(x);
```

```
public void foo(Employee e) {  
    e = new Employee();  
    e.setSalary (1_000_000.00); // Que sucede aquí?  
}
```

- El valor de x no cambia como resultado de la llamada al método foo:





# Referencias pasados como objetos

```
4 public class ObjectPassTest {  
5     public static void main(String[] args) {  
6         ObjectPassTest test = new ObjectPassTest();  
7         Employee x = new Employee ();  
8         x.setSalary(120_000.00);           salary set to  
9         test.foo(x);                      120_000  
10        System.out.println ("Employee salary: "  
11                + x.getSalary());  
12    }  
13  
14    public void foo(Employee e) {  
15        e.setSalary(130_000.00);           What will  
16        e = new Employee();                 x.getSalary() return  
17        e.setSalary(140_000.00);           at the end of the  
18    }                                     main() method?
```



# Cómo compilar y ejecutar

Los archivos de clase Java se deben compilar antes de ejecutarse.

Para compilar un archivo de origen Java, utilice el compilador Java (javac).

```
javac -cp <path to other classes> -d <complier output path> <path to source>.java
```

- Puede utilizar la variable de entorno CLASSPATH al directorio superior a la ubicación de la jerarquía de paquetes.
- Tras compilar el archivo .java de origen, se genera un archivo .class.
- Para ejecutar la aplicación Java, ejecútela con el intérprete Java (java):

```
java -cp <path to other classes> <package name>.<classname>
```



# Compilación y ejecución: ejemplo

- Suponga que la clase que aparece en las notas está en el directorio D:\test\com\example:

```
javac -d D:\test D:\test\com\example\HelloWorld.java
```

- Para ejecutar la aplicación, utilice el intérprete y el nombre de clase totalmente cualificado:

```
java -cp D:\test com.example.HelloWorld
Hello World!
```

```
java -cp D:\test com.example.HelloWorld Tom
Hello Tom!
```

- La ventaja de un IDE como NetBeans es que la gestión del classpath, la compilación y la ejecución de la aplicación Java se manejan mediante la herramienta.

# Cargador de clase Java

Durante la ejecución de un programa Java, Java Virtual Machine carga los archivos de clase Java compilados con una clase Java propia denominada el “cargador de clases” (java.lang.ClassLoader).

Al instanciar un objeto, se llama al cargador de clases:

```
public class Test {  
    public void someOperation() {  
        Employee e = new Employee();  
        //...  
    }  
}
```

Al cargador de clases  
se le llama para  
“cargar” esta clase en  
la memoria.

```
Test.class.getClassLoader().loadClass("Employee");
```



# Recolección de basura

Cuando se crea una instancia de un objeto con la palabra clave new, se asigna memoria al objeto. El ámbito de una referencia de objeto depende de si se ha instanciado el objeto:

```
public void someMethod() {  
    Employee e = new Employee();  
    // operations on e  
}
```

El alcance del objeto e finaliza aquí.

- Cuando finaliza someMethod, ya no se puede acceder a la memoria a la que hace referencia e.
- El recolector de basura de Java reconoce cuándo ya no se puede acceder a una instancia y libera automáticamente esta memoria.



# Resumen

En esta lección, debe haber aprendido a hacer lo siguiente:

- Crear clases Java simples
  - Crear variables primitivas
  - Manipular cadenas
  - Usar las sentencias de bifurcación if-else y switch
  - Iterar con bucles
  - Crear matrices
- Usar campos, constructores y métodos Java
- Usar las sentencias package e import



# Quiz

Which is the printed result in the following fragment?

```
public float average (int[] values) {  
    float result = 0;  
    for (int i = 1; i < values.length; i++)  
        result += values[i];  
    return (result/values.length);  
}  
// ... in another method in the same class  
int[] nums = {100, 200, 300};  
System.out.println (average(nums));
```

- a. 100.00
- b. 150.00
- c. 166.66667
- d. 200.00



# Quiz

In the following fragment, which two statements are false?

```
package com.oracle.test;
public class BrokenClass {
    public boolean valid = "false";
    public String s = "A new string";
    public int i = 40_000.00;
    public BrokenClass() {}  
}
```

- a. An import statement is missing.
- b. The boolean valid is assigned a String.
- c. String s is created.
- d. BrokenClass method is missing a return statement.
- e. You need to create a new BrokenClass object.
- f. The integer value i is assigned a double.



# Quiz

What is displayed when the following code snippet is compiled and executed?

```
String s1 = new String("Test");
String s2 = new String("Test");
if (s1==s2)
    System.out.println("Same");
if (s1.equals(s2))
    System.out.println("Equals");
```

- a . Same
- b . Equals
- c . Same Equals
- d . Compiler Error



# CJAVA

siempre para apoyarte

#CJavaNoPara

Gracias