

---

---

# P00 en Java

UNQ - PO2

---

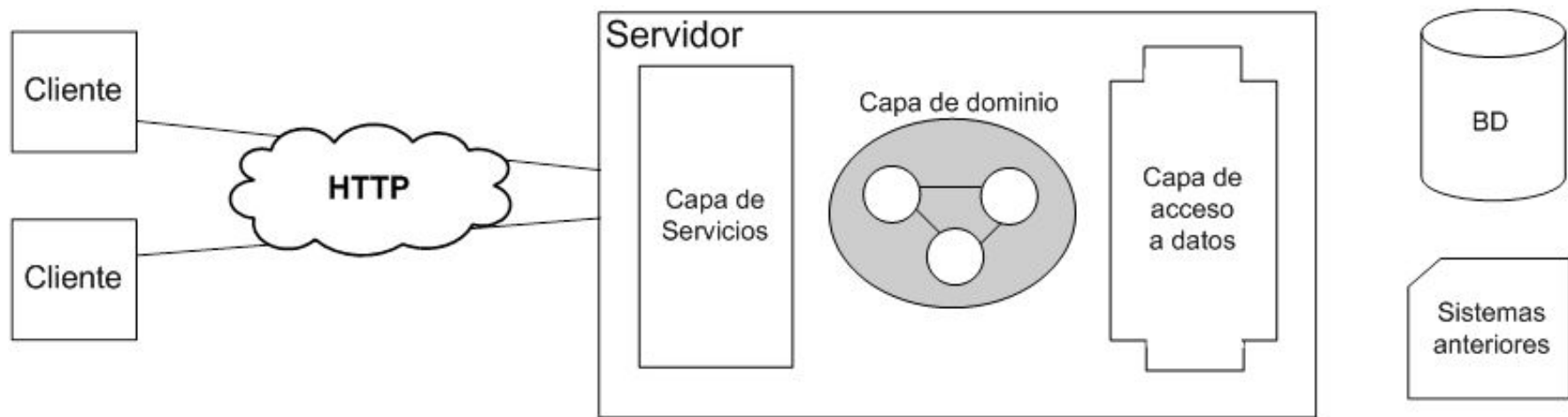
---

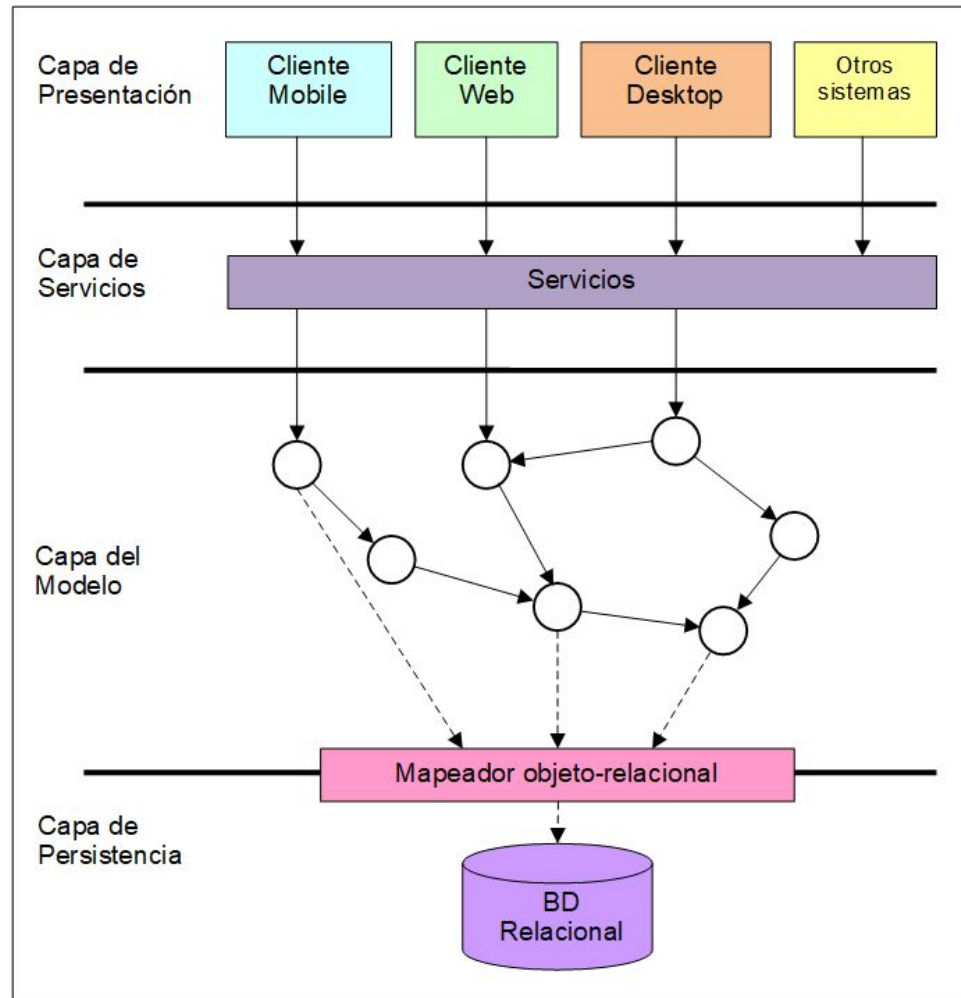
# Temario de las próximas dos clases

- Arquitectura. ¿En que parte de la arquitectura iría lo que desarrollamos en esta materia?
- Parte 1
  - Repaso de lenguajes de programación
  - Repaso de PO
  - Cómo Java implementa conceptualmente los conceptos vistos en el repaso
- Partes 2. Componentes de la plataforma Java
  - Lenguaje
  - JDK
  - JRE
- Instalación y Armado del ambiente ([video tutorial](#))
- TP3 - Repaso POO e intro a Java
- Pilares del lenguaje Java
- TP4 - POO en Java

# Arquitectura

En qué parte de la arquitectura iría lo que desarrollamos en la materia?





# **PARTE 1 - REPASO LENGUAJES DE PROGRAMACIÓN, POO E INTRO A POO EN JAVA**

# Temario

- Contenido en las clases
  - Variables
  - Métodos
  - Constructores
- Niveles de Accesibilidad
- Paquetes
- Tests

# Introducción a Java

Es un lenguaje de programación de propósito general desarrollado en 1991 y publicado en 1995.

Adelanto del final de la clase: jdk y Eclipse.



James Gosling



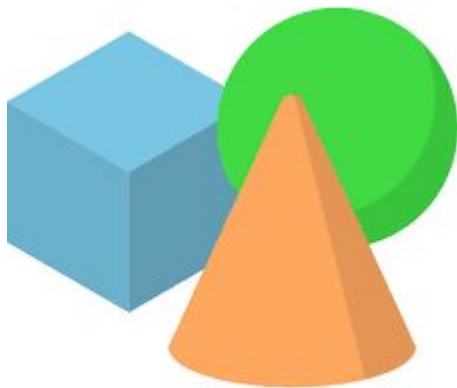
# Objetivos originales de Java

- Orientado a objetos
- Estáticamente tipado
- Simple
- Con soporte para concurrencia
- Con soporte para distribución
- Garbage collection
- Portabilidad



James Gosling

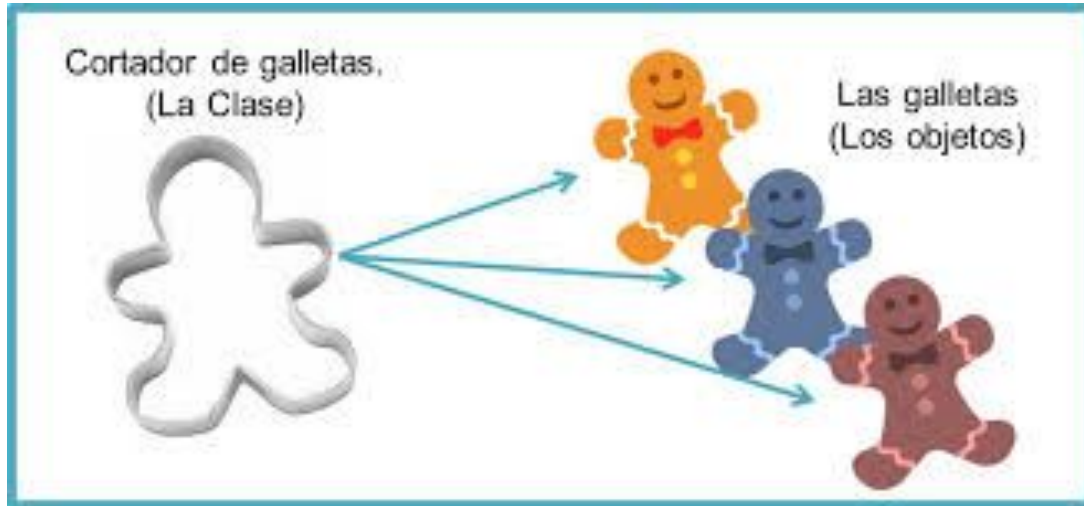
# Objetivos originales del lenguaje



...orientado a objetos

- Clase: Molde
- Objeto: estado (valores para los atributos) y comportamiento
- Protocolo
- Encapsulamiento
- Herencia
- Polimorfismo
- Binding dinámico
- Abstracción

# Repaso P00 - Clase



atributos  
protocolo  
comportamiento

# Java: Clases

```
package ar.edu.unq.po2;

public class Punto {

    //Atributos.
    //El conjunto de valores que tiene la instancia en un
    //momento determinado conforman el estado del objeto
    int x;
    int y;

    //Comportamiento representado con métodos
    public int getX() {
        return x;
    }

    public int getY() {
        return y;
    }

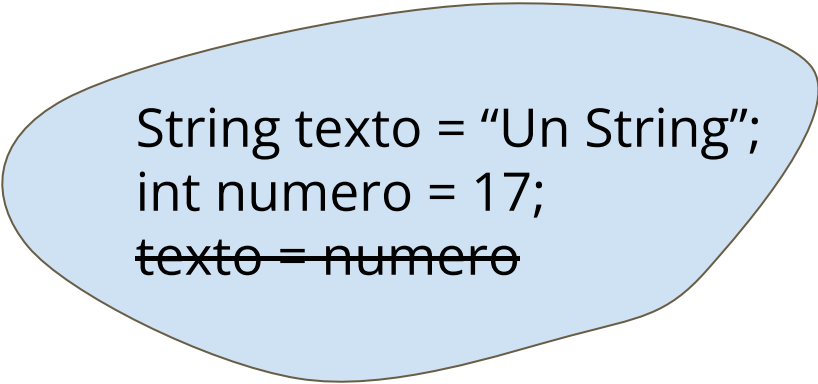
    public void setX(int x) {
        this.x = x;
    }

    public void setY(int y) {
        this.y = y;
    }

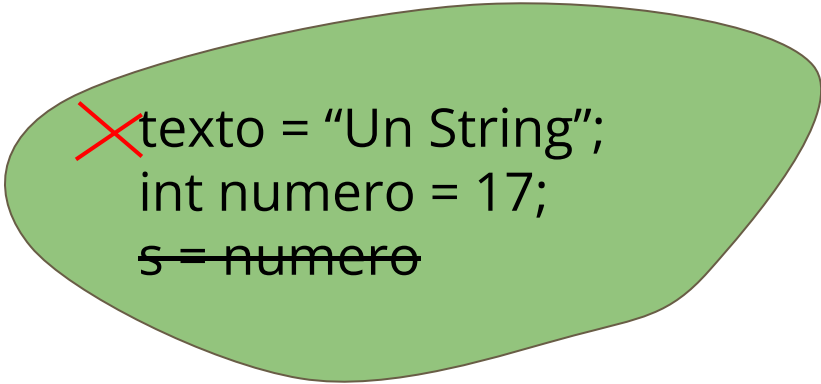
    public Punto abs() {}
}
```

# Objetivos originales del lenguaje

...estáticamente tipado



```
String texto = "Un String";  
int numero = 17;  
texto = numero
```

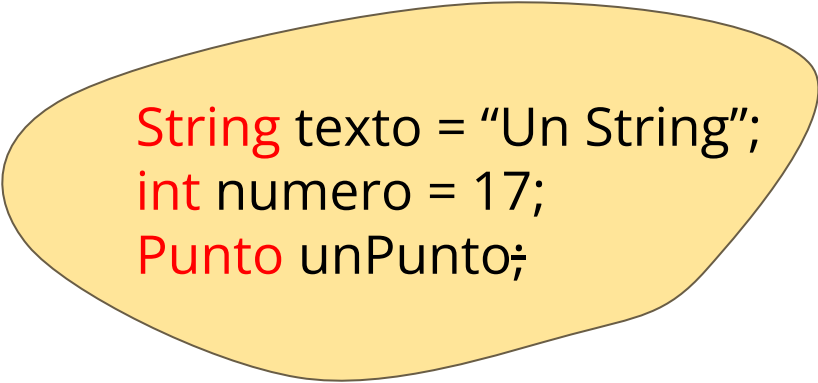


```
× texto = "Un String";  
int numero = 17;  
s = numero
```

Mostrar ejemplo en lenguaje dinámicamente tipado

# Objetivos originales del lenguaje

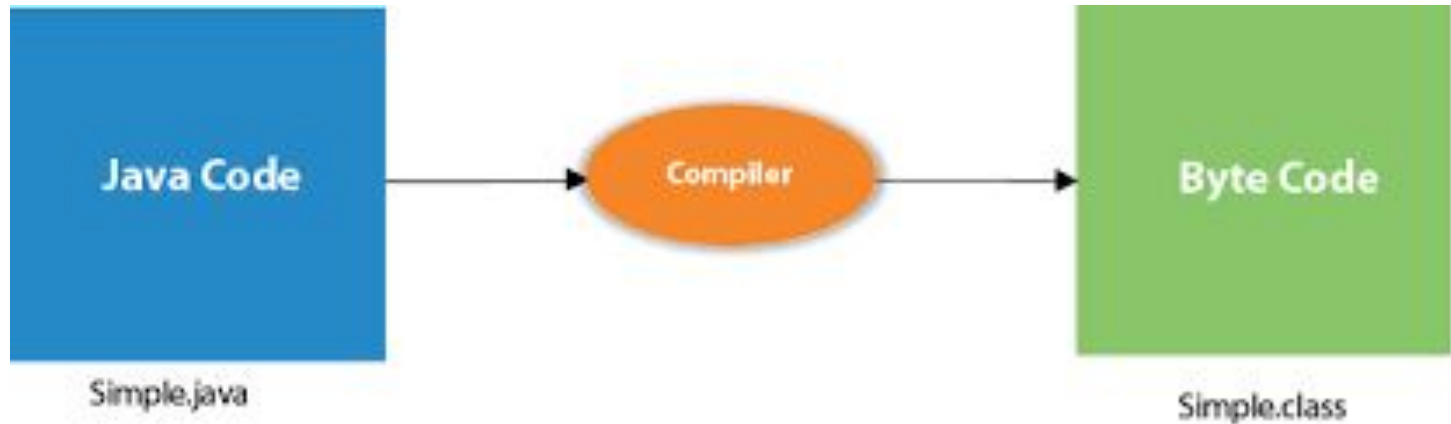
...Con qué puedo tipar mis variables?



```
String texto = "Un String";  
int numero = 17;  
Punto unPunto;
```

# Objetivos originales del lenguaje

...Qué pasa si el tipo no coincide con el valor que asigno? Cuándo se chequea?



# Paquetes

- Agrupa clases con características comunes.
- Convención para el Formato del nombre. De lo más general a lo más particular. Ejemplo: ar.edu.unq.po2....

You form a unique package name by first having (or belonging to an organization that has) an Internet domain name, such as `oracle.com`. You then reverse this name, component by component, to obtain, in this example, `com.oracle`, and use this as a prefix for your package names, using a convention developed within your organization to further administer package names. Such a convention might specify that certain package name components be division, department, project, machine, or login names.

- Se corresponde con folders en el file system.
- Tienen una estructura jerárquica
- Útil para manejar niveles de accesibilidad
- Evita colisión de nombres



# Repaso P00 - P00 en Java

Instanciación de clases

# Java: Instanciación de clases

```
3= /**
4  * Representa un punto en el espacio
5  *
6  */
7  public class Punto {
8
9
10     int x;
11     int y;
12
13
14     //Constructores
15     public Punto(int valorX, int valorY) {
16         this.setValues(valorX, valorY);
17     }
18
19 }
```

class name	Point
super class	Object
instance var	x y
class var	pi
class messages and methods	

newX:xvalue Y:yvalue  
^ self new x: xvalue  
y: yvalue

x: anXNumber y: aYNumber

Set the x and y coordinate to anXNumber and aYNumber, respectively

aPoint <- Point newX:1 Y:2

# Constructores

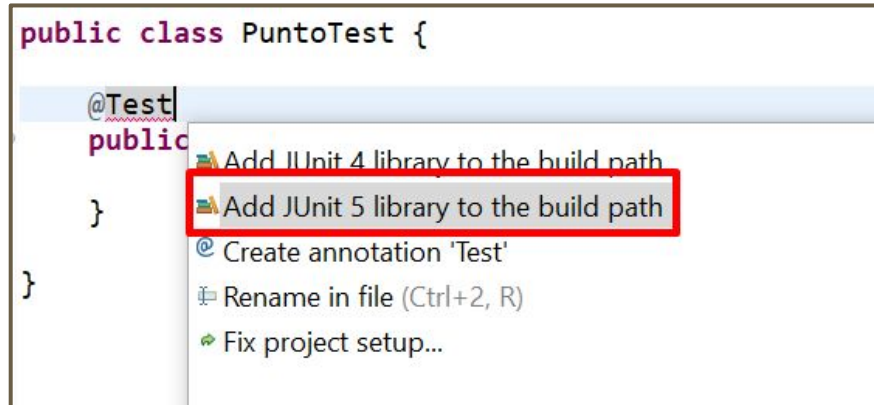
- ¿A toda clase le tengo que definir un constructor?
- ¿Puedo tener más de un constructor?
- Si defino nuevos constructores, sigo teniendo disponible el constructor por defecto?
- ¿Qué pasa si no le pongo el modificador public al constructor?
- ¿Como instancia una clase?

# Tests

- Source folder hermano de src llamado test
- Misma estructura de paquetes que las clases
- Las clases de test llevan el mismo nombre que la clase que testean con la palabra Test al final.

# Tests

- Los métodos que hacen las validaciones se deben marcar con `@Test`. La primera vez importar la librería de JUnit 5 (Jupiter)



# Tests

- El framework de test (JUnit Jupiter) provee métodos de validación. Se llaman asserts.
- Son métodos estáticos que se pueden importar utilizando `import static org.junit.jupiter.api.Assertions.*;`
- Por ahora les va a alcanzar con `assertTrue`, `assertEquals`.

# Tests

- Los assert pueden llevar opcionalmente un mensaje de error como parámetro. Sean claros con los mensajes.
- No puede conocer el orden de ejecución. Cada test debe ser programado de forma independiente a los demás.

# Tests

- Métodos marcados con `@BeforeEach` se ejecuta antes de cada test. Idem los `@AfterEach`.
- Si un mismo objeto lo tengo que usar en más de un método? Dónde se declara? Dónde se inicializa?
- Promoveremos el uso de TDD.



# Repaso P00

Cómo sería la relación mensaje método cuando tengo una relación polimórfica?

# Repaso P00

## Encapsulamiento

¿Qué busco con el encapsulamiento?

# Repaso P00

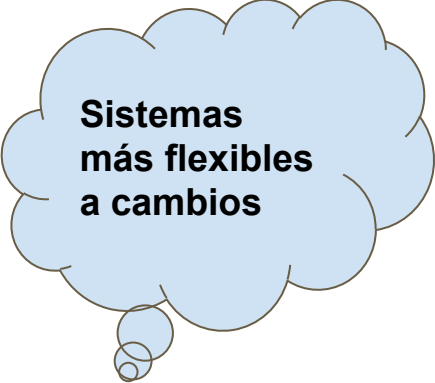
## Encapsulamiento

- Promover sistemas más flexibles al cambio
- Minimizar el riesgo de estados internos inconsistentes

# Java: Herramientas para el Encapsulamiento

**CLASE**  
**PuntoColoreado**

```
3  /**
4   * Representa un punto colerado en el espacio
5   *
6   */
7  public class PuntoColoreado {
8
9      //Estado: representado como variables de instancia
10     public int x;
11     public int y;
12     //Color en inglés
13     public String color;
14 }
```



**Sistemas  
más flexibles  
a cambios**

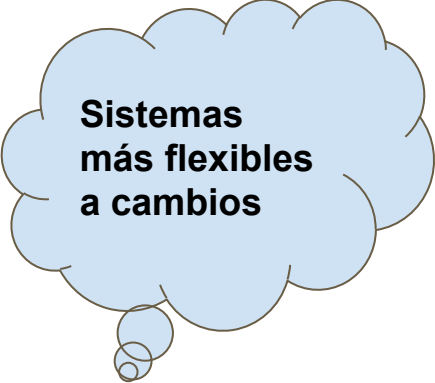
# Java: Herramientas para el Encapsulamiento

## CLASE PuntoColoreado

```
public class PuntoColoreado {  
  
    private int x;  
    private int y;  
  
    //Color en inglés. Por ejemplo: red, black  
    public String color;  
  
    public PuntoColoreado(int x, int y) {  
        super();  
        this.x = x;  
        this.y = y;  
    }  
}
```

## CLASE ClientePuntoColoreado

```
public void rompeEncapsulamientoDePuntoColoreado() {  
  
    PuntoColoreado puntoRojo = new PuntoColoreado(10,10);  
    puntoRojo.color="Red";  
  
}
```



Sistemas  
más flexibles  
a cambios

# Java: Herramientas para el Encapsulamiento

## CLASE ClientePuntoColoreado

```
public void rompeEncapsulamientoDePuntoColoreado2() {  
    PuntoColoreado puntoMarron = new PuntoColoreado(10,10);  
    puntoMarron.color="Brown";  
}
```

## CLASE ClientePuntoColoreado2

```
public void rompeEncapsulamientoDePuntoColoreado3() {  
    PuntoColoreado puntoAzul = new PuntoColoreado(10,10);  
    puntoAzul.color="Blue";  
}
```

## CLASE ClientePuntoColoreado3

```
public void rompeEncapsulamientoDePuntoColoreado() {  
    PuntoColoreado puntoRojo = new PuntoColoreado(10,10);  
    puntoRojo.color="Red";  
}
```

# Java: Herramientas para el Encapsulamiento

```
3= /**
4  * Representa un punto colerado en el espacio
5  *
6  */
7  public class PuntoColoreado {
8
9      //Estado: representado como variables de instancia
10     public int x;
11     public int y;
12     //Color en inglés
13     public String color;
14 }
```



```
4
3= /**
4  * Representa un punto colerado en el espacio
5  *
6  */
7  public class PuntoColoreado {
8
9      //Estado: representado como variables de instancia
10     public int x;
11     public int y;
12     //RGB en Notación Hexadecimal. Ejemplo: #ffffff
13     public String color;
14 }
```

# Java: Herramientas para el Encapsulamiento

```
public void rompeEncapsulamientoDePuntoColoreado2() {  
    PuntoColoreado puntoMarron = new PuntoColoreado(10,10);  
    puntoMarron.color="Brown";  
}
```



```
public void rompeEncapsulamientoDePuntoColoreado3() {  
    PuntoColoreado puntoAzul = new PuntoColoreado(10,10);  
    puntoAzul.color="Blue";  
}
```

```
public void rompeEncapsulamientoDePuntoColoreado() {  
    PuntoColoreado puntoRojo = new PuntoColoreado(10,10);  
    puntoRojo.color="Red";  
}
```





# Java: Herramientas para el Encapsulamiento

```
3= /**
4  * Representa un punto colerado en el espacio
5  *
6  */
7  public class PuntoColoreado {
8
9      //Estado: representado como variables de instancia
10     private int x;
11     private int y;
12     //Color en inglés
13     private String color;
14
15=    public void setColor(String color) {
16        this.color = color;
17    }
18
19=    public String getColor() {
20        return color;
21    }
22 }
```

# Java: Herramientas para el Encapsulamiento

## CLASE ClientePuntoColoreado

```
public void rompeEncapsulamientoDePuntoColoreado() {  
    PuntoColoreado puntoRojo = new PuntoColoreado(10,10);  
    puntoRojo.color;  
}
```

The field PuntoColoreado.color is not visible

2 quick fixes available:

- Change visibility of 'color' to 'package'
- Replace puntoRojo.color with getter

Press 'F2' for focus

```
3 /**  
4  * Representa un punto coloreado en el espacio  
5  *  
6  */  
7 public class PuntoColoreado {  
8  
9     //Estado: representado como variables de instancia  
10    private int x;  
11    private int y;  
12    //Color en inglés  
13    private String color;  
14  
15    public void setColor(String color) {  
16        this.color = color;  
17    }  
18  
19    public String getColor() {  
20        return color;  
21    }  
22 }
```

```
public void rompeEncapsulamientoDePuntoColoreado() {  
    PuntoColoreado puntoRojo = new PuntoColoreado(10,10);  
    puntoRojo.setColor("Red");  
}
```

# Java: Herramientas para el Encapsulamiento

```
public void rompeEncapsulamientoDePuntoColoreado2() {  
    PuntoColoreado puntoMarron = new PuntoColoreado(10,10);  
    puntoMarron.setColor("Brown");  
}
```

```
public void rompeEncapsulamientoDePuntoColoreado3() {  
    PuntoColoreado puntoAzul = new PuntoColoreado(10,10);  
    puntoAzul.setColor("Blue");  
}
```

```
public void rompeEncapsulamientoDePuntoColoreado() {  
    PuntoColoreado puntoRojo = new PuntoColoreado(10,10);  
    puntoRojo.setColor("Red");  
}
```

# Java: Herramientas para el Encapsulamiento

```
1 package ar.edu.unq.po2.tp1;
2
3 /**
4  * Representa un punto colerado en el espacio
5  *
6  */
7 public class PuntoColoreado {
8
9     //Estado: representado como variables de instancia
10    private int x;
11    private int y;
12    //RGB en Notación Hexadecimal. Ejemplo:
13    private String color;
14
15    public void setColor(String color) {
16        this.color = fromEnglishToHex(color);
17    }
18 }
```

# Java: Herramientas para el Encapsulamiento

```
public void rompeEncapsulamientoDePuntoColoreado2() {  
    PuntoColoreado puntoMarron = new PuntoColoreado(10,10);  
    puntoMarron.setColor("Brown");  
}
```



```
public void rompeEncapsulamientoDePuntoColoreado3() {  
    PuntoColoreado puntoAzul = new PuntoColoreado(10,10);  
    puntoAzul.setColor("Blue");  
}
```

```
public void rompeEncapsulamientoDePuntoColoreado() {  
    PuntoColoreado puntoRojo = new PuntoColoreado(10,10);  
    puntoRojo.setColor("Red");  
}
```



# Repaso POO: Ocultar estructura interna

- Alcanza con disponer las variables privadas? No! El diseño también debe ser acorde.



## Evaluación de protocolos de una clase



Sea la clase Rectángulo con 4 variables de instancia (esquinaSuperiorIzquierda, esquinaSuperiorDerecha, esquinaInferiorIzquierda y esquinaInferiorDerecha). Elija uno de los protocolos presentados a continuación y justifique su elección. Recuerde que el protocolo es el conjunto de mensajes que entiende una clase o tipo.

## Encuesta

**Minimizar el  
riesgo de  
estados  
internos  
inconsistentes**

### Opción 1)

```
Class Rectangulo>>new  
Rectangulo>>esquinaSuperiorIzquierda: unPunto  
Rectangulo>>esquinaSuperiorDerecha: unPunto  
Rectangulo>>esquinaInferiorIzquierda: unPunto  
Rectangulo>>esquinaInferiorDerecha: unPunto
```

### Opcion 2)

```
Class Rectangulo>>newEnOrigenEsquinaSuperiorIzquierda: unPunto alto: unNumero ancho:unNumero  
Rectangulo>>reubicarEsquinaSuperiorIzquierdaEn: unPunto  
Rectangulo>>ancho: unNumero  
Rectangulo>>alto: unNumero
```

# Repaso POO: Ocultar estructura interna

- Alcanza con disponer las variables privadas? No! El diseño también debe ser acorde.



## Evaluación de protocolos de una clase



Sea la clase Rectángulo con 4 variables de instancia (esquinaSuperiorIzquierda, esquinaSuperiorDerecha, esquinaInferiorIzquierda y esquinaInferiorDerecha). Elija uno de los protocolos presentados a continuación y justifique su elección. Recuerde que el protocolo es el conjunto de mensajes que entiende una clase o tipo.

## Encuesta

Minimizar el  
riesgo de  
estados  
internos  
inconsistentes

Opción 1)

```
public class RectanguloNoEncapsulado
```

```
Class Rectangulo>>new
```

```
Rectangulo>>esquinaSuperiorIzquierda: unPunto
```

```
Rectangulo>>esquinaSuperiorDerecha: unPunto
```

```
Rectangulo>>esquinaInferiorIzquierda: unPunto
```

```
Rectangulo>>esquinaInferiorDerecha: unPunto
```

Opcion 2)

```
public class Rectangulo
```

```
Class Rectangulo>>newEnOrigenEsquinaSuperiorIzquierda: unPunto alto: unNumero ancho:unNumero
```

```
Rectangulo>>reubicarEsquinaSuperiorIzquierdaEn: unPunto
```

```
Rectangulo>>ancho: unNumero
```

```
Rectangulo>>alto: unNumero
```

# Repaso POO: Ocultar estructura interna

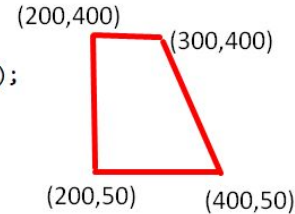
Alcanza con disponer las variables privadas? No! El diseño también debe ser acorde.

OPCION1

```
public void dibujarRectanguloNoEncapsulado(Graphics g) {  
    RectanguloNoEncapsulado r = new RectanguloNoEncapsulado();  
    r.setEsquinaSuperiorDerecha(new Punto(150,400));  
    r.setEsquinaInferiorDerecha(new Punto(150,50));  
    r.setEsquinaSuperiorIzquierda(new Punto(50,400));  
    r.setEsquinaInferiorIzquierda(new Punto(50,50));  
    r.dibujar(g);  
}
```

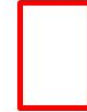


```
public void dibujarMALRectanguloNoEncapsulado(Graphics g) {  
    RectanguloNoEncapsulado r = new RectanguloNoEncapsulado();  
    r.setEsquinaSuperiorDerecha(new Punto(300,400));  
    r.setEsquinaInferiorDerecha(new Punto(400,50));  
    r.setEsquinaSuperiorIzquierda(new Punto(200,400));  
    r.setEsquinaInferiorIzquierda(new Punto(200,50));  
    r.dibujar(g);  
}
```



OPCION2

```
public void dibujarRectanguloEncapsulado(Graphics g) {  
    Rectangulo r = new Rectangulo(new Punto(450,400),350,100);  
    r.dibujar(g);  
}
```



```
public void dibujarMALRectanguloEncapsulado(Graphics g) {  
    //NO PUEDO!!!  
}
```



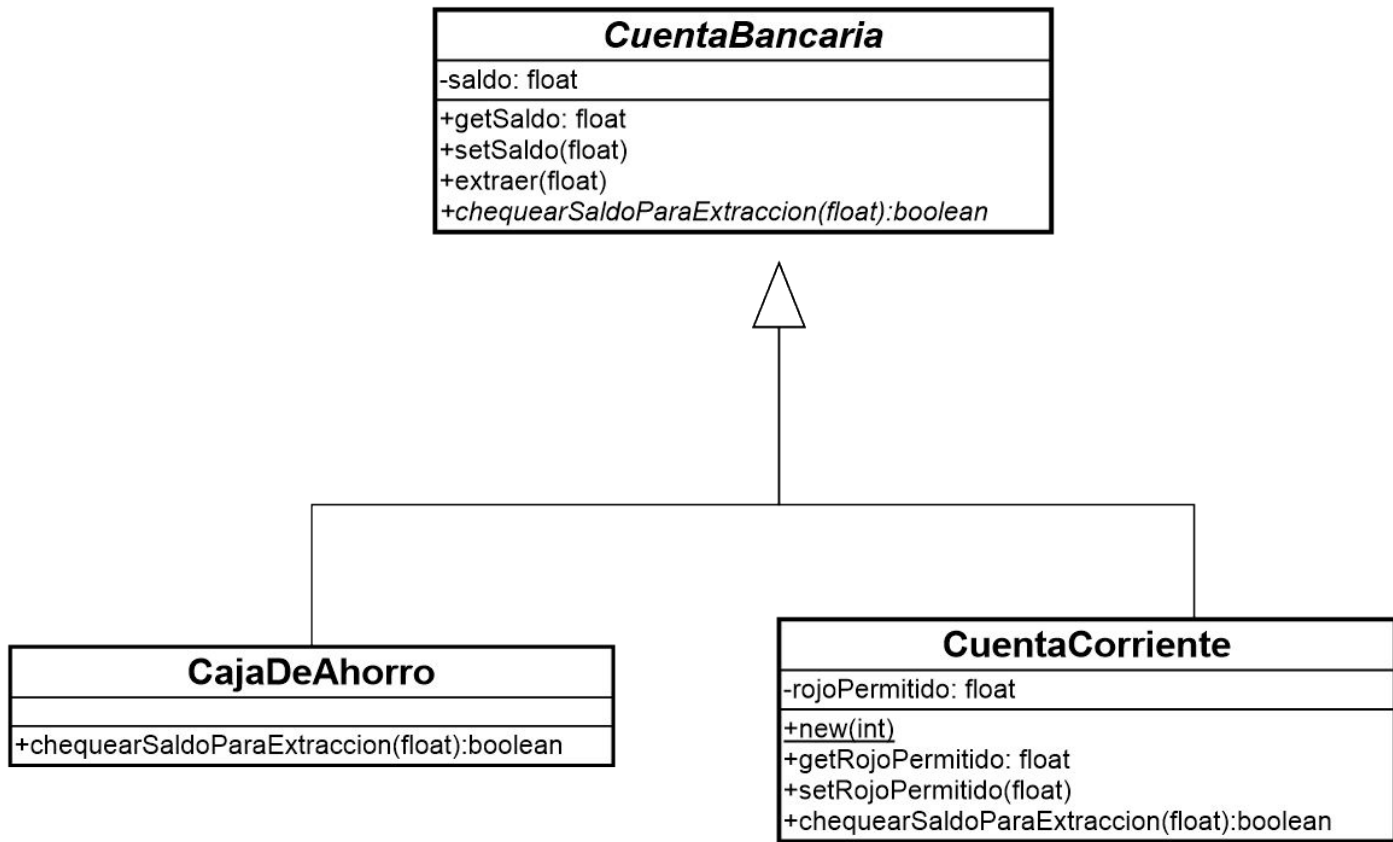
# Repaso P00: Ocultar estructura interna

¿Qué formas de representar el rectángulo encontraron?

# Repaso P00

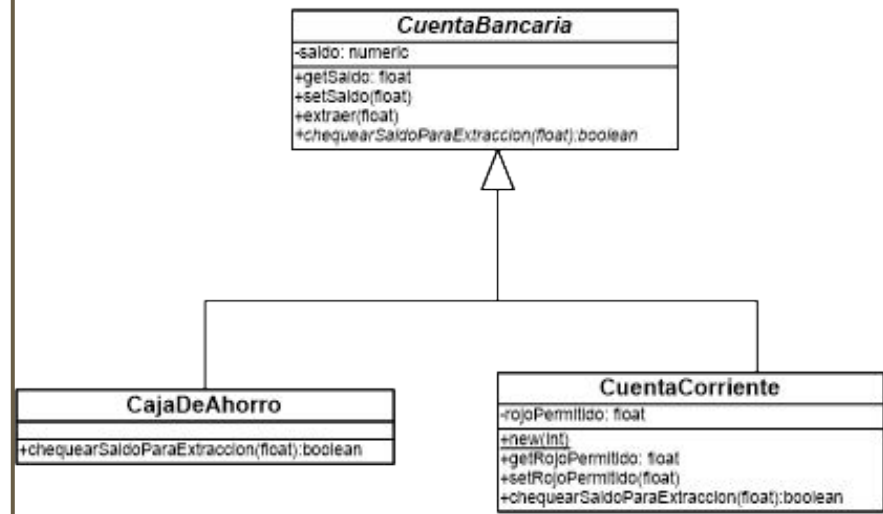
**Herencia**

# P00 en Java: Herencia



# P00 en Java: Herencia

Ejercicio 4 - Opción 4



```
package ar.edu.unq.po2.tp1.herencia;

public class CajaDeAhorro extends CuentaBancaria{

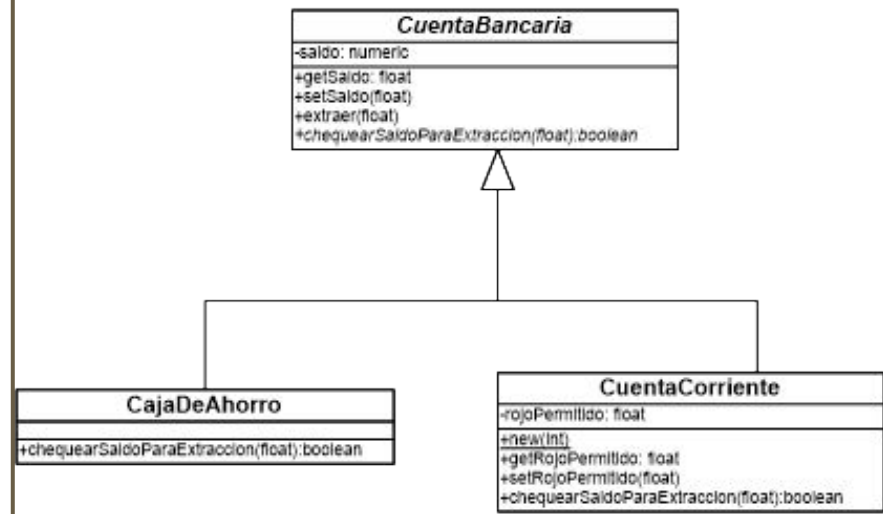
    @Override
    boolean chequearSaldoParaExtraccion(float unMonto) {
        return this.getSaldo()>=unMonto;
    }

}
```

¿Es obligatorio el override?

# P00 en Java: Herencia

Ejercicio 4 - Opción 4



```
package ar.edu.unq.po2.tp1.herencia;

public class CajaDeAhorro extends CuentaBancaria{

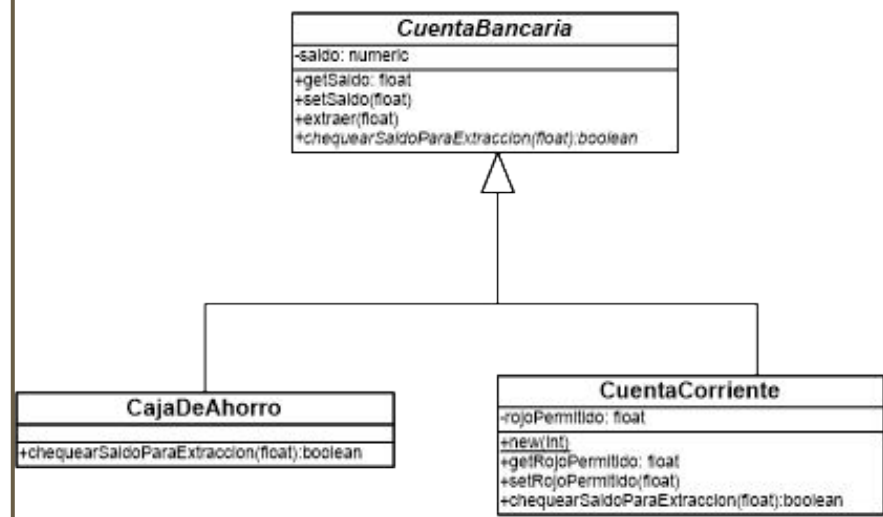
    @Override
    boolean chequearSaldoParaExtraccion(float unMonto) {
        return this.getSaldo()>=unMonto;
    }

}
```

¿Es obligatorio el override?

# P00 en Java: Herencia

Ejercicio 4 - Opción 4



```
package ar.edu.unq.po2.tp1.herencia;

public class CuentaCorriente extends CuentaBancaria{

    private int rojoPermitido;

    @Override
    boolean chequearSaldoParaExtraccion(float unMonto) {
        return this.getSaldo()+this.getRojoPermitido()>=unMonto;
    }

    protected int getRojoPermitido() {
        return rojoPermitido;
    }

    protected void setRojoPermitido(int rojoPermitido) {
        this.rojoPermitido = rojoPermitido;
    }

    ...
}
```

# P00 en Java: Herencia

- ¿Diferencia entre sobrescribir y sobrecargar un método?

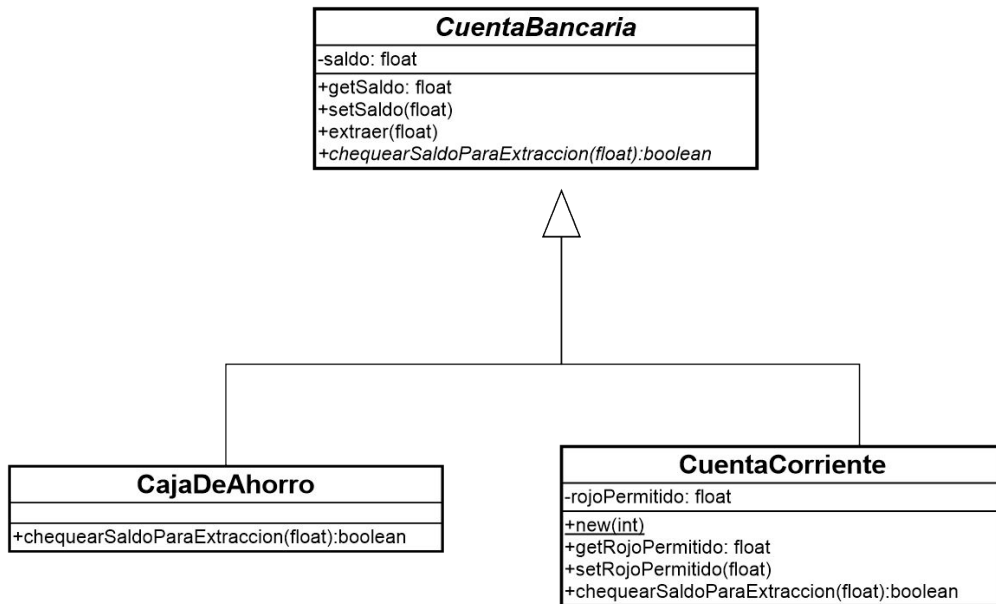
# P00 en Java: Herencia

¿Los constructores se heredan?



# P00 en Java: Herencia

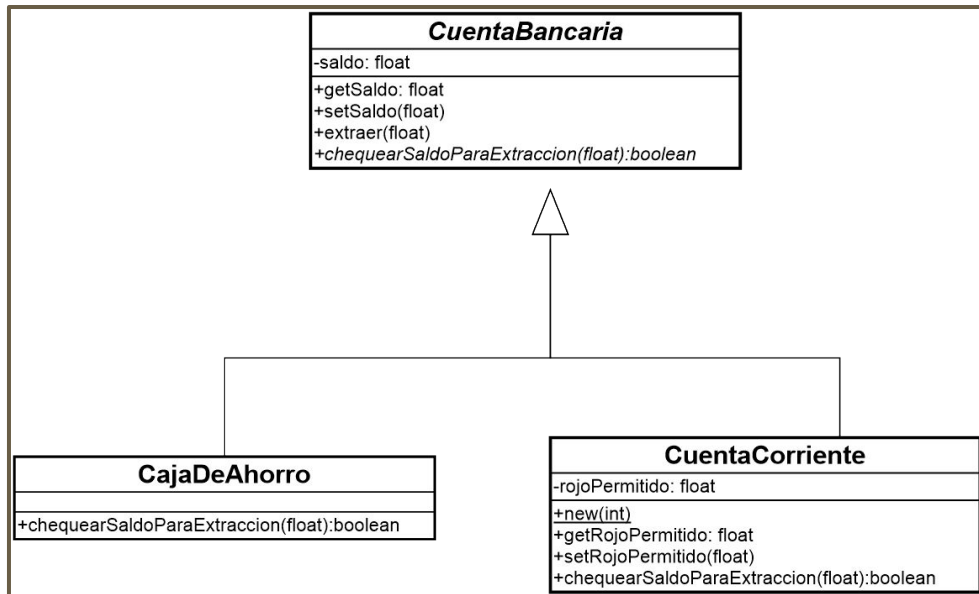
UML: tengo que volver a escribir los metodos en las subclases?



# Repaso P00

## Clases y métodos abstractos

# P00 en Java: Herencia. Clases métodos abstractos.



```
package ar.edu.unq.po2.tp1.herencia;

public abstract class CuentaBancaria {

    private float saldo=0;

    public void extraer(float unMonto) {
        if (chequearSaldoParaExtraccion(unMonto)) {
            this.setSaldo(this.getSaldo()-unMonto);
        }
    }

    abstract boolean chequearSaldoParaExtraccion(float unMonto);

    public float getSaldo() {
        return saldo;
    }

    public void setSaldo(float saldo) {
        this.saldo = saldo;
    }

}
```

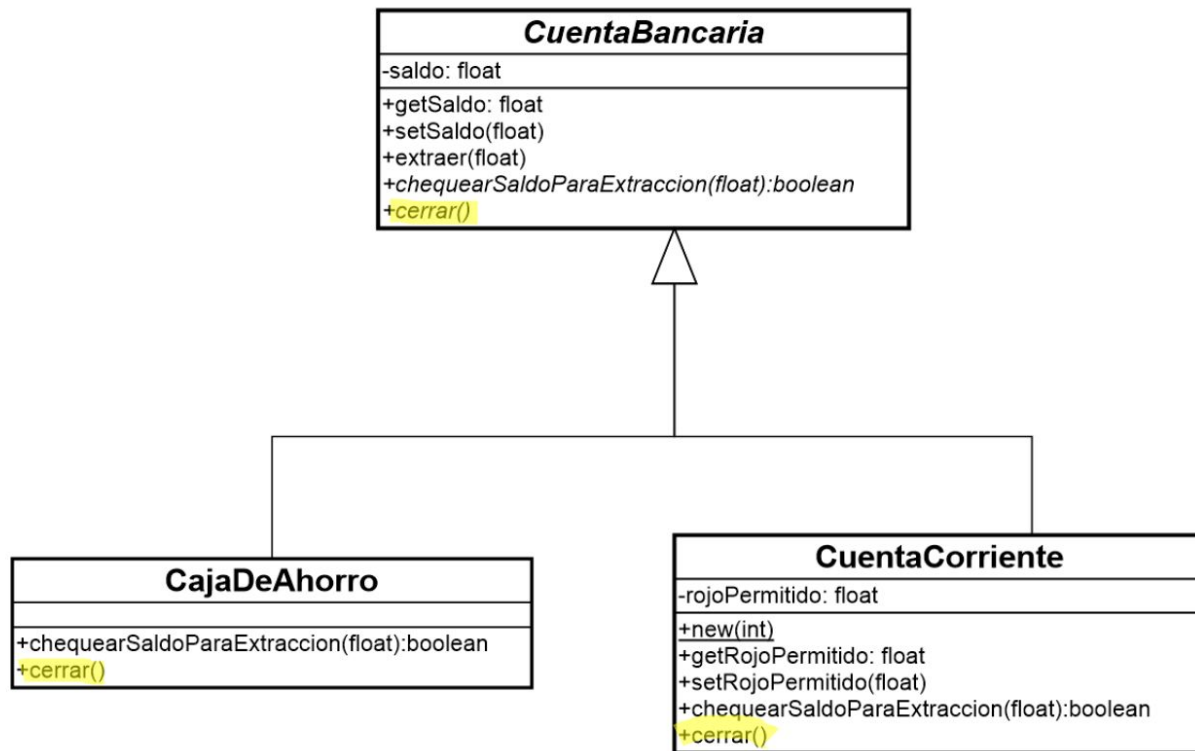
# Clases y métodos abstractos

- Una clase abstracta puede tener métodos con comportamiento (no abstractos)?
- Una clase concreta (no abstracta) puede tener métodos abstractos?

# Repaso P00

## Polimorfismo

# P00 en Java: polimorfismo



# P00 en Java: Polimorfismo

```
public abstract class CuentaBancaria {  
  
    public abstract void cerrar();  
}
```

```
public class CuentaCorriente extends CuentaBancaria{  
  
    @Override  
    public void cerrar() {  
        //La Cuenta corriente tiene una lógica propia para su cierre  
        //Asumimos que es totalmente diferente a la de la CajaDeAhorro  
    }  
}
```

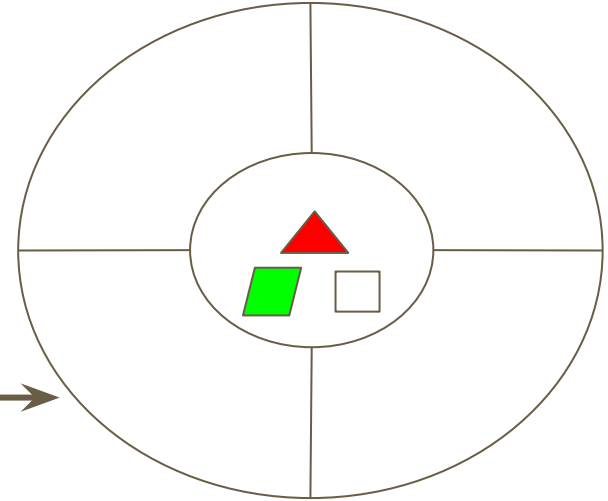
```
public class CajaDeAhorro extends CuentaBancaria{  
  
    @Override  
    public void cerrar() {  
        //La Caja de Ahorro tiene una lógica propia para su cierre  
        //Asumimos que es totalmente diferente a la de la CuentaCorriente  
    }  
}
```

```
public class Banco {  
  
    public void cerrarCuenta(int numeroCuenta) {  
        CuentaBancaria cuenta = this.buscarCuenta(numeroCuenta);  
        cuenta.cerrar();  
    }  
}
```

**BANCO**

mensaje  
**cerrar()**

**CUENTA BANCARIA**





```
public class Banco {  
  
    public void cerrarCuenta(int numeroCuenta) {  
        CuentaBancaria cuenta = this.buscarCuenta(numeroCuenta);  
        cuenta.cerrar();  
    }  
}
```

## CUENTA BANCARIA

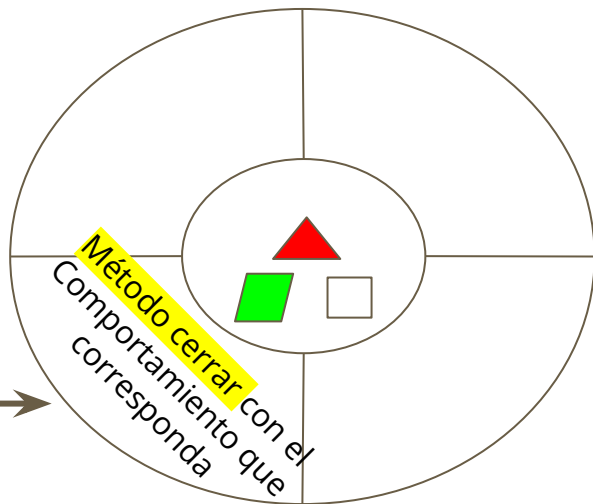
### BANCO

mensaje

**cerrar()**

mensaje

**cerrar()**



MÉTODO cerrar  
CA

CA

MÉTODO cerrar  
CC

CC

# P00 en Java: Polimorfismo

```
public class Banco {  
  
    public void cerrarCuenta(int numeroCuenta) {  
        CuentaBancaria cuenta = this.buscarCuenta(numeroCuenta);  
        cuenta.cerrar();  
    }  
}
```

**MENSAJE**

```
public class CajaDeAhorro extends CuentaBancaria{
```

```
@Override
```

```
public void cerrar() {
```

**METODO**

```
//La Caja de Ahorro tiene una lógica propia para su cierre
```

```
//Asumimos que es totalmente diferente a la de la CuentaCorriente
```

```
}
```

```
public class CuentaCorriente extends CuentaBancaria{
```

```
@Override
```

```
public void cerrar() {
```

**METODO**

```
//La Cuenta corriente tiene una lógica propia para su cierre
```

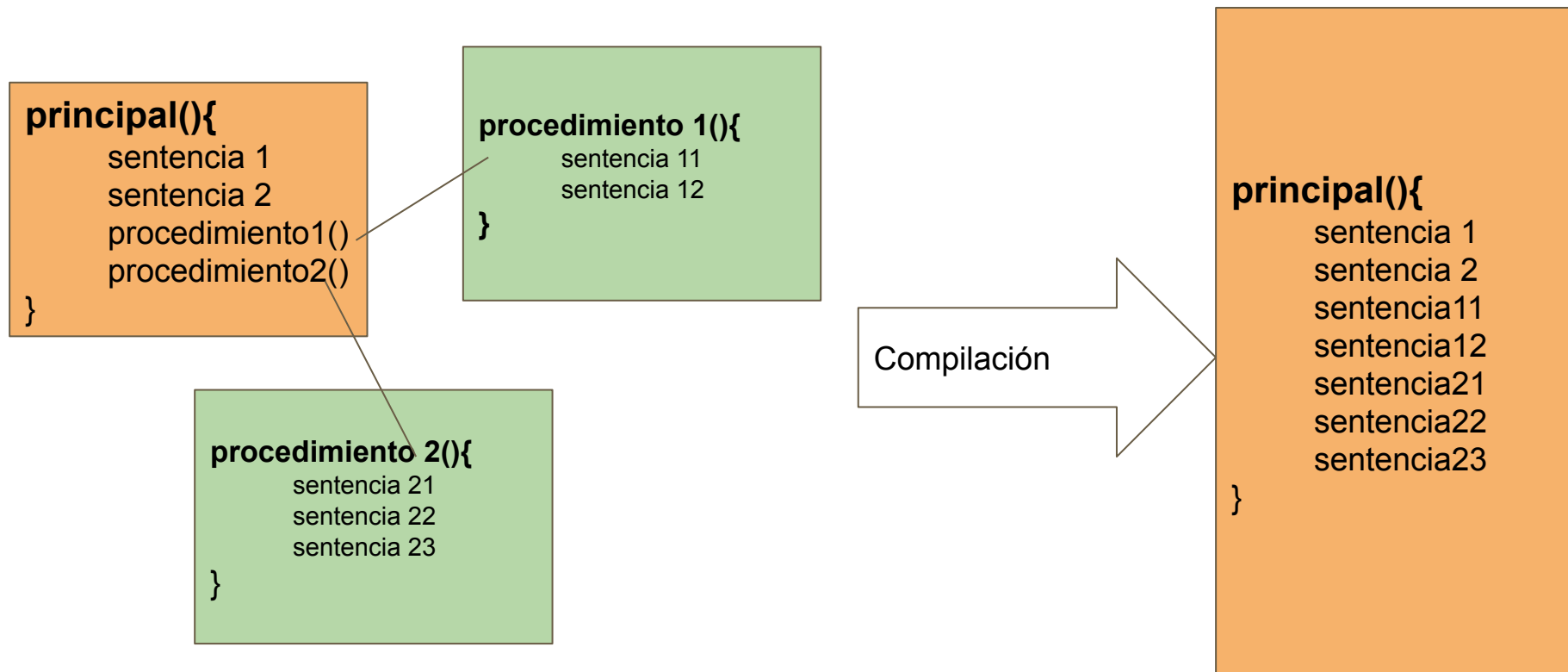
```
//Asumimos que es totalmente diferente a la de la CajaDeAhorro
```

```
}
```

# Repaso P00

**Binding dinámico**

# P00 en Java: Binding estático vs. Binding dinámico



# P00 en Java: Binding dinámico

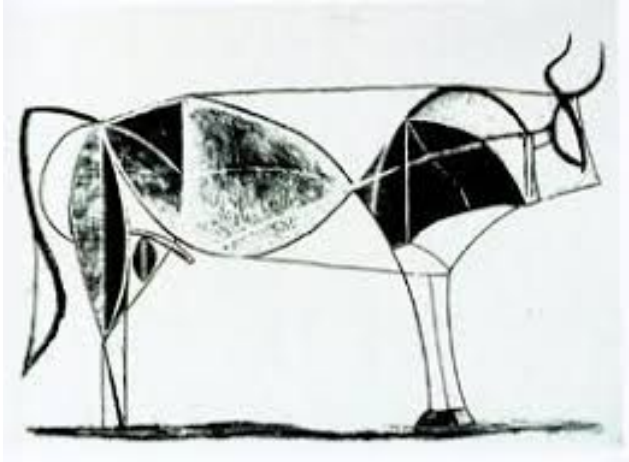
```
public class Banco {  
  
    List<CuentaBancaria> cuentas = new ArrayList<CuentaBancaria>();  
  
    public void cerrarCuentas() {  
        //cuentas.forEach((CuentaBancaria cuenta) -> cuenta.cerrar());  
        for (CuentaBancaria cuentaBancaria : cuentas) {  
            cuentaBancaria.cerrar();  
        }  
    }  
}
```

**MENSAJE**

```
public class CajaDeAhorro extends CuentaBancaria{  
  
    @Override  
    public void cerrar() { METODO  
        //La Caja de Ahorro tiene una lógica propia para su cierre  
        //Asumimos que es totalmente diferente a la de la CuentaCorriente  
    }  
}
```

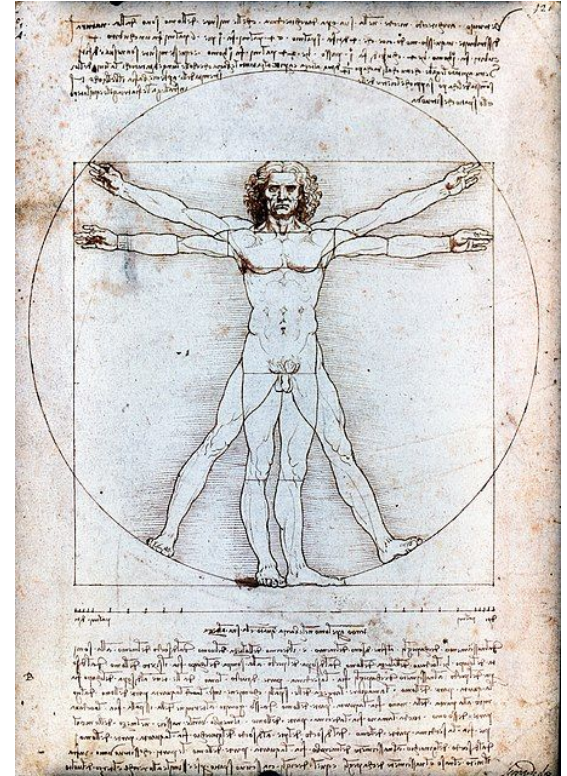
```
public class CuentaCorriente extends CuentaBancaria{  
  
    @Override  
    public void cerrar() { METODO  
        //La Cuenta corriente tiene una lógica propia para su cierre  
        //Asumimos que es totalmente diferente a la de la CajaDeAhorro  
    }  
}
```

# Repaso POO - Abstracción



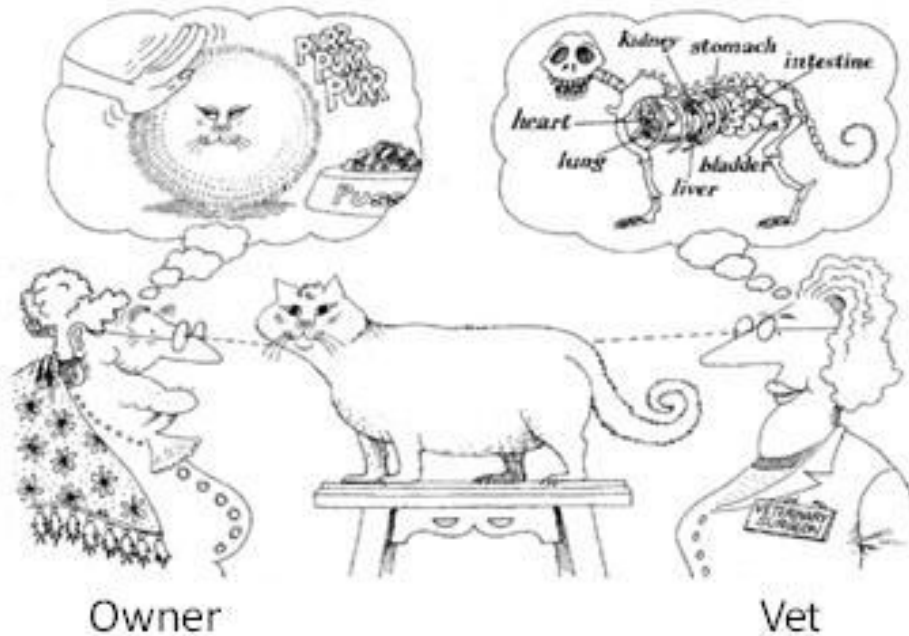
Pablo Picasso, Bull - plate December 1945

**Recomendación: Chapter 2 (Abstraction) del libro  
“Budd - Intro to OOP”**



Hombre Vitruvio, da Vinci 1490

# Repaso P00 - Abstracción



# Conceptos vistos

- Clases
- Constructores
- Mensaje vs. método
- Encapsulamiento
- Abstracción
- Herencia
- Clases abstractas y métodos abstractos
- Polimorfismo



# Objetivos originales de Java

- Orientado a objetos
- Estáticamente tipado
- Simple
- Con soporte para concurrencia
- Con soporte para distribución
- Garbage collection
- Portabilidad



James Gosling

# Objetivos originales del lenguaje

...simple



# Objetivos originales del lenguaje

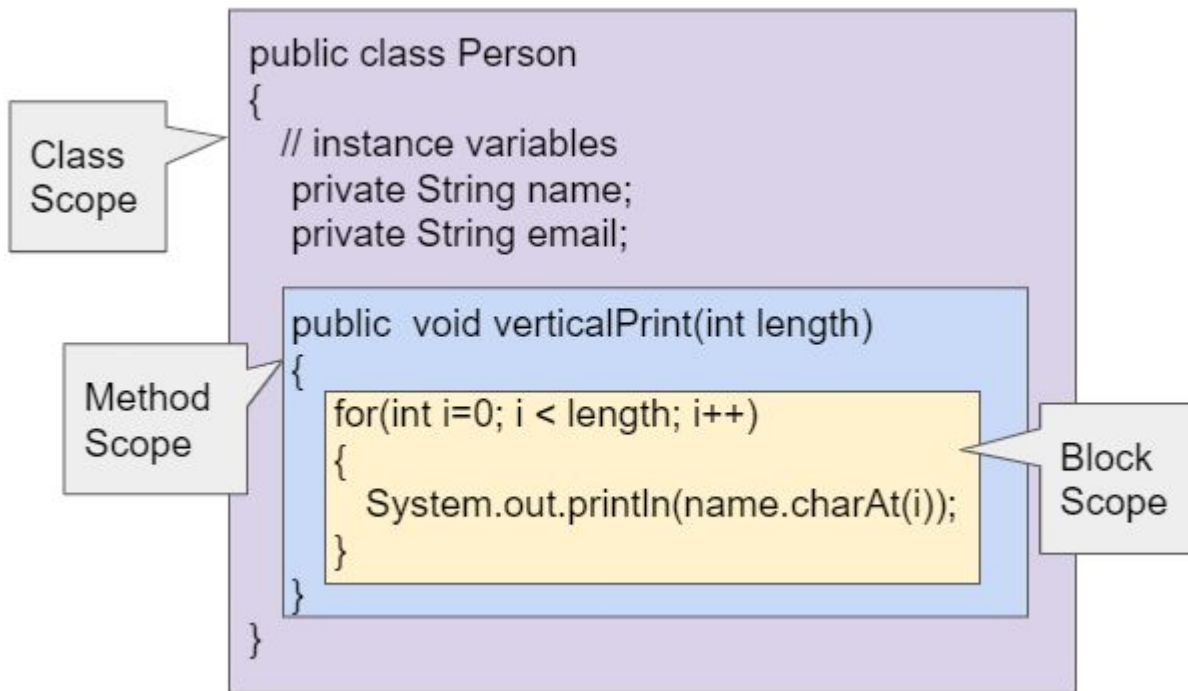
...garbage collection



# Objetivos originales del lenguaje

...garbage collection y scopes

```
public class Ciclista {  
    public float calcularVelocidad()  
    {  
        float velocidad=1;  
        //calcula de velocidad  
        return velocidad;  
    }  
}
```



# Objetivos originales del lenguaje

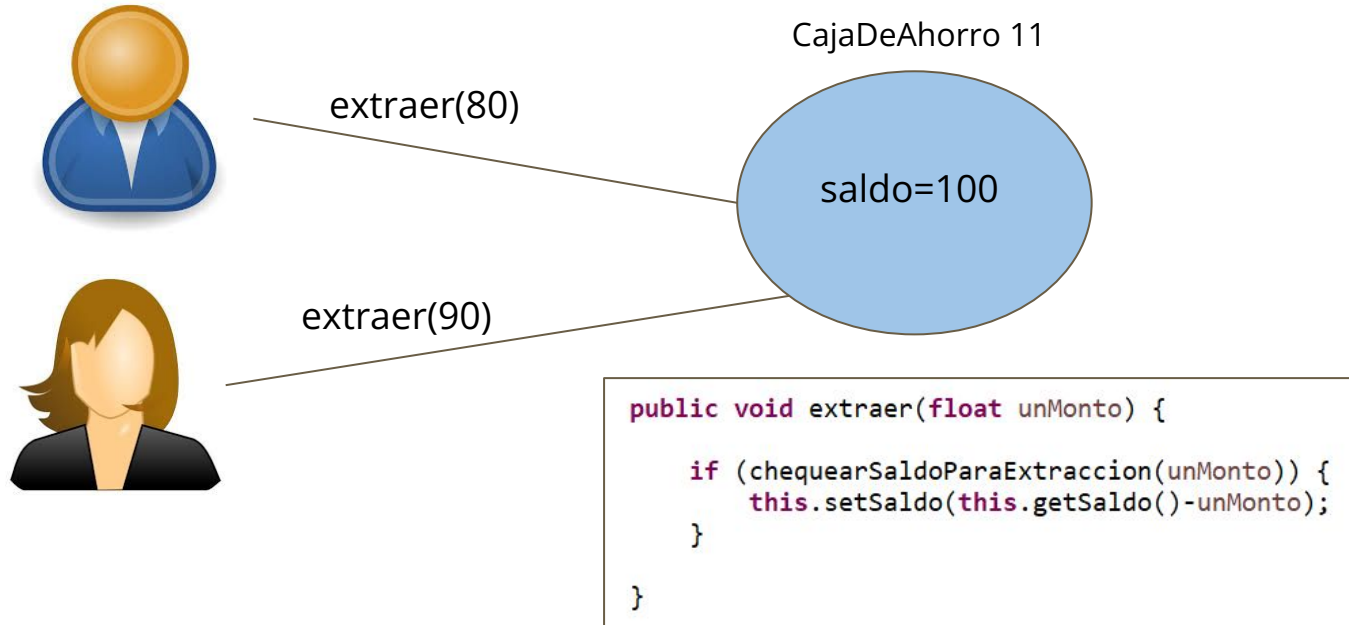
...con soporte para concurrencia

Si no la controlo:  
posibles Inconsistencias de datos

Si la controlo mal: deadlock




# Objetivos originales del lenguaje: Soporte para manejo de concurrencia



# Objetivos originales del lenguaje: Soporte para manejo de concurrencia

CajaDeAhorro 11

saldo=100

```
public void extraer(float unMonto) {  
    if (chequearSaldoParaExtraccion(unMonto)) {  
         extraer(80)  
  
        this.setSaldo(this.getSaldo()-unMonto);  
    }  
}
```

# Objetivos originales del lenguaje: Soporte para manejo de concurrencia

CajaDeAhorro 11

saldo=100

```
public void extraer(float unMonto) {  
    if (chequearSaldoParaExtraccion(unMonto)) {  
  
        this.setSaldo(this.getSaldo()-unMonto);  
  
    }  
}
```



**extraer(80)**



# Objetivos originales del lenguaje: Soporte para manejo de concurrencia

CajaDeAhorro 11

saldo=100

```
public void extraer(float unMonto) {
```

```
    if (chequearSaldoParaExtraccion(unMonto)) {
```



**extraer(90)**

```
        this.setSaldo(this.getSaldo()-unMonto);
```



**extraer(80)**

```
    }
```

```
}
```

# Objetivos originales del lenguaje: Soporte para manejo de concurrencia

CajaDeAhorro 11

saldo=100

```
public void extraer(float unMonto) {  
    if (chequearSaldoParaExtraccion(unMonto)) {  
  
        this.setSaldo(this.getSaldo()-unMonto);  
  
    }  
}
```



**extraer(80)**



**extraer(90)**

# Objetivos originales del lenguaje: Soporte para manejo de concurrencia

CajaDeAhorro 11

saldo=10

```
public void extraer(float unMonto) {  
    if (chequearSaldoParaExtraccion(unMonto)) {  
  
        this.setSaldo(this.getSaldo()-unMonto);  
  
    }  
}
```



extraer(80)



extraer(90)



# Objetivos originales del lenguaje: Soporte para manejo de concurrencia

CajaDeAhorro 11

saldo=-70

```
public void extraer(float unMonto) {  
    if (chequearSaldoParaExtraccion(unMonto)) {  
  
        this.setSaldo(this.getSaldo()-unMonto);  
  
    }  
}
```



**extraer(90)**



**extraer(80)**



# Objetivos originales de Java

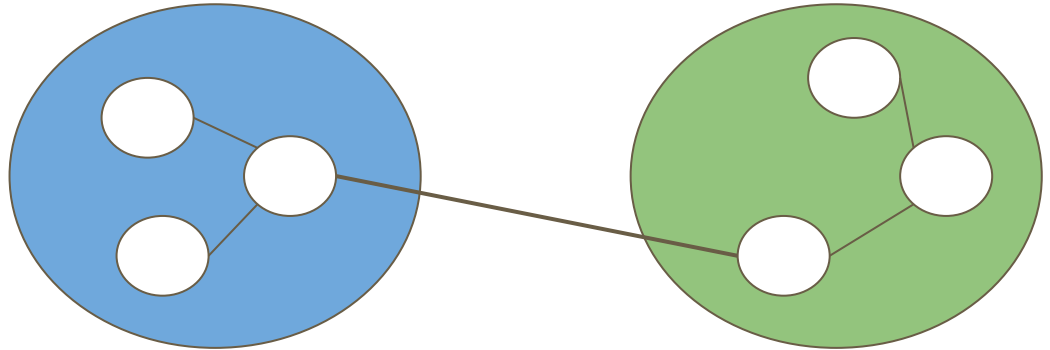
- Orientado a objetos
- Estáticamente tipado
- Simple
- Garbage collection
- Con soporte para concurrencia
- Con soporte para distribución
- Portabilidad



James Gosling

# Objetivos originales del lenguaje

...con soporte para distribución



# Objetivos originales de Java

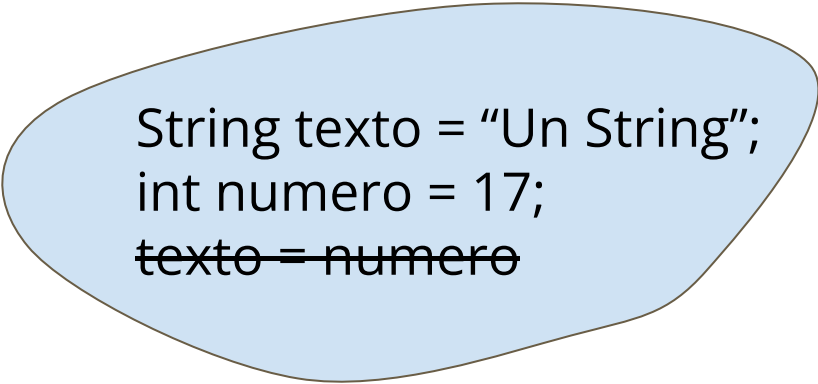
- Orientado a objetos
- Simple
- Garbage collection
- Con soporte para concurrencia
- Con soporte para distribución
- Estáticamente tipado
- Portabilidad



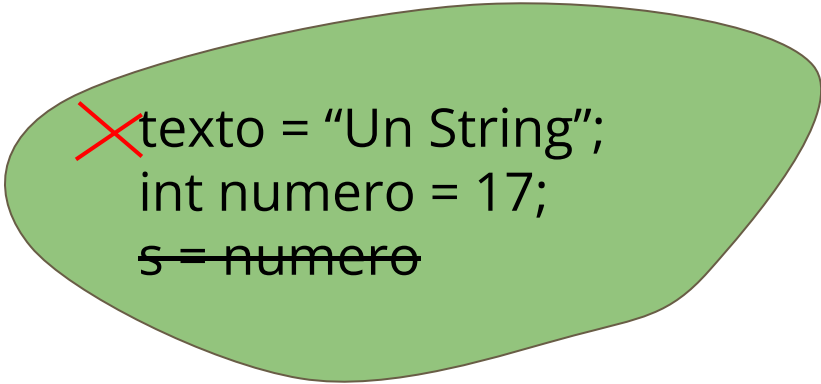
James Gosling

# Objetivos originales del lenguaje

...estáticamente tipado



```
String texto = "Un String";  
int numero = 17;  
texto = numero
```



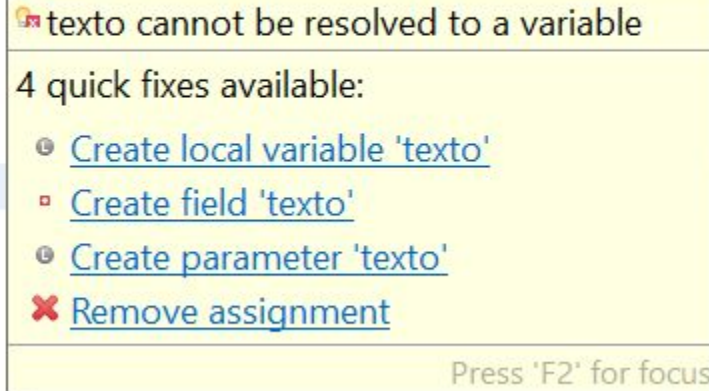
```
× texto = "Un String";  
int numero = 17;  
s = numero
```



# Objetivos originales del lenguaje

...estáticamente tipado. Los tipos se chequean en tiempo de compilación

```
public void estaticamenteTipado() {  
    texto="String";  
}
```



✖ texto cannot be resolved to a variable

4 quick fixes available:

- ④ [Create local variable 'texto'](#)
- ▣ [Create field 'texto'](#)
- ④ [Create parameter 'texto'](#)
- ✖ [Remove assignment](#)

Press 'F2' for focus

# Objetivos originales de Java

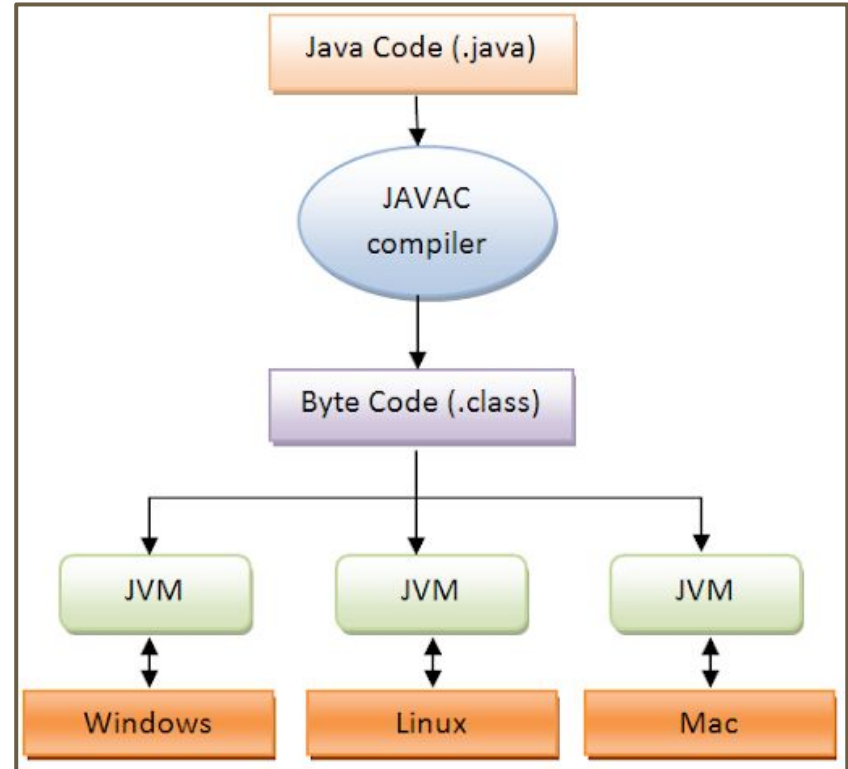
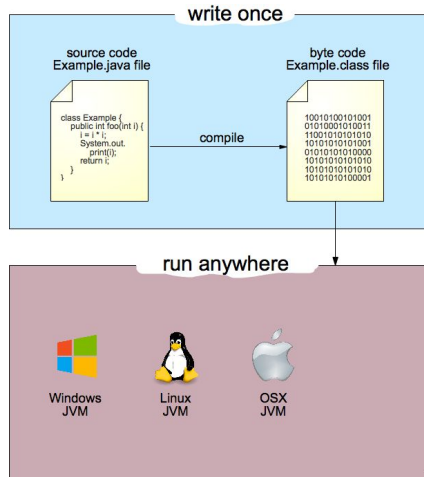
- Orientado a objetos
- Simple
- Garbage collection
- Con soporte para concurrencia
- Con soporte para distribución
- Estáticamente tipado
- Portabilidad



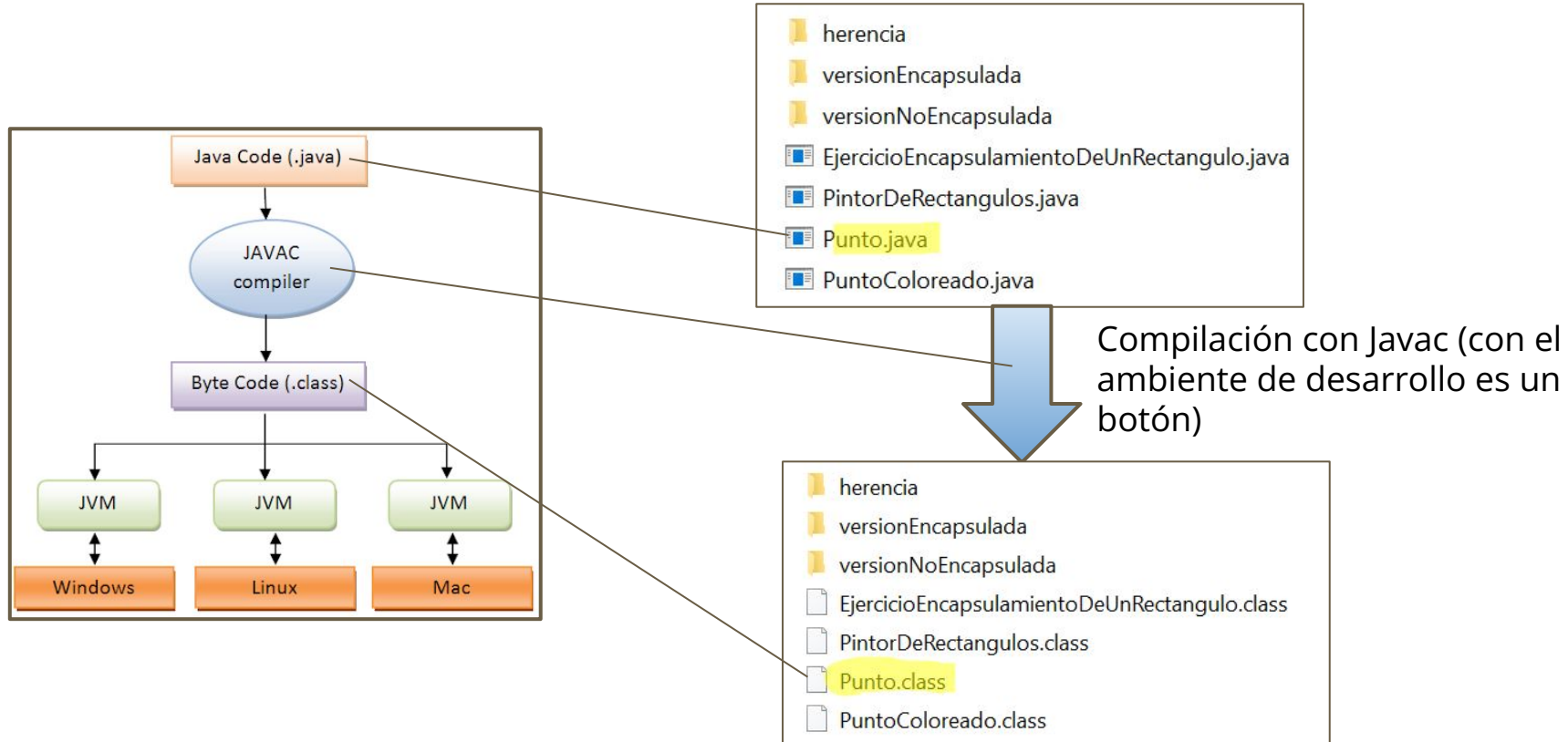
James Gosling

# Objetivos originales del lenguaje

...portabilidad



# Objetivos originales del lenguaje: Portabilidad



## **PARTE 2 - COMPONENTES DE LA PLATAFORMA**

- **1-Lenguaje**
- **2-JDK: Java development toolkit**
- **3-JRE: Java Runtime Environment**

# ¿De qué se compone Java?: 1-Lenguaje



## The Java® Language Specification *Java SE 19 Edition*

James Gosling  
Bill Joy  
Guy Steele  
Gilad Bracha  
Alex Buckley  
Daniel Smith  
Gavin Bierman

# ¿De qué se compone Java?: 1-Lenguaje

## 8.1. Class Declarations

A class declaration specifies a new named reference type.

There are two kinds of class declarations: *normal class declarations* and *enum declarations*.

*ClassDeclaration:*

[\*NormalClassDeclaration\*](#)

[\*EnumDeclaration\*](#)

*NormalClassDeclaration:*

[\*{ClassModifier}\*](#) class [\*TypeIdentifier\*](#) [\*\[TypeParameters\]\*](#) [\*\[Superclass\]\*](#) [\*\[Superinterfaces\]\*](#) [\*ClassBody\*](#)

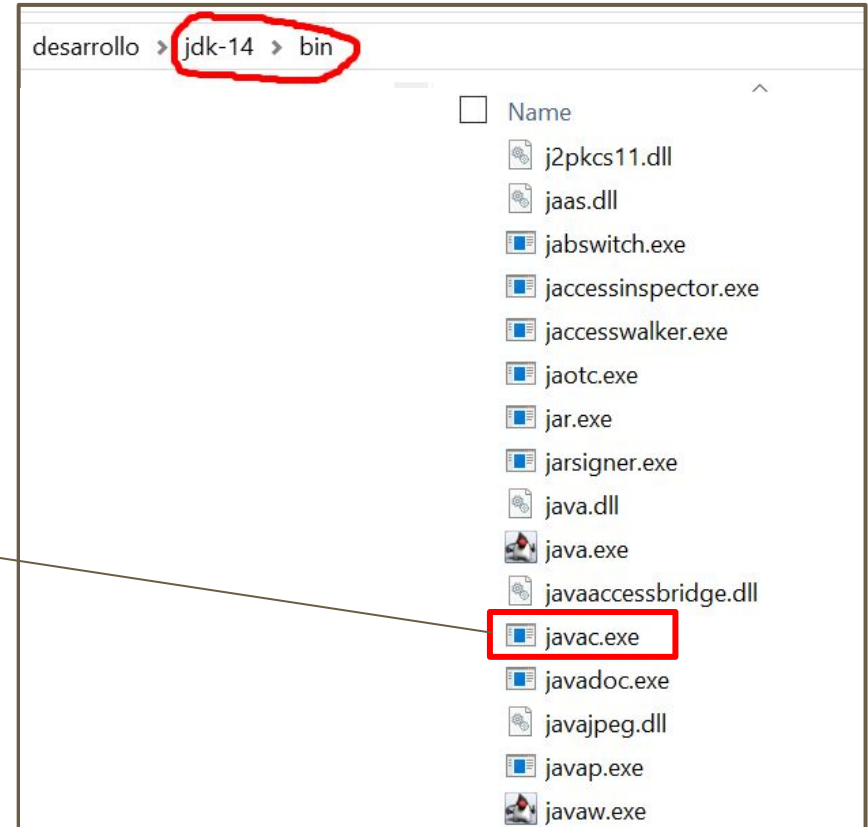
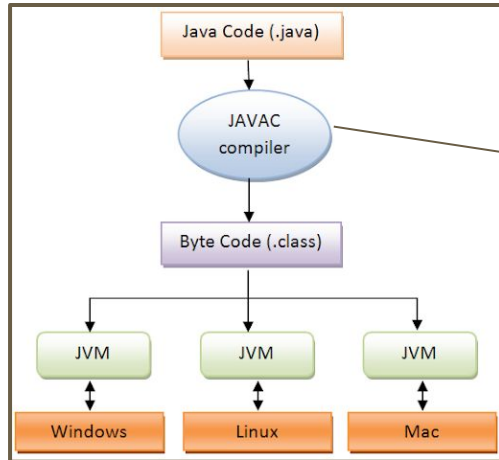
The rules in this section apply to all class declarations, including enum declarations. However, special rules apply to enum declarations with regard to class modifiers, inner classes, and superclasses; these rules are stated in [§8.9](#).

The *TypeIdentifier* in a class declaration specifies the name of the class.

**It is a compile-time error if a class has the same simple name as any of its enclosing classes or interfaces.**

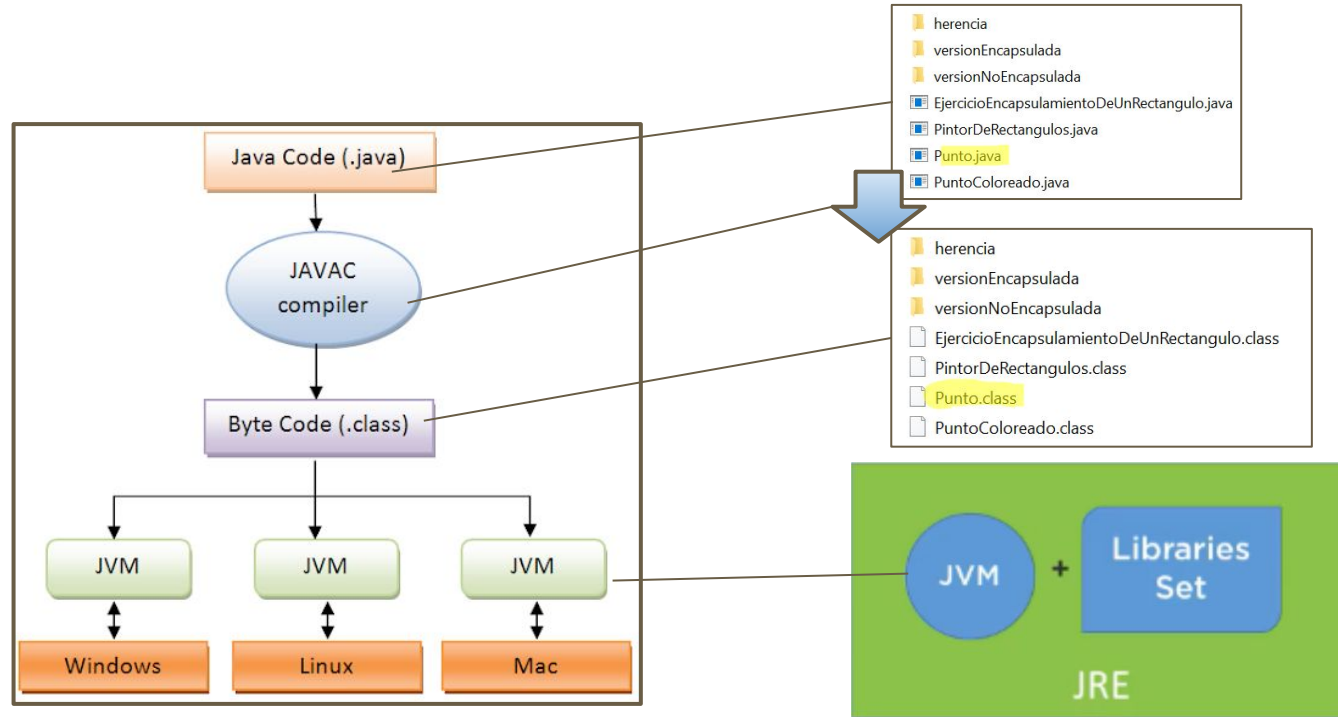
The scope and shadowing of a class declaration is specified in [§6.3](#) and [§6.4](#).

# ¿De qué se compone Java?: 2-JDK (Toolkit de desarrollo)





# ¿De qué se compone Java?: 3-JRE (PLATAFORMA DE EJECUCIÓN DE PROGRAMAS JAVA)



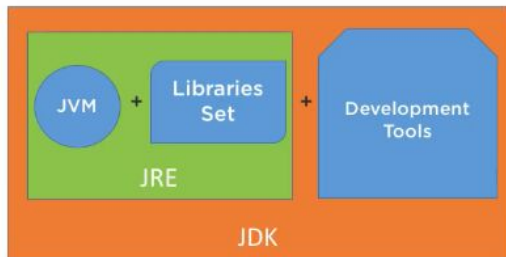
Varias implementaciones de JVM:  
[http://en.wikipedia.org/wiki/List\\_of\\_Java\\_virtual\\_machines](http://en.wikipedia.org/wiki/List_of_Java_virtual_machines)

# ¿Entonces qué necesitamos para programar en Java?

Aprender y aplicar buenas prácticas y criterio para diseñar y desarrollar software de calidad



JDK (Ya incluye la JRE)



Java Development Kit

Ambiente de desarrollo

A screenshot of the Eclipse IDE interface. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The main editor area shows a Java file named 'ImportExperimentsFromTCGATest.java'. The code includes package declarations, imports, and a class definition. The class 'ImportExperimentsFromTCGATest' extends 'RPackageTestCase' and contains a 'setUp()' method. The code is as follows:

```
1 package edu.unlp.medicina.rpackages.bioplatR;
2
3 import java.util.ArrayList;
4
5
6 /**
7  * It is necessary to install in R: Rserve and BioplatR. Then startup the Rserve on port 6336 thr
8  *
9  */
10
11 public class ImportExperimentsFromTCGATest extends RPackageTestCase {
12
13     String cancer_study_id_for_testing = "brca_tcga";
14     String subset_id = "brca_tcga_mrna";
15     List<String> p53_pathway_genes = new ArrayList<String>();
16     List<String> clinical_attribute_names = new ArrayList<String>();
17     String a_mrna_profile_id = "brca_tcga_mrna_median_zscores";
18
19     @Override
20     protected void setUp() {
21         //super.setUp();
22         p53_pathway_genes.add("TP53");
23         p53_pathway_genes.add("MDM2");
24         p53_pathway_genes.add("MDM4");
25         p53_pathway_genes.add("CDKN2A");
26         p53_pathway_genes.add("CDKN2B");
27         p53_pathway_genes.add("TP53BP1");
28
29         clinical_attribute_names.add("DFS_MONTHS");
30         clinical_attribute_names.add("DFS_STATUS");
31     }
32 }
```

Otros: IntelliJ IDEA, NetBeans, BlueJ

# Ambiente de desarrollo (Eclipse)

## □ Qué aprovechar de un ambiente de desarrollo?

- Administración de workspaces
- Syntax highlighting.
- Auto Syntax Checking.
- Quick fix
- Content assistant o IntelliSense (Ctrl Space)
- Code generation
- Herramientas de debug.
- Herramientas de Testing
- Integración con controladores de versiones.
- Vista de problemas

# Instalación de Java

<https://jdk.java.net/19/>

## OpenJDK JDK 19.0.2 General-Availability Release

This page provides production-ready open-source builds of the [Java Development Kit](#), version 19, an implementation of the [Java SE 19 Platform](#) under the [GNU General Public License](#), version 2, with the [Classpath Exception](#).

Commercial builds of JDK 19.0.2 from Oracle, under a [non-open-source license](#), can be found at the [Oracle Technology Network](#).

<https://www.eclipse.org/downloads/>

## Documentation

- [Features](#)
- [Release notes](#)
- [API Javadoc](#)

## Builds

<b>Linux/AArch64</b>	<a href="#">tar.gz (sha256)</a>	194649390 bytes
<b>Linux/x64</b>	<a href="#">tar.gz (sha256)</a>	195931906
<b>macOS/AArch64</b>	<a href="#">tar.gz (sha256)</a>	190625723
<b>macOS/x64</b>	<a href="#">tar.gz (sha256)</a>	192580989
<b>Windows/x64</b>	<a href="#">zip (sha256)</a>	194455638