03 Operadores y Herencia

everis
an NTT DATA Company

Operadores

If –then - else

```
If (isMoving) currentSpeed++;
If (isMoving) {currentSpeed++;System.out.println(“Se mueve”);}
 if (isMoving) {currentSpeed--;}
else {System.err.println(“No se mueve!");}
```

Buenas practicas:

Separar los If’s con las llaves aunque java permita la escritura continua como el primer ejemplo.

everis
an NTT DATA Company

Operadores

Switch

```
switch (month) {
case 1:
case 3:
case 5:
case 12:numDays = 31; break;
case 4:
case 6:
case 11:numDays = 30; break;
default:System.out.println("Invalid month.");
break;
}
```

```
switch (string) {
    case "texto1":
        // Hacer algo
        break;
    case "texto2":
        // Hacer otra cosa
        break;
    case "texto3":
        // Hacer otra cosa distinta
    case "texto4":
        // Además de la anterior, hacer otra cosa
        break;
    default:
        // Comportamiento por defecto
}
```

everis
an NTT DATA Company

Operadores

While – do while

```
while (count < 11) {
 System.out.println("Count is: " + count);
  count++;
}
```

```
do {statement(s)} while (expression);
```

```
while (indice < strings.length) {
    // Hacer algo
}
```

```
do {
    // Hacer algo
} while (indice < strings.length);
```

everis
an NTT DATA Company

Operadores

For – for each

for (*initialization*; *termination*; *increment*) {

statement(s)

}

```
for(int i=1; i<11; i++){
System.out.println("Count is: " + i);
}
```

```
for (int i = 0; i < strings.length; i++) {
    // Hacer algo
}
```

```
for (int i = 0, j = strings.length; i < j; i++) {
    // Hacer algo
}
```

```
for (String string : strings) {
    // Hacer algo
}
```

Operadores

Salida

En cualquier iteración de un bucle es posible “romper” la ejecución de las siguientes maneras:

- **break**: Termina la ejecución del bucle y continúa con la siguiente instrucción después de este.
- **continue**: Termina la iteración actual del bucle y continúa con la siguiente.
- **return**: Sale del método actual.
- **throw**: Lanza una excepción.

everis
an NTT DATA Company

Operadores

Ejercicios

1. Escribir un método que acepte un parámetro float como parámetro que represente una calificación y que devuelva un String con la calificación escrita. Ejemplos:
   - calificacion(4.99f) devuelve “Suspenso”
   - calificacion(6f) devuelve “Aprobado”
2. Escribir un método que obtenga como parámetro un entero que represente el número de mes y que devuelva un String con el nombre del mes asociado al número. Ejemplos:
   - textoMes(1) devuelve “Enero”.
   - textoMes(12) devuelve “Diciembre”.
   - textoMes(0) o textoMes(13) devuelve “Error: no existe el mes 0” o “Error: no existe el mes 13” respectivamente
3. Escribir un método que obtenga como parámetro un entero que represente el número de mes y que devuelva un String con todos los meses que faltan para final de año. Ejemplos:
   - textoMeses(12) devuelve “Diciembre”.
   - textoMeses(10) devuelve “Octubre, Noviembre, Diciembre”.
   - textoMeses(0) devuelve “Error: no existe el mes 0”.

Herencia

Es un aspecto fundamental del paradigma orientado a objetos.

Toda clase Java por defecto hereda de la clase ***Object***.

Sintaxis:

class Hija extends Padre

Herencia

Si clase **Hija** hereda de clase **Padre:**

*La clase **Hija*** contiene implícitamente todos los métodos y propiedades de ***Padre*** que sean *public* o *protected*.

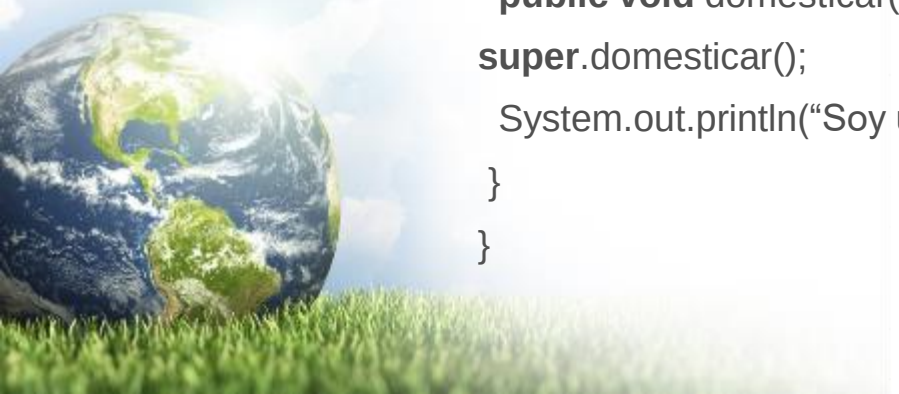
Y en caso de ambas pertenecer al mismo paquete, también los métodos y propiedades con modificador de acceso por

defecto.

Herencia

Ejemplo

```
public class Animal{
        long edad;
        public void hacer_dormir() {};
        public void domesticar()  {   System.out.println("Domestícame.");  }
        }
public class Perro extends Animal{
        public void sacar_a_pasear() {};
        public void domesticar()   // SOBREESCRITURA DEL METODO "domesticar"  {
        super.domesticar();
        System.out.println("Soy un perro, así que es fácil domesticarme.");
        }
        }
```

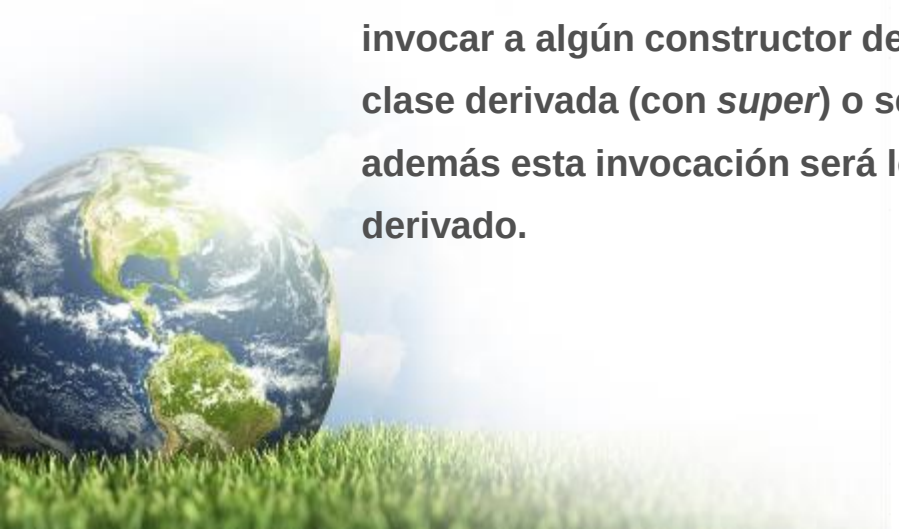


Herencia

Inicialización de clases en herencia

Java automáticamente inserta llamadas al constructor por defecto, sin parámetros de la clase base en el constructor de la clase derivada, siempre que exista.

Si no existiese constructor sin argumentos en la clase base, hay que invocar a algún constructor de la clase base en cada constructor de la clase derivada (con *super*) o se obtendrá un error de compilación. Y además esta invocación será lo primero que se realice en el constructor derivado.



Upcasting

Ejemplo

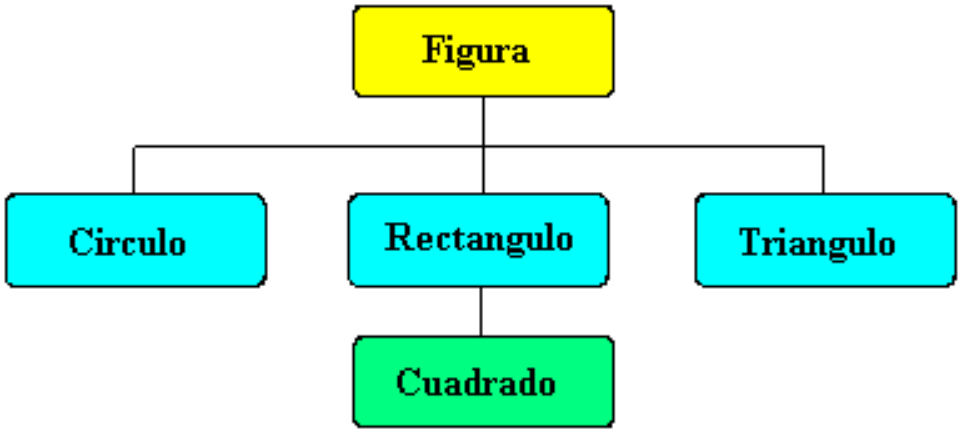
```
import java.util.*;

class Instrument {
  public void play() {}
  static void afinar (Instrument i) {}
}

public class DeViento extends Instrument {
  public static void main(String[] args) {
    DeViento flauta = new DeViento();
    Instrument.afinar(flauta); // Upcasting
  }
}
```

everis
an NTT DATA Company

Polimorfismo

Definición

Es la característica de POO que permite que un objeto pueda ser tratado como un objeto de otra clase.



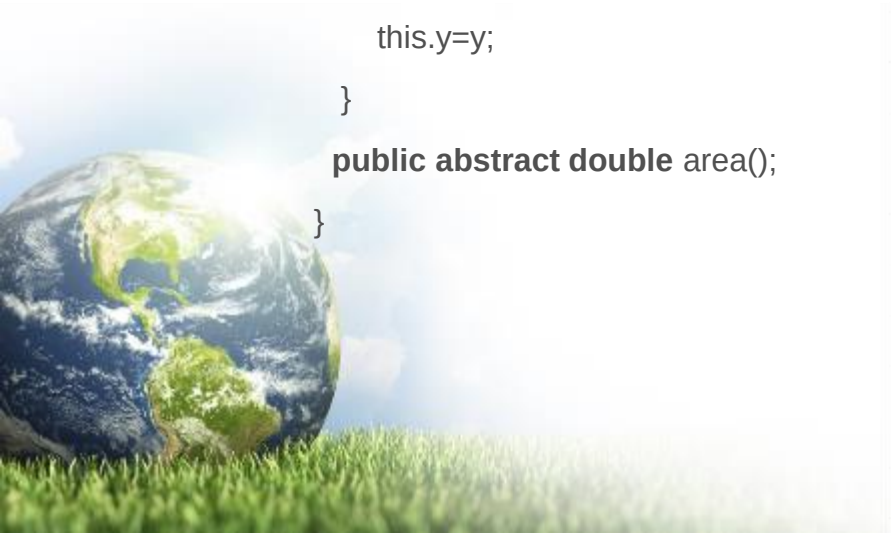
everis
an NTT DATA Company

Polimorfismo

Ejemplo

```
public abstract class Figura {
  protected int x;
  protected int y;
  public Figura(int x, int y) {
      this.x=x;
      this.y=y;
   }
  public abstract double area();
}
```

```
class Circulo extends Figura{
    protected double radio;
    public Circulo(int x, int y, double radio){
       super(x,y);
       this.radio=radio;
    }
    public double area(){
       return Math.PI*radio*radio;
    }
}
```

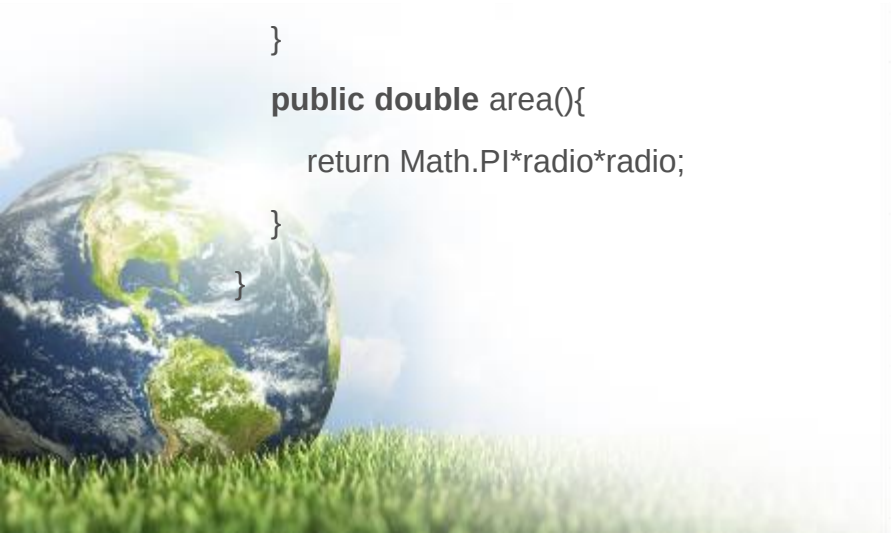


everis
an NTT DATA Company

Polimorfismo

Ejemplo

```
class Circulo extends Figura{
    protected double radio;
    public Circulo(int x, int y, double radio){
        super(x,y);
        this.radio=radio;
    }
    public double area(){
        return Math.PI*radio*radio;
    }
}
```



```
class Cuadrado extends Rectangulo{

    public Cuadrado(int x, int y, double dimension) {
        super(x, y, dimension, dimension);
    }
```

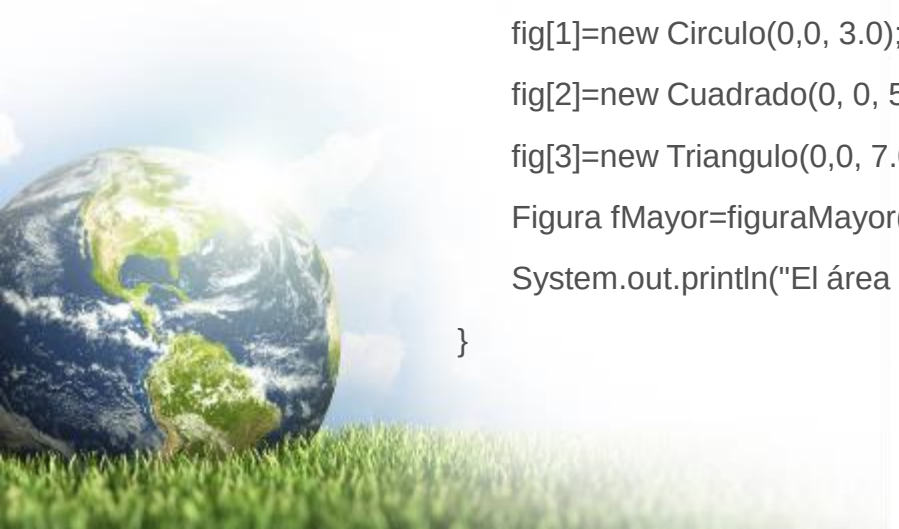
```
class Triangulo extends Figura{

    protected double base, altura;

    public Triangulo(int x, int y, double base, double altura){
        super(x, y);
        this.base=base;
        this.altura=altura;
    }
    public double area(){
        return base*altura/2;
    }
}
```

everis
an NTT DATA Company

Polimorfismo

Ejemplo

```
public class FigurasApp1 {
    public static void main(String[] args) {
//array de objetos
        Figura fig[]=new Figura[4];
        fig[0]=new Rectangulo(0,0, 5.0, 2.0);
        fig[1]=new Circulo(0,0, 3.0);
        fig[2]=new Cuadrado(0, 0, 5.0);
        fig[3]=new Triangulo(0,0, 7.0, 12.0);
        Figura fMayor=figuraMayor(fig);
        System.out.println("El área mayor es "+fMayor.area());
 }
```



everis
an NTT DATA Company

Especificadores de Acceso

FINAL

Cuando es aplicado sobre una propiedad de tipo primitivo, estamos definiendo una *constante*. El valor de esa propiedad no puede cambiar.

final int PI = 3.14

Cuando es aplicado sobre una referencia a un objeto, una vez que la referencia apunta a una instancia de la clase, la misma no puede ser modificada para que apunte a otra instancia. Sin embargo, el objeto SI puede ser modificado.

final Animal ref_obj;

ref_obj = new Animal();

everis
an NTT DATA Company

Especificadores de Acceso

Métodos FINAL

En un método, los argumentos *final* no pueden

ser modificados.

Cuando es aplicado sobre la declaración de un método de una clase, ese método no podrá ser sobreescrito por ninguna clase derivada.

Todos los métodos *private*, implícitamente son *final*. Aunque en este caso el compilador no se quejará si se intenta sobreescribir el método.

everis
an NTT DATA Company

Especificadores de Acceso

Clase FINAL

Cuando es aplicado a una clase, se está impidiendo que ninguna otra clase pueda heredar de ella.

Todos los métodos de una clase *final*, son implícitamente *final*.

Los campos pueden ser *final* o no.

final class NoMeHeredesPorFavor

everis
an NTT DATA Company

Especificadores de Acceso

Synchronize

A synchronized block in Java is synchronized on some object. All synchronized blocks synchronized on the same object can only have one thread executing inside them at the same time. All other threads attempting to enter the synchronized block are blocked until the thread inside the synchronized block exits the block.

```
public synchronized void add(int value){ this.count += value; }
```

This y Super

http://www.arquitecturajava.com/java-constructores-y-super/

this

Al acceder a variables de instancia de una clase, la palabra clave this hace referencia a los miembros de la propia clase en el objeto actual; es decir, this se refiere al objeto actual sobre el que está actuando un método determinado y se utiliza siempre que se quiera hace referencia al objeto actual de la clase.

super

Si se necesita llamar al método padre dentro de una clase que ha reemplazado ese método, se puede hacer referencia al método padre con la palabra clave super: