



IMPLEMENTACIÓN DE LA RELACIÓN DE HERENCIA EN JAVA

Marzo / 2020

PROGRAMACIÓN ORIENTADA A OBJETOS
INGENIERÍA INFORMÁTICA

Objetivos

Identificar una relación de herencia entre clases en un diagrama UML.

Identificar la sintaxis que manifiesta herencia entre clases en el lenguaje de programación Java.

Objetivos

Identificar la clase padre y la clase hija en una relación de herencia en el lenguaje de programación Java.

Implementar una relación de herencia en el lenguaje de programación de Java.

Caracterizar el uso de la palabra reservada **super**.

Sumario

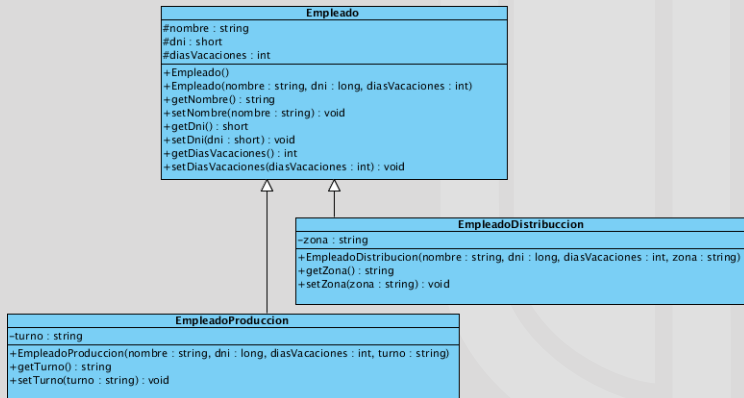
- ◇ Relación de Herencia.
- ◇ Sintaxis para expresar relación de herencia en Java.
- ◇ La palabra reservada **super**.

Bibliografía

- ◇ *Como programar en Java.*
- ◇ *Aprenda Java como si estuviera en primero.*

Motivación

Como expresar e implementar la relación que se muestra en la imagen usando Java?



Relaciones entre clases

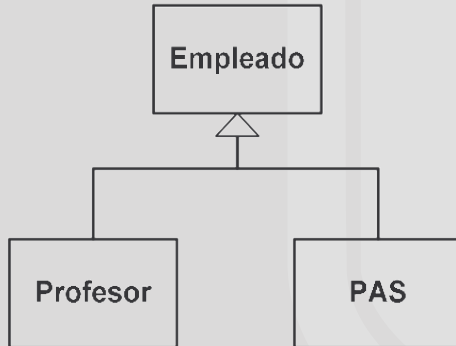
Herencia

La relación entre una superclase y sus subclases.

Objetos de distintas clases pueden tener atributos similares y exhibir comportamientos parecidos (p.ej. animales, mamíferos...).

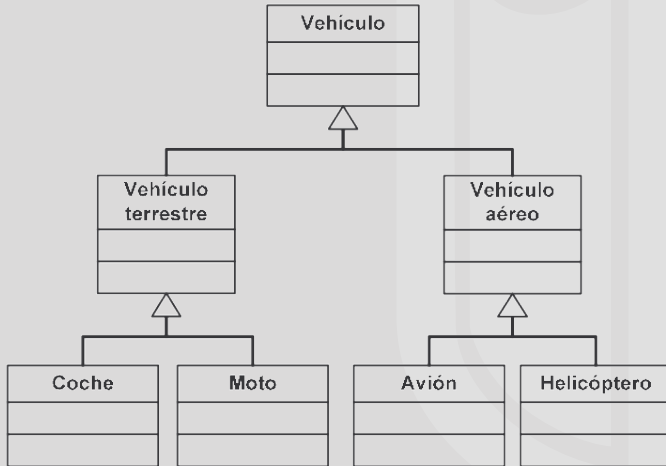
Relaciones entre clases

Herencia



Relaciones entre clases

Herencia



Relaciones entre clases

Herencia

El comportamiento de una categoría más general es aplicable a una categoría particular.

Las subclases heredan características de las clases de las que se derivan y añaden características específicas que las diferencian.

Relaciones entre clases

Herencia

En el diagrama de clases, los atributos, métodos y relaciones de una clase se muestran en el nivel más alto de la jerarquía en el que son aplicables.

Relaciones entre clases

Herencia

Es la relación de especialización/generalización (o de herencia) entre dos clases. Esta relación se considera propia de los lenguajes de POO.

Herencia en Java

Implementación

Para explicar como se expresa la relación de herencia en el lenguaje de programación Java vamos a ver desde los cambios o adecuaciones desde:

- La clase padre o super-clase.

- La clase hija o clase derivada.

Herencia en Java

La clase padre o super-clase

En la clase padre debemos cambiar la visibilidad **private** de aquellos atributos y métodos que queremos que sus clases hijas **"hereden"** o que tengan acceso a estos por **protected**.

Herencia en Java

La clase padre o super-clase

```
public class Empleado {  
    protected String nombre;  
    protected short dni;  
    protected int diasVacaciones;  
    public Empleado(){  
        this.nombre=""; this.dni=0;  
        this.diasVacaciones=0;  
    }  
    public Empleado(String nombre, short dni,  
    int diasVacaciones){  
        this.nombre=nombre; this.dni=dni;  
        this.diasVacaciones=diasVacaciones;  
    }  
    //....  
}
```

Herencia en Java

La clase hija o clase derivada

Para expresar una herencia en las clases hijas o clases derivadas, existen dos elementos fundamentales:

- La declaración de la clase.

- Los constructores de la clase.

Herencia en Java

La clase hija o clase derivada

Para indicar que una clase deriva de otra se utiliza la palabra **extends**

Síntaxis

```
public class <ClaseHija> extends <ClasePadre> {  
    //.... Implementacion de la clase  
}
```

Herencia en Java

La clase hija o clase derivada

```
public class EmpleadoProduccion extends Empleado {  
    /* Implementacion de la clase EmpleadoProduccion  
       */  
}
```

```
public class EmpleadoDistribucion extends Empleado  
{  
    /*Implementacion de la clase EmpleadoDistribucion  
       */  
}
```

Herencia en Java

La clase hija o clase derivada

En los constructores de las clases hijas o derivadas tienen la responsabilidad no solo de inicializar sus propios atributos, sino los atributos que hereda de su padre, abuelo, etc.

Para hacer esto existen dos variantes:

- Idea trivial

- Utilización del **super**

Herencia en Java

La clase hija o clase derivada

Idea trivial

```
public class EmpleadoProduccion extends Empleado {  
    //....  
    public EmpleadoProduccion(String nombre,  
        short dni, int diasVacaciones, String turno){  
        this.nombre=nombre;  
        this.dni=dni;  
        this.diasVacaciones=diasVacaciones;  
        this.turno=turno;  
    }  
    //...  
}
```

Herencia en Java

La clase hija o clase derivada

Utilización del super

```
public class EmpleadoProduccion extends Empleado {  
    //....  
    public EmpleadoProduccion(String nombre,  
        short dni, int diasVacaciones, String turno){  
        super(nombre,dni,diasVacaciones);  
        this.turno=turno;  
    }  
    //...  
}
```

Herencia en Java

La clase hija o clase derivada

Utilización del super (2)

```
public class EmpleadoProduccion extends Empleado {  
    //....  
    public EmpleadoProduccion(String nombre,  
        short dni, int diasVacaciones, String turno){  
        super();  
        this.turno=turno;  
    }  
    //...  
}
```

Herencia en Java

Super

El constructor de una clase hija o derivada puede llamar al constructor de su super-clase o clase padre por medio de la palabra **super**, seguida de los argumentos apropiados entre paréntesis.

Herencia en Java

Super

De esta forma, un constructor sólo tiene que inicializar por sí mismo las variables no heredadas.

Herencia en Java

Super

Los métodos de la super-clase que han sido redefinidos pueden ser todavía accedidos por medio de la palabra **super** desde los métodos de la clase derivada, aunque con este sistema sólo se puede subir un nivel en la jerarquía de clases.

Conclusiones

- ◇ Una clase solo puede heredar de una sola clase en Java. No existe herencia multiple como en C++.
- ◇ Una clase padre puede tener perfectamente un atributo o método que no sea visible para sus clases hijas si esto fuera necesario.

UNIVERSIDAD DE MATANZAS

cosechando el saber

FIN