Cheatography

Herencia en Java Cheat Sheet

by diegojserrano via cheatography.com/71015/cs/19624/

Concept

La herencia es una relación entre clases

Permite definir una clase a partir de otra

Es una relación asimétrica, las dos clases involucradas tienen roles diferentes

Una de las clases se denomina Base y la otra Derivada

También se las conoce como Padre e Hija respectivamente

La clase derivada se la define como una variante o caso particular de la clase Base

Aplicación

La herencia se utiliza cuando dos clases son suficientemente similares como para programarse como una clase, pero algunos de los objetos tienen características que los diferencian o que representan casos particulares.

La relación de herencia siempre debe poder interpretarse con la expresión "es un"

Entre las clases Empleado y Gerente puede plantearse una relación de herencia, ya que un Gerente siempre **es un** Empleado.

Entre Empresa y Empleado no tiene sentido la herencia, un Empleado no **es una** Empresa. (Aunque puede haber otro tipo de reación entre esas clases, p.ej. Composición)

Herencia simple

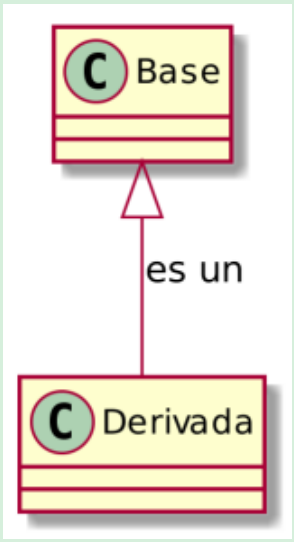
Java posee herencia simple, es decir que cada clase puede tener una única clase base

Sin embargo, cada clase puede tener múltiples clases derivadas

Las clases derivadas pueden tener a su vez otras que deriven de ellas

A causa de esto, el conjunto de relaciones de herencia de una clase se lo denomina **Árbol de herencia**

Diagrama de clases



Miembros de la clase derivada

Los objetos de la clase derivada poseen todos los miembros definidos en la clase base

A su vez la clase derivada puede poseer otros miembros (atributos y métodos)

A causa de esto es que la cláusula que establece la herencia se denomina `extends` (extiende), ya la clase derivda "es más grande" que la clase base

En el código de la clase derivada se puede acceder a los miembros visibles de la clase base como si fueran miembros propios directamente o a través de la referencia `this`

Cheatography

Herencia en Java Cheat Sheet

by diegojserrano via cheatography.com/71015/cs/19624/

Modificador de acceso protected

Además de los modificadores de acceso `public` y `private` se le agrega el modificador `protected`

Un miembro marcado como protected se lo denomina protegido

Los miembros protegidos son accesibles desde las clases derivadas pero no desde otras

Por estar ocultos desde otras clases no relacionadas por medio de la herencia, se requiere programar los métodos get y set.

Sintaxis: cláusula extends

Para establecer la herencia en el código debe agregarse al bloque class **de la derivada** la cláusula **extends** y a continuación el nombre de la clase base

```
public class Base {...}

public class Derivada extends Base{...}
```

Todas las clases que no posean cláusula extends no establecen ninguna clase base, en este caso derivan implícitamente de una clase especial llamada Object

Referencia super

La referencia `super` es similar a `this`, pero sólo puede acceder a los miembros heredados

Sólo debe usarse cuando hay que diferenciar entre un método programado en la clase base que puede generar conflicto con los de la clase derivada

Tiene dos usos principales:

- Para llamar a los constructores de la clase base.

- Para llamar a la versión existente en la clase base de un método sobreescrito en la derivada.

Constructores

Los constructures no se heredan

Las clases derivadas deben proveer sus propios constructores

Cuando se instancia un objeto de una clase derivada, se usa `new` con el nombre de la derivada y los parámetros de algún constructor de la misma

Primero se ejecuta el constructor de la clase base y luego el de la derivada

Si el constructor de la clase derivada no llama a ningún constructor de la clase base, se ejecuta el constructor sin parámetros

Para invocar al constructor de la clase base en forma explícita se agrega como **primera línea** del constructor de la clase derivada una llamada a `super();`

Para usar un constructor de la clase base con parámetros se indica dentro de la lista de parámetros de super()

El constructor de la derivada requiere parámetros tanto para los atributos propios como para los de la clase base

```
public class Derivada extends Base{
   public Derivada(param1, param2, param3){
      super(param1,param2);
      this.atributo3 = param3;
   }
}
```

By **diegojserrano**

cheatography.com/diegojserrano/

Published 13th May, 2019.
Last updated 13th May, 2019.