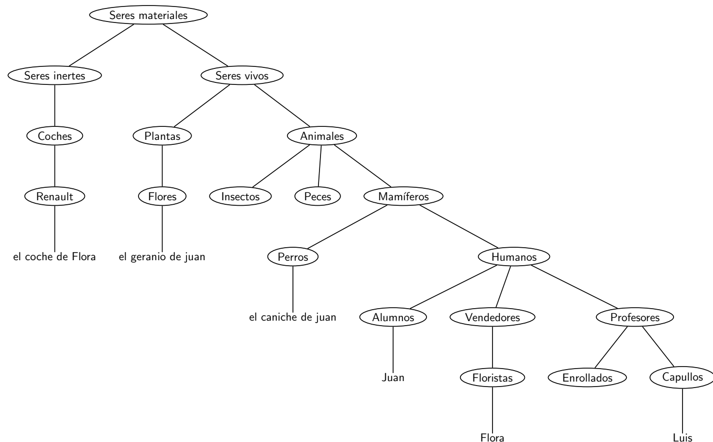
Herencia

Luis Fernando Llana Díaz

Departamento de Sistemas Informáticos y Computación
Universidad Complutense de Madrid

16 de abril de 2007

Herencia: relación *es un*



Herencia: relación *es un* II

Recorrido de la jerarquía

Según se recorre la jerarquía los elementos que están por debajo heredan características de los elementos superiores:

- Todos los objetos tienen masa y volúmen.
- Todos los seres vivos nacen, crecen, etc. . . .
- Todos los mamíferos maman cuando son pequeños, etc. . . .
- Todos los perros ladran.

Herencia: relación *es un* III

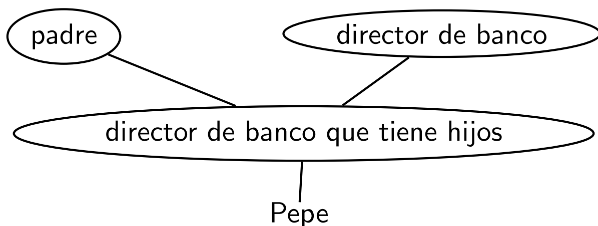
Por tanto el perro de juan

- tienen masa y volúmen.
- nace, crece, etc. . . .
- mamó cuando fue pequeño, etc. . . .
- ladra.

Características heredadas

- Existen características que no cambian (peso y volúmen).
- Caracteríticas que se añaden.
- Características que se modifican, se refinan o cambian.

Herencia múltiple



Herencia en Java

- Interfaces.
- Clases abstractas.
- Extensión de clases.
- Implementación de interfaces.

Clase Object

```
public class Patata {          1
   .......                     2
   .......                     3
   .......                     4
}                              5
```

```
public class Patata extends Object {     1
   .......                               2
   .......                               3
   .......                               4
}                                        5
```

Clase Object

La clase Object es la superclase de todas las clases en Java.
Dispone, entre otros, de los métodos:

- boolean equals(Object obj)
- String toString()

Estos métodos se deben sobreescribir en las subclases.

Clase Fecha

```
public boolean equals(Object obj) {
    if (! (obj instanceof Fecha)) return false;
    return diasDesdeInicio==((Fecha)obj).diasDesdeInicio;
}
```

```
public String toString(){
    FechaTerna f = new FechaTerna(diasDesdeInicio);
    return f.toString();
}
```

Extensión de clases

Extensión de clase

```
package fecha;
public class FechaFueraDeRango extends RuntimeException {
  public FechaFueraDeRango(String s) {
      super(s);
    }
}
```

Extensión de interfaz

```
public class Fecha implements Comparable{
  .........
  .........
  .........
}
```

Clases, clases abstracas e interfaces

Clase normal: todalmente definida, puede haber objetos de esa clase.

Clase abstracta: parcialmente definida, puede tener algún atributo, métodos totalmente definidos y métodos no definidos.

Interfaz: sólo define algún método, pero sin implementar.

Clase Abstracta

```
public abstract class Patata{                                   1
  private int atributo;                                         2
  public abstract modifica(int x);                              3
}                                                               4
```

Interfaz

El interfaz `Comparable` del API de Java *debe ser* de la forma

```
package java.lang;
public interface Comparable {
  public int compareTo(Object o);
}
```

Herencia múltiple

```
public class Patata {                    1
  public int x;                          2
  public void incrementa(){              3
    x=x+1;                               4
  }                                      5
}                                        6
```

```
public class Tomate {                    1
  public int x;                          2
  public void incrementa(){              3
    x=x+10;                              4
  }                                      5
}                                        6
```

```
public class Lechuga extends Patata, Tomate {                              1
  public int y;                                                            2
}                                                                          3
......                                                                     4
......                                                                     5
Lechuga lechuga = new Lechuga();                                           6
lechuga.incrementa(); // ¿incrementa x = x+1 ó x = x+10 ?                  7
```

Herencia múltiple en Java

En Java existe versión restringida de herencia

- Se puede extender sólo 1 clase.
- Se puende implementar más de 1 interfaz.

```
public class Trazador extends Frame                                    1
                      implements WindowListener,ObjetoComparable {     2
```

Objeto super

Invocación a la superclase

```
public class Estudiante{                                         1
  private String nombre;                                         2
  public Estudiante(String nombre) {                             3
    this.nombre = nombre;                                        4
  }                                                              5
  public void estudia(double tiempo){....}                       6
}                                                                7
```

```
public class EstudiantePerezoso extends Estudiante{              1
   public EstudiantePerezoso(String nombre) {                    2
     super(nombre);                                              3
   }                                                             4
   public void trabaja(double tiempo){                           5
     super.estudia(tiempo/3);                                    6
     descansa(tiempo/3);                                         7
     super.estudia(tiempo/3);                                    8
   }                                                             9
}                                                                10
```

Objeto this

```
public class Fecha {                                                        1
    ....................                                                    2
    public Fecha(int dia, int mes, int anyo) {                              3
        compruebaFecha(dia, mes , anyo);                                    4
        diasDesdeInicio=calculaDiasDesdeInicio(dia, mes, anyo);             5
    }                                                                       6
                                                                            7
    public Fecha() {                                                        8
        this(1,1,1998);                                                     9
    }                                                                       10
    ....................                                                    11
}                                                                           12
```

```
 class FechaTerna {                                                         1
   private int dia;                                                         2
   private int mes;                                                         3
   private int anyo;                                                        4
   ......................                                                   5
   public FechaTerna(int dias) {                                            6
        this.dia = quedan+1;                                                7
        this.mes = mes+1;                                                   8
        this.anyo = anyo;                                                   9
    }                                                                       10
    ........                                                                11
}                                                                           12
```

Figuras planas

```
package geometria.figurasPlanas;                                        1
public interface FiguraPlana{                                           2
    public boolean intersecaCon(FiguraPlana otra);                      3
    public double distancia(Punto p);                                   4
}                                                                       5
```

```
package geometria.figurasPlanas;                                        1
public abstract class Superficie implements FiguraPlana{                2
    public abstract boolean estaDentro(Punto P);                        3
    public abstract double superficie();                                4
}                                                                       5
```

Puntos

```
public class Punto implements FiguraPlana{                                   1
    public Punto(double x, double y){                                        2
        posX=x; posY=y;                                                      3
    }                                                                        4
    public double distancia(Punto otro){                                     5
        Vector v=new Vector(this,otro);                                      6
        return v.modulo();                                                   7
    }                                                                        8
    public boolean equals(Object o){                                         9
        if (o instanceof Punto) {                                            10
            Punto otro = (Punto)o;                                           11
            return (this.posX==otro.posX) && (this.posY==otro.posY);         12
        }                                                                    13
        else return false;                                                   14
    }                                                                        15
    public boolean intersecaCon(FiguraPlana otra){                           16
        if (otra instanceof Punto) {                                         17
            return this.equals(otra);                                        18
        }                                                                    19
        else {                                                               20
            return otra.intersecaCon(this);                                  21
        }                                                                    22
    }                                                                        23
}                                                                            24
```

Método equals

Definido en Object, todos los objetos son subclase suya: todos tienen definido el método equals.

```
public boolean equals(Object o){
    if (o instanceof Punto) {
        Punto otro = (Punto)o;
        return (this.posX==otro.posX) && (this.posY==otro.posY);
    }
    else return false;
}
```

Interfaces Comparable y Cloneable

```
public class Fecha implements Comparable, Cloneable{                    1
  ..................................                                    2
    public int compareTo(Object o) {                                    3
        if (!(o instanceof Fecha))                                      4
            throw new ClassCastException("Se requiere un objeto de clase fecha.Fecha."+   5
                                         "He recibido algo de clase "+o.getClass().getNa  6
        return diasDesdeInicio-((Fecha)o).diasDesdeInicio;              7
    }                                                                   8
    public Fecha clone() {                                              9
        Fecha fecha = new Fecha();                                      10
        fecha.diasDesdeInicio = this.diasDesdeInicio;                   11
        return fecha;                                                   12
    }                                                                   13
  ........................                                              14
}                                                                       15
```

Interfaz Comparable

```
package java.lang;                                    1
public interface Comparable<T> {                      2
  public int compareTo(T o);                          3
}                                                     4
```

Se usa para:

- ordenación.
- colecciones ordenadas.

```
public class Fecha implements Comparable<Fecha>, Cloneable{   1
    ............                                              2
    ............                                              3
    public int compareTo(Fecha o) {                           4
        return diasDesdeInicio-o.diasDesdeInicio;             5
    }                                                         6
}                                                             7
```

Interfaz `Comparable`

- *Autoriza* el uso del método `clone` de la clase object.
- Copia los atributos, no hace un `clone` de los atributos.
- Si se quiere hacer público hace falta reescribirlo.

Estático y dinámico

Estático

Conocido en tiempo de compilación.

Dinámico

Lo que se conoce en tiempo de ejecución.

Tipo estático

```
public class Fecha {                                    1
  ....................                                  2
    public boolean equals(Object obj) {                 3
      ..............                                    4
      ..............                                    5
    }                                                   6
  ....................                                  7
}                                                       8
```

¿Qué sabemos de `obj`?

El objeto al que hace referencia `obj` pertenece a un subclase de `Object`.

Tipo dinámico

```
public class Fecha {
  ......................
    public boolean equals(Object obj) {
      ...............
      ...............
    }
  ......................
      Fecha f1 = new Fecha(31,JULIO,2006);
      Fecha f2 = new Fecha(31,JULIO,2005);
      boolean b = f1.equals(f2);
  ........
}
```

¿Qué sabemos de obj?

f2 es de clase Fecha. Cuando se hace la llamada obj va a ser una referencia a un objeto de clase Fecha.

Calificativo static

Los métodos y atributos marcados como static

- son inherentes a la clase,
- todos los objetos los comparten ,
- se pueden invocar directamente desde la clase.

```
public class Fecha {                                1
  ....................                              2
  ....................                              3
  public final static int ENERO=0;                  4
  ................                                  5
}                                                   6
........                                            7
    Fecha f =  new Fecha(1,Fecha.ENERO,2005);       8
```

Si un atributo es estático

- Se comparte por todos los objetos de la clase

Calificativo `static`

```
public class Fecha {                                                                      1
  ....................                                                                    2
  ....................                                                                    3
      private static int calculaDiasDesdeInicio(int elDia, int elMes, int elAnyo){        4
        int dias;                                                                         5
                                                                                          6
        dias=calulaDiasHastaPrimeroDe(elAnyo);                                            7
        dias=dias+calculaDiasHastaPrimeroDelMes(elMes,elAnyo);                            8
        dias=dias+elDia-1; // el 1/1/1601 es el dia 0                                     9
        return dias;                                                                      10
    }                                                                                     11
                                                                                          12
  ....................                                                                    13
                                                                                          14
}                                                                                         15
```

Si un método es estático

- Se gana en eficiencia (tiempo/memoria)
- No puede hacer referencia a atributos no estáticos.

Permisos de acceso

Los métodos y atributos de una clase pueden ser

- `public`: accesibles desde cualquier clase.
- `protected`: accesibles desde clases que estén en el mismo paquete o desde subclases.
- `private` sólo es accesible desde la propia clase.