Universidad Zaragoza

Fundamentos de Informática

Lección 7. Programación Orientada a Objetos

Curso 2010-2011

José Ángel Bañares y Pedro Álvarez
2/11/2010. Dpto. Informática e Ingeniería de Sistemas.

Universidad Zaragoza
1542

Índice de contenidos

- Implementación de nuevos tipos de datos
  - Identificación de Objetos y atributos (field)
  - Definición de una Clase
  - Definición de Constructores
  - Definición de Accesos (getters) y Modificadores (setters)
- Reutilización de clases vía herencia

Universidad Zaragoza

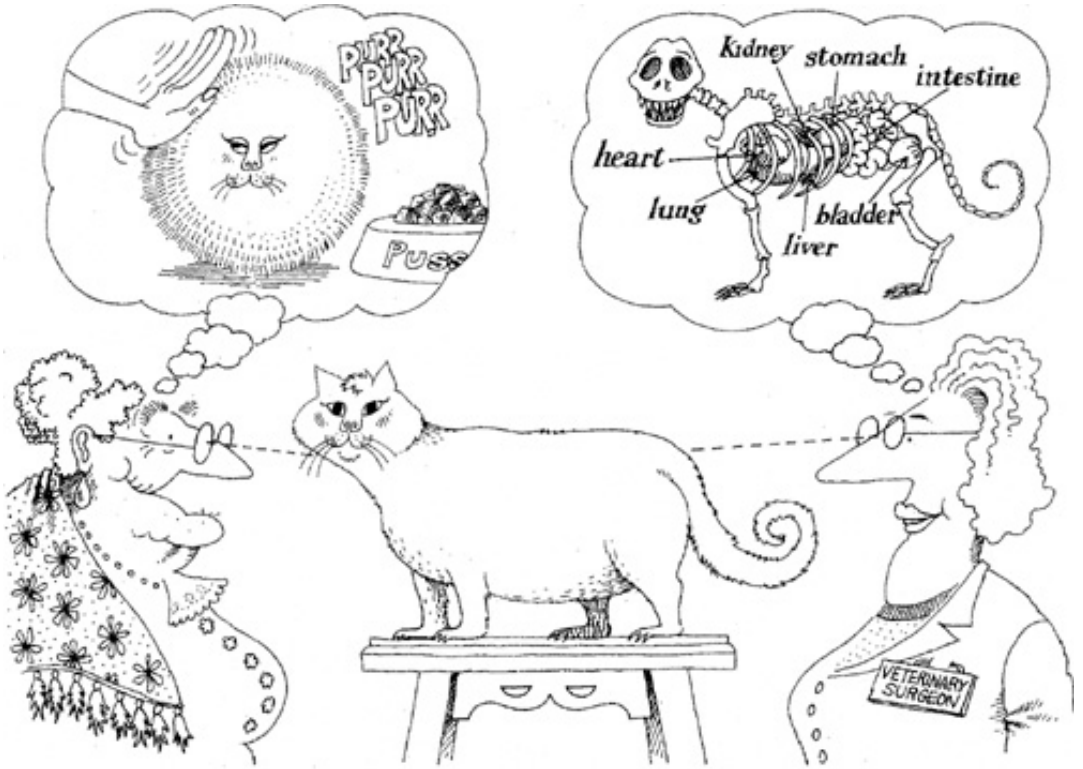
Abstracción

- Abstraer, o el "método de dar **nombre**" a conceptos sobre los que construir nuestros programas
  - Podemos asociar un nombre con una **dirección de memoria**: Nombre de la **variable**)
  - Podemos asociar un nombre con una "estructura" o forma de interpretar **uno o más bytes**: **Tipo de dato**
  - Podemos asociar un nombre con un **grupo de instrucciones** java: **método**
  - **Podemos asociar nombres a nuevas abstracciones no soportadas por el lenguaje: nuevos tipos de datos agrupando variables relacionadas y funciones**

Universidad Zaragoza
1542

Abstracción

- La **Abstracción** se centra en las características esenciales del objeto desde la perspectiva del observador.



Object-Oriented Analysis and Design with Applications Grady Booch, Second Edition. Benajmiin/Cummings.

Universidad Zaragoza

Identificación de Objetos y atributos

- Definición de **nuevas Estructuras de datos**
  - Estructura de datos = Agrupación de datos
    - **Array** = Estructuras de datos de acceso indexado
    - **Clase** = Agrupación de datos de diferente tipo. Diferentes atributos.
  - Clase: Agrupación de datos asociados a una entidad
    - Acceso por nombre de atributo

      Por ejemplo, un **estudiante** tiene :

      Nombre: una cadena

      NotaCursos (nota media de cada curso)

Universidad Zaragoza
1542

Clases que implementan Tipos de datos (registro)

```
public class Estudiante{
 // visibilidad tipo nombre [= expresión]
  public String nombre = "Anonimo";
  public double[] notaCursos;

}
```

Universidad Zaragoza

Clases que implementan Tipos de datos (registro)

```
public class Estudiante{
 // visibilidad tipo nombre [= expresión]
  public String nombre = "Anonimo";
  public double[] notasCursos;

}
```

- **public**: La clase permite mostrar los atributos, lo que obliga al usuario a conocer su implementación.
- La clase es un **registro**: agrupación de datos, a los que se accede por el nombre del campo.

Universidad Zaragoza

Clases que implementan Tipos de datos

Estudiante:Antonio Pérez
Curso 1:7.0
Curso 2:7.5
Curso 3:7.0
Curso 4:0.0

```
public class Estudiante{
  public String nombre = "Anonimo";
  public double[] notasCursos;

}
```

```
class ProgramaEstudiante{
  public static void main(String[] args){
    double[] notas={6.0,7.0,7.5,7,0};
    Estudiante est1=new Estudiante();
    est1.nombre="Antonio Pérez";
    est1.notasCursos= notas;
    System.out.println("Estudiante:"+ est1.nombre);
    for (int i=1; i< est1.notasCursos.length;i++)
      System.out.println("Curso "+i+":"+est1.notasCursos[i]);
  }
}
```

Universidad Zaragoza

Clases que implementan Tipos de datos (Tipo abstracto de dato)

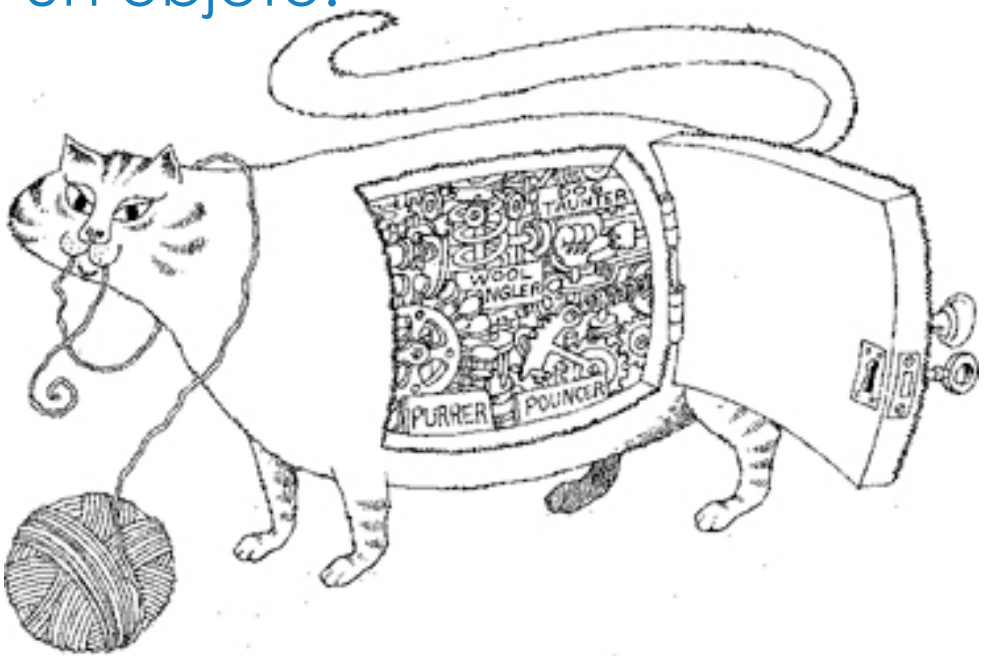
```
public class Estudiante{
 // visibilidad tipo nombre [= expresión]
  private String nombre = "Anonimo";
  private double[] notaCursos;

}
```

- **private**: La clase permite **encapsular** (ocultar) la implementación (Definición de un nuevo tipo abstracto de datos, TAD)
  - El usuario del Tipo de dato desconoce que atributos y como están implementados
  - Sólo se permite manipular los atributos con los métodos definidos en la clase

Universidad Zaragoza
1542

Encapsulación

- La encapsulación oculta los detalles de implementación de un objeto.



Object-Oriented Analysis and Design with Applications Grady Booch, Second Edition. Benajmiin/Cummings.

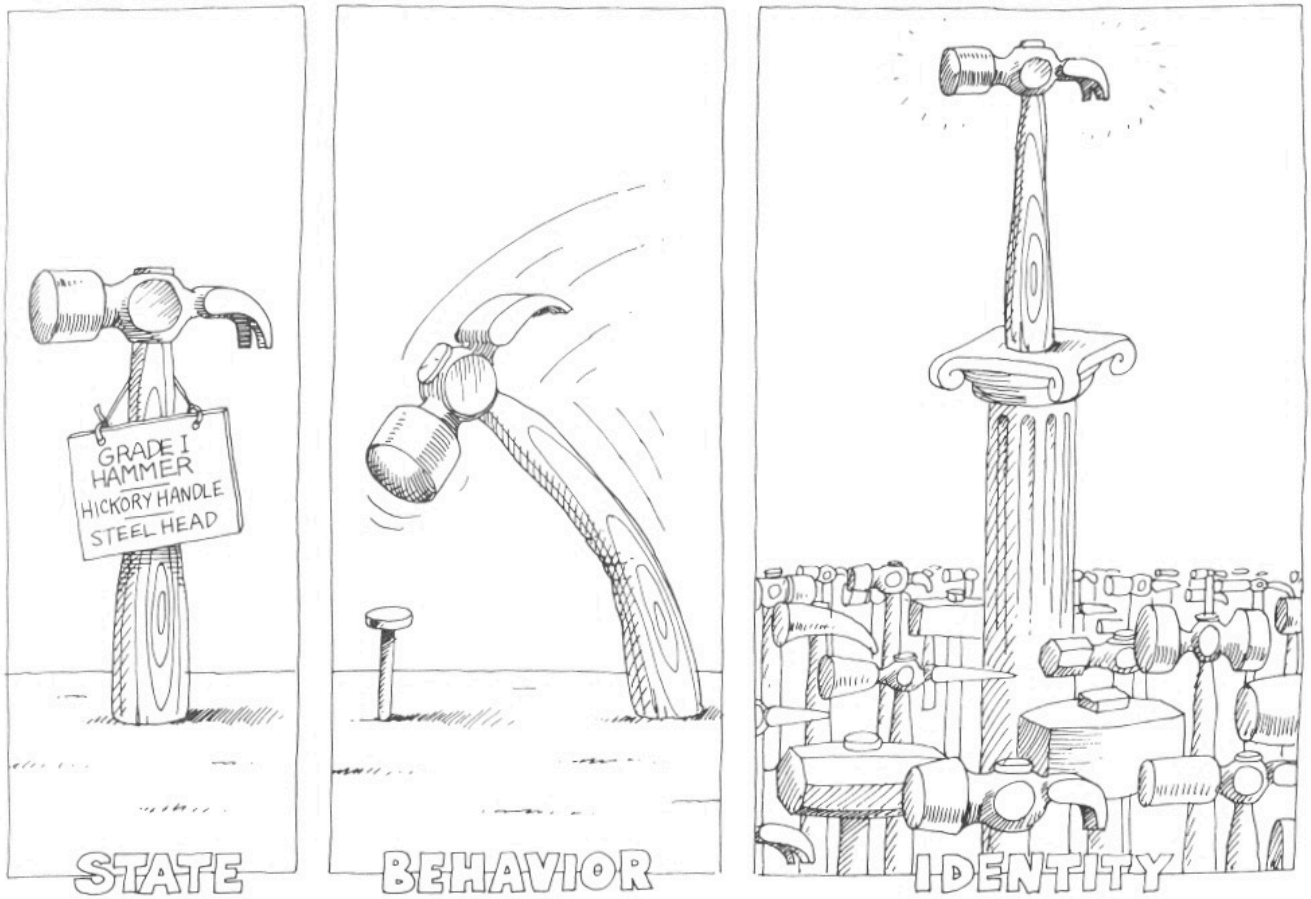
Universidad Zaragoza

Clases que implementan tipos de datos

- Las clases son "**factorías**" de objetos
  - Cada instancia tiene su identidad y su estado independiente
    - **Constructores** (Crean instancias de objeto, inicializando el estado el objeto de distintas formas)
    - **Métodos** (que ocultan los detalles de implementación)
    - **Atributos**/fields (que representan el estado del objeto)

Universidad Zaragoza

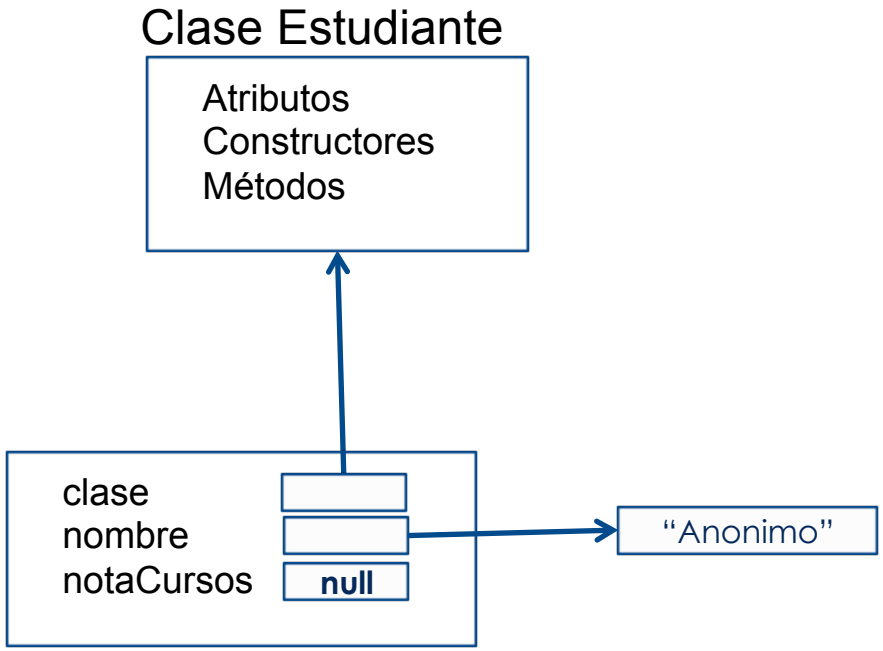
Objetos con Identidad y Estado



Universidad Zaragoza

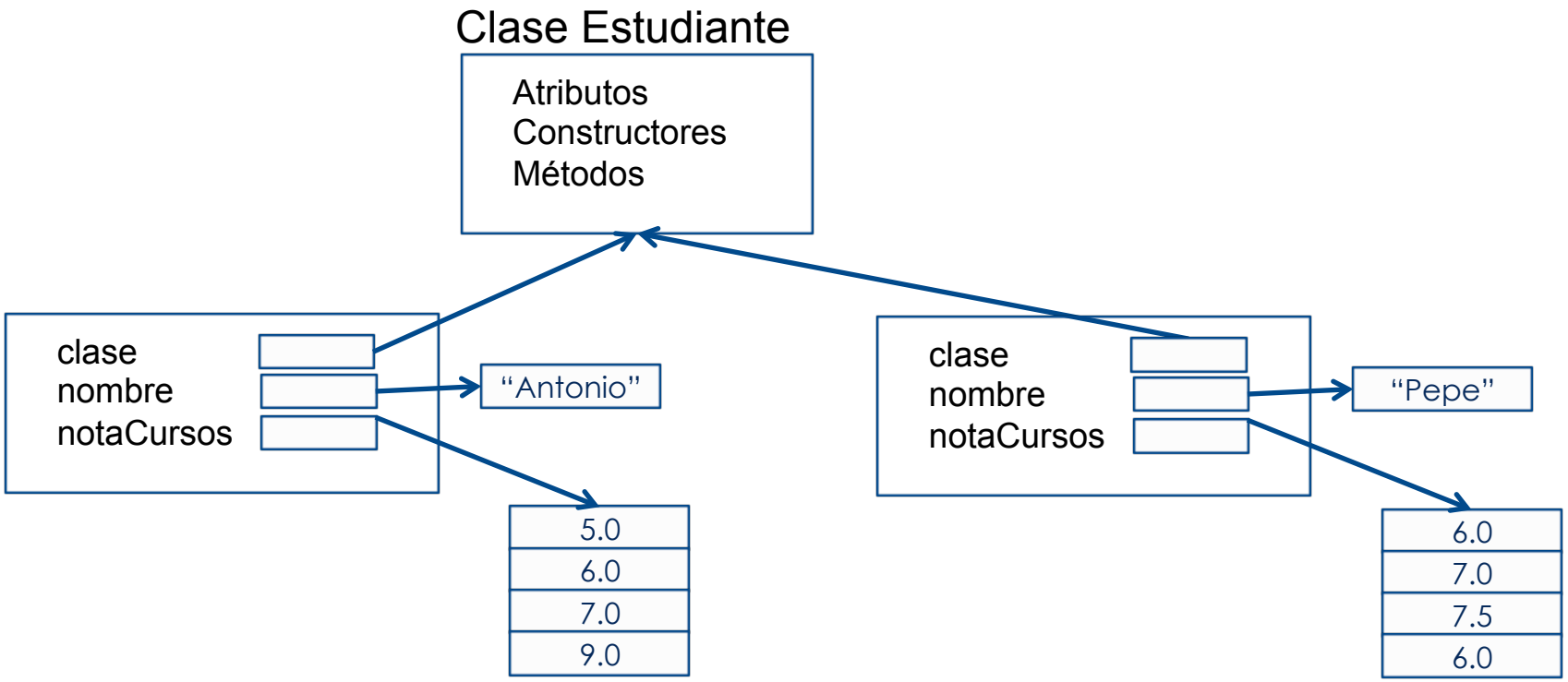
Constructor por defecto

```
Estudiante = new Estudiante();
```



Universidad Zaragoza

Clases: Factoría de objetos con identidad y estado



Universidad Zaragoza

Definición de constructores

```
public class Estudiante{
  ///////// atributos ////////////////
  private String nombre = "Anonimo";
  private double[] notasCursos;

  ///////// constructores /////////
  public Estudiante (String nombre){
      this.nombre = nombre;
  }
 public Estudiante(){}
  ///////// métodos ////////////////
  public String toString(){
      return "Estudiante: "+ this.nombre;
  }

}
```

Universidad Zaragoza

Definición de constructores

```
class ProgramaEstudiante{
  public static void main(String[] args){

     Estudiante est1 = new Estudiante();
     Estudiante est2 = new Estudiante("Antonio");

     System.out.println(est1);
     System.out.println(est2);
     System.out.println(new Estudiante("Luis"));

  }
}
```

```
Estudiante: Anonimo
Estudiante: Antonio
Estudiante: Luis
```

Universidad Zaragoza

Sobrecarga de constructores

```
public class Estudiante{
  private String nombre = "Anonimo";
  private double[] notasCursos;
  ////////// constructores //////////
  public Estudiante (String nombre){
      this.nombre = nombre;
  }
  public Estudiante (String nombre, double[] lasNotas){
      this.nombre = nombre;
      this.notasCursos=lasNotas;
  }
  public Estudiante(){   }
  ////////// métodos ////////////////
  public String toString(){
      return "Estudiante: "+ this.nombre;
  }
}
```

Universidad Zaragoza

Sobrecarga de constructores

```
class ProgramaEstudiante{
  public static void main(String[] args){
      double[] notas = {5.5,7.0,6.0,8.0};

    Estudiante est1 = new Estudiante();
     Estudiante est2 = new Estudiante("Antonio");
     Estudiante est3 = new Estudiante("Juan",notas);

     System.out.println(est1);
     System.out.println(est2);
     System.out.println(est3);
  }
}
```

```
Estudiante: Anonimo
Estudiante: Antonio
Estudiante: Juan
```

Universidad Zaragoza

Métodos

```
...

  public double mediaNotas(){
      double media=0.0;
      if (this.notasCursos != null &&
           this.notasCursos.length>0){
          double sum=0.0;
          for (int i = 0;i < this.notasCursos.length;i++)
           sum+=this.notasCursos[i];
          media = sum/notasCursos.length;
      }
      return media;
  }
}
```

Estudiante: Anonimo
Estudiante: Antonio
Estudiante: Juan

Universidad Zaragoza

Métodos

```
class ProgramaEstudiante{
  public static void main(String[] args){
     double[] notas = {5.5,7.0,6.0,8.0};

    Estudiante est = new Estudiante("Juan",notas);

    System.out.println(est + ". Nota media:" + est.mediaNotas());

  }
}
```

Estudiante: Juan. Nota media:6.625

Universidad Zaragoza

Métodos (Dar valor atributos/set)

```
public void setNotasCursos(int indice, double nuevaNota){
    if (this.notasCursos== null)
      this.notasCursos=new double[4];
    if (nuevaNota >=0 && nuevaNota<=10 &&
        indice >=0 && indice <=3){
       this.notasCursos[indice]=nuevaNota;
    }
    else System.out.println("indice o nota erronea");
 }

 public void setNombre(String nombre){
   this.nombre=nombre;
 }
```

Universidad Zaragoza

Métodos (Dar valor atributos/set)

```
public void setNotasCursos(int indice, double nuevaNota){
    if (this.notasCursos== null)
      this.notasCursos=new double[4];
    if (nuevaNota >=0 && nuevaNota<=10 &&
        indice >=0 && indice <=3){
       this.notasCursos[indice]=nuevaNota;
    }
    else System.out.println("indice o nota erronea");
 }

 public void setNombre(String nombre){
   this.nombre=nombre;
 }
```

Universidad Zaragoza

Métodos (Dar valor atributos/set)

```
class ProgramaEstudiante{
  public static void main(String[] args){

    Estudiante est = new Estudiante();
     est.setNombre("Maria");
     est.setNotasCursos(0, 5.5);
     est.setNotasCursos(1, 7.0);
     est.setNotasCursos(2, 6.0);
     est.setNotasCursos(3, 8.0);

    System.out.println(est + " Nota media:" + est.mediaNotas());

  }
}
```

Estudiante: Maria Nota media:6.625

Universidad Zaragoza

Métodos (Leer valor atributos/get)

```
 public String getNombre(){
        return this.nombre;
  }

  public double getNotasCursos(int i){
        return this.notasCursos[i];
}
```

¿Deberíamos tener un método que devolviera un vector de un atributo privado?

Universidad Zaragoza

Métodos (Leer valor atributos/get)

```
class ProgramaEstudiante{
  public static void main(String[] args){
    Estudiante est = new Estudiante();
     est.setNombre("Maria");
     est.setNotasCursos(0, 5.5);
     est.setNotasCursos(1, 7.0);
     est.setNotasCursos(2, 6.0);
     est.setNotasCursos(3, 8.0);

     System.out.println("Estudiante:" + est.getNombre());

    System.out.println("Notas:" );
    for (int i=0;i<4;i++)
     System.out.println("Curso "+(i+1)+":"+
                        est.getNotasCursos(i));
    System.out.println("Nota Media:"+est.mediaNotas());
  }
}
```

```
Estudiante:Maria
Notas:
Curso 1:5.5
Curso 2:7.0
Curso 3:6.0
Curso 4:8.0
Nota Media:6.625
```

Universidad Zaragoza

Reutilizar clases vía **herencia**

```
public class EstudianteCPS extends Estudiante{
    private int NIP;
    public EstudianteCPS(String nombre, int NIP){
        super.setNombre(nombre);
        this.NIP=NIP;
    }
    public void setNIP (int nip){
        this.NIP=nip;
    }
    public int getNIP (){
        return this.NIP;
    }
    public String toString(){
          return "Estudiante del CPS: "+ this.getNombre() +
                  " NIP:"+ this.getNIP();
    }
}
```

Universidad Zaragoza

Reutilizar clases vía herencia

```
class ProgramaEstudiante{
  public static void main(String[] args){

    EstudianteCPS est = new EstudianteCPS("Jose",51999);

     est.setNotasCursos(0, 5.5);
     est.setNotasCursos(1, 7.0);
     est.setNotasCursos(2, 6.0);
     est.setNotasCursos(3, 8.0);

    System.out.println(est + " Nota media:" + est.mediaNotas());

  }
}
```

```
Estudiante del CPS: Jose NIP:51999 Nota media:6.625
```

Universidad Zaragoza

Reutilizar clases vía herencia

```
public class EstudianteCPS extends Estudiante{
    private int NIP;
    public EstudianteCPS(String nombre, int NIP){
        super.setNombre(nombre);
        this.NIP=NIP;
    }
    public void setNIP (int nip){
        this.NIP=nip;
    }
    public int getNIP (){
        return this.NIP;
    }
    public String toString(){
         return "Estudiante del CPS: "+ this.nombre() +
                 " NIP:"+ this.getNIP();
    }
}
```

Los atributos private no son accesibles en las subclases

Universidad Zaragoza

Reutilizar clases vía herencia

```
public class Estudiante{
 // visibilidad tipo nombre [= expresión]
  protected String nombre = "Anonimo";
  private double[] notaCursos;

}
```

Los atributos protected son accesibles en las subclases, no en otras clases

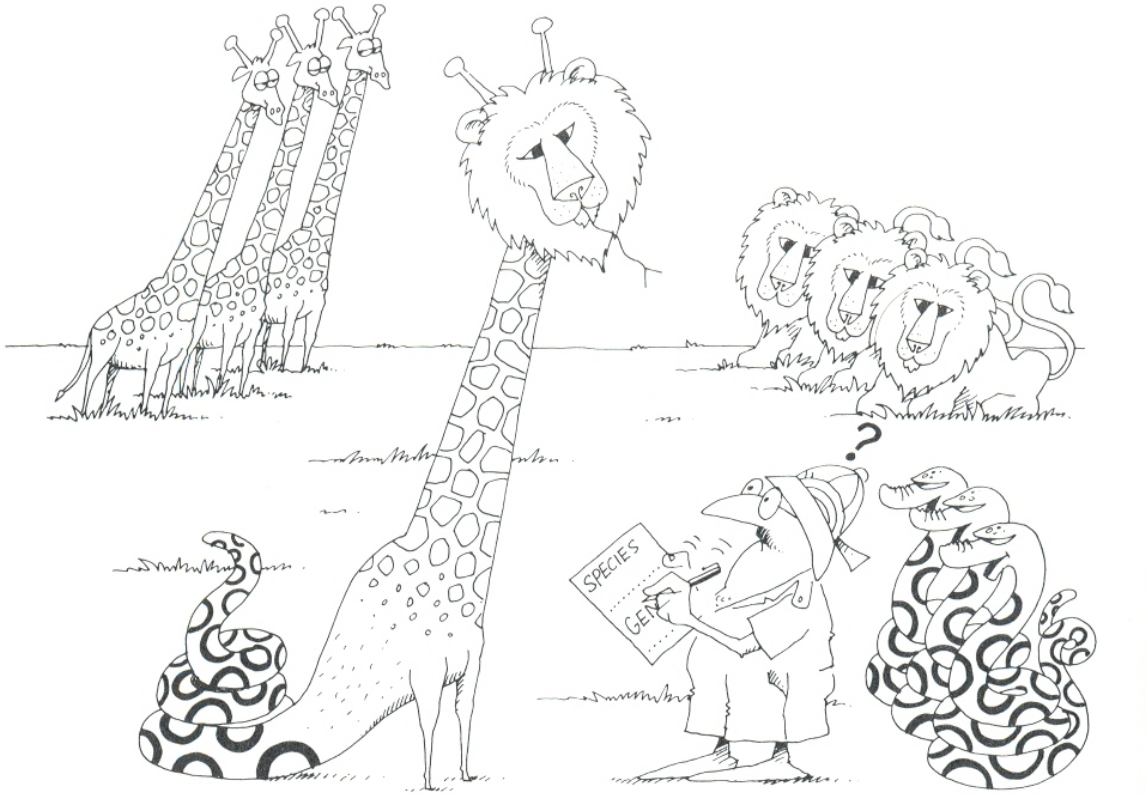
OK

```
public class EstudianteCPS extends Estudiante{
    …
    public String toString(){
         return "Estudiante del CPS: "+ this.nombre() +
                 " NIP:"+ this.getNIP();
    }
}
```

Universidad Zaragoza
1542

La **herencia** nos permite reutilizar código

Object-Oriented Analysis and Design with Applications Grady Booch, Second Edition. Benajmiin/Cummings.



Universidad Zaragoza

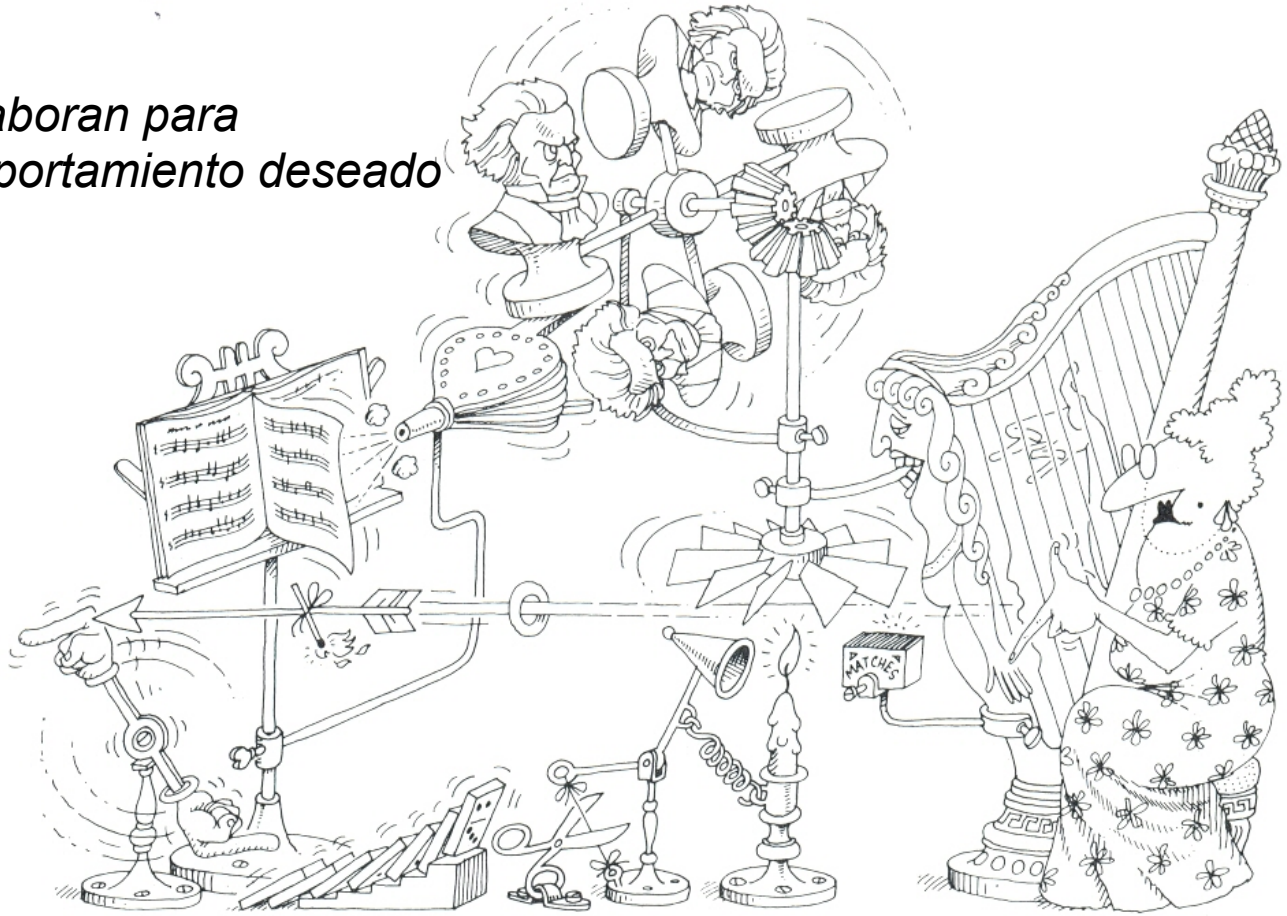
Los Objetos **agregan** datos (otros objetos)

Object-Oriented Analysis and Design with Applications Grady Booch, Second Edition. Benajmiin/Cummings.

Universidad Zaragoza
1542

Los POO reflejan "**mecanismos**"

Los objetos colaboran para soportar el comportamiento deseado



Object-Oriented Analysis and Design with Applications Grady Booch, Second Edition. Benajmiin/Cummings.

Universidad Zaragoza
1542

Resumen

- **Elementos de la programación orientada a objeto**
  1. Abstracción
  2. Encapsulación
  3. Modularidad
  4. Jerarquía: Agregación/herencia

Universidad Zaragoza
1542