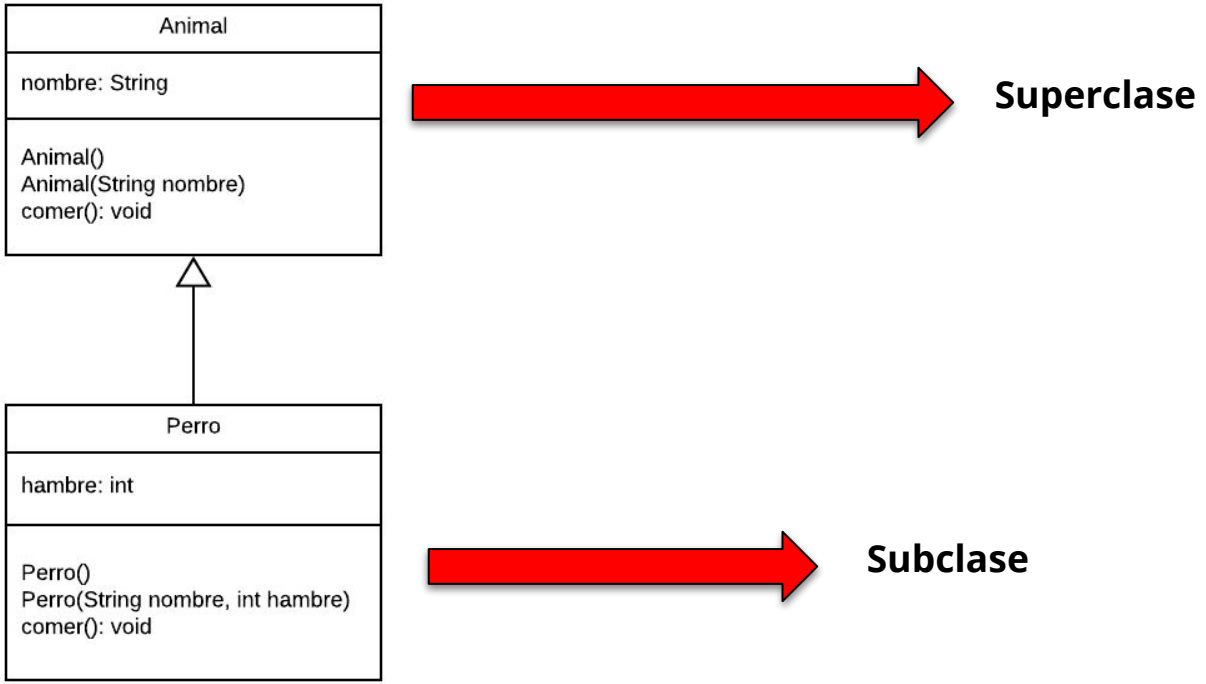
Clase 7: Herencia & Polimorfismo

Programación y Laboratorio III

Agenda

- **Ejemplo UML**
- Palabra reservada `super`
  - Ejemplo
- Modificador de acceso: `protected`
- Ejecución dinámica de métodos
- Polimorfismo
  - Definición
  - Ejemplo
  - Warning
- UML - Ejemplo completo para practicar

Ejemplo UML


Animal
nombre: String
Animal()
Animal(String nombre)
comer(): void
Superclase
Perro
hambre: int
Perro()
Perro(String nombre, int hambre)
comer(): void
Subclase

Agenda

- ~~Ejemplo UML~~
- **Palabra reservada `super`**
  - Ejemplo
- Modificador de acceso: `protected`
- Ejecución dinámica de métodos
- Polimorfismo
  - Definición
  - Ejemplo
  - Warning
- UML - Ejemplo completo para practicar

Palabra reservada: `super`

- Se utiliza para invocar **métodos de la superclase**.

  ```
  super.metodo();
  ```

- En el caso de los constructores, debe ser la primer línea en el constructor de la subclase.

  ```
  super();
  ```

  ó

  ```
  super(lista de parámetros);
  ```

Palabra reservada: `super(2)`

- Si el constructor en la subclase no invoca explícitamente al constructor de la superclase, el compilador de Java lo inserta automáticamente.

```
public Perro(String nombre) {
        [espacio reservado para que el compilador agregue super(); ]
        this.nombre = nombre;
}
```

Agenda

- ~~Ejemplo UML~~
- **Palabra reservada `super`**
  - **Ejemplo**
- Modificador de acceso: `protected`
- Ejecución dinámica de métodos
- Polimorfismo
  - Definición
  - Ejemplo
  - Warning
- UML - Ejemplo completo para practicar

Ejemplo - Palabra reservada: super

```
public class Animal {

      //Variable de instancia
      private String nombre;

      //Constructor por defecto
      public Animal() {
      }

      //Constructor con parámetros
      public Animal(String nombre) {
         this.nombre = nombre;
      }

      //Método
      public void comer() {
         System.out.println("Estoy comiendo");
      }
}
```

Nombre de la clase: primer letra mayúscula y en singular

Los constructores tienen el mismo nombre de la clase!!!

Método concreto (no es abstracto)

Ejemplo - Palabra reservada: super (2)

```
public class Perro extends Animal {

    //Variable de instancia
    private int hambre;

    //Constructor por defecto
    public Perro() {
    }

    //Constructor con parámetros
    public Perro(String nombre, int hambre) {
        super(nombre);
        this.hambre = hambre;
    }

    //Método sobreescrito
    @Override
    public void comer() {
        super.comer();
        hambre --;
    }
}
```

Hay una herencia, por lo tanto Perro es una subclase de Animal

Hereda la variable de instancia “nombre” y agrega una nueva

Llama al constructor de la superclase Animal

Se sobreescribe el método de la superclase

Se llama al método comer() de la superclase Animal

Agenda

- ~~Ejemplo UML~~
- ~~Palabra reservada super~~
  - ~~Ejemplo~~
- **Modificador de acceso: `protected`**
- Ejecución dinámica de métodos
- Polimorfismo
  - Definición
  - Ejemplo
  - Warning
- UML - Ejemplo completo para practicar

Modificador de acceso: protected

```
public class Animal {

    //Variable de instancia
    private String nombre;

    //Constructor por defecto
    public Animal() {
    }

    //Constructor con parámetros
    public Animal(String nombre) {
       this.nombre = nombre;
    }

    //Método
    public String getNombre() {
       return nombre;
    }
}
```

Nombre de la clase: primer letra mayúscula y en singular

El modificador de acceso “private” indica que la variable de instancia “nombre” puede ser leída desde la clase Animal. El resto de las clases que quieran leerla deben acceder a través del getter.

Los constructores tienen el mismo nombre de la clase!!!

Método “getter” para leer desde otro objeto el nombre de la variable “nombre”

Modificador de acceso: protected (2)

```
public class Perro extends Animal {

    public void comer() {
        System.out.println("Me llamo " + this.getNombre() + " y estoy comiendo");
    }
}
```

Hay una herencia, por lo tanto Perro es una subclase de Animal

See lee la variable de instancia "nombre" a través del método getter

Modificador de acceso: protected (3)

```
public class Animal {

    //Variable de instancia
    protected String nombre;

    //Constructor por defecto
    public Animal() {
    }

    //Constructor con parámetros
    public Animal(String nombre) {
       this.nombre = nombre;
    }

    //Método
    public String getNombre() {
       return nombre;
    }

}
```

Nombre de la clase: primer letra mayúscula y en singular

El modificador de acceso “protected” indica que la variable de instancia “nombre” puede ser leída desde las subclases.

Los constructores tienen el mismo nombre de la clase!!!

Modificador de acceso: protected (4)

Hay una herencia, por lo tanto Perro es una subclase de Animal

```
public class Perro extends Animal {

    public void comer() {
       System.out.println("Me llamo " + nombre + " y estoy comiendo");
    }

}
```

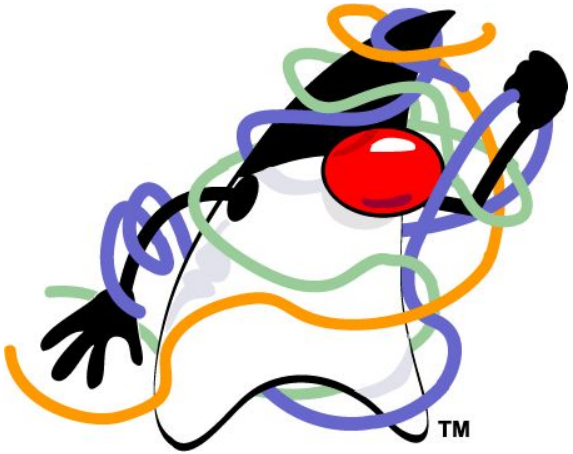
See lee la variable de instancia "nombre" sin necesidad de acceder por el método getter

Agenda

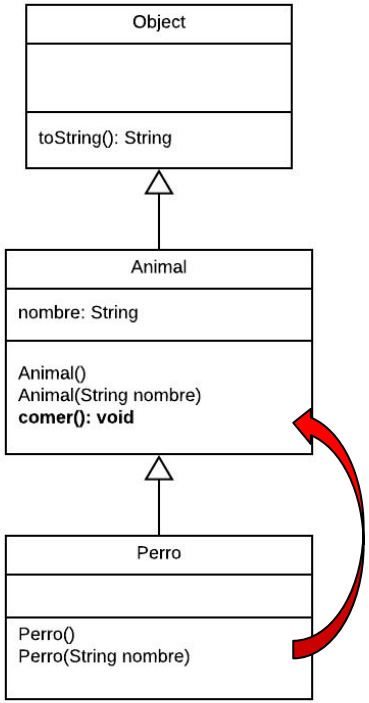
- ~~Ejemplo UML~~
- ~~Palabra reservada super~~
  - ~~Ejemplo~~
- ~~Modificador de acceso: protected~~
- **Ejecución dinámica de métodos**
- Polimorfismo
  - Definición
  - Ejemplo
  - Warning
- UML - Ejemplo completo para practicar

Ejecución dinámica de métodos

- Los métodos sobrescritos de las subclases tienen precedencia sobre los métodos de las subclases.
- La búsqueda del método comienza al final de la jerarquía, entonces la última redefinición de un método es la que se ejecuta primero.



Ejecución dinámica de métodos (2)



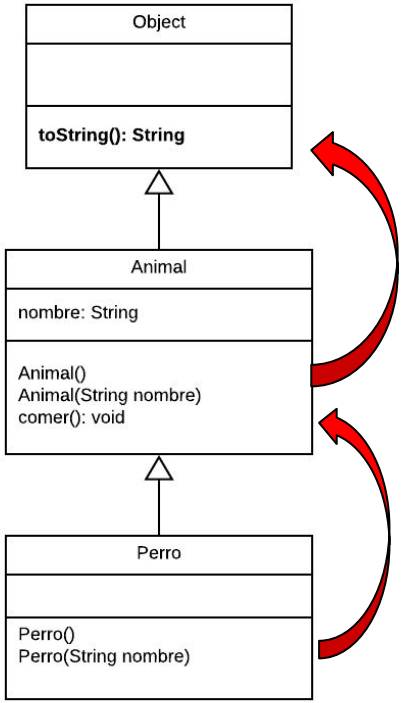
Ejemplo 1: Se quiere ejecutar el método comer() de la clase Perro.

```
public static void main(String [] args) {
   Perro perro = new Perro("Ayudante de Santa");
   perro.comer();
}
```

Para acceder a los métodos de la clase Perro, es necesario crear una instancia.

1) Se busca el método **comer()** en la subclase Perro.
2) Como no se encuentra la sobreescritura, se sube en la jerarquía hasta que se encuentra y se ejecuta.

Ejecución dinámica de métodos (3)



Ejemplo 2: Se quiere ejecutar el método toString() de la clase Perro.

```
public static void main(String [] args) {
   Perro perro = new Perro("Ayudante de Santa");
   System.out.println(perro.toString());
}
```

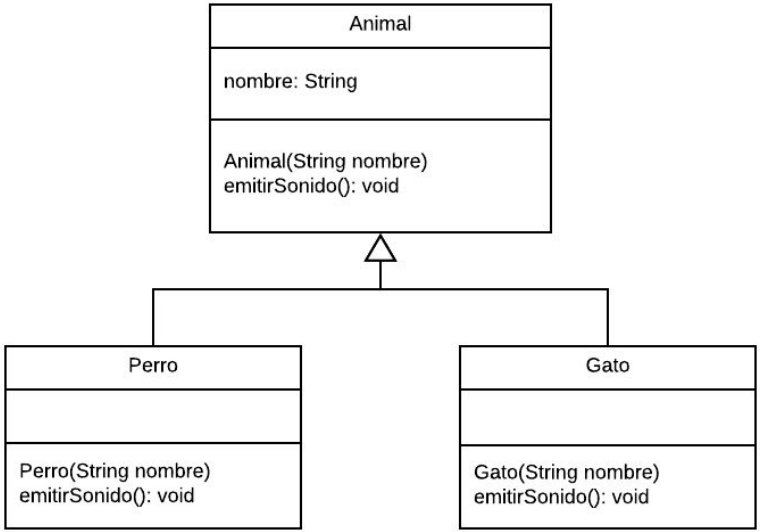
Para acceder a los métodos de la clase Perro, es necesario crear una instancia.

1) Se busca el método **toString()** en la subclase Perro.
2) Como no se encuentra la sobreescritura, se sube en la jerarquía hasta que se encuentra y se ejecuta.

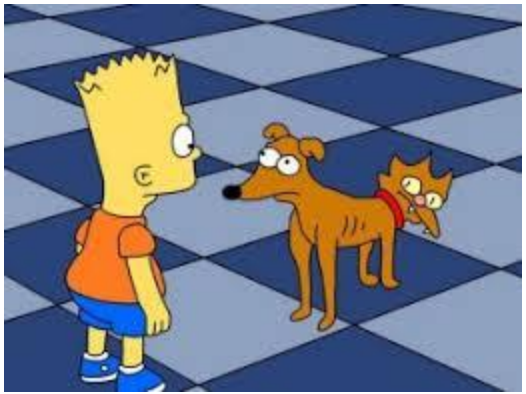
Agenda

- ~~Ejemplo UML~~
- ~~Palabra reservada super~~
  - ~~Ejemplo~~
- ~~Modificador de acceso: protected~~
- ~~Ejecución dinámica de métodos~~
- **Polimorfismo**
  - **Definición**
  - Ejemplo
  - Warning
- UML - Ejemplo completo para practicar

Polimorfismo - Definición



- Se refiere a la propiedad por la que es posible enviar mensajes sintácticamente iguales a objetos de tipos distintos.
- Está ligado a la herencia.



Agenda

- ~~Ejemplo UML~~
- ~~Palabra reservada super~~
  - ~~Ejemplo~~
- ~~Modificador de acceso: protected~~
- ~~Ejecución dinámica de métodos~~
- **Polimorfismo**
  - ~~Definición~~
  - **Ejemplo**
  - Warning
- UML - Ejemplo completo para practicar

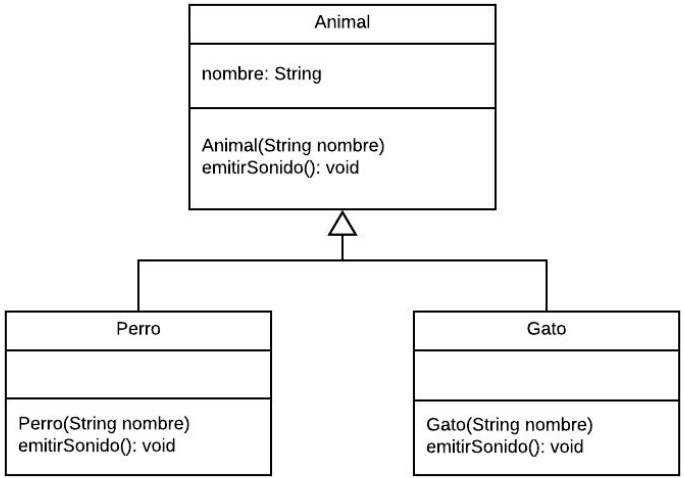
Polimorfismo - Ejemplo

```
public abstract class Animal {

      //Variable de instancia
      protected String nombre;

      //Constructor con parámetros
      public Animal(String nombre) {
         this.nombre = nombre;
      }

      public abstract void emitirSonido();

}
```

```
public class Perro extends Animal {
      @Override
      public void emitirSonido() {
            System.out.println(“Guau!”);
      }
}
```

```
public class Gato extends Animal {
      @Override
      public void emitirSonido() {
            System.out.println(“Miau!”);
      }
}
```

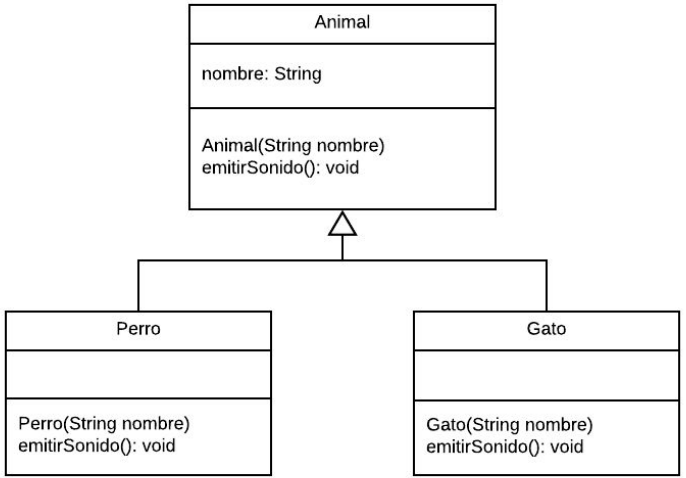
Polimorfismo - Ejemplo (2)




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

     Perro [] perros = new Perro[2];

     Perro perro1 = new Perro(“Ayudante de Santa”);
     Perro perro2 = new Perro(“Procer”);

     perros[0] = perro1;
     perros[1] = perro2;

     for (int i = 0; i < perros.lenght; i++) {
         System.out.println(perros[i].emitirSonido());
     }
}
```

Se invoca al método de la subclase Perro

Polimorfismo - Ejemplo (3)




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

    Animal [] animales = new Animal[2];

    Perro perro = new Perro("Ayudante de Santa");
    Gato gato = new Gato("Bola de nieve I");

    animales[0] = perro;
    animales[1] = gato;

    for (int i = 0; i < animales.lenght; i++) {
        System.out.println(animales[i].emitirSonido());
    }
}
```

Se invoca al método de la subclase Gato cuando animales[1].emitirSonido();

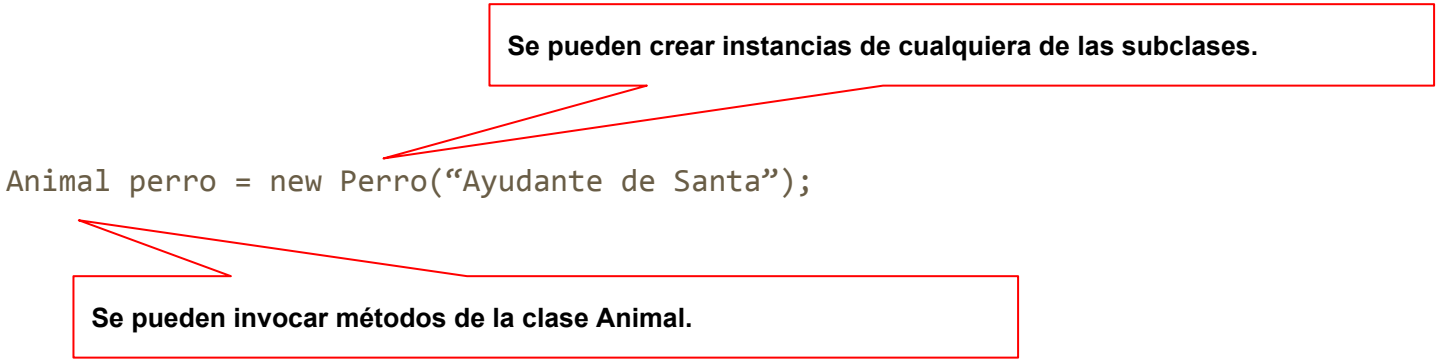
Se invoca al método de la subclase Perro cuando animales[0].emitirSonido();

Agenda

- ~~Ejemplo UML~~
- ~~Palabra reservada super~~
  - ~~Ejemplo~~
- ~~Modificador de acceso: protected~~
- ~~Ejecución dinámica de métodos~~
- **Polimorfismo**
  - ~~Definición~~
  - ~~Ejemplo~~
  - **Warning**
- UML - Ejemplo completo para practicar

Polimorfismo - Warning!

- Los métodos que se pueden invocar son los que contiene la clase del tipo declarado.



- Si hay métodos declarados en la subclase (Perro) que no pertenecen a la superclase (Animal), no se pueden invocar.

Ejemplo - Polimorfismo Warning!

```
public abstract class Animal {

      //Variable de instancia
      protected String nombre;

      //Constructor con parámetros
      public Animal(String nombre) {
         this.nombre = nombre;
      }

      public abstract void emitirSonido();

}
```

```
public class Perro extends Animal {

      @Override
      public void emitirSonido() {
          System.out.println(“Guau!”);
      }

      public void agarrarPelota() {
          System.out.println(“Agarré la pelota”);
      }
}
```

Comportamiento definido para la clase Perro. No existe en la clase Animal.

Ejemplo - Polimorfismo Warning! (2)

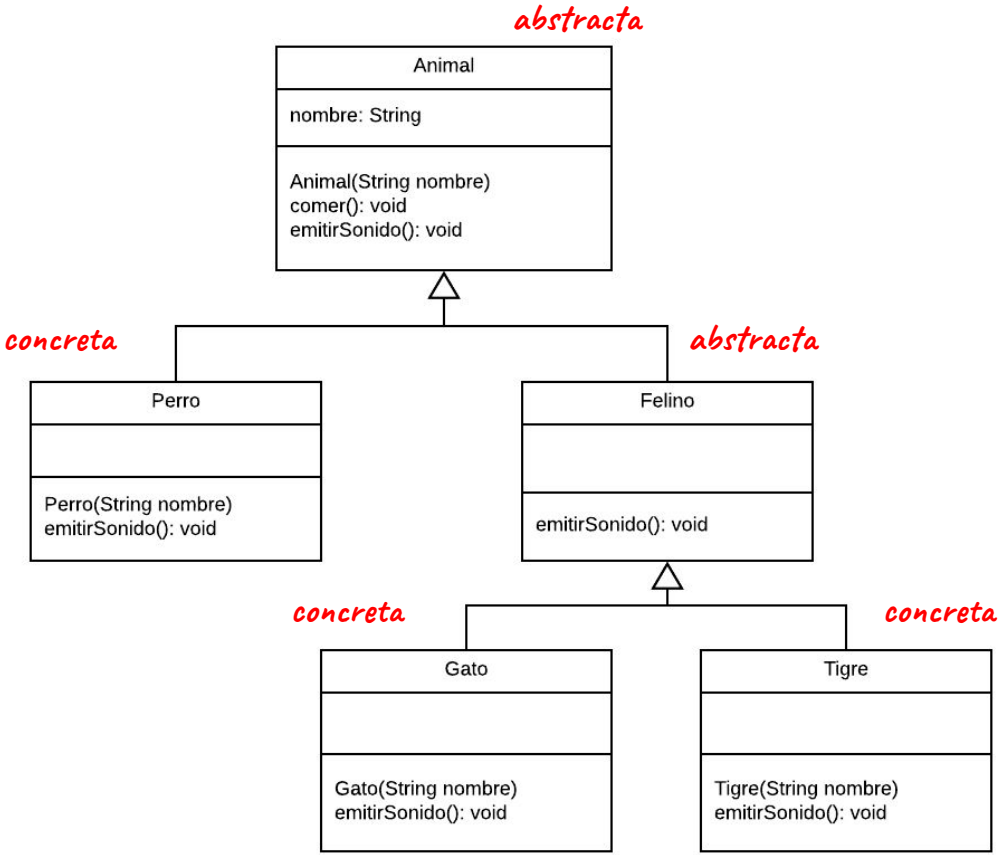
```
Animal perro = new Perro(“Ayudante de Santa”);

perro.agarrarPelota();
```

El método no está definido en la clase Animal!!

Agenda

- ~~Ejemplo UML~~
- ~~Palabra reservada super~~
  - ~~Ejemplo~~
- ~~Modificador de acceso: protected~~
- ~~Ejecución dinámica de métodos~~
- ~~Polimorfismo~~
  - ~~Definición~~
  - ~~Ejemplo~~
  - ~~Warning~~
- **UML - Ejemplo completo para practicar**

UML - Ejemplo completo



Bibliografía oficial

- https://docs.oracle.com/javase/tutorial/java/IandI/super.html
- https://docs.oracle.com/javase/tutorial/java/IandI/objectclass.html
- https://docs.oracle.com/javase/tutorial/java/IandI/polymorphism.html