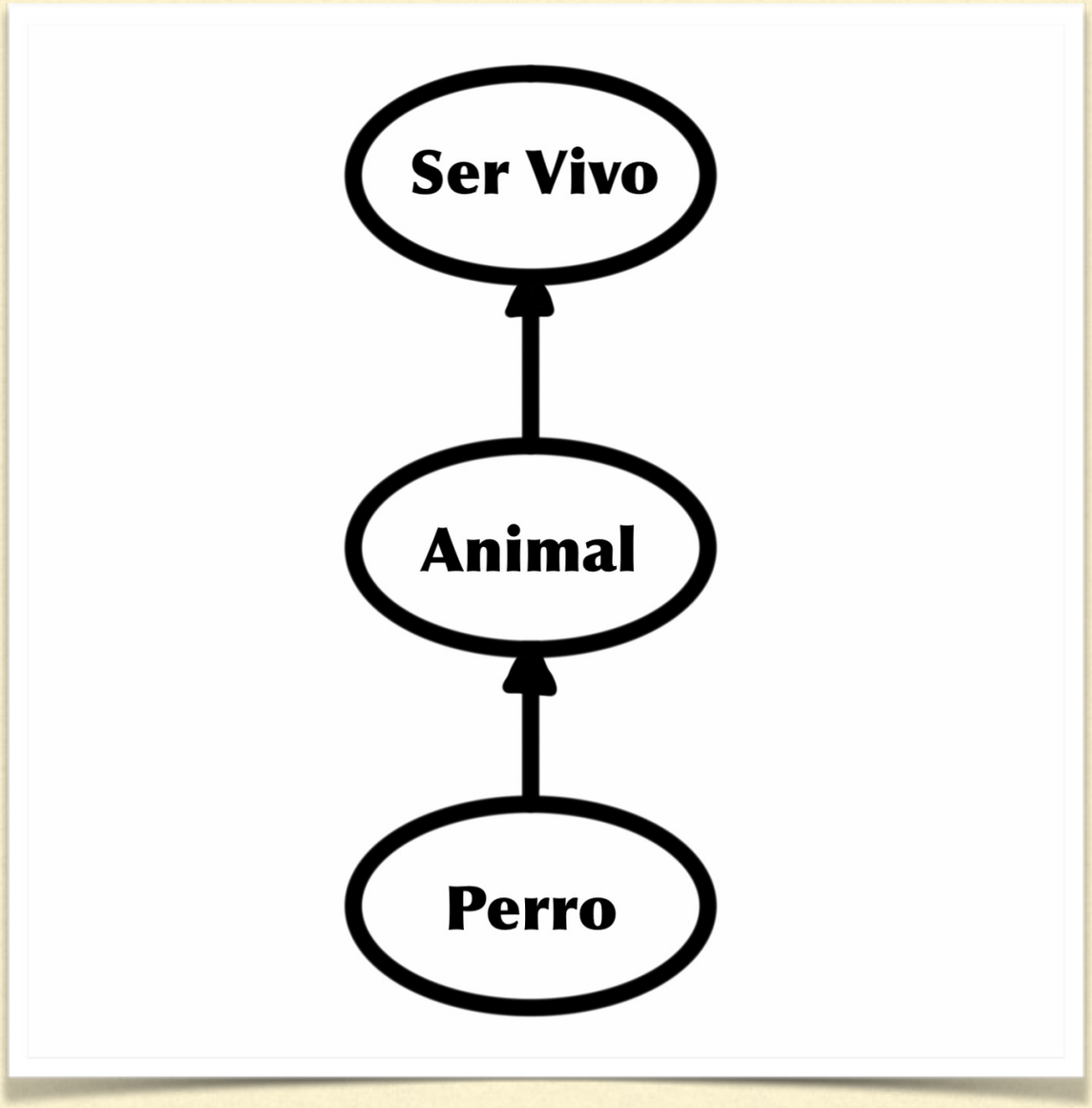
JAVA INTERSEMESTRAL AM 2019

POLIMORFISMO

- Es la capacidad de un objeto en Java, de ser tratado de distintas formas, como si fuera de otra clase.

- Está muy relacionado a la herencia, ya que solo es posible gracias a ella.

POLIMORFISMO



- En este caso, un perro es, a su vez, un animal y un ser vivo.
- Un perro puede ser tratado tanto como animal, como ser vivo.
- Este es un ejemplo de polimorfismo.

POLIMORFISMO

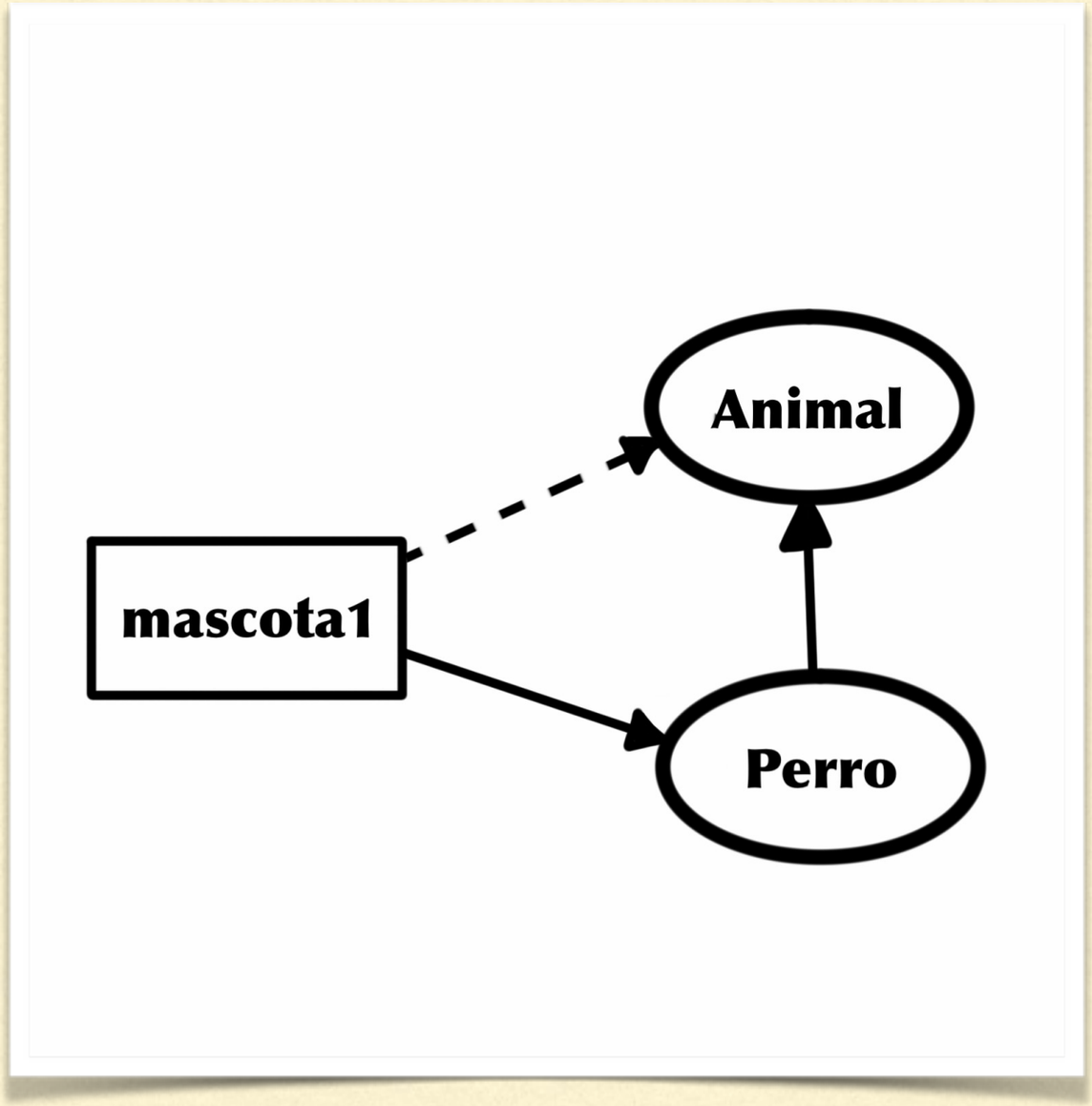
- En Java, una subclase puede ser tratada como si fuera un objeto de su superclase.

- Uno de los ejemplos más comunes y claros se dan cuando declaramos un objeto de una superclase pero lo instanciamos como un objeto de una subclase de la misma.

```
Animal mascota1 = new Perro();
```

POLIMORFISMO

- La clase Perro hereda de la clase Animal.
- Como "mascota1" se instanció como un objeto de la clase perro, es también un objeto de la clase Animal, pero más particular.
- "mascota1" tendría los métodos y atributos de la clase Animal más los métodos y atributos que se agreguen en la clase Perro.



CONVERSIONES

- Cuando se trabaja con herencia, puede llegar a ser necesario tratar una subclase como si fuera una superclase o viceversa, para esto se usan las conversiones.

- Cuando una conversión es de una subclase a una superclase, se dice que es una conversión "hacia arriba" o **"upcasting"**.

- Cuando una conversión es de una superclase a una subclase, se dice que es una conversión "hacia abajo" o **"downcasting"**.

- Una conversión no cambia las características del objeto, únicamente cambia la etiqueta con la cual se maneja dicho objeto.

UPCASTING

- Al hacer una conversión hacia arriba no se tiene que hacer el casteo manualmente.

```
Animal mascota = (Animal) new Perro();

Animal mascota = new Perro();
```

DOWNCASTING

- Al hacer una conversión hacia abajo el casteo siempre se tiene que hacer manualmente.

```
Perro mascota1 = new Perro();

Animal animal = mascota1;

Perro mascota2 = (Perro) animal;
```

CLASES Y MÉTODOS ABSTRACTOS

- Una clase abstracta es una clase que reúne las características generales de otras clases más particulares.

- Funciona para unificar estas características y que otras clases puedan heredarlas de ella.

- Un método abstracto funciona de manera similar, ya que es un método general entre clases, pero cada clase lo implementa de manera diferente.

CLASES ABSTRACTAS

- Al declarar una clase abstracta, se deberá incluir la palabra reservada “abstract”.

```
public abstract class Animal{...}
```

- Una clase abstracta tiene métodos y atributos, con la diferencia de que mínimo uno de sus métodos debe de ser abstracto.

CLASES ABSTRACTAS

- La herencia funciona igual en las clases abstractas que con las clases normales.
- Siempre se va a buscar heredar de una clase abstracta.
- UNA CLASE ABSTRACTA NO SE PUEDE INSTANCIAR.

MÉTODOS ABSTRACTOS

- Estos métodos no tienen cuerpo, únicamente se declaran, incluyendo la palabra reservada “abstract”.

```
public abstract void hacerSonido();
```

- Solo se pueden declarar métodos abstractos en una clase abstracta.

- Cuando una clase los hereda, éstos tienen que sobreescribirse forzosamente.

INTERFACES

- Una interfaz es una colección de métodos abstractos. Funciona como una clase abstracta, pero no tiene atributos, únicamente puede tener métodos abstractos y constantes.

- Una interfaz se declara utilizando la palabra reservada “interface”, en vez de la palabra “class”.

```
public interface Cazador{

      public static final constante = valor;

      public abstract void cazar();

}
```

INTERFACES

- En Java no se permite la herencia múltiple, pero esta se puede simular con interfaces, ya que sí se permite que una clase use varias interfaces.

- Para usar una interfaz en una clase se agrega al declararla usando la palabra reservada “implements”.

```
public class Perro implements Cazador{...}
```

- Al igual que con las clases abstractas, todos sus métodos abstractos se deben sobreescribir forzosamente.