Resumen de guía 3

Java

Conceptos básicos

- Relaciones
  - Dependencia (usa)
  - Asociación (tiene)
    - Composición
    - Agregación
  - Herencia (es un)
- Manejo de Fechas

Relaciones entre objetos

El paradigma orientado a objetos, se basa en la creación de objetos que interactúan entre sí para resolver problemas a partir de la definición de clases. Los objetos colaboran e intercambian información, existiendo distintos tipos de relaciones entre ellos.

A nivel de diseño, podemos distinguir entre 5 tipos de relaciones básicas entre clases de objetos:

dependencia, asociación, agregación, composición y herencia

Relaciones entre objetos

Dependencia **USA**

Asociación - Agregación **TIENE**

Asociación - Composición **COMPONE**

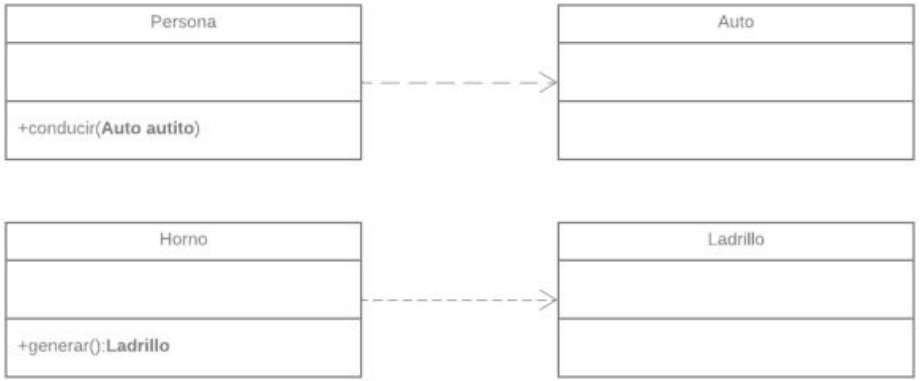
Herencia **ES**

Implementa **ES**

Dependencia

En el mundo real la dependencia significa la necesidad de tener elementos acoplados en los cuales unos necesitan de otros para su funcionamiento, los sistemas deben ser diseñados con bajos niveles de acoplamiento.

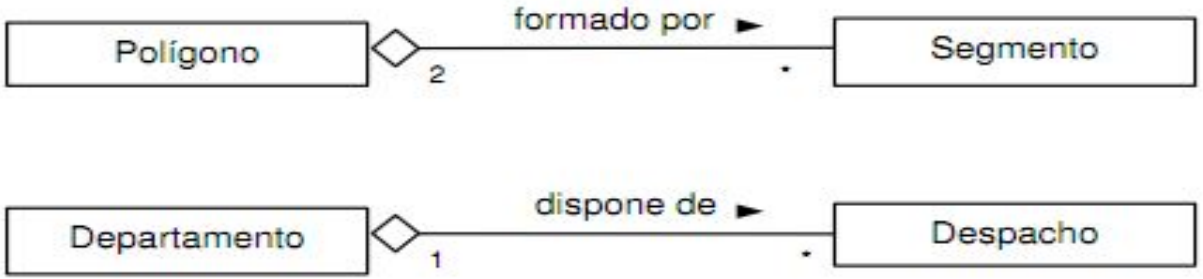
Una clase depende de otra, cuando: uno de los parámetros o el tipo de retorno de cualquiera de los métodos de la clase dependiente es del tipo de la clase independiente.



Agregación

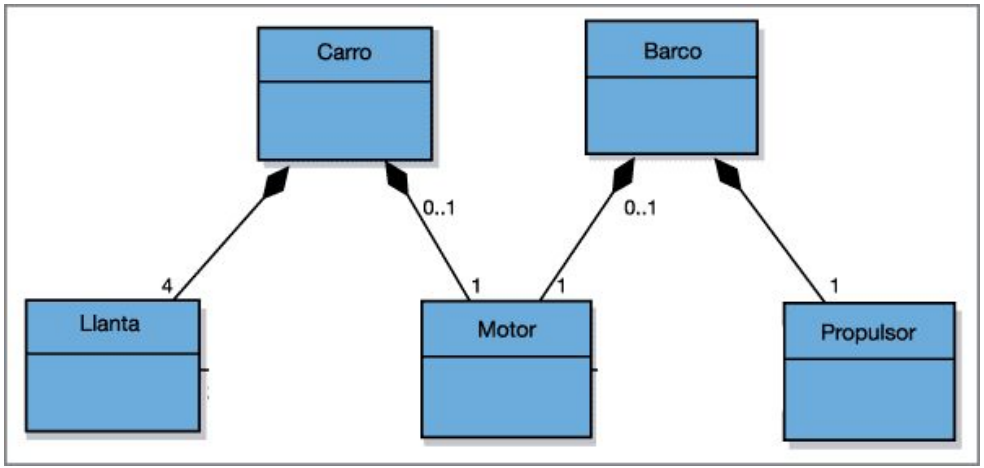
Refleja la relación entre dos clases independientes que se mantiene durante la vida de los objetos de dichas clases o al menos durante un tiempo prolongado. Representa una relación del tipo "tiene un".

Una asociación se implementa en Java, introduciendo referencias a objetos de una clase como atributos en la otra.



Composición

Es un tipo de agregación que añade el matiz de que la clase “todo” controla la existencia de las clases “parte”. Es decir, normalmente la clase “todo” creará al principio las clases “parte” y al final se encargará de su destrucción.

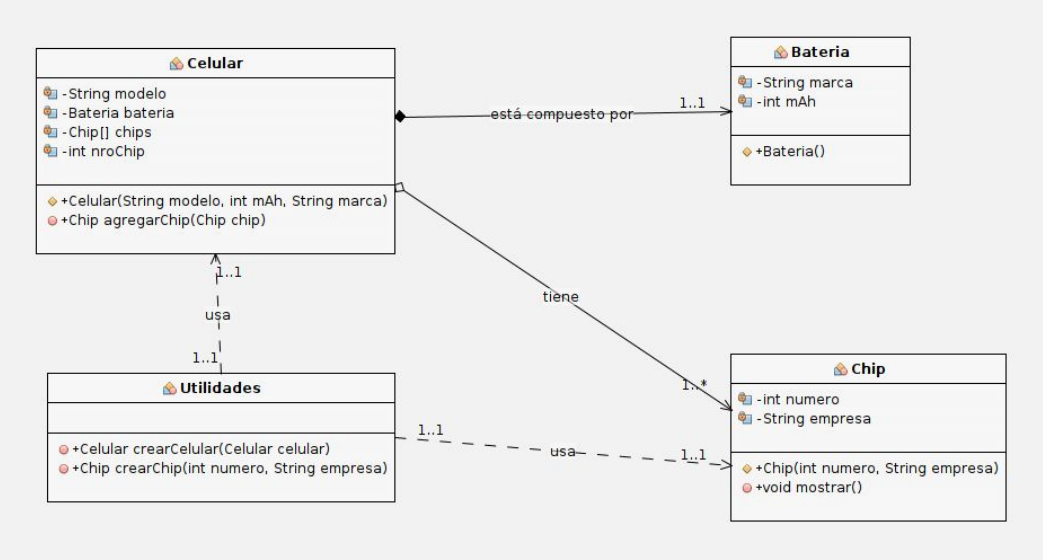


Ejemplo con los tres tipos de relaciones

ASOCIACIÓN: La clase Utilidades usa a las clases Celular y chip.

AGREGACIÓN La clase Celular tiene un Chip

COMPOSICIÓN: tiene (está compuesta) por una Batería.



Análisis de la Composición

La asociación tipo composición es la relación más fuerte que existe. La “clase todo” Cuenta es la clase de la que depende la clase Bateria. Cuando un objeto tipo Celular es eliminado también se eliminará su Bateria.

Cada Bateria se crea (new) desde dentro de la clase Celular.

```
public class Celular {
    private String modelo;
    private Bateria bateria;
    private Chip[] chips;
    private int nroChips;

    public Celular(String modelo, int mAh, String marca) {
        this.modelo=modelo;
        bateria=new Bateria(marca, mAh);
        chips=new Chip[2];
        nroChips=0;
    }

    public String getModelo() {...3 lines }

    public void setModelo(String modelo) {...3 lines }

    public Chip[] getChips() {...3 lines }

    public void setChips(Chip chip) {
        if(nroChips<2){
           this.chips[nroChips] = chip;
           nroChips++;
        }
        else
            System.out.println(x: "No hay capacidad para más chips");
    }
    @Override
    public String toString() {
        return "Celular{" + "modelo=" + modelo + ", bateria=" + bateria + ", "
                + "chips=" + chips[0].getNumero()+"-"+chips[1].getNumero() +
                ", nroChips=" + nroChips + '}';
    }

}
```

Composición

Agregacion

Análisis de la Agregación

En la clase Celular existe un atributo tipo Chip, pero el mismo se crea desde fuera de la clase Celular, no en la clase Celular.

El Chip se recibe por parámetro, de tal modo que si el objeto tipo Celular desaparece el objeto tipo Chip sigue existiendo.

Herencia

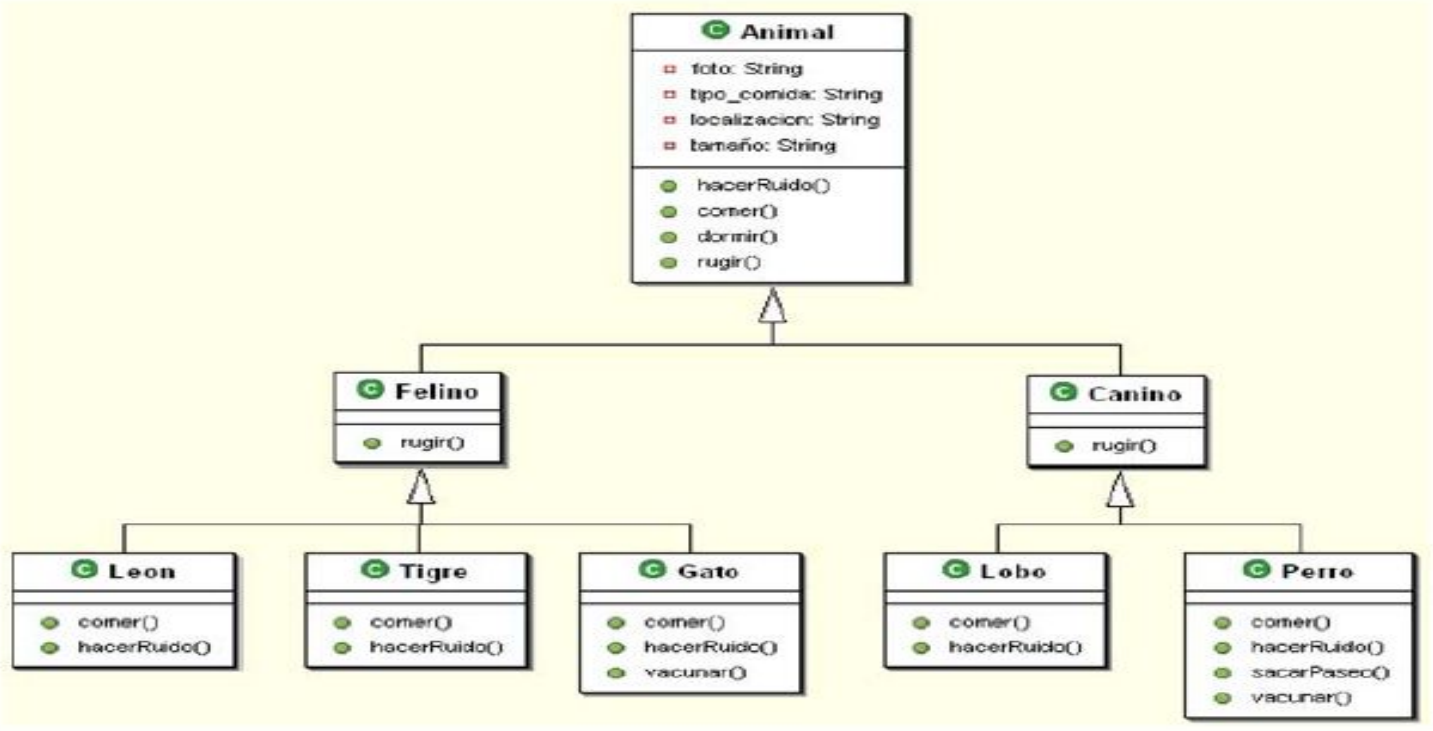
Relación de herencia

- Se basa en la existencia de relaciones de generalización/especialización entre clases.
- Las clases se disponen en una jerarquía, donde una clase hereda los atributos y métodos de las clases superiores en la jerarquía.
- Una clase puede tener sus propios atributos y métodos adicionales a lo heredado.
- Una clase puede modificar los atributos y métodos heredados.

Relación de herencia

- Las clases por encima en la jerarquía a una clase dada, se denominan superclases.
- Las clases por debajo en la jerarquía a una clase dada, se denominan subclases.
- Una clase puede ser superclase y subclase al mismo tiempo.
- Tipos de herencia:
  - Simple.
  - Múltiple (no soportada en Java)

Ejemplo


Animal
foto: String
tipo_comida: String
localizacion: String
tamaño: String
hacerRuido()
comer()
dormir()
rugir()
Felino
rugir()
Canino
rugir()
Leon
comer()
hacerRuido()
Tigre
comer()
hacerRuido()
Gato
comer()
hacerRuido()
vacunar()
Lobo
comer()
hacerRuido()
Perro
comer()
hacerRuido()
sacarPaseo()
vacunar()

Ejemplos LocalDate

```java
import java.time.LocalDate;
import java.time.Month;

public class Ejemplo {
    public static void main(String[] args){
        LocalDate fechaActual=LocalDate.now();
        int year = fechaActual.getYear();
        Month mes = fechaActual.getMonth();
        int dia = fechaActual.getDayOfMonth();

        LocalDate fechaNacimiento = LocalDate.of(year:1990, month:12, dayOfMonth:31);
    }
}
```

```java
import java.time.LocalDate;
import java.time.temporal.ChronoUnit;

public class Main {
    public static void main(String[] args) {
        LocalDate fechaActual = LocalDate.now();
        LocalDate proxaño = fechaActual.plus(amountToAdd: 1, unit:ChronoUnit.YEARS);
        LocalDate proxmes = fechaActual.plus(amountToAdd: 1, unit:ChronoUnit.MONTHS);
        LocalDate proxsem = fechaActual.plus(amountToAdd: 1, unit:ChronoUnit.WEEKS);
        LocalDate mañana = fechaActual.plus(amountToAdd: 1, unit:ChronoUnit.DAYS);
        LocalDate proxHora = fechaActual.plus(amountToAdd: 1, unit:ChronoUnit.HOURS);
        LocalDate proxmin = fechaActual.plus(amountToAdd: 1, unit:ChronoUnit.MINUTES);
        LocalDate proxseg = fechaActual.plus(amountToAdd: 1, unit:ChronoUnit.SECONDS);

        System.out.println("Próximo año: " + proxaño);
        System.out.println("Próximo mes: " + proxmes);
        System.out.println("Próxima semana: " + proxsem);
        System.out.println("mañana: " + mañana);
        System.out.println("Próxima hora: " + proxHora);
        System.out.println("Próximo minuto: " + proxmin);
        System.out.println("Próximo segundo: " + proxseg);
    }
}
```