

## 1. Java

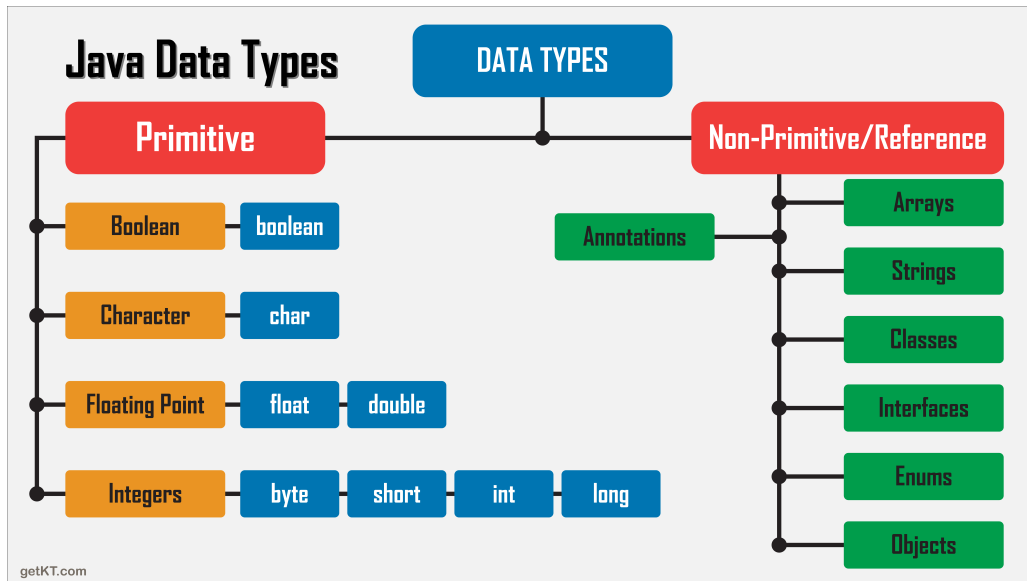


Figura 1: Tipos de datos en Java [4]

### 1.1. Tipos de clases en JAVA

- **Clases:** Son las clases comunes y corrientes, que se pueden instanciar.
- **Clases Abstractas:** Son clases que no se pueden instanciar, pero que sirven para heredar.
- **Interfaces:** Son clases que no se pueden instanciar, pero que sirven para heredar.
- **Clases Anónimas:** Son clases que no tienen nombre, y se usan para sobrescribir métodos.
- **Clases finales:** Son clases que no se pueden heredar.

### 1.2. Relaciones entre clases

Relaciones en UML: [1]

1. **Asociación:** Indica que una propiedad de una clase contiene una referencia a una instancia (o instancias) de otra clase.

La asociación es la relación más utilizada entre una clase y otra clase, lo que significa que existe una conexión entre un tipo de objeto y otro tipo de objeto. Las combinaciones y agregaciones también pertenecen a las relaciones asociativas, pero las relaciones entre clases de afiliaciones son más débiles que las otras dos.

Hay cuatro tipos de asociaciones: asociaciones bidireccionales, asociaciones unidireccionales, autoasociación y asociaciones de números múltiples.

Por ejemplo: coches y conductores, un coche corresponde a un conductor en particular y un conductor puede conducir varios coches.



Figura 2: Asociación

- a) **Agregación:** La relación entre el todo y la parte, y el todo y la parte se pueden separar. Las relaciones agregadas también representan la relación entre el todo y una parte de la clase, los objetos miembros son parte del objeto general, pero el objeto miembro puede existir independientemente del objeto general. Tiempo de vida *independiente*.

Por ejemplo, los conductores de autobús y la ropa y los sombreros de trabajo son parte de la relación general, pero se pueden separar. La ropa de trabajo y los sombreros se pueden usar en otros conductores. Los conductores de autobuses también pueden usar otra ropa de trabajo y sombreros.

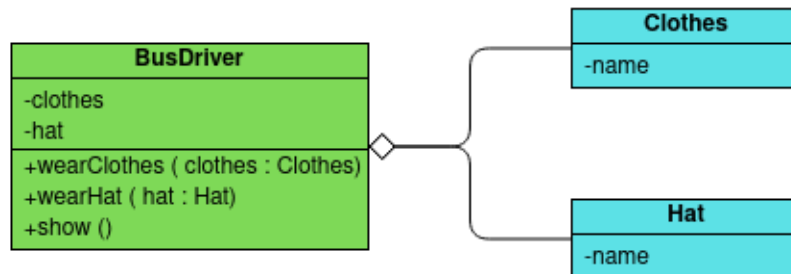


Figura 3: Agregación

Otro ejemplo puede ser el de un auto y sus ruedas, el auto depende si o si de tener cuatro ruedas.

- b) **Composición:** La relación entre el todo y la parte, pero el todo y la parte no se pueden separar. La relación de combinación representa la relación entre el todo y la parte de la clase, y el total y la parte tienen una duración constante. Una vez que el objeto general no existe, algunos de los objetos no existirán y todos morirán en la misma vida. Tiempo de vida *dependiente*.

Por ejemplo, una persona está compuesta por una cabeza y un cuerpo. Los dos son inseparables y coexisten.

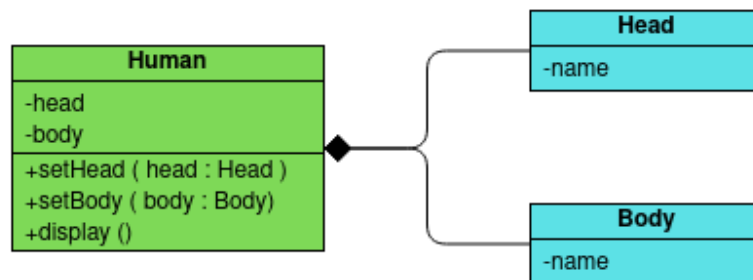


Figura 4: Composición

2. **Herencia/Generalización:** La herencia también se denomina generalización y se utiliza para describir la relación entre las clases padre e hijo. Una clase principal también se denomina clase base y una subclase también se denomina clase derivada.

En la relación de herencia, la subclase hereda todas las funciones de la clase principal y la clase principal tiene todos los atributos, métodos y subclases. Las subclases contienen información adicional además de la misma información que la clase principal.

a

Por ejemplo: autobuses, taxis y automóviles son automóviles, todos tienen nombres y todos pueden estar en la carretera.

3. **Dependencia:** Suponga que un cambio en la clase A provoca un cambio en la clase B, luego diga que la clase B depende de la clase A. En la mayoría de los casos, las dependencias se reflejan en los métodos de una clase que utilizan el objeto de otra clase como parámetro. Una relación de dependencia es una relación de “uso”. Un cambio en una cosa en particular puede afectar a otras cosas que la usan, y usar una dependencia cuando es necesario indicar que una cosa usa otra.

Por ejemplo: El auto depende de la gasolina. Si no hay gasolina, el automóvil no podrá conducir.

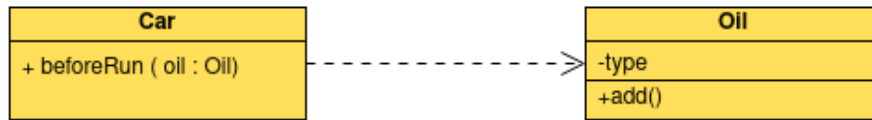


Figura 5: Dependencia

4. **Realización/Implementación:** La implementación (Implementación) se utiliza principalmente para especificar la relación entre las interfaces y las clases de implementación.

Una interfaz (incluida una clase abstracta) es una colección de métodos. En una relación de implementación, una clase implementa una interfaz y los métodos de la clase implementan todos los métodos de la declaración de la interfaz.

Por ejemplo: los automóviles y los barcos son vehículos, y el vehículo es solo un concepto abstracto de una herramienta móvil, y el barco y el vehículo realizan las funciones móviles específicas.

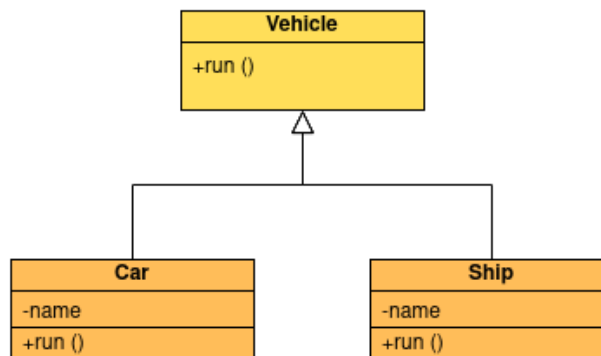


Figura 6: Realización/Implementación

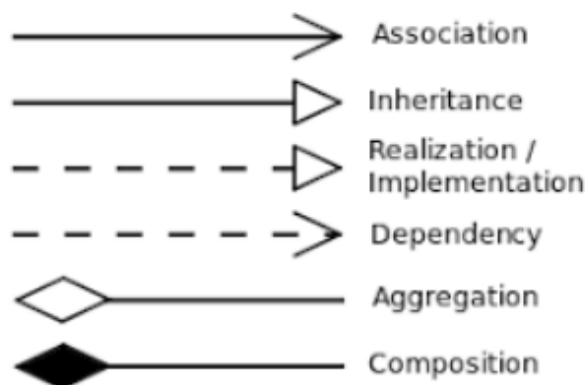


Figura 7: UML Relaciones entre clases

### 1.3. Interface

Interfaces: [\[3\]](#).

- Son una **colección de métodos abstractos** con propiedades (atributos) **constantes**.
- Una interfaz **solamente puede extender o implementar otras interfaces** (la cantidad que quiera).
- Da a conocer qué se debe hacer (métodos) **pero sin mostrar su implementación** (solo puede tener métodos abstractos).
- Solo puede tener **métodos** con **métodos públicos** (no pueden ser protected o private).
- Solo puede tener "variables" public static final (o sea constantes).
- La palabra tener **abstract** en la definición de métodos no es obligatoria.
- Generalmente las interfaces indican el **PUEDE HACER** de un objeto.