Clases y Objetos

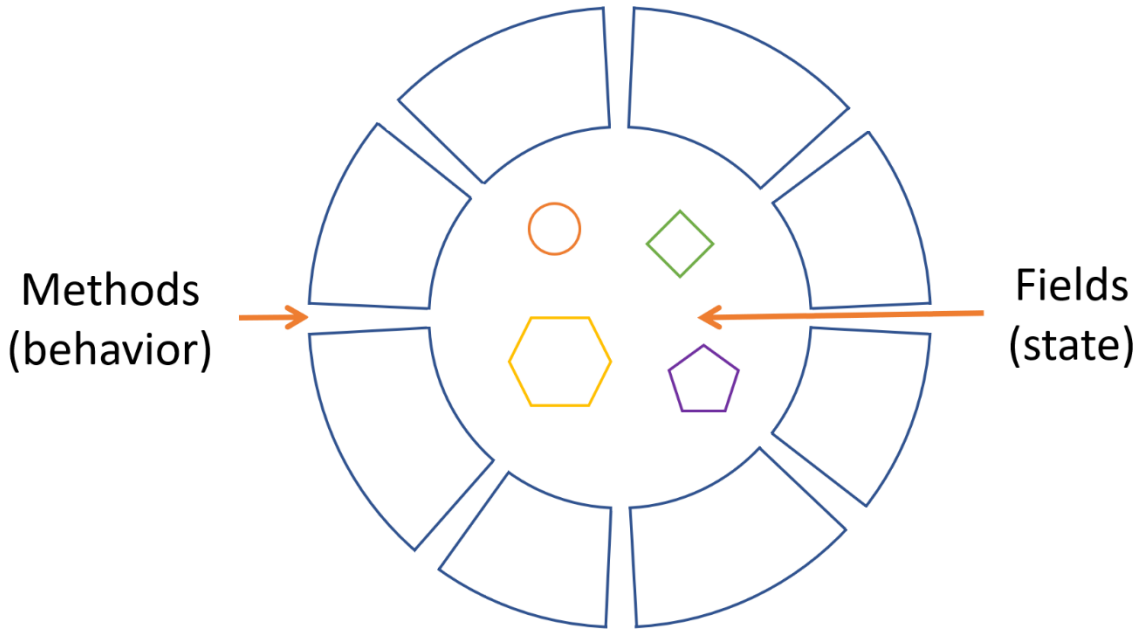
Si nunca ha utilizado un lenguaje de programación orientado a objetos, deberá aprender algunos conceptos básicos antes de comenzar a escribir código. Esta sección le presentará los objetos, las clases, la herencia, las interfaces y los paquetes. Cada discusión se centra en cómo se relacionan estos conceptos con el mundo real, al mismo tiempo que proporciona una introducción a la sintaxis del lenguaje de programación Java.

¿Qué es un objeto?

Un objeto es un conjunto de software que contiene estados y comportamientos relacionados. En esta sección se explica cómo se representan el estado y el comportamiento dentro de un objeto, se presenta el concepto de encapsulación de datos y se explican los beneficios de diseñar el software de esta manera.

Los objetos comparten dos características: todos tienen estado y comportamiento. Los perros tienen estado (nombre, color, raza, hambre) y comportamiento (ladrar, ir a buscar algo, mover la cola). Las bicicletas también tienen estado (marcha actual, cadencia de pedaleo actual, velocidad actual) y comportamiento (cambiar de marcha, cambiar la cadencia de pedaleo, aplicar los frenos). Identificar el estado y el comportamiento de los objetos del mundo real es una excelente manera de comenzar a pensar en términos de programación orientada a objetos.

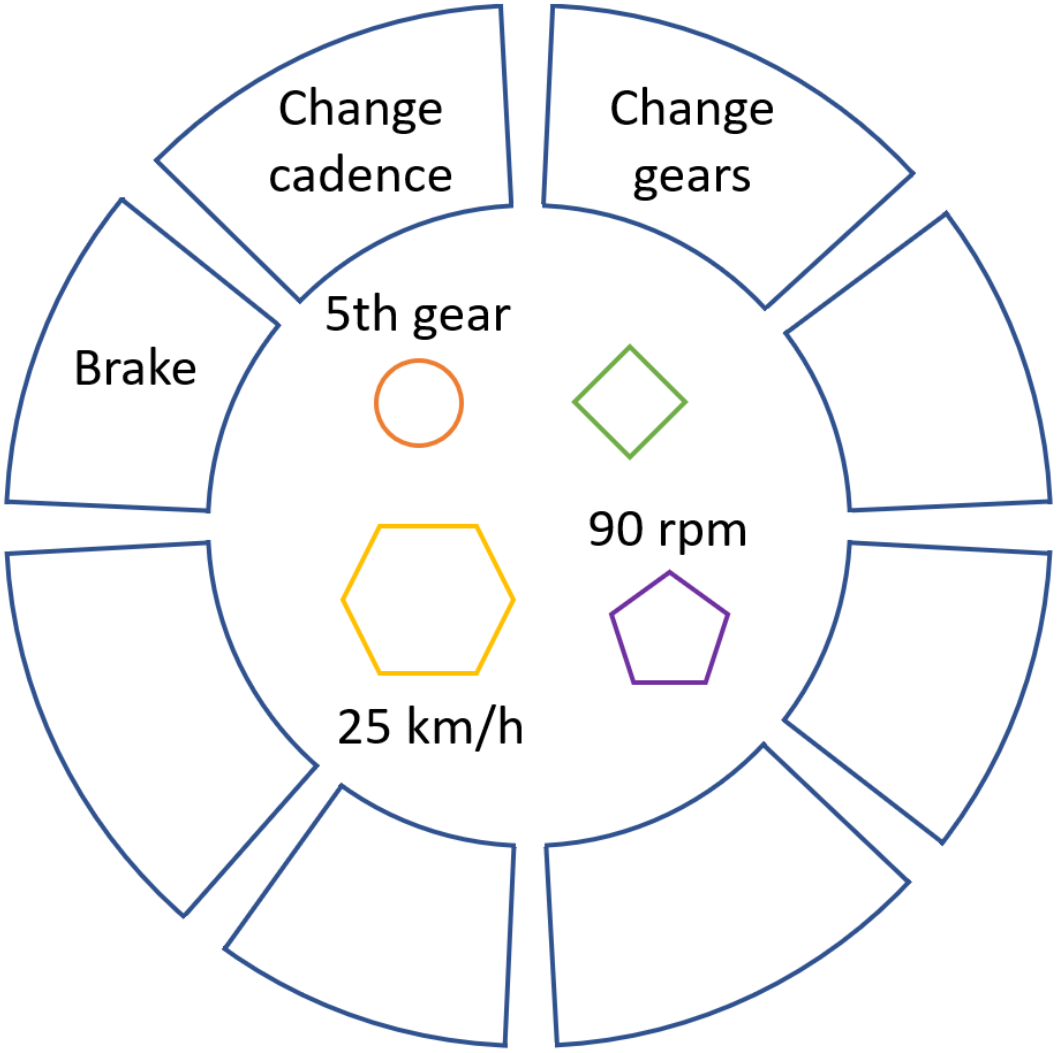
Tómese un minuto ahora mismo para observar los objetos del mundo real que se encuentran en su área inmediata. Para cada objeto que vea, hágase dos preguntas: "¿En qué estados posibles puede estar este objeto?" y "¿Qué comportamiento posible puede realizar este objeto?". Asegúrese de anotar sus observaciones. A medida que lo haga, notará que los objetos del mundo real varían en complejidad; su lámpara de escritorio puede tener solo dos estados posibles (encendido y apagado) y dos comportamientos posibles (encender, apagar), pero su radio de escritorio puede tener estados adicionales (encendido, apagado, volumen actual, estación actual) y comportamiento (encender, apagar, aumentar el volumen, disminuir el volumen, buscar, escanear y sintonizar). También puede notar que algunos objetos, a su vez, también contendrán otros objetos. Estas observaciones del mundo real son un punto de partida para comprender el mundo de la programación orientada a objetos.



Un objeto de software

Los objetos de software constan de estado y comportamiento relacionado. Un objeto almacena su estado en *campos* (variables en algunos lenguajes de programación) y expone su comportamiento a través de *métodos* (funciones en algunos lenguajes de programación). Los métodos operan sobre el estado interno de un objeto y sirven como mecanismo principal para la comunicación entre objetos. Ocultar el estado interno y exigir que toda interacción se realice a través de los métodos de un objeto se conoce como *encapsulación de datos*, un principio fundamental de la programación orientada a objetos.

Consideremos una bicicleta, por ejemplo:



Una bicicleta modelada como un objeto de software

Al atribuir un estado (velocidad actual, cadencia de pedaleo actual y marcha actual) y proporcionar métodos para cambiar ese estado, el objeto mantiene el control sobre cómo se le permite al mundo exterior usarlo. Por ejemplo, si la bicicleta solo tiene 6 marchas, un método para cambiar de marcha podría rechazar cualquier valor que sea menor que 1 o mayor que 6.

Agrupar código en objetos de software individuales proporciona una serie de beneficios, entre ellos:

1. **Modularidad:** el código fuente de un objeto se puede escribir y mantener independientemente del código fuente de otros objetos. Una vez creado, un objeto se puede pasar fácilmente dentro del sistema.
2. **Ocultamiento de información:** al interactuar únicamente con los métodos de un objeto, los detalles de su implementación interna permanecen ocultos al mundo exterior.
3. **Reutilización de código:** si ya existe un objeto (tal vez escrito por otro desarrollador de software), puede utilizarlo en su programa. Esto permite que los especialistas implementen, prueben y depuren objetos complejos y específicos de la tarea, que luego puede ejecutar en su propio código.
4. **Conectividad y facilidad de depuración:** si un objeto en particular resulta problemático, simplemente puede eliminarlo de su aplicación y conectar un objeto diferente como reemplazo. Esto es análogo a

solucionar problemas mecánicos en el mundo real. Si se rompe un perno, lo reemplaza, no toda la máquina.

¿Qué es una clase?

En sus aplicaciones, a menudo encontrará muchos objetos individuales, todos del mismo tipo. Puede haber miles de otras bicicletas en existencia, todas de la misma marca y modelo. Cada bicicleta se construyó a partir del mismo conjunto de planos y, por lo tanto, contiene los mismos componentes. En términos orientados a objetos, decimos que su bicicleta es una instancia de la *clase de objetos* conocidos como bicicletas.
Una *clase* es el plano a partir del cual se crean los objetos individuales.

La siguiente clase **Bicycle** es una posible implementación de una bicicleta:

```
class Bicycle {

    int cadence = 0;
    int speed = 0;
    int gear = 1;

    void changeCadence(int newValue) {
         cadence = newValue;
    }

    void changeGear(int newValue) {
         gear = newValue;
    }

    void speedUp(int increment) {
         speed = speed + increment;
    }

    void applyBrakes(int decrement) {
         speed = speed - decrement;
    }

    void printStates() {
         System.out.println("cadence:" +
             cadence + " speed:" +
             speed + " gear:" + gear);
    }
}
```

La sintaxis del lenguaje de programación Java puede parecerle nueva, pero el diseño de esta clase se basa en la discusión previa sobre los objetos de bicicleta. Los campos **cadence**, **speed**, y **gear** representan el estado del objeto, y los métodos (**changeCadence(), changeGear(), speedUp()** etc.) definen su interacción con el mundo exterior.

Es posible que hayas notado que la clase **Bicycle** no contiene ningún **main()**método. Esto se debe a que no es una aplicación completa; es solo el plano de las bicicletas que se pueden usar en una aplicación. La responsabilidad de crear y usar nuevos objetos **Bicycle** corresponde a otra clase de tu aplicación.

Aquí hay una clase **BicycleDemo** que crea dos objetos **Bicycle** separados e invoca sus métodos:

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

        // Create two different
        // Bicycle objects
        Bicycle bike1 = new Bicycle();
        Bicycle bike2 = new Bicycle();

        // Invoke methods on
        // those objects
        bike1.changeCadence(50);
        bike1.speedUp(10);
        bike1.changeGear(2);
        bike1.printStates();

        bike2.changeCadence(50);
        bike2.speedUp(10);
        bike2.changeGear(2);
        bike2.changeCadence(40);
        bike2.speedUp(10);
        bike2.changeGear(3);
        bike2.printStates();
    }
}
```

El resultado de esta prueba imprime la cadencia del pedal final, la velocidad y la marcha de las dos bicicletas:

```
cadence:50 speed:10 gear:2
cadence:40 speed:20 gear:3
```

Bibliografía: https://dev.java/learn/oop/