Introdução à programação Orientada a Objetos

PCII - Programação Orientada a Objetos em Java

FEG - UNESP - 2021

Sumário

Orientação a Objetos

O que são objetos?

- Objetos do mundo real: lousa, apagador, mesa, professor, aluno, cão, bicicleta, carro.
- Duas características essenciais de objetos do mundo real: estado e comportamento.
- Identificar o estado e comportamento dos objetos é a melhor maneira para começar a pensar em termos de programação orientada a objetos.

Objetos do mundo real

Identidade: **cão**

Estado:	Comportamentos:
Nome	Latir
Raça	Balançar Rabo
Cor	Fingir de Morto
Fome	Alimentar

Objetos do mundo real

Identidade: **bicicleta**

Estado:	Comportamentos:
Velocidade	Aumentar cadência
Cadência	Trocar marcha
Marcha	Acelerar
Modelo	Frear

Objetos do mundo real

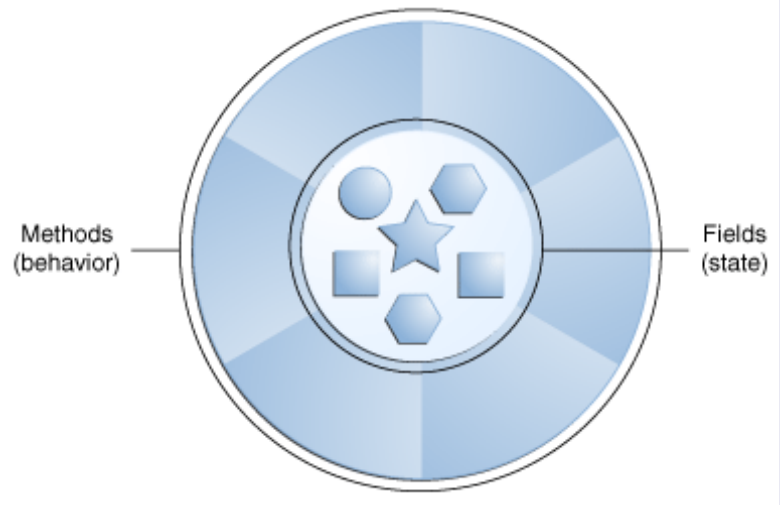
- Objetos diferentes possuem complexidades diferentes.
- Lanterna:
  - *Estados*: ligado e desligado.
  - *Comportamentos*: ligar e desligar.
- Rádio:
  - *Estados*: ligado, desligado, volume atual, estação atual.
  - *Comportamentos*: ligar, desligar, aumentar volume, diminuir volume, procurar estação, sintonizar.

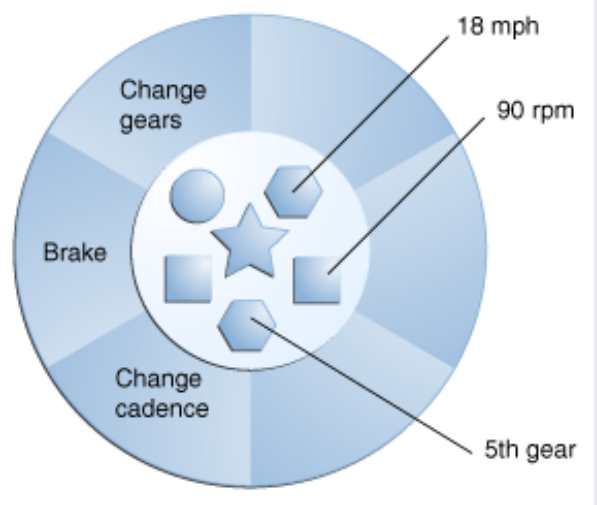
Orientação a Objetos

Objetos de software

- Objetos em software também possuem estados e comportamentos.
- Um objeto armazena seu estado em atributos (*variáveis*) e assumem comportamentos através de métodos (*funções*).
- Os métodos de um objeto operam sobre seu estado interno e são os principais mecanismos da intercomunicação objeto-objeto.
- A possibilidade de esconder o estado interno de um objeto e obrigar que todas as interações com um objeto sejam realizadas por métodos é conhecido como encapsulamento.

Objetos de software





Fonte: http://docs.oracle.com/javase/tutorial/java/concepts/object.html

POO – Programação Orientada a Objetos

Desenvolver um programa computacional constituído por um conjunto de objetos relacionados que interagem de forma a resolver um problema.

POO – Programação Orientada a Objetos

- Etapas fundamentais do design orientado a objetos de um software:
  - Analisar os requisitos que descrevem o sistema desejado.
  - Determinar os objetos necessários para implementar o sistema.
  - Determinar os atributos que os objetos terão.
  - Determinar os comportamentos que esses objetos exibirão.
  - Especificar como ocorre a interação entre os objetos para atender aos requisitos do sistema.

POO – Programação Orientada a Objetos

- Benefícios da programação orientada a objetos:
  - Abstração
  - Modularidade
  - Encapsulamento
  - Reuso de código
  - Alteração e depuração

Abstração

- O conceito de *abstração* ou *modelagem* significa decompor um sistema complicado em suas partes fundamentais, descrevendo-as em uma linguagem simples e precisa.
- Os componentes do sistema e suas funcionalidades são identificadas e descritas.
- Essa descrição passa a ser uma abstração ou um modelo do sistema, que pode ser implementado como um objeto em uma linguagem orientada a objetos.

Modularidade

- Sistemas modernos de software são constituídos por diversos componentes (objetos) distintos que interagem entre si.
- A modularidade prevê que o código-fonte de um objeto pode ser escrito e mantido de modo independente de outros objetos.
- Essa independência deve estar associada a uma estrutura de organização dos diferentes objetos, para que eles possam interagir corretamente, fazendo com que todo sistema funcione de forma adequada.

Orientação a Objetos

Encapsulamento

- A interação de um objeto com o meio externo é realizada exclusivamente através de seus métodos.
- Os detalhes de implementação são mantidos escondidos do meio externo.

Reuso de código

- Frequentemente precisamos de objetos já implementados em outras situações e por outros desenvolvedores.
- Esses objetos podem ser incorporados em um novo programa.
- Objetos complexos e de propósito específico podem ser implementados e testados exaustivamente antes de serem incorporados ao programa computacional.

Alteração e depuração

- Se um objeto tornar-se problemático, é possível simplesmente fazer a remoção e substituição por outro objeto que esteja funcional.
- *“se uma máquina possui uma engrenagem defeituosa, não troque a máquina, mas somente a engrenagem”*.

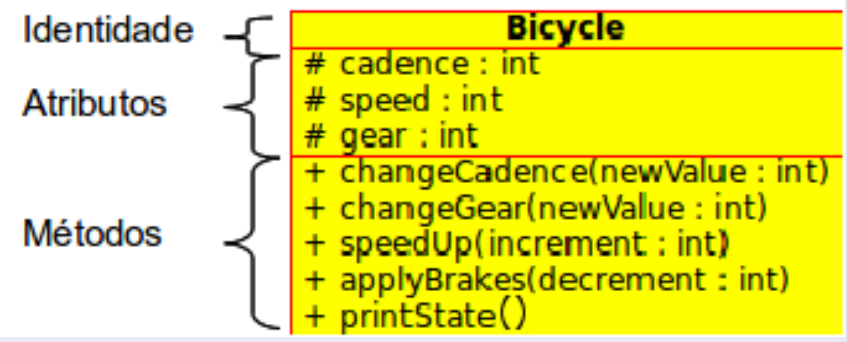
O que são classes?

- No mundo real, é possível encontrar diversos exemplares de objetos de um mesmo tipo.
- Existem muitas unidades de bicicleta de um mesmo tipo, dotadas dos mesmos componentes e feitas a partir de um mesmo desenho (modelo).
- Em termos de orientação a objetos, dizemos que um objeto bibicleta é uma instância de uma classe que modela bicicletas.
- Uma classe é uma receita através da qual objetos são criados.

O que são classes?

- A modelagem de classes pode ser auxiliada por meio de diagramas de classes da linguagem UML.
- A UML é uma linguagem de representação gráfica para a modelagem de sistemas orientados a objetos.
- Essa linguagem é rica em recursos e possui diversos diagramas para a descrição estrutural, comportamental e de interações.

Diagrama UML para a classe Bicicleta



Software Umbrello para elaboração de diagramas UML (*Unified Modelling Language*): http://umbrello.kde.org/

Orientação a Objetos

```java
/* Definição de uma classe Bicicleta */
public class Bicycle {

    // Atributos (estado) de uma bicicleta
    protected int cadence = 0; // unidade: rpm
    protected int speed = 0;   // unidade: km/h
    protected int gear = 1;

    // Métodos (comportamentos) de uma bicicleta
    public void changeCadence(int newValue) {
         cadence = newValue;
    }

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

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

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

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

}
```

Classes em Java

- Ainda sem conhecer a sintaxe de Java, é possível perceber que a classe `Bicycle` segue o mesmo modelo descrito para bicicletas.
- Os atributos `cadence`, `speed` e `gear` representam o estado do objeto.
- Os métodos `changeCadence()`, `changeGear()`, `speedUp()` etc. definem o modo como o objeto interage com o meio externo.

Classes em Java

- A classe `Bicycle` consiste somente em uma receita para a criação de objetos do tipo Bicicleta.
- A classe `BicycleDemo` a seguir mostra a instanciação de dois objetos da classe `Bicycle` e a manipulação de atributos pela chamada de seus métodos.

Orientação a Objetos

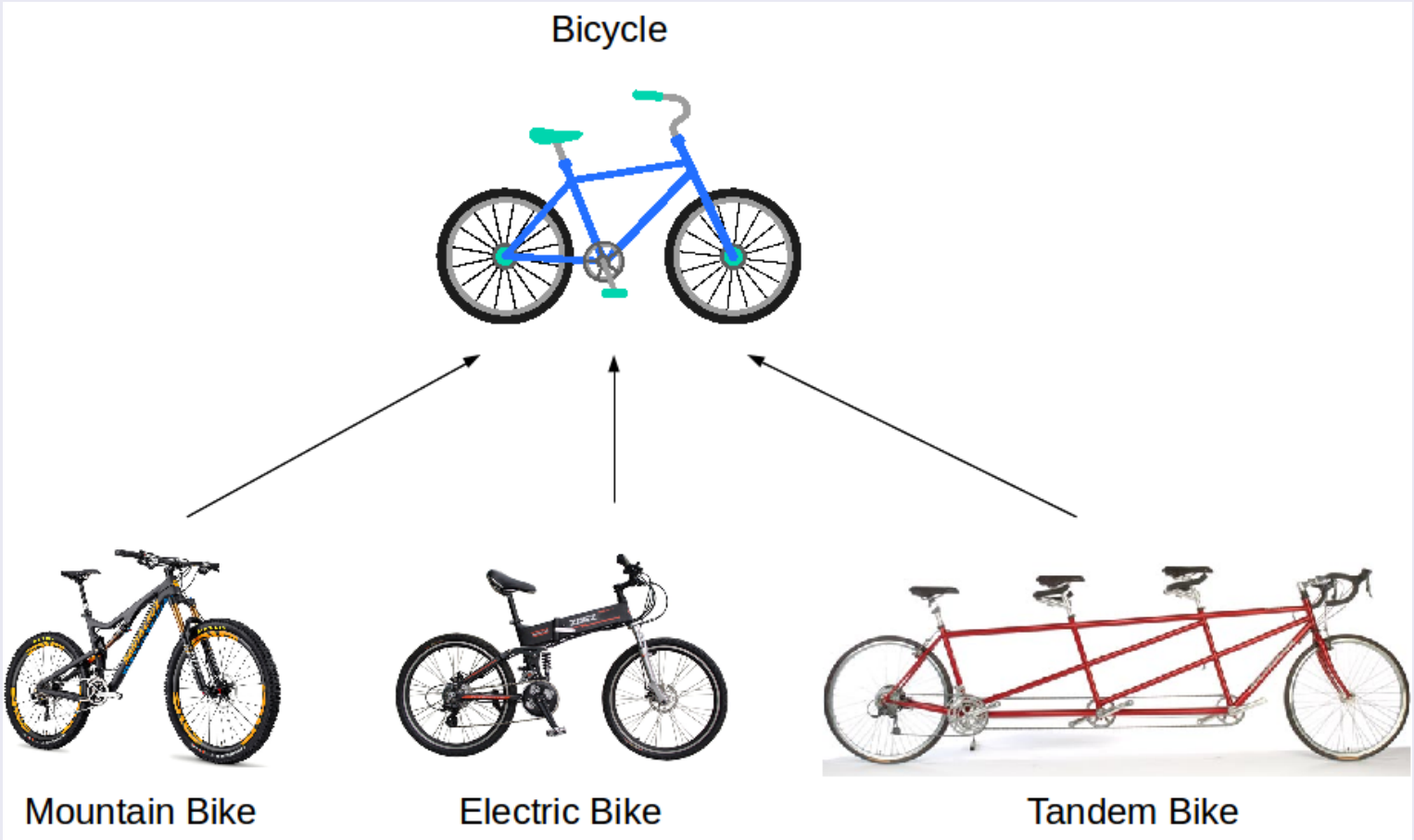
```
/* Instanciando objetos da classe Bicicleta */
class BicycleDemo {
    public static void main(String[] args) {

        // Criando dois objetos Bicicleta
        Bicycle bike1 = new Bicycle();
        Bicycle bike2 = new Bicycle();

        // Chamando métodos dos objetos instanciados
        bike1.changeCadence(50);
        bike1.speedUp(10);
        bike1.changeGear(2);
        bike1.printState();

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

```
// Saída da execução da classe BicycleDemo
cadence:50 speed:10 gear:2
cadence:40 speed:20 gear:3
```

O que é herança entre objetos?

- Objetos de tipos diferentes muitas vezes possuem propriedades em comum.
- Exemplos: mountain bikes, bicicletas elétricas e bicicletas tandem.
- Esses objetos têm em comum atributos como: velocidade, cadência e marcha atual.
- No entanto, esses objetos também possuem atributos que os especializam.

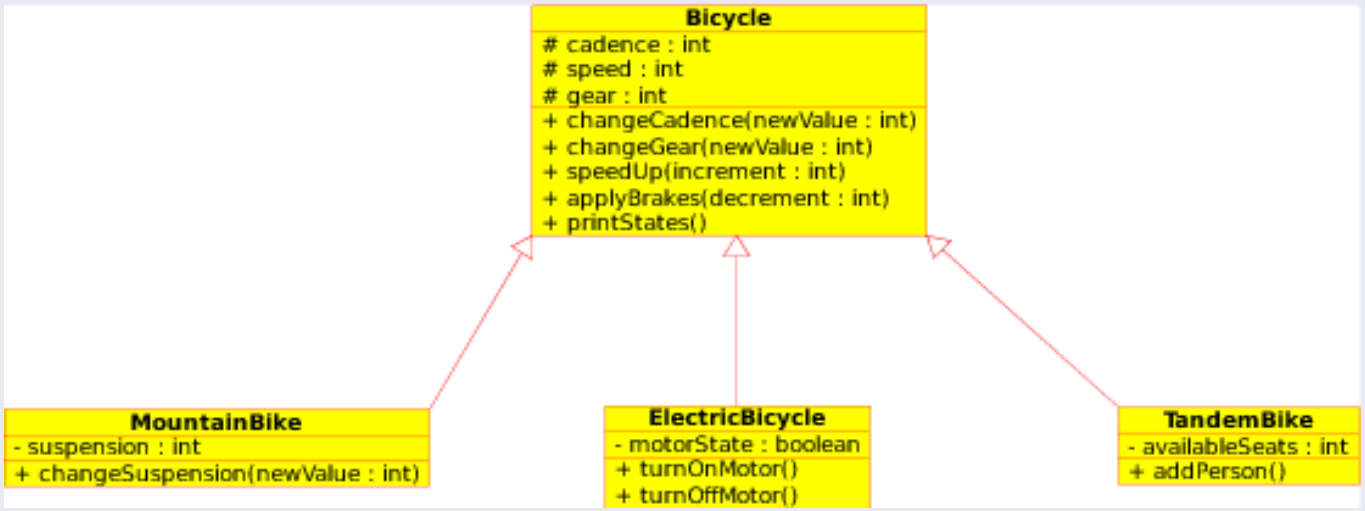
Exemplo de associação entre objetos



Herança

- A programação orientada a objetos permite que as classes herdem estados e comportamentos de outras classes.
- Exemplo: Bicicleta pode ser modelada como uma superclasse das (sub)classes *mountain bike*, *bicicleta elétrica* e *bicicleta tandem*.
- Particularmente em Java, uma classe pode ter uma única superclasse (imediata) e cada superclasse pode ter um número indefinido de subclasses.

Diagrama UML para a classe Bicicleta e suas subclasses



Software Umbrello para elaboração de diagramas UML (*Unified Modelling Language*): http://umbrello.kde.org/

Herança

- O código a seguir exemplifica a sintaxe de declaração de uma subclasse.

```
/* Exemplo de declaração da subclasse ElectricBicycle  */
class ElectricBicycle extends Bicycle {

    private boolean motorState;

    public void turnOnMotor() {
        motorState = true;
    }

    public void turnOffMotor() {
        motorState = false;
    }
}
```

Herança

- A subclasse `BicicletaEletrica` possui os atributos e métodos da superclasse `Bicicleta`, mas o código centraliza exclusivamente na implementação das características da subclasse.
- Isso torna o código da subclasse mais fácil de entender.
- A documentação do estado e comportamento da superclasse deve ser cuidadosa pois os atributos e métodos da superclasse não aparecem no código-fonte das subclasses.

Referências

1. Java: Como Programar, Paul Deitel & Heivey Deitel; Pearson; 8a. Ed.
2. ECKEL, B. Thinking in Java. 2a. Ed.
   http://mindview.net/Books
3. The Java Tutorials (Oracle)
   http://docs.oracle.com/javase/tutorial/
4. Introduction to Computer Science using Java
   http://chortle.ccsu.edu/java5/index.html