面向对象

@M了个J
李明杰

https://github.com/CoderMJLee
http://cnblogs.com/mjios

码拉松

小码哥教育 SEEMYGO

类的定义、对象的创建

```
public class Dog {
    // 成员变量
    public int age;
    public double weight;
    // 方法
    public void run() {
        System.out.println(age + "_" + weight + "_run");
    }
    public void eat(String food) {
        System.out.println(age + "_" + weight + "_eat_" + food);
    }
}
```

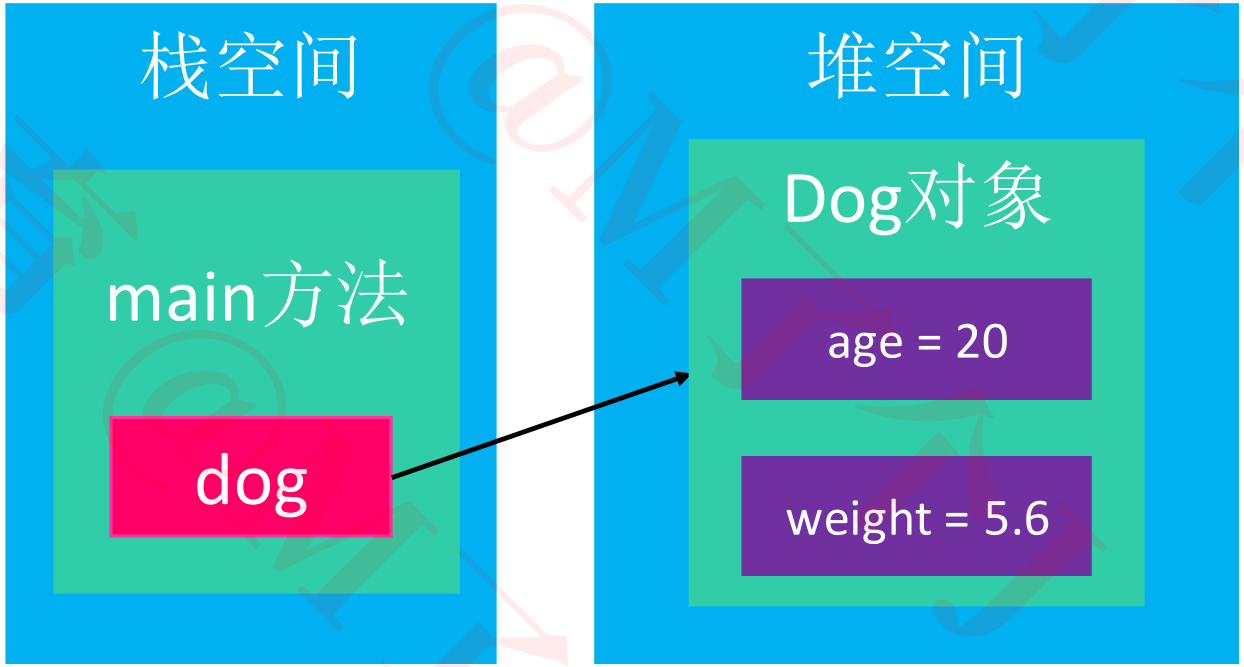
```
public static void main(String[] args) {
    Dog dog = new Dog();
    dog.age = 20;
    dog.weight = 5.6;
    dog.run();
    dog.eat("appel");
}
```

- 成员变量（Member Variable）也叫做字段（Field）

对象的内存

- Java 中所有对象都是 new 出来的，所有对象的内存都是在堆空间，所有保存对象的变量都是引用类型

```
public static void main(String[] args) {
    Dog dog = new Dog();
    dog.age = 20;
    dog.weight = 5.6;
    dog.run();
    dog.eat("appel");
}
```

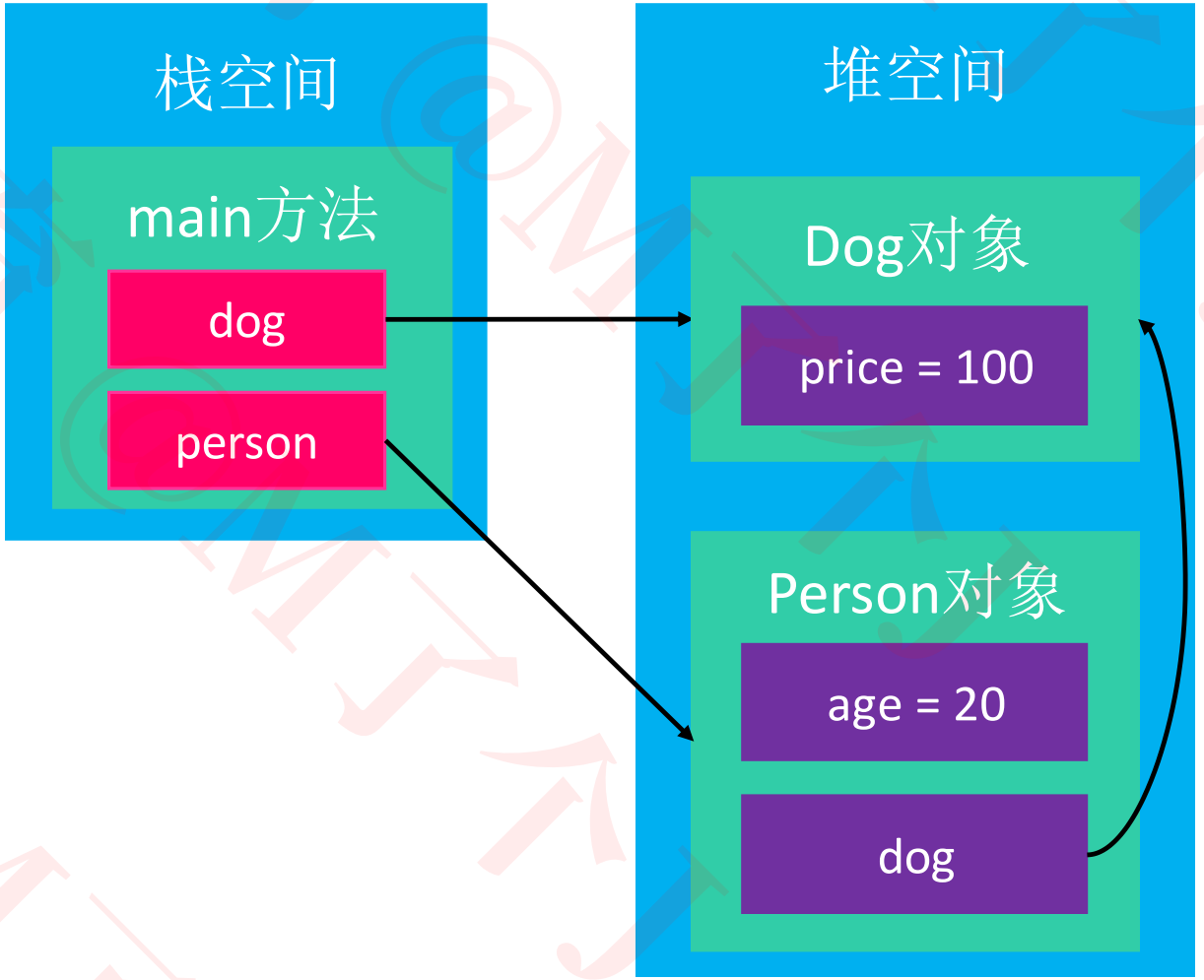


- Java 运行时环境有个垃圾回收器（garbage collector，简称GC），会自动回收不再使用的内存
  - 当一个对象没有任何引用指向时，会被GC回收掉内存

小码哥教育 SEEMYGO

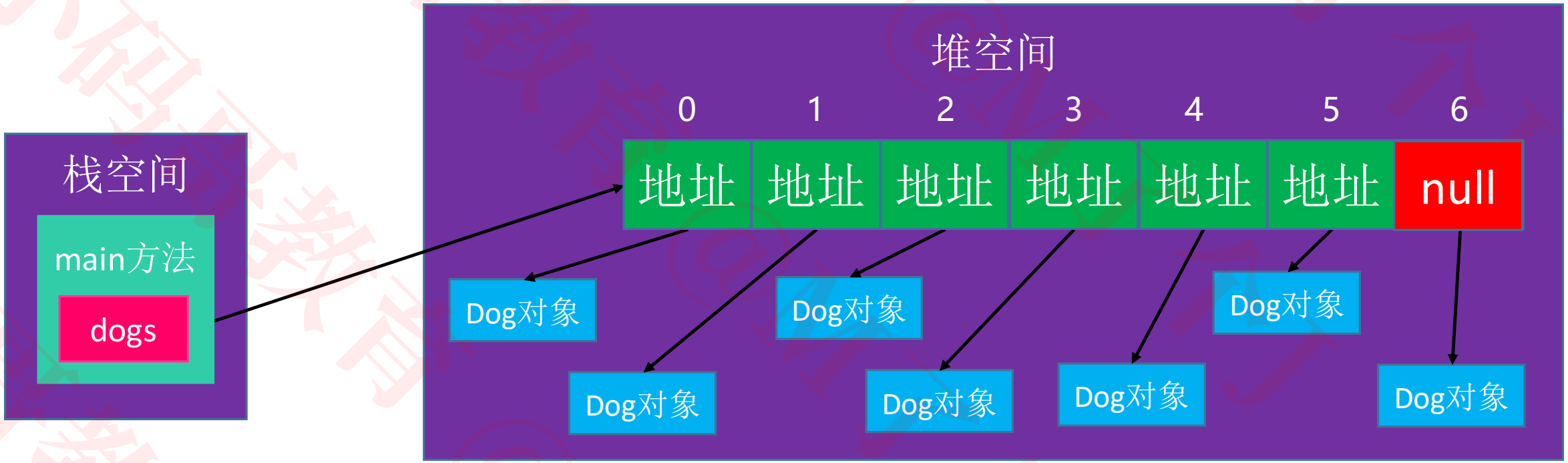
复杂对象的内存

```
public class Dog {
    public int price;
}

public class Person {
    public int age;
    public Dog dog;
}

public static void main(String[] args) {
    Dog dog = new Dog();
    dog.price = 100;

    Person person = new Person();
    person.age = 20;
    person.dog = dog;
}
```



对象数组的内存

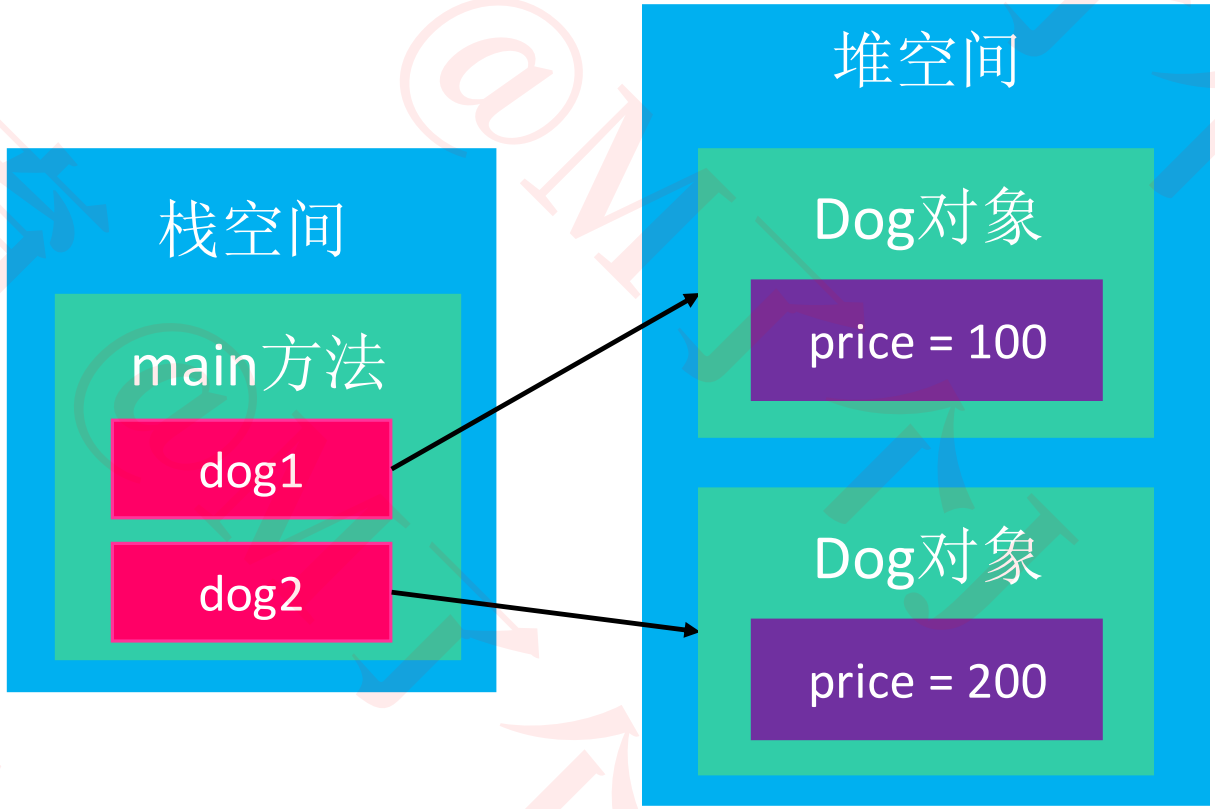
```
public static void main(String[] args) {
    Dog[] dogs = new Dog[7];
    for (int i = 0; i < dogs.length; i++) {
        dogs[i] = new Dog();
    }
    dogs[6] = null;
}
```



小码哥教育 SEEMYGO

思考：方法存储在哪里？

```
public class Dog {
    public int price;
    public void run() {
        System.out.println(price + "_run");
    }
    public void eat() {
        System.out.println(price + "_eat");
    }
}

public static void main(String[] args) {
    Dog dog1 = new Dog();
    dog1.price = 100;
    dog1.run();
    dog1.eat();

    Dog dog2 = new Dog();
    dog2.price = 200;
    dog2.run();
    dog2.eat();
}
```



Java程序的内存划分

- Java 虚拟机在执行 Java 程序时会将内存划分为若干个不同的数据区域，主要有
  - PC 寄存器（Program Counter Register）：存储 Java 虚拟机正在执行的字节码指令的地址
  - Java 虚拟机栈（Java Virtual Machine Stack）：存储栈帧
  - 堆（Heap）：存储 GC 所管理的各种对象
  - 方法区（Method Area）：存储每一个类的结构信息（比如字段和方法信息、构造方法和普通方法的字节码等）
  - 本地方法栈（Native Method Stack）：用来支持 native 方法的调用（比如用 C 语言编写的方法）

构造方法（Constructor）

- 构造方法，也叫构造器，能够更方便地创建一个对象
  - 方法名必须和类名一样
  - 没有返回值类型
  - 可以重载

```
public class Dog {
    public int age;
    public double weight;
    public Dog() {}
    public Dog(int age) {
        this.age = age;
    }
    public Dog(int age, double weight) {
        this.age = age;
        this.weight = weight;
    }
}

Dog dog1 = new Dog();
Dog dog2 = new Dog(18);
Dog dog3 = new Dog(20, 6.66);
```

- 建议每个 Java 类都提供无参的构造方法

this

- this 是一个指向当前对象的引用，常见用途是
  - 访问当前类中定义的成员变量
  - 调用当前类中定义的方法（包括构造方法）
- this 的本质是一个隐藏的、位置最靠前的方法参数
- 只能在构造方法中使用 this 调用其他构造方法
- 如果在构造方法中调用了其他构造方法
  - 构造方法调用语句必须是构造方法中的第一条语句

```
public class Dog {
    public String name;
    public int age;
    public int price;
    public Dog(String name, int age, int price) {
        this.name = name;
        this.age = age;
        this.price = price;
    }
    public Dog(String name) {
        this(name, 0, 0);
    }
}
```

小码哥教育 SEEMYGO

默认构造方法（Default Constructor）

- 如果一个类没有自定义构造方法，编译器会自动为它提供无参数的默认构造方法
- 一旦自定义了构造方法，默认构造方法就不再存在

```
public class Dog {
    public int age;
    public Dog(int age) {
        this.age = age;
    }
}

Dog dog1 = new Dog(10); // ok
Dog dog2 = new Dog(); // error
```

包（package）

- Java 中的包就是其他编程语言中的命名空间，包的本质是文件夹，常见作用是
  - 将不同的类进行组织管理、访问控制
  - 解决命名冲突

- 命名建议
  - 为保证包名的唯一性，一般包名都是以公司域名的倒写开头，比如 com.baidu.*
  - 全小写（以避免与某些类名或者接口名冲突）

- 类的第一句代码必须使用 package 声明自己属于哪个包
  - 比如 package com.mj.model;

```
package com.mj.model;

public class Dog {

}
```

包名的细节

- 如果公司域名有非法字符，建议添加下划线（_）来使包名合法化

域名	软件包名称前缀
my-name.example.org	org.example.my_name
example.int	int_.example
123name.example.com	com.example._123name

如何使用一个类

■ 要想正常使用一个类，必须得知道这个类的具体位置（在哪个包），有 3 种常见方式来使用一个类

① 使用类的全名

```
com.mj.model.Dog dog = new com.mj.model.Dog();
```

② 使用 import 导入指定的类名

```
import com.mj.model.Dog;

Dog dog = new Dog();
```

③ 使用 import 导入整个包的所有类

```
import com.mj.model.*;

Dog dog = new Dog();
```

导入的细节

- 为了方便，Java 编译器会为每个源文件自动导入 2 个包
  - `import java.lang.*;`
  - ✓ `java.lang` 包提供了 Java 开发中最常用的一些类型
  - `import 源文件所在包.*;`

- `import aa.bb.*;`
  - 仅仅是 `import` 了直接存放在 `aa.bb` 包中的类型
  - 并不包含 `import aa.bb.xx.*;`

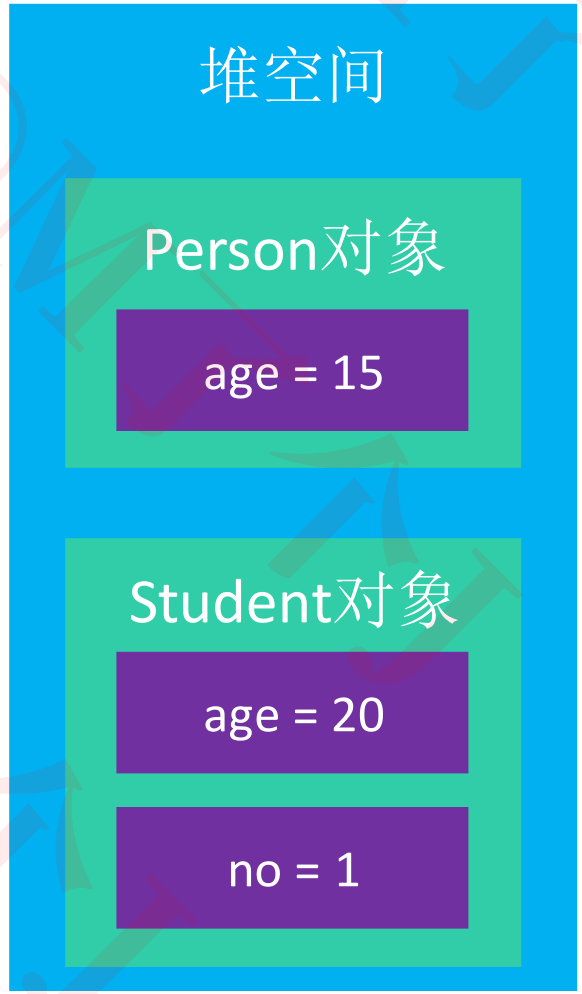
- Eclipse 中导包的快捷键：Ctrl + Shift + O，也可以使用 Ctrl + 1 修复错误来导包

继承（Inheritance）

```
public class Person {
    public int age;
    public void run() {
        System.out.println(age + "_run");
    }
}

public class Student extends Person {
    public int no;
    public void study() {
        System.out.println(age + "_" + no + "_study");
    }
}
```

```
Person person = new Person();
person.age = 15;
person.run();
```

```
Student student = new Student();
student.age = 20;
student.no = 1;
student.run();
student.study();
```



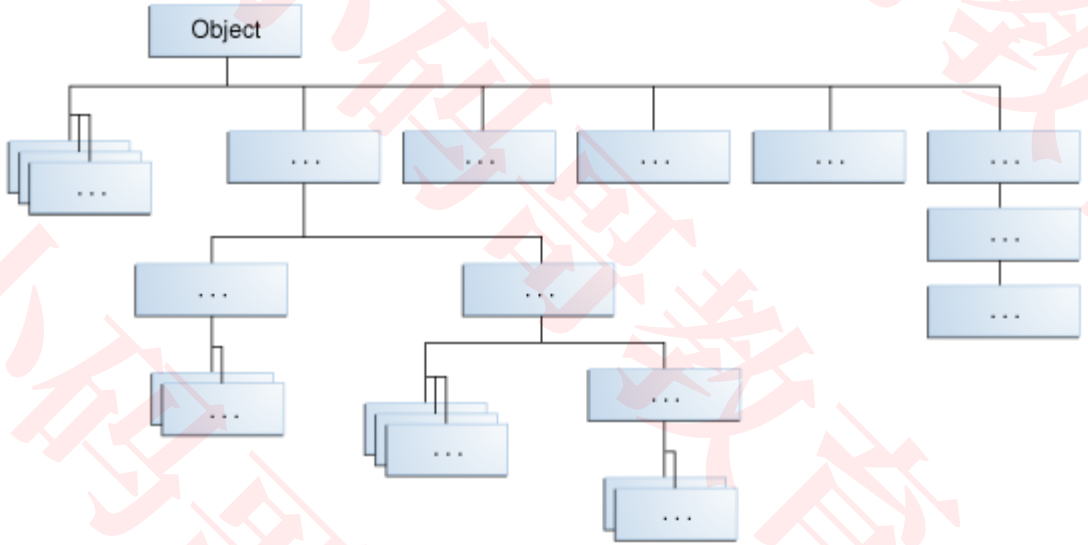
小码哥教育 SEEMYGO

思考

- 子类对象的内存中，是否包含父类中定义的 `private` 成员变量？
  - 依然包含

Object

- 任何类最终都继承自 java.lang.Object，一般称它为基类



小码哥教育 SEEMYGO

同名的成员变量

- 子类可以定义跟父类同名的成员变量（但不推荐这么做）

```
public class Person {
    public int age = 1;
}

public class Student extends Person {
    public int age = 2;
    public void show() {
        System.out.println(age); // 2
        System.out.println(this.age); // 2
        System.out.println(super.age); // 1
    }
}

new Student().show();
```

方法的重写（Override）

- 重写：子类的实例方法签名与父类一样。也叫做覆盖、覆写

```java
public class Animal {
    public void speak() {
        System.out.println("Animal - speak");
    }
    public void run() {
        System.out.println("Animal - run");
    }
}
```

```java
public class Dog extends Animal {
    @Override
    public void speak() {
        super.speak();
        run();
        this.run();
        super.run();
        System.out.println("Dog - speak");
    }
    @Override
    public void run() {
        System.out.println("Dog - run");
    }
}
```

```java
new Dog().speak();
/*
Animal - speak
Dog - run
Dog - run
Animal - run
Dog - speak
 */
```

重写的注意点

- 子类 override 的方法权限必须 ≥ 父类的方法权限

- 假设子类 override 的方法返回值类型是 A，父类的方法返回值类型是 B
  - 那么 A == B 或者 A 是 B 的子类型

- 子类的类方法签名和父类一样，不能称之为：重写

小码哥教育 SEEMYGO

super

- super 的常见用途是
  - 访问父类中定义的成员变量
  - 调用父类中定义的方法（包括构造方法）

```
public class Person {
    public int age;
    public Person(int age) {
        this.age = age;
    }
}

public class Student extends Person {
    public int no;
    public Student(int no) {
        super(0);
        this.no = no;
    }
}
```

构造方法的细节

- 子类的构造方法必须先调用父类的构造方法，再执行后面的代码

- 如果子类的构造方法没有显式调用父类的构造方法
  - 编译器会自动调用父类无参的构造方法（若此时父类没有无参的构造方法，编译器将报错）

注解（Annotation）

- 3 个常见的注解
  - `@Override`：告诉编译器这是一个重写后的方法
  - `@SuppressWarnings("警告类别")`：让编译器不生成警告信息
    - ✓ `@SuppressWarnings({ "rawtypes", "unused" })`
    - ✓ `@SuppressWarnings("unused")`
  - `@Deprecated`：表示这个内容已经过期，不推荐使用

访问控制（Access Control）

- Java 中有 4 个级别的访问权限，从高到低如下所示
  - public：在任何地方都是可见的
  - protected：仅在自己的包中、自己的子类中可见
  - 无修饰符（package-private）：仅在自己的包中可见
  - private：仅在自己的类中可见

修饰符	Class	Package	Subclass	World
public	✔	✔	✔	✔
protected	✔	✔	✔	✘
无修饰符	✔	✔	✘	✘
private	✔	✘	✘	✘

- 使用注意
  - 上述 4 个访问权限都可以修饰类的成员，比如成员变量、方法、嵌套类（Nested Class）等
  - 只有 public、无修饰符（package-private）可以修饰顶级类（Top-level Class）
  - 上述 4 个访问权限不可以修饰局部类（Local Class）、局部变量
  - 一个 Java 源文件中可以定义多个顶级类，public 顶级类的名字必须和文件名一样

小码哥教育 SEEMYGO

封装

```
public class Person {
    private int age;
    private String name;
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}
```

- 成员变量 private 化，提供 public 的getter、setter

toString方法

- 当打印一个对象时，会自动调用对象的 toString 方法，并将返回的字符串打印出来

- toString 方法来源于基类 java.lang.Object，默认实现如下所示

```
public String toString() {
    return getClass().getName() + "@" + Integer.toHexString(hashCode());
}
```

- Eclipse 中有一个可以自动生成 getter、setter、constructor、toString 等常用代码的快捷键
  - Shift + Alt + S