LightHouseIT

2.1 Java 面向对象编程基础

- 面向对象编程基本概念
- Java 类和对象
- 静态变量和方法

LightHouseIT

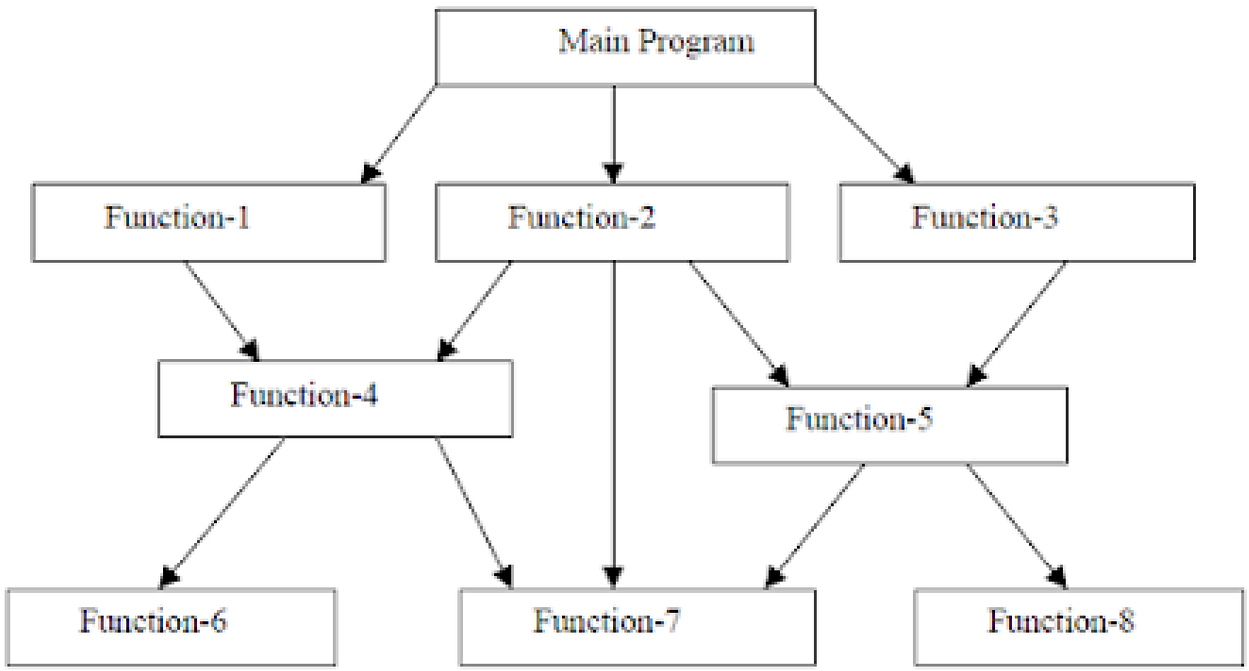
面向过程编程

- 我们到目前为止学过的都是面向过程[手続き型]编程
  （Procedure Oriented Programming）

- 定义各种方法：
method1
method2
method3
…

- 数据：
int
float
char
String
…

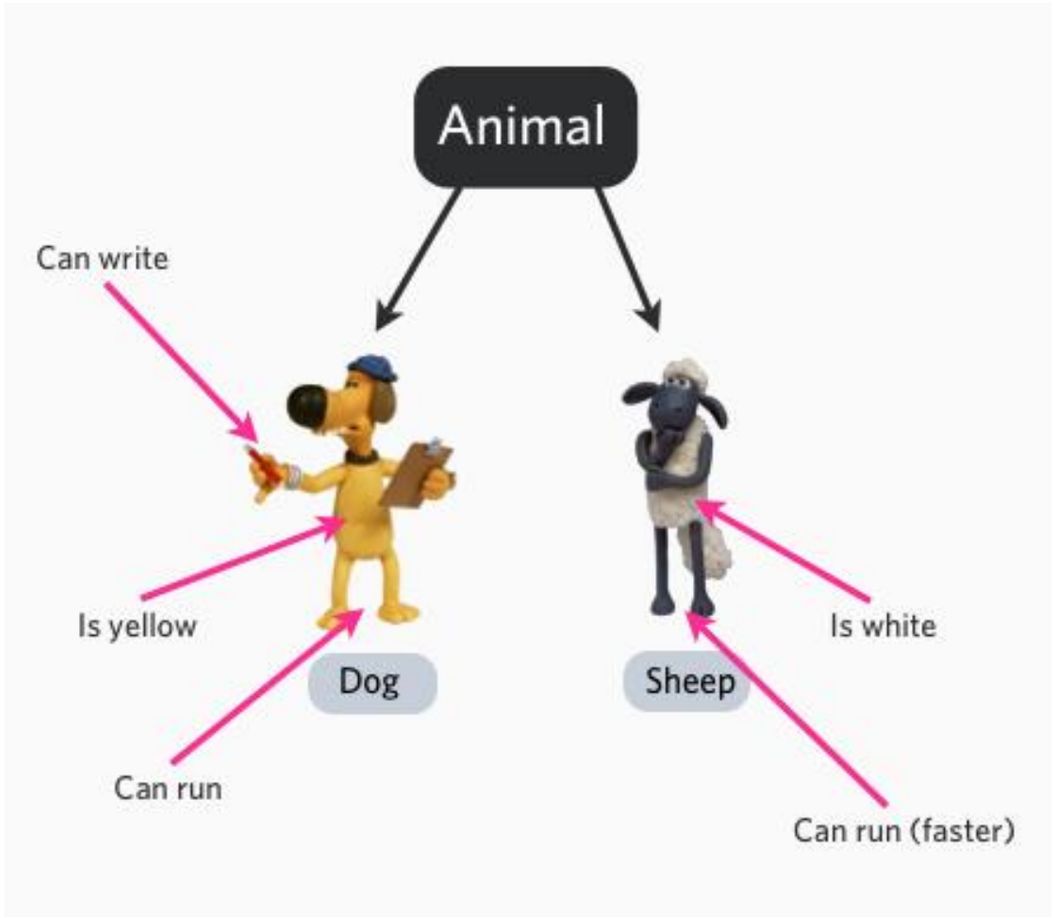
- 以及方法、数据间的操作顺序。

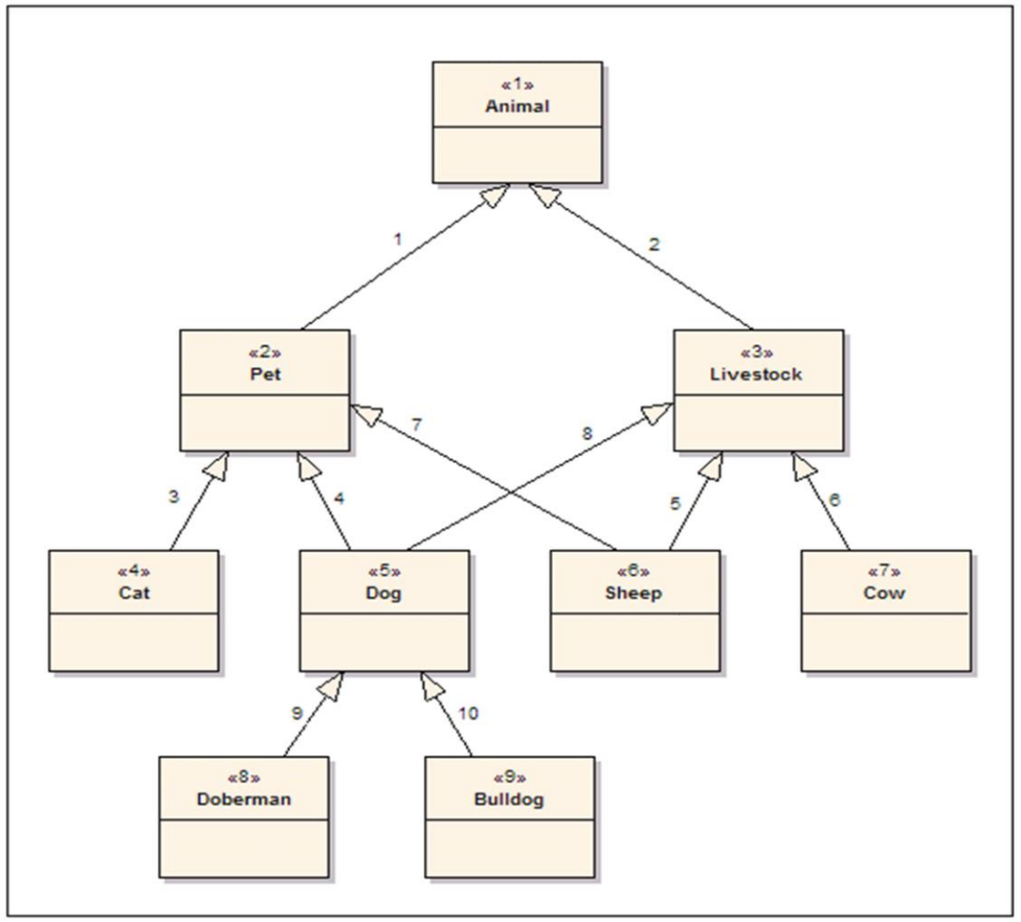


Structure of procedural oriented programs

面向对象编程

- 今天要学习另一种编程思想：**面向对象**[オブジェクト指向]编程（Object Oriented Programming）。





面向过程 vs 面向对象

- 在面向方法编程时，我们考虑计算机解决问题的所有步骤，并将它们一一列举，写成程序。

- 在面向对象编程时，我们先考虑系统中有哪些事物，它们分别具有哪些**属性**和**功能**，再具体编写这些功能的完成方法，并使用这些事物来解决问题。

- 在解决复杂现实系统中的问题的时候，使用面向对象编程不仅能让我们代码编写更简单、更符合人的思维方式，更能提升代码本身的可读性。

面向过程 vs 面向对象

Example

问题：输出一个班级所有同学的姓名和学号。

面向过程编程：

1. 创建两个个数组，分别存入所有同学的姓名和学号。
2. 让 i 从 0 变化到人数 – 1，输出每一个数组的第 i 个元素。

面向对象编程：

1. 每一个学生具有姓名、学号两个属性，并有一个输出自己信息的方法。
2. 创建一个学生的数组，存入所有学生的数据。
3. 遍历这个数组并让所有学生输出自己的信息。

对象的定义

- 在面向对象编程时，我们代码处理的目标从具体的数据变成了**对象**[オブジェクト]。对象既可以是一个现实中存在的物品或生物，也可以是一个理论上的概念。

Example

一辆汽车可以是一个对象。
一个人可以是一个对象。
一个正方形可以是一个对象。
一个班级可以是一个对象。

外部和内部对象

- 对象可以指两种概念:
  - *外部对象*: 系统所涉及的现实物理世界中的物品、概念。
  - *内部对象*: 用编程语言实现的计算机上的表现。

- 外部对象 → 内部对象转变流程:
  1. 分析阶段: 认识到问题所涉及的外部对象。
  2. 设计阶段: 变换成内部对象的表现。
  3. 实现阶段: 用程序表达内部对象。

Example

比如我们需要做一个动物园的系统:

1. 认识到系统需要使用动物对象。
2. 每一个动物需要有姓名，这是一个字符串；年龄，这是一个数字；以此类推，设计其他属性和行为。
3. 实际使用 Java 代码实现以上设计。

设计对象的例子

Example

在动物园系统中，我们要使用到很多猫的对象。
想一想，一只猫需要具有哪些特性？

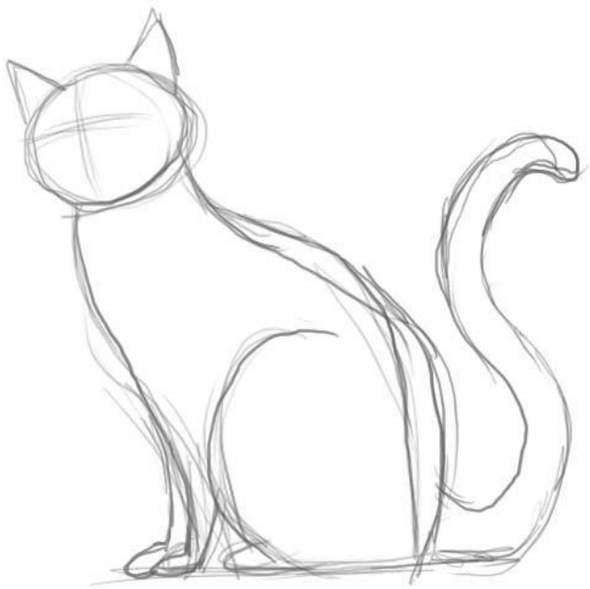
属性和行为

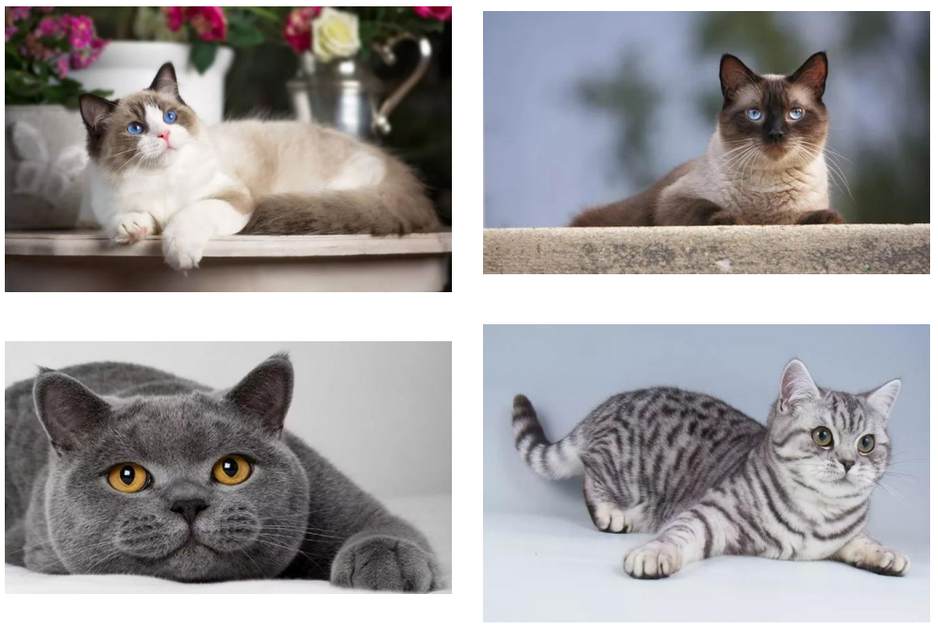
- 属性：猫所具有的性质或状态。
  - 性质：猫毛长度，皮毛颜色，名字……
  - 状态：吃饭了没，起床了没，抓到老鼠了没……
- 行为：猫所能执行的动作或功能
  - 动作：吃饭，发出叫声，抓老鼠……
  - 功能：知道它的名字，知道它吃饭了没……

类

- 我们可以注意到，这些猫的对象都具有类似的特性：属性的类型总是相同的；方法也是相同的。不同的只是属性的具体值。

- 我们可以把这些共通的性质总结成猫**类**[クラス]。一个类就相当于一个模板、蓝图。当我们要创建一只猫时，可以根据猫类直接确定它有哪些属性和行为，我们只需要把属性设定成相应数值即可。





类和对象的例子

Example

猫的类:
猫具有名字，年龄，颜色等属性。
猫具有吃东西，抓老鼠等方法。

猫的对象1：名字是 Alice，6 岁，白色。

猫的对象2：名字是 Bob，8 岁，蓝色。

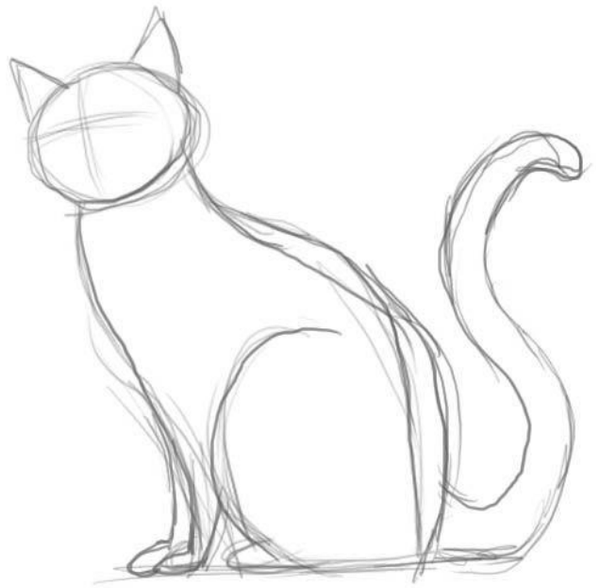
.....

接次页

接前页

猫的类

定义猫所拥有的属性和行为



猫的对象1

猫的对象2

猫的对象3

猫的对象4

Question and answer

LightHouseIT

类的创建

- 要创建一个类，使用关键字 **class**：

```
1 public class Cat {
2     // 类的定义
3 }
```

- 类的定义包括类的属性和行为。在 Java 中，类的属性被称作**成员变量**[フィールド]，而行为则被称为**方法**[メソッド]。
- 一个 Java 文件中可以创建多个类，但请不要这样做。

成员变量的定义

- **成员变量**的定义与局部变量类似:

```
1 public class Cat {
2     String name;
3     int age;
4 }
```

- 你同样可以在定义时初始化变量:

```
1 public class Cat {
2     String name = "Unkown";
3     int age = 0;
4 }
```

方法的定义

- 我们之前已经了解过**方法**，但一般的类的方法定义和之前略有不同：

```
1 class Cat {
2     void meow() {
3         System.out.println("meow~");
4     }
5 }
```

- 实际上，方法定义时只需要在名称前写上返回类型，之前使用的 static 关键字我们会在稍后说明。

对象的创建

- 我们已经创建了名为 Cat 的类，现在可以使用它来创建对象了。
- 要创建对象，使用关键字 **new**：

```
new Cat();
```

- 你可以将创建好的对象存入变量内。这个变量需要存储一个 Cat 的对象，因此它的类型就是 “Cat”：

```
Cat alice = new Cat();
```

对象的使用

- 在创建好对象后，就可以使用对象的成员变量和方法。
- 要使用对象的成员变量和方法，使用“.”运算符：

```
1 Cat alice = new Cat();
2 System.out.println(alice.name); // 使用成员变量的值
3 alice.age = 5; // 改变成员变量的值
4 alice.eat("catfood"); // 使用方法
```

使用本类的成员变量和方法

- 一个类的方法可以直接使用本类的成员变量和方法：

```
1 class Cat {
2     // ...
3
4     void eat(String food) {
5         System.out.println(name + " eat " + food);
6         meow(); // 发出猫叫
7     }
8 }
```

- 实际运行的过程中，被使用的将是**当前**对象的成员变量和方法：

```
1 Cat alice = new Cat();
2 alice.name = "Alice";
3 alice.eat("cat food"); // => Alice eat cat food, meow~
```

Note

成员变量和方法可以在定义的代码之前使用。

构造方法

- **构造方法**[コンストラクター]（Constructor）是在对象被创建时会被自动调用的方法。如果你有一些固定的初始化操作需要在每一个对象被创建时被执行，那就可以考虑将它写入构造方法中。

- 要定义构造方法，定义一个名称与类名相同的方法，并不写返回类型：

```
class Cat {
    // ...

    Cat() {
        name = "Default Name";
        age = 1;
    }
}
```

Note

构造方法中的代码将会在成员定义时的初始化**之后**执行。

构造方法的参数

- 构造方法可以有任意个参数。我们可以让构造方法接受一些参数以设置属性的初始值：

```
1 Cat(String name_, int age_) {
2     name = name_;
3     age = age_;
4 }
```

- 在创建对象时，我们可以输入对应的参数：

```
1 Cat alice = new Cat("Alice", 5);
2 System.out.println(alice.name); // => Alice
3 System.out.println(alice.age); // => 5
```

- 你也可以重载构造器，使它接受不同类型的参数。

默认构造器

- 如果没有定义任何构造器，Java 会自动定义一个默认的无参构造器，不做任何事。
- 如果手动定义了**任何**构造器，默认构造器将不会存在!

```
1 Cat alice = new Cat("Alice", 5); // Cat 类定义了一个有参数的构造器
2 Cat bob = new Cat(); // 语法错误：Cat 没有无参构造器
```

Try 01011 11010 01011

Cat.java

Question and answer

目录

LightHouseIT

静态变量和方法

- 还记得之前定义方法时的 static 关键字吗？“static” 代表的是 “**静态**[静的]”。Java 中，变量和方法都可以是静态的。静态指变量或方法属于某个类，而不是具体的对象。

- 使用 static 关键字可以定义静态变量和方法：

```
1 class A {
2     static int staticVariable = 10;
3     static int staticMethod(int a) {
4         return a + staticVariable;
5     }
6 }
```

接次页

接前页

- 要使用静态变量或方法，使用 “.” 运算符链接**类名**和变量 / 方法名：

```
1 System.out.println(A.staticVariable); // => 10
2 A.staticVariable = 5;
3 System.out.println(A.staticMethod(10)); // => 15
```

接次页

接前页

- 同类中的方法也可以直接使用该类中的静态方法和变量:

```
1 class A {
2     static int staticVariable = 10;
3     static int staticMethod(int a) {
4         return a + staticVariable;
5     }
6
7     void method() {
8         System.out.println(staticVariable);
9         staticVariable = staticMethod(10);
10    }
11 }
```

接次页

接前页

- 在该类的任意对象后加上“.”，同样可以调用该类的静态变量或方法，但是不推荐这样做：

```
1 A a = new A();
2 a.staticVariable = 5;
3 System.out.println(a.staticMethod(10)); // => 15
```

静态和非静态

- 由于静态方法不属于某个具体的对象，在**静态**方法中不能直接调用**非静态**的方法或变量（成员变量）：

```
1 class A {
2     void method1() { }
3     int a = 0;
4
5     static void method2() {
6         method1(); // => 报错
7         System.out.println(a); // => 报错
8     }
9 }
```

Try
Student.java

- 现在你明白我们之前定义方法时为什么要写 static 了吗？

静态和非静态：总结

Sum Up

我们可以将与静态方法相关的调用表现归纳如下：

代码所在方法	调用非静态变量或方法	调用静态变量或方法
非静态	可调用	可调用
静态	不可调用	可调用

Question and answer

总结

Sum Up

1. 面向对象的基本概念。

2. Java 中的类和对象：
   ① 定义类、成员变量和方法；
   ② 定义构造方法和创建对象；
   ③ 使用成员变量和方法；
   ④ 定义和使用静态变量和静态方法。

LightHouseIT

THANK YOU!