编写类

Java Reference

Fields, constructor & methods

类 (class) 是对象的蓝图。它的字段 (field) 存储数据, 构造方法 (constructor) 初始化一个新对象, 方法 (method) 是它能做的动作。`this.name` 表示" 这个对象的 `name`"。用 `new` 创建对象。

```
public class Main {
    public static void main(String[] args) {
        Dog d = new Dog("Rex", 3);
        System.out.println(d.describe());   // Rex is 3 years old
        d.haveBirthday();
        System.out.println(d.describe());   // Rex is 4 years old
    }
}

class Dog {
    private String name;
    private int age;

    public Dog(String name, int age) {   // constructor
        this.name = name;
        this.age = age;
    }

    public String describe() {
        return name + " is " + age + " years old";
    }

    public void haveBirthday() {
        age++;
    }
}
```

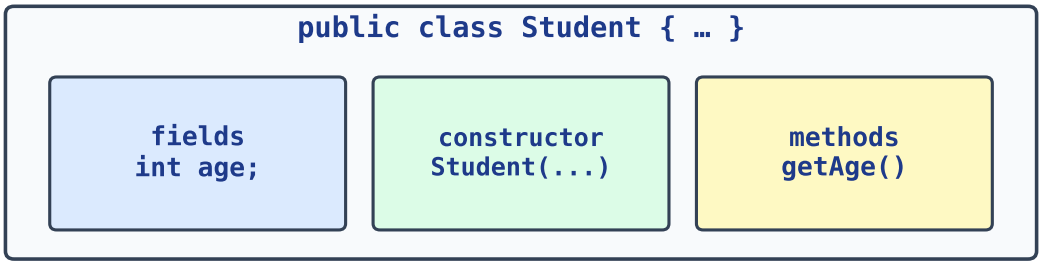
`static` 成员属于类本身，而不属于某个对象。静态方法用类名来调用 — 就像 `Math.max`；静态字段由所有对象共享：

```
public class Main {
    public static void main(String[] args) {
        System.out.println(Counter.made());   // 0
        new Counter();
        new Counter();
        System.out.println(Counter.made());   // 2
    }
}

class Counter {
    private static int count = 0;      // shared by ALL Counter objects

    public Counter() { count++; }

    public static int made() {         // called on the class:
↪   Counter.made()
        return count;
    }
}
```



Exam tip: constructor name matches the class; fields store state; methods do work

Fields store state; constructor builds; methods act

Encapsulation

封装控制对对象状态的访问。将字段标记为 `private` 以阻止无关类的直接访问；访问器读取状态，修改器更改状态。setter 是 mutator 的一种。`private` 不验证值：构造方法和修改方法需要自己的检查或明确的先决条件。本例假设非负 `start` 和在 `int` 范围内的加法；其 deposit 方法拒绝非正金额。

```
public class Main {
    public static void main(String[] args) {
        Account a = new Account(100);
        a.deposit(50);
        a.deposit(-999);                          // rejected by the guard
        System.out.println(a.getBalance());       // 150
    }
}

class Account {
    private int balance;                          // hidden from outside

    public Account(int start) {
        balance = start;
    }

    public void deposit(int amount) {
        if (amount > 0) balance += amount;        // this guard checks the
↪   deposit amount
    }

    public int getBalance() {                     // accessor (getter)
        return balance;
    }
}
```

常见错误

- 构造方法与类同名, 而且没有返回类型 (连 `void` 都没有)。
- 用 `this.field` 区分字段和同名的参数。
- 把字段设为 `private`, 通过 getter/setter 方法访问 (封装)。