1. 一定要保证数据私有

绝对不要破坏封装性。数据的表示形式很可能会改变，但它们的使用方法却不会经常变化。当数据保持私有时，表示形式的变化不会对类的使用者产生影响，而且也更容易检测 bug。

2. 一定要对数据进行初始化

最好不要依赖于系统的默认值，而是应该显式地初始化所有的数据。

3. 不要在类中使用过多的基本类型

将多个相关的基本类型变量替换为其他的类，这样会使类更易于理解，也更易于修改。例如：

```
// 可以用 Address 类替换这四个实例字段，这样可以很容易地处理地址的变化
private String street;
private String city;
private String state;
private int zip;
```

4. 不是所有的字段都需要单独的字段访问器和字段更改器

在对象中，常常包含一些不希望别人获得或设置的实例字段，这时就不需要字段访问器和字段更改器。有的实例字段被设置后就不会再修改，这样的实例字段就不需要字段更改器。

5. 分解有过多职责的类

如果明显地可以将一个复杂的类分解成两个更为简单的类，就应该将其分解。但也不要走极端，如果设计 10 个类，每个类只有一个方法，就有些矫枉过正了。

6. 类名和方法名要能够体现它们的职责

类名的惯例：类名应该是一个名词（如`Order`），或者是前面有形容词修饰的名词（如`RushOrder`），或者是有动名词修饰的名词（如`BillingAddress`）。

方法名的标准惯例：访问器方法用小写 get 开头，更改器方法用小写 set 开头。

7. 优先使用不可变的类

不可变的类指没有方法能够修改对象的状态，例如`java.time`包中的类都是不可变的。

可变对象的问题在于，如果多个线程试图同时更新一个对象，就会发生并发更改，其结果是不可预料的。如果类是不可变的，就可以安全地在多个线程间共享其对象。

设计类时，也要尽可能让类是不可变的。当然，并不是所有类都应当是不可变的，要视情况而定。