6.1 抽象

在第一周就有一个Shape类的例子。这个类有很多的子类，每个子类也都实现了父类的方法。实际上父类Shape只是一个抽象的概念而并没有实际的意义。如果请你画一个圆，你知道该怎么画；如果请你画一个矩形，你也知道该怎么画。但是如果我说：“请画一个形状，句号”。你该怎么画？同样，我们可以定义Circle类和Rectangle类的draw()，但是Shape类的draw()呢？

Shape类表达的是一种概念，一种共同属性的抽象集合，我们并不希望任何Shape类的对象会被创建出来。那么，我们就应该把这个Shape类定义为抽象的。我们用abstract关键字来定义抽象类。抽象类的作用仅仅是表达接口，而不是具体的实现细节。抽象类中可以存在抽象方法。抽象方法也是使用abstract关键字来修饰。抽象的方法是不完全的，它只是一个方法签名而完全没有方法体。

如果一个类有了一个抽象的方法，这个类就必须声明为抽象类。如果父类是抽象类，那么子类必须覆盖所有在父类中的抽象方法（即必须实现抽象类中的抽象方法函数），否则子类也成为一个抽象类。一个抽象类可以没有任何抽象方法，所有的方法都有方法体，但是整个类是抽象的。设计这样的抽象类主要是为了防止制造它的对象出来。

6.2 数据表现与分离---细胞自动机

表现和数据分离

程序的业务逻辑和表现无关，

表现可以是图形的也可以是文本的，表现可以是当地的也可以是远程的。

View只管根据field画图形，field只管数据的存放，一旦数据更新以后，通知view重新画出整个画面：

不去精心设计哪个局部需要更新

这样简化了程序逻辑

是在计算机运算速度提高的基础上实现的

责任驱动的设计：

将程序要实现的功能分配到合适的类/对象中去是设计中非常重要的一环。

网格化：

图形界面本身有更高的解析度

将画面网格化以后，数据就更容易处理了。

6.3 接口

接口interface，在java中和class地位一样

接口是纯抽象类

所有成员函数都是抽象函数

所有成员变量都是public static final

接口规定了长什么样，但是不管里面有什么

继承类用extends，接口用implements；

类可以实现很多接口

接口可以继承接口，但不能继承类

接口不能实现接口

面向接口的编程方式：

设计程序时先定义接口，再实现类

任何需要再函数间传入传出的一定是接口而不是具体的类

是Java成功的关键之一，因为极适合多人同时写一个大程序

也是Java被批评的点之一，因为代码膨胀起来很快