9 面向对象编程I：思想

一. 结构化编程的问题.

1. 不易读.
   - 全局变量
2. 不易维护.
   - 细节更改
   - 需求更改

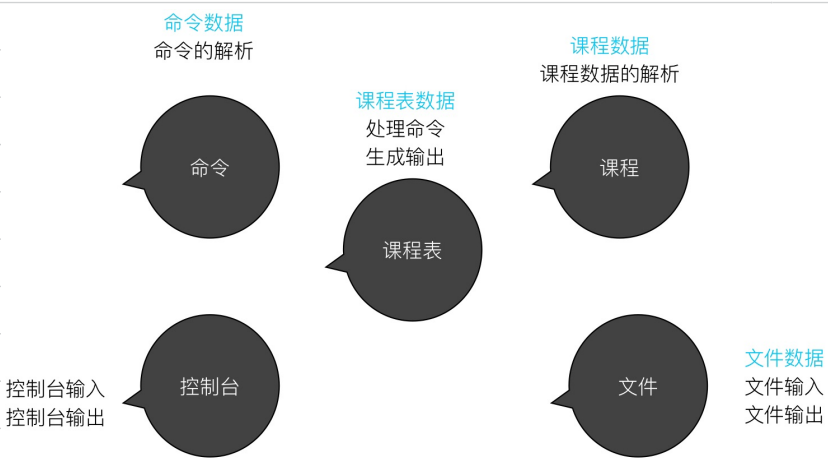
o 解决方案：

(1) 在有限的范围内修改：数据和操作在一起（封装）

(2) 扩展：运行时动态链接（多态）

二. 面向对象思想.

1. 实体职责：数据职责 + 行为职责



2. 类：职责的抽象 （抽象）

   对象：职责的实现 （具体）

o 视角的转变：函数之间的调用 → 有职责的对象之间的交互与协作

行为视角：结构化方法.

数据视角：数据为中心方法.

职责视角：面向对象方法.

三. 类和对象

对象是面向对象中的术语，既表示客观世界问题空间中的某个具体事物，又表示软件系统解空间中的基本元素。在软件系统中，对象具有唯一的标识符，对象包括属性和方法。属性就是需要记忆的信息，方法就是对象能够提供的服务。

- 每个对象都保存着描述当前特征的信息。
- 对象状态的改变必须通过调用方法实现。
- 对象的状态不能完全描述一个对象。每个对象都有一个唯一的身份(identity)。
- 每个对象的标识永远是不同的，状态常常也存在着差异。

获得对象：

(1)寻找候选对象

①找名词：类(对象)与属性。

②找动词：行为。

(2)精化对象：

①去除：冗余、不相干、模糊的概念。

②转化：没有行为的对象→某个类的属性。

类这个术语被用来描述相同事物的集合，它以概要的方式描述了相同事物集合中的所有元素，但却允许类中的每个实体元素可以在非本质特征上变化。面向对象程序设计语言使用类来描述对象，并且通过类方法来定义它们的行为。

- 类（Class）这个术语是对具有共同具体属性的对象的描述。
- 类是一个描述或蓝图（被表示成一段代码），用于定义组成某类特定对象的所有特性。
- 编程中使用类的思想与现实世界中把东西进行分类的思想相一致，这是一种方便而明确的事物组织方式。
- 一旦定义了一个类，就可以接着得到这个类的对象或实例。
- 实例变量的值由类的每个实例提供。
- 当创建类的实例时， 就建立了这种类型的一个对象，然后系统为类定义的实例变量分配内存。

类与对象的联系：

- 类是对某个对象的定义。
- 它包含有关对象动作方式的信息，包括它的名称、方法、属性和事件。
- 当引用类的代码运行时，类的一个新的实例，即对象，就在内存中创建了。虽然只有一个类，但能从这个类在内存中创建多个相同类型的对象。

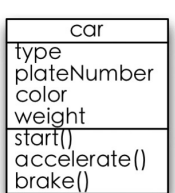
类与对象的职责：

- 所谓职责，我们可以理解它为功能。
- 每个类应当只有单一职责。
  - 当你发现有两个变化会要求我们修改这个类，那么你就要考虑撤分这个类了。
- 给对象分配责任的策略：
  - 覆盖到所有重要的方面
  - 寻找需要执行的动作以及需要维护和生成的信息

创建类的原因：

- 对现实世界中的对象建模
- 对抽象对象建模
- 降低复杂度
- 隔离复杂度
- 隐藏实现细节
- 限制变化所影响的范围
- 创建中心控制点

类图：



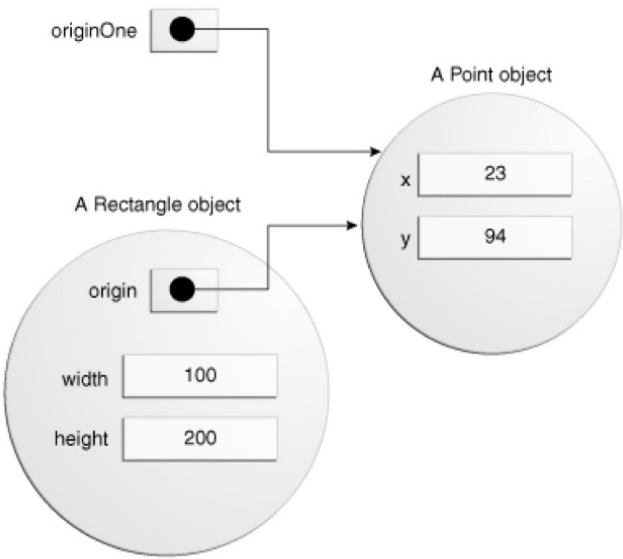
类的定义：

```
public class Rectangle {
    public int width = 0;
    public int height = 0;
    public Point origin;

    // four constructors
    public Rectangle() {
        origin = new Point(0, 0);
    }
    public Rectangle(Point p) {
        origin = p;
    }
    public Rectangle(int w, int h) {
        origin = new Point(0, 0);
        width = w;
        height = h;
    }
    public Rectangle(Point p, int w, int h) {
        origin = p;
        width = w;
        height = h;
    }

    // a method for moving the rectangle
    public void move(int x, int y) {
        origin.x = x;
        origin.y = y;
    }

    // a method for computing the area
    // of the rectangle
    public int getArea() {
        return width * height;
    }
}
```



变量和方法的访问：

- 引用变量.变量名
  - 1 . int height = new Rectangle().height;
  - 2. Rectangle rect = new Rectangle();
  - int height = rect.height;
- 引用变量.方法名（）；
  - System.out.println("Area of rectOne: " + rectOne.getArea());
  - ...
  - rectTwo.move(40, 72);