09-面向对象编程 I
思想

刘 钦
南京大学软件学院

Outline

- 结构化编程的问题
- 面向对象思想
- 类和对象
- 案例分析

Problems of Structured Programming

- Not easy to read
- Not easy to maintain

Not Easy to Read -- 全局变量

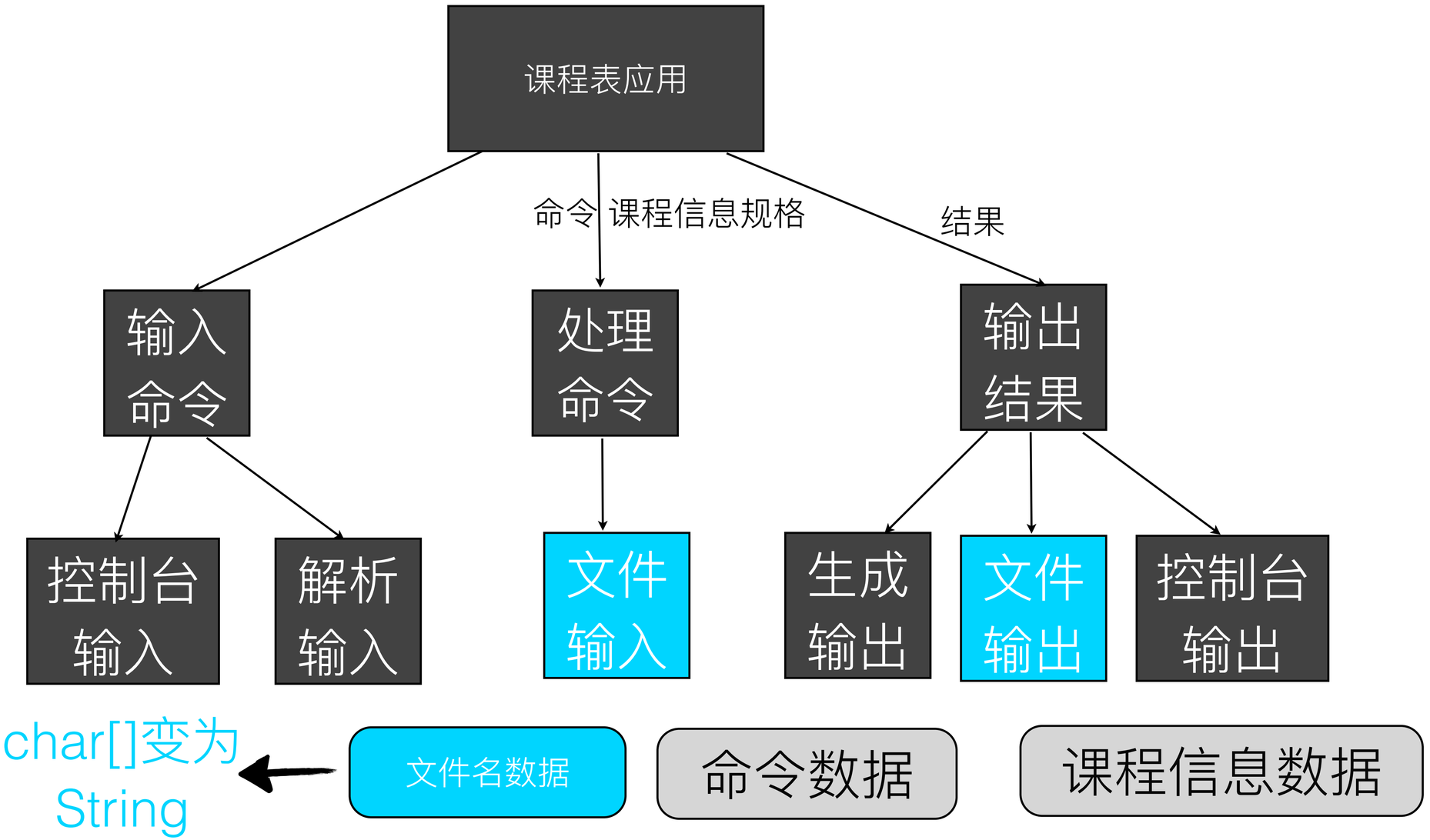
```
public class CourseSchedule {

    static public String fileName = "CurriculumSchedule";

    public static void main(String[] args){
        String input = "",output = "";
        String command;
        String courseInfo;
        int cmd = -1;
```

第11行

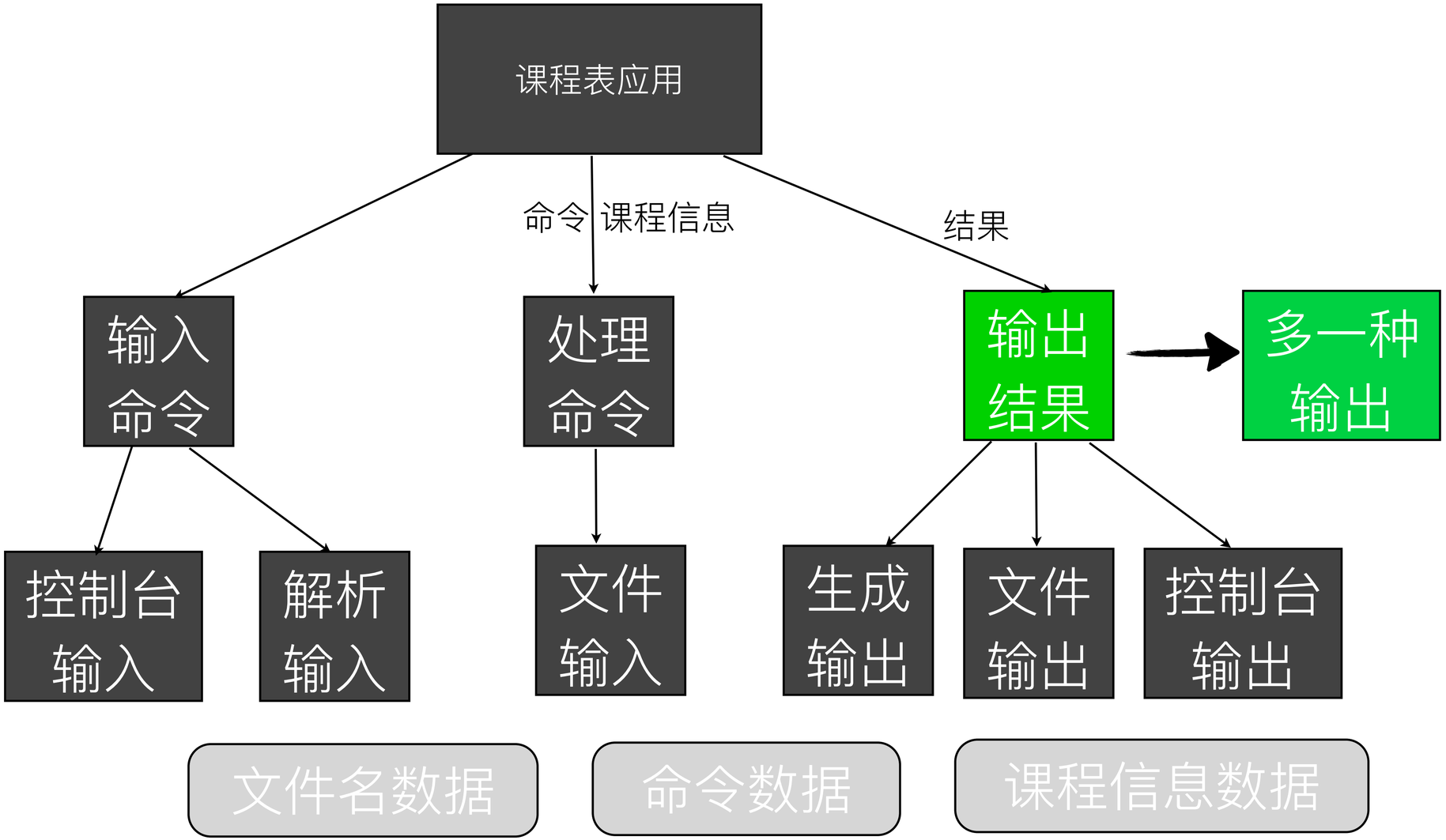
```
//      System.out.println("courseInfo:"+day+time+name+location);
        try{

            BufferedReader br1=new BufferedReader(new FileReader(fileName));
            String line;
            while((line=br1.readLine())!=null){
                String day2;
                String time2;
                String name2;
                String location2;
```

第137行

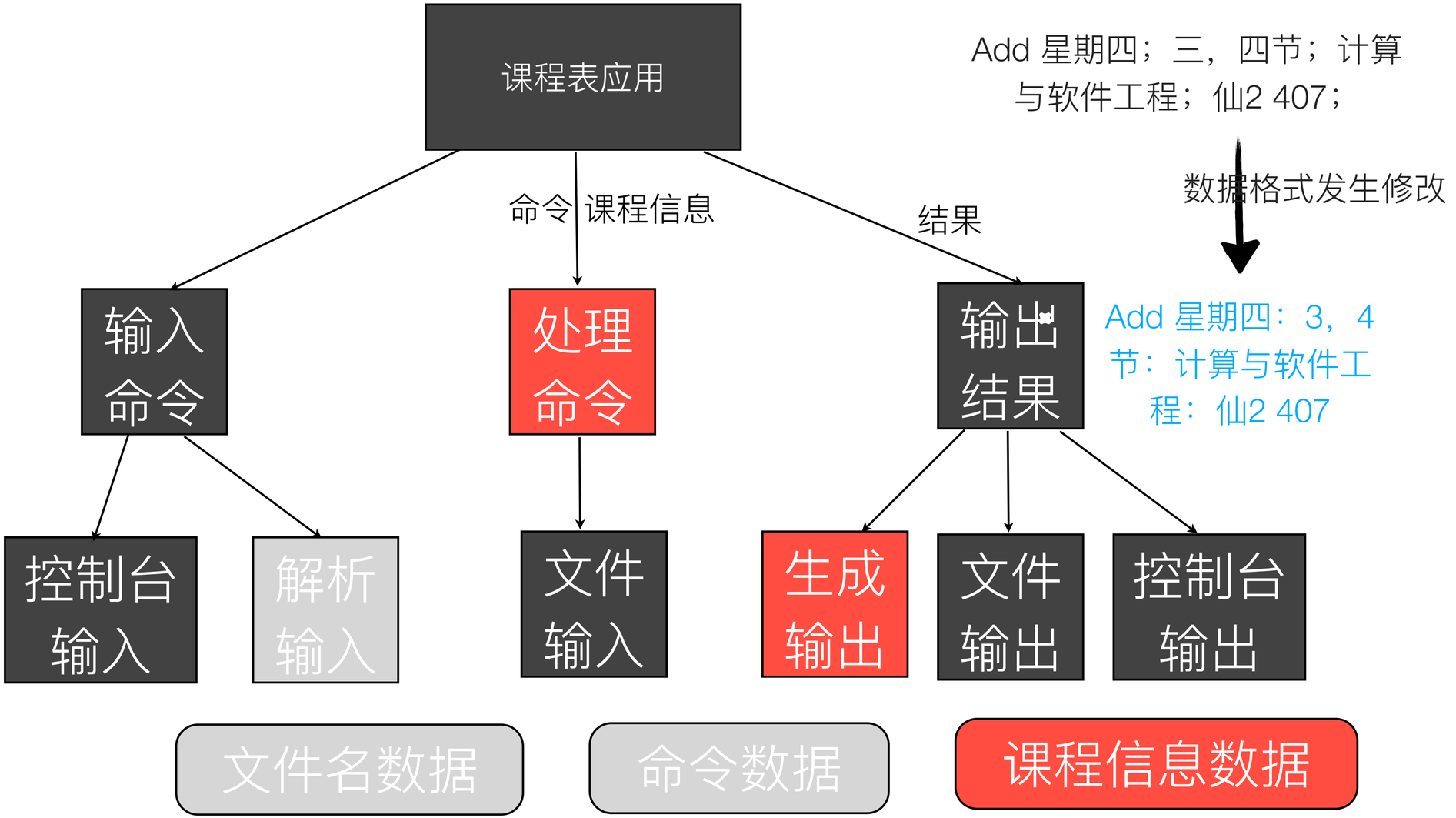
Not Easy to Maintain -- 实现变更



Not Easy to Maintain -- 需求增加



Not Easy to Maintain -- 需求更改



Add 星期四；三，四节；计算与软件工程；仙2 407；

数据格式发生修改

Add 星期四：3，4节：计算与软件工程：仙2 407

大范围的修改
Nightmare!

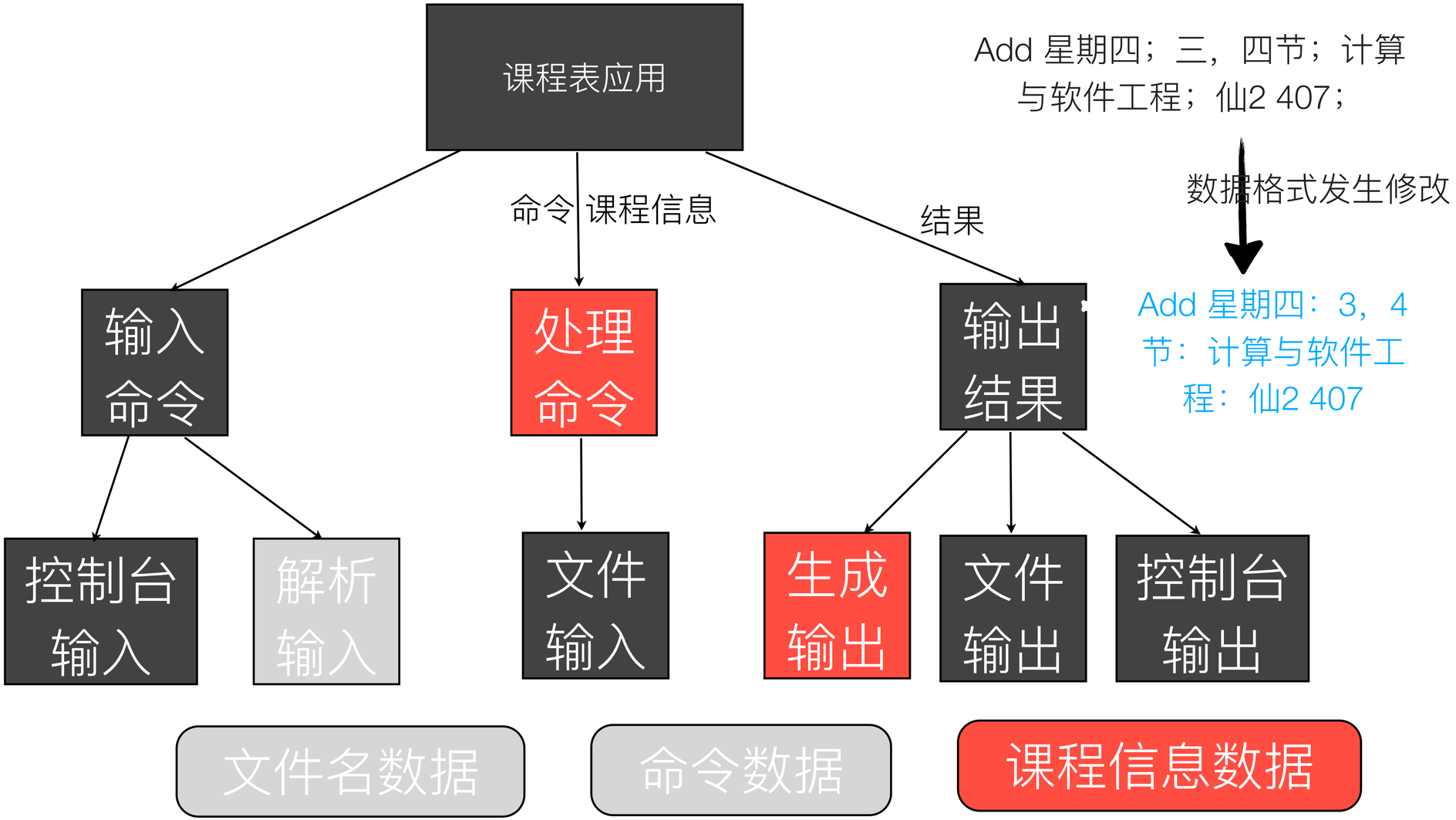
问题

Q: 不修改，怎么应对变更?

A: 1. 在有限的范围内修改

2. 扩展

大范围的修改

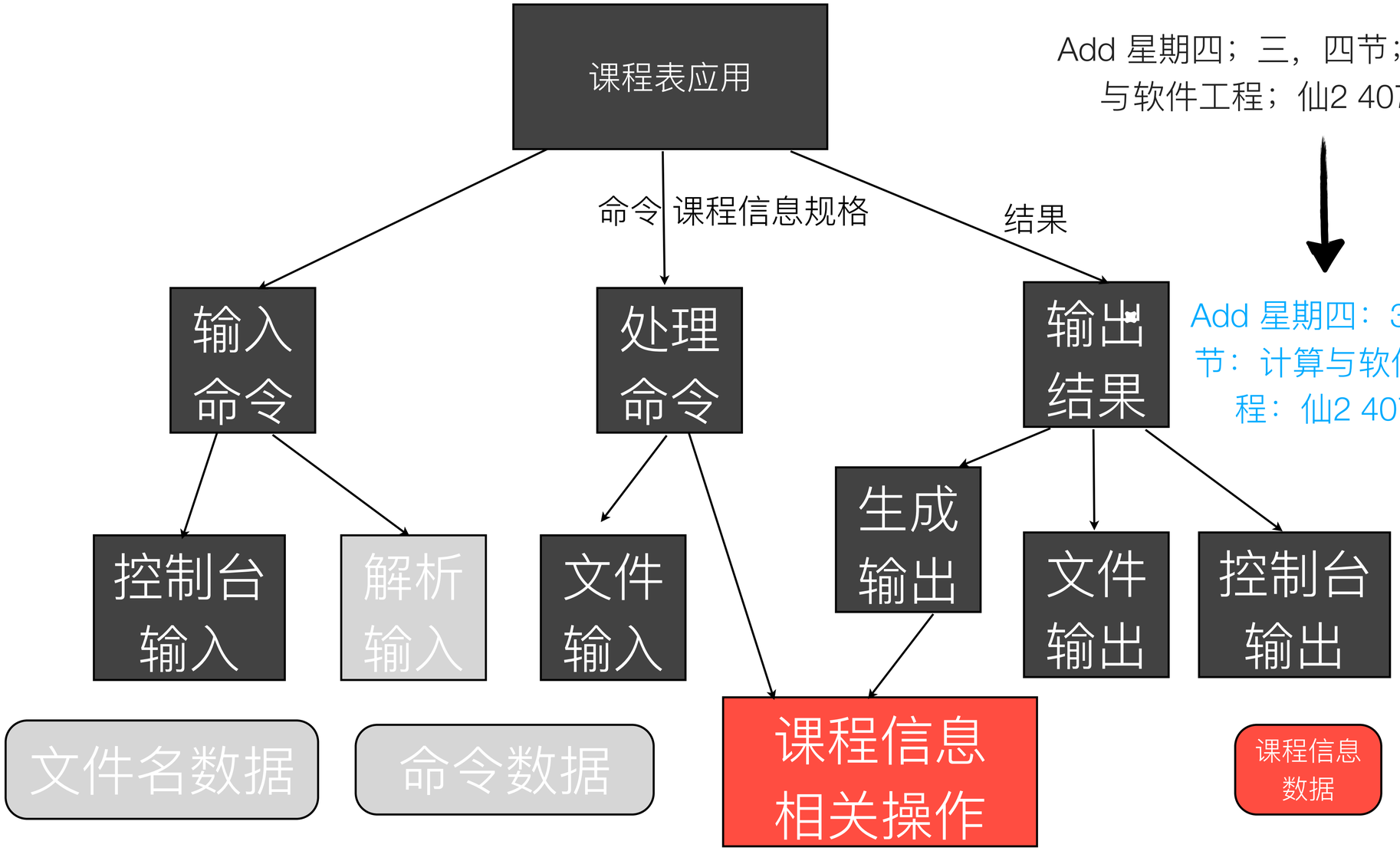


Add 星期四；三，四节；计算与软件工程；仙2 407；

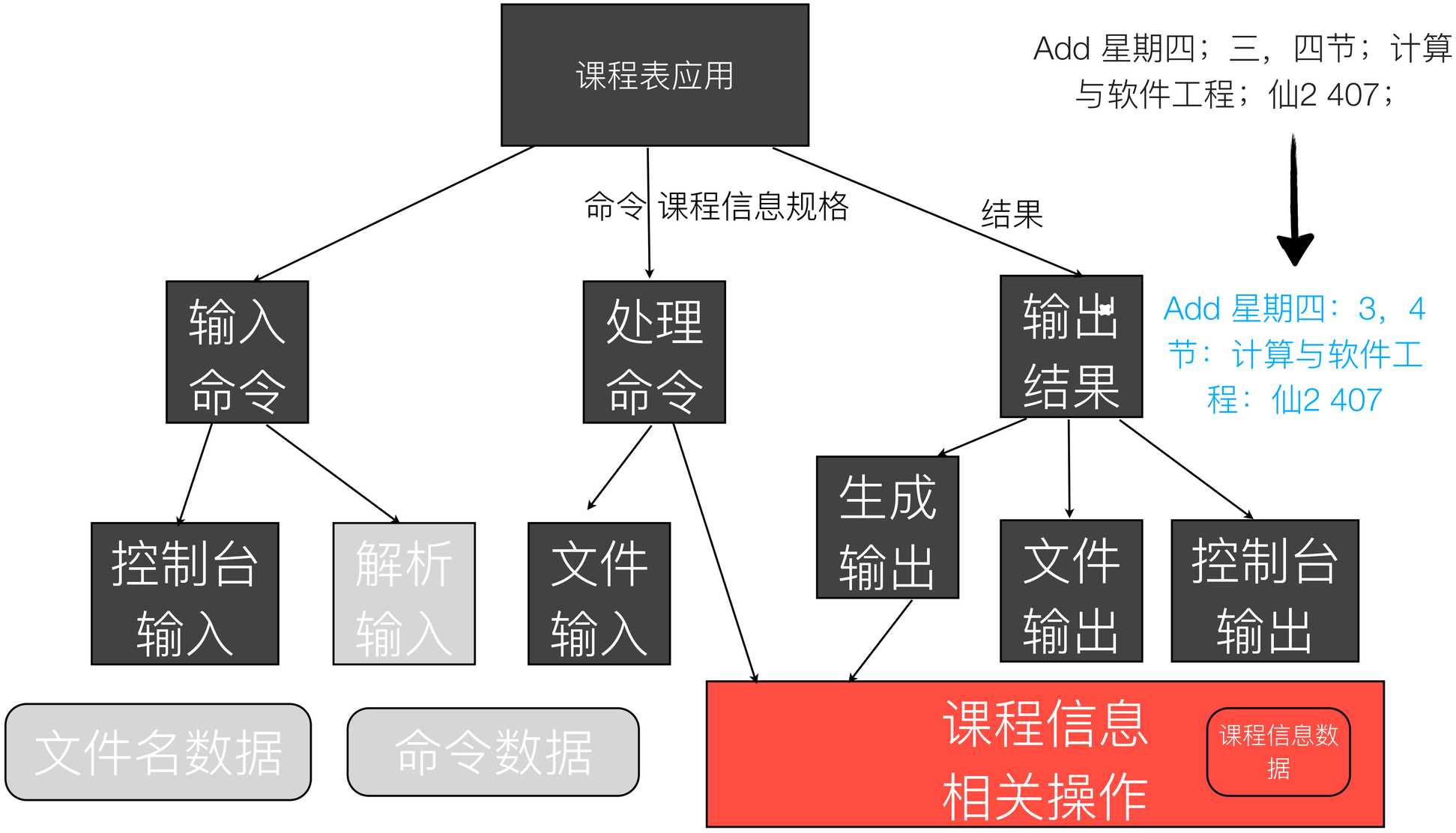
数据格式发生修改

Add 星期四：3，4节：计算与软件工程：仙2 407

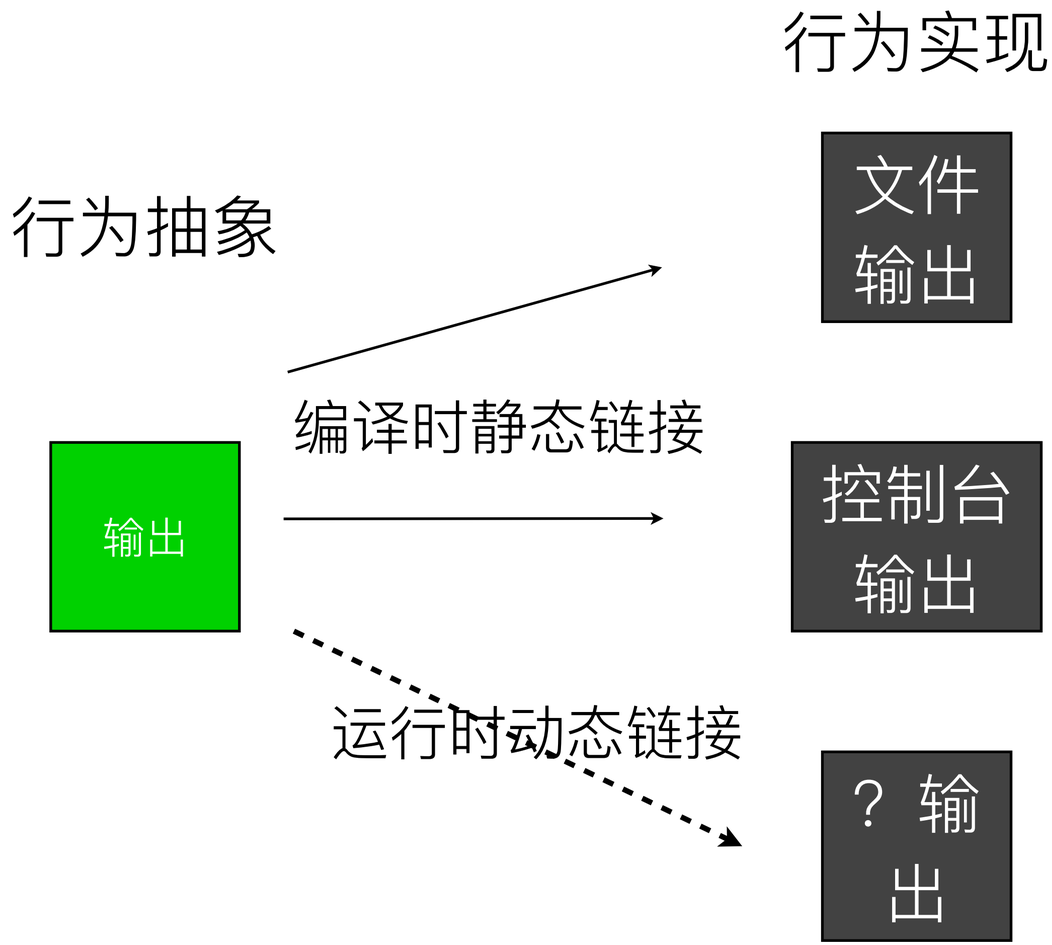
在有限的范围内修改--相关操作在一起



在有限的范围内修改--数据与操作在一起（封装）



扩展--运行时动态链接（多态）



Outline

- 结构化编程的问题
- 面向对象思想
- 类和对象
- 案例分析

职责

职责

- 数据职责和行为职责
- 在一起

在一起

- 这个类的数据和行为能体现一个职责么?
  - class Person{
    - String name;
    - int getAge(){}
  - }

在一起

- 这个类的数据和行为能体现一个职责么?
  - class Person{
    - Date birthday;
    - Int getAge(){}
  - }

在一起

- 这个类的数据和行为能体现一个职责么?
  - class Person{
    - String name;
    - Date birthday;
    - public getName(){}
    - public getAge(){}
  - }

再看课程表应用中有哪些职责?

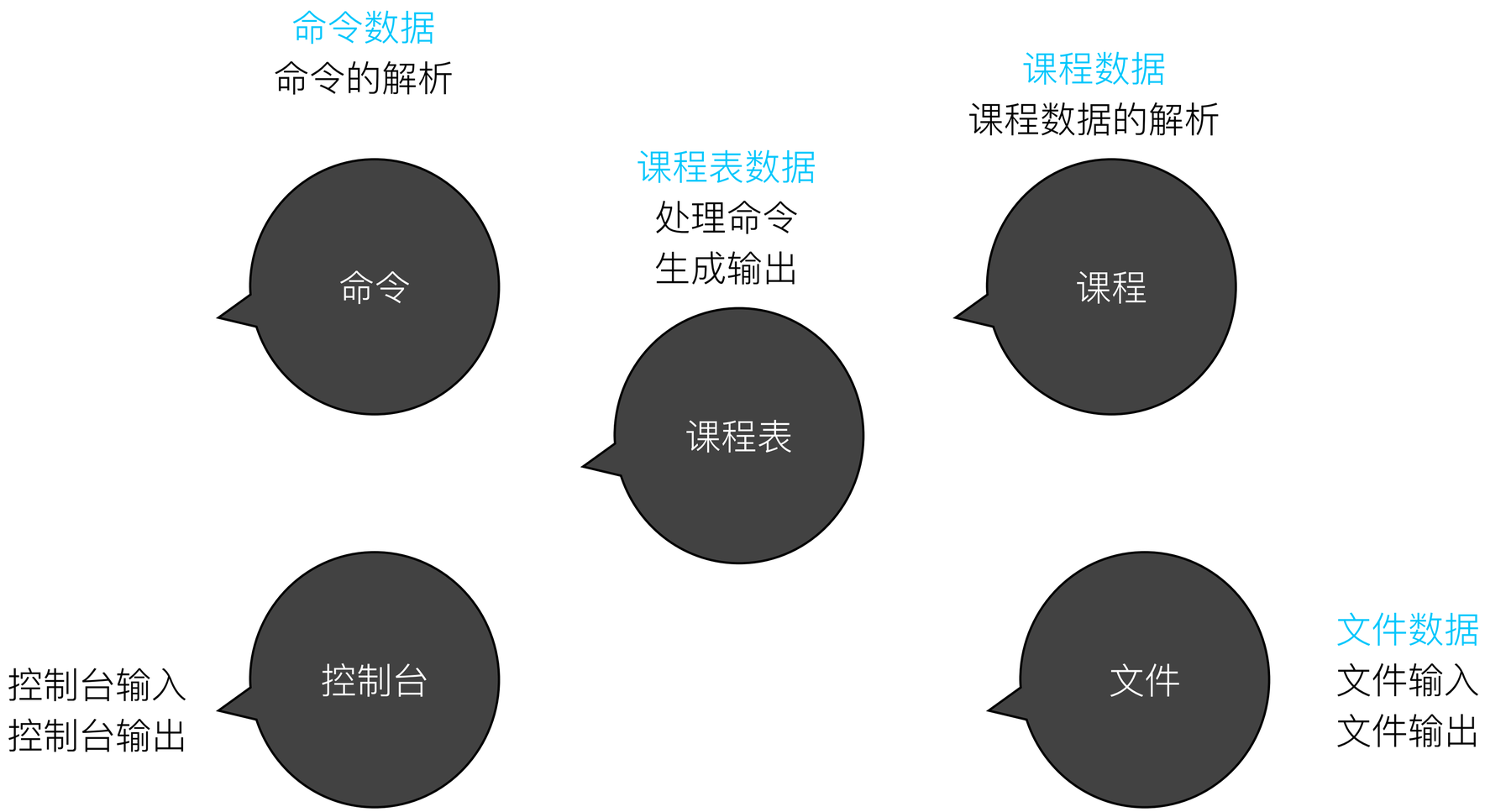
行为职责

- 命令的解析
- 生成输出
- 课程数据的解析
- 控制台输入
- 处理命令
- 文件输入
- 控制台输出
- 文件输出

数据职责

- 课程表数据
- 课程数据
- 命令数据
- 文件数据

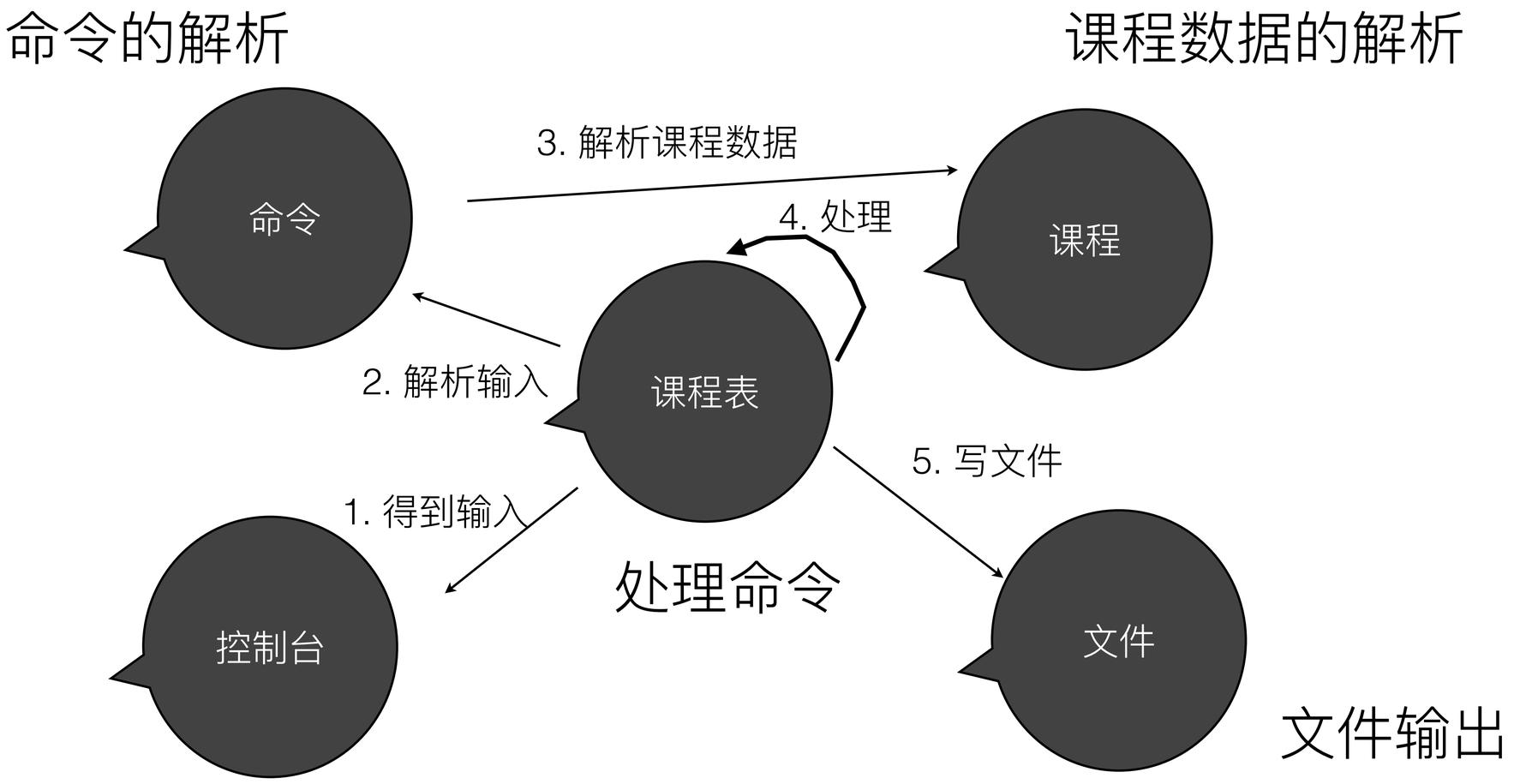
职责的分配


命令数据
命令的解析
课程数据
课程数据的解析
课程表数据
处理命令
生成输出
命令
课程表
课程
控制台输入
控制台输出
控制台
文件
文件数据
文件输入
文件输出

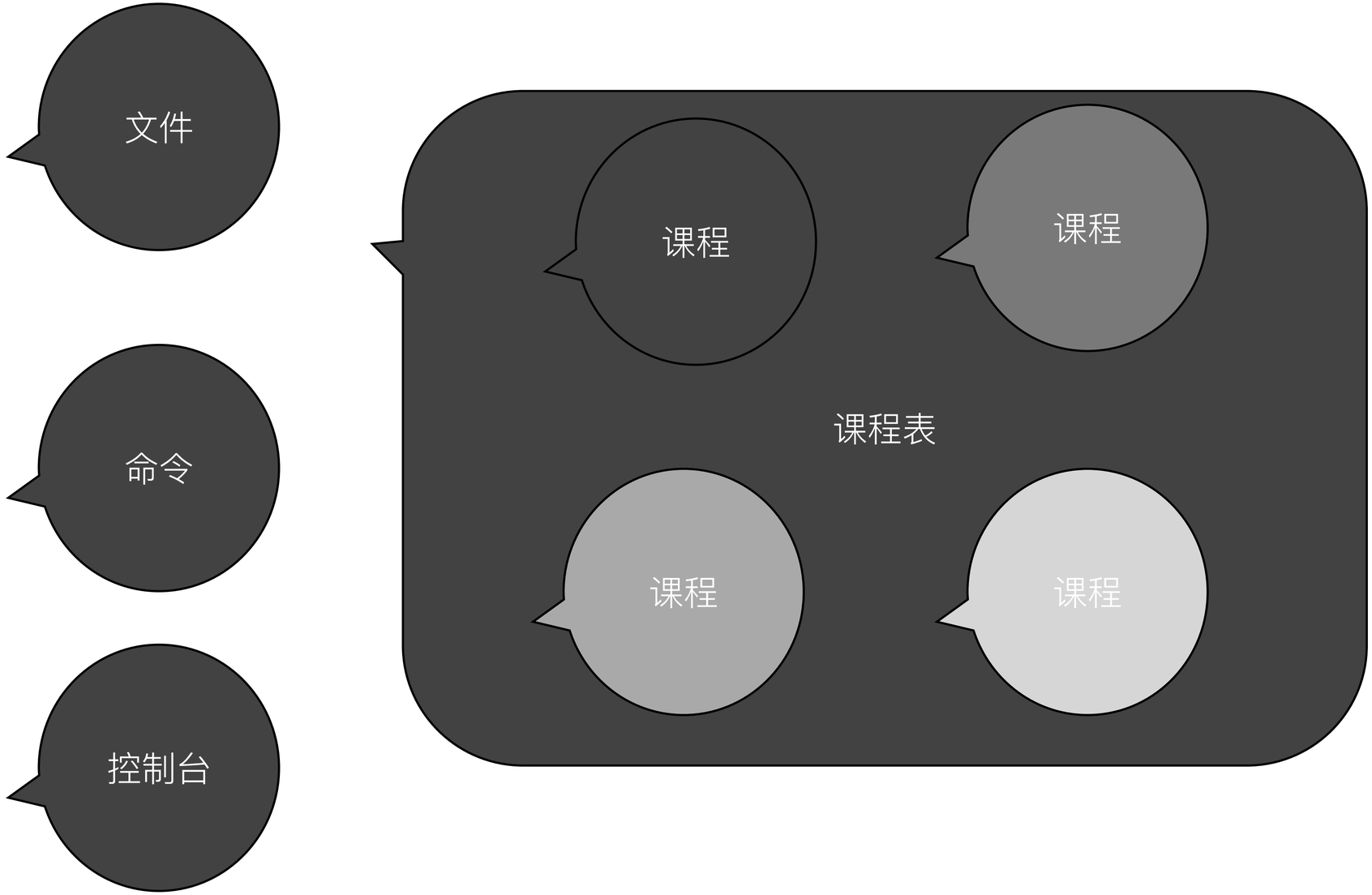
类 - 职责的抽象

光有一个个职责，能不能完成
我们的任务?

他们是怎么交互的?


命令的解析
课程数据的解析
命令
3. 解析课程数据
课程
4. 处理
2. 解析输入
课程表
5. 写文件
1. 得到输入
处理命令
控制台
文件
文件输出
控制台输入

课程还有很多个?



对象 - 职责的实现

类和对象的关系

- 抽象与具体

你的世界观改变了！

以前是函数之间的调用，现在是有职责的对象之间的交互

视角的变化

- 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

Outline

- 结构化编程的问题
- 面向对象思想
- 类和对象
- 案例分析

什么是对象

- 对象是面向对象 中的术语，既表示客观世界问题空间(Namespace)中的某个具体的事物，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(Properties)和方法(Methods)，属性就是需要记忆的信息，方法就是对象能够提供的服务。

什么是对象

- 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。
- 每个对象的标识永远是不同的，状态常常也存在着差异。

如何获得对象

- 寻找候选对象
  - 找名词 -- 类（对象）与属性
  - 找动词 -- 行为
- 精化对象
  - 去除
    - 冗余
    - 不相干
    - 模糊的概念
  - 转化
    - 没有行为的对象-》某个类的属性

什么是类?

- 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，但却允许类中的每一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。

什么是类?

- 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。

类与对象

- 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

- 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。

职责

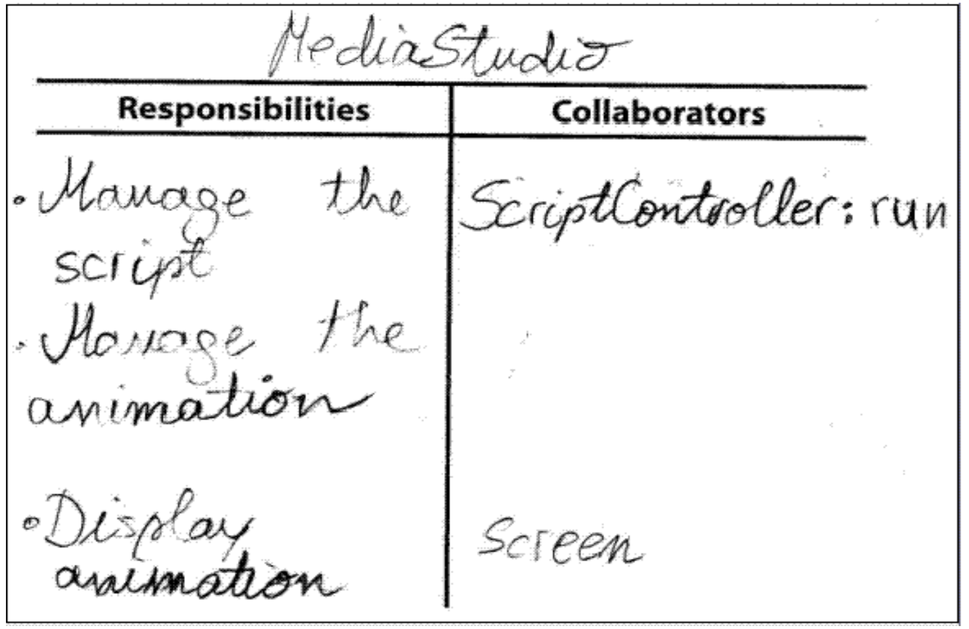
- 所谓职责，我们可以理解它为功能。
- 每个类应当只有单一职责。
  - 当你发现有两个变化会要求我们修改这个类，那么你就要考虑撤分这个类了。
- 给对象分配责任的策略：
  - 覆盖到所有重要的方面
  - 寻找需要执行的动作以及需要维护和生成的信息

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

面向对象分析

- 用例分析
- CRC卡
- 非正式英语描述

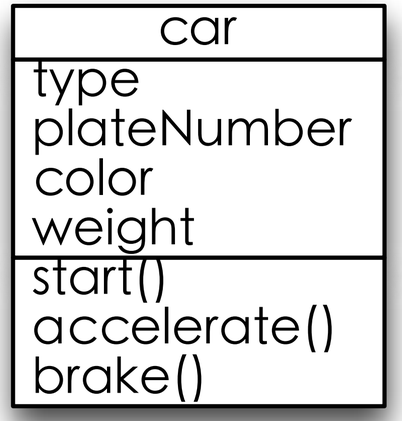


MediaStudio

Responsibilities	Collaborators
• Manage the script	ScriptController: run
• Manage the animation	
• Display animation	Screen

类的定义和类图

- class MyClass {
- 	// field, constructor, and
- 	// method declarations
- }



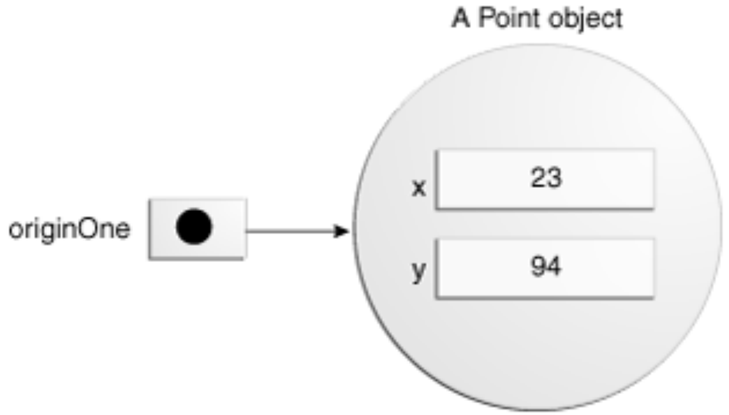
类的定义

```
public class Bicycle {

    private int cadence;
    private int gear;
    private int speed;

    public Bicycle(int startCadence, int startSpeed, int startGear) {  //构造方法
        gear = startGear;
        cadence = startCadence;
        speed = startSpeed;
    }

    public int getCadence() {
        return cadence;
    }

    public void setCadence(int newValue) {
        cadence = newValue;
    }

    。。。

    public int getSpeed() {
        return speed;
    }

    public void applyBrake(int decrement) {
        speed -= decrement;
    }

    public void speedUp(int increment) {
        speed += increment;
    }
}
```

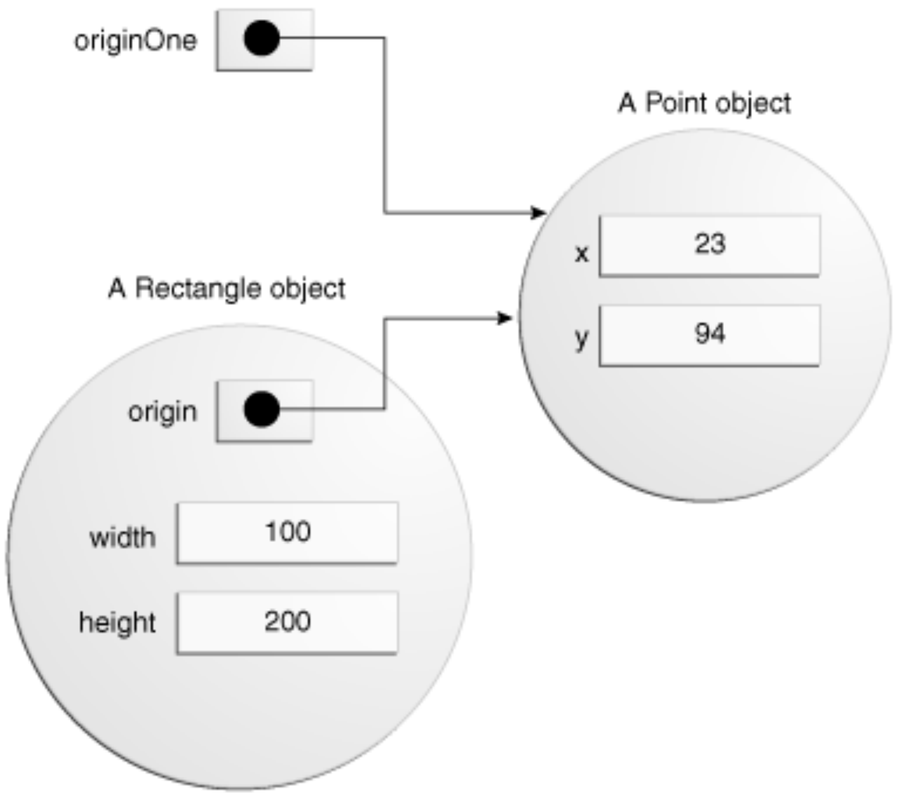
类的定义

```
public class Point {
    public int x = 0;
    public int y = 0;
    //constructor
    public Point(int a, int b) {
        x = a;
        y = b;
    }
}
```

```
Point originOne = new Point(23, 94);
```



```
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);

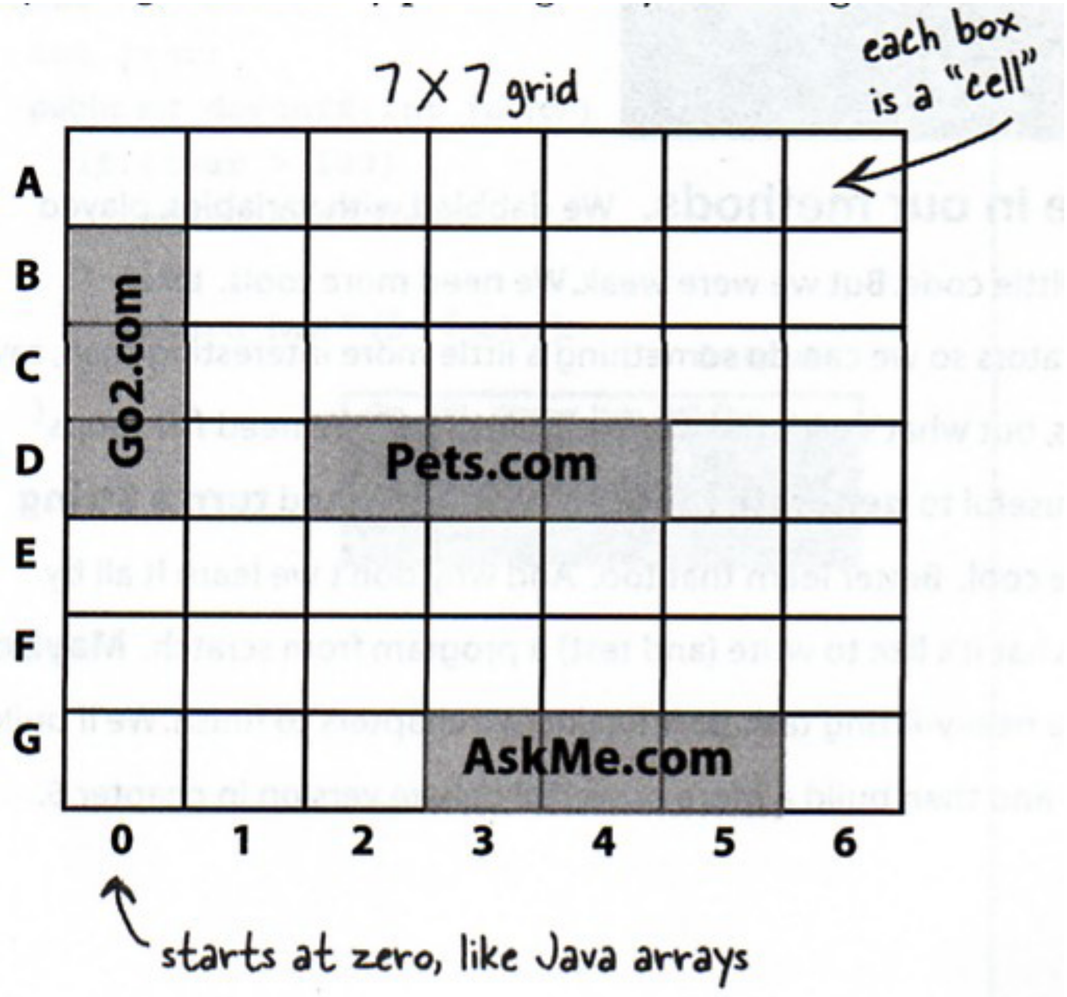
Outline

- 结构化编程的问题
- 面向对象思想
- 类和对象
- 案例分析

案例分析

A Real “Sink a Dot Com”

- Goal: Sink all Dot Coms in the fewest number of guesses
- Setup: Three Dot Coms on 7x7 grid
- How you play:
  - Guess -> Hit, miss or kill?;
  - Guess -> Hit, miss or kill?;
  - …
  - Win, Show your number of guesses



Sink Dot Com

part of a game interaction

```
File Edit Window Help Sell
%java DotComBust
Enter a guess  A3
miss
Enter a guess  B2
miss
Enter a guess  C4
miss
Enter a guess  D2
hit
Enter a guess  D3
hit
Enter a guess  D4
Ouch! You sunk Pets.com   : (
kill
Enter a guess  B4
miss
Enter a guess  G3
hit
Enter a guess  G4
hit
Enter a guess  G5
Ouch! You sunk AskMe.com   : (
```

What have been changed

- Number of dotcom:
  - 1 -> 3
- Position of dotcom:
  - 1D -> 2D
- Algorithm of the game:
  - Easy -> Complicated

用面向对象方法我们怎么办?

职责分配，协作

过程

- 寻找职责
- 职责分配
- 对象的创建
- 对象的协作

寻找职责

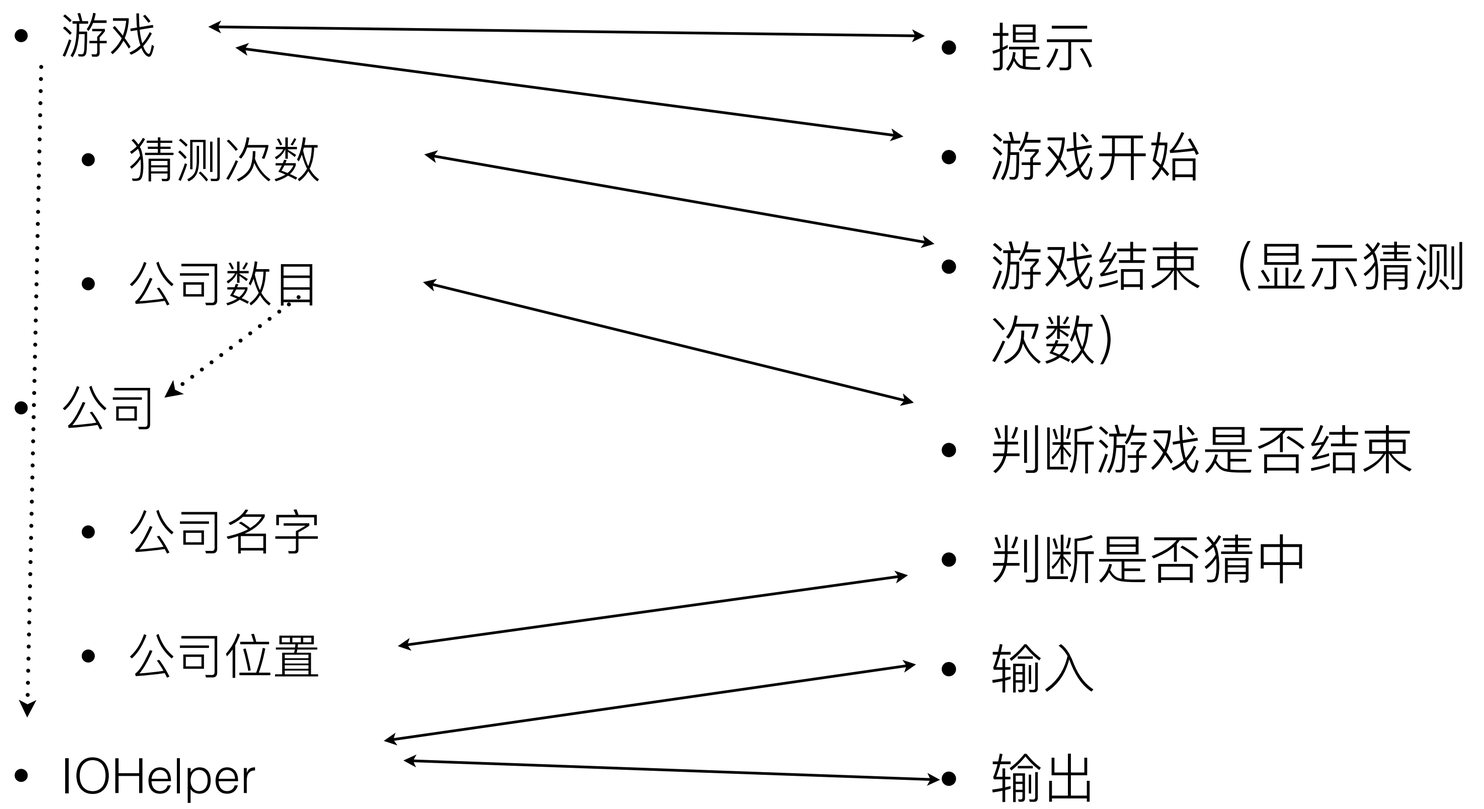
- 寻找数据职责
  - 找名词
- 寻找行为职责
  - 找动词
- 确定职责
  - 去除冗余、无关
  - 检查是否满足“封装”的条件

名词与动词

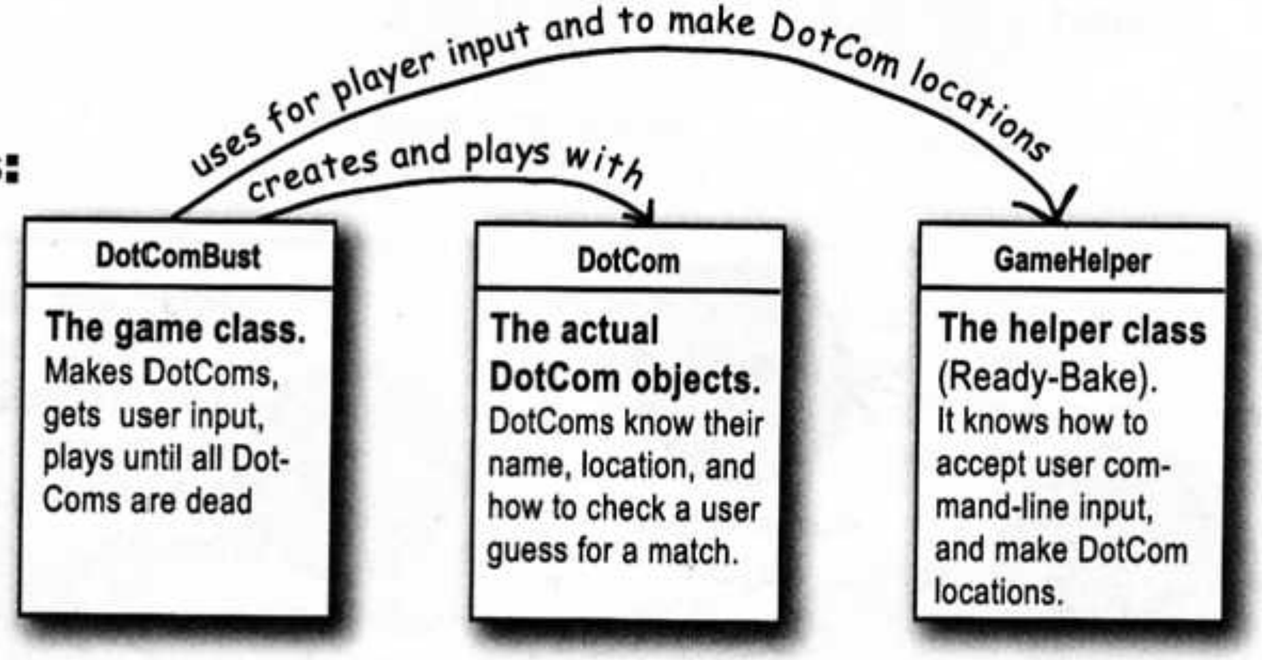
- 游戏
- 猜测次数
- 棋盘大小
- 公司
- 公司数目
- 公司名字
- 公司位置

- 提示
- 猜测
- 判断是否猜中
- 判断游戏是否结束
- 输入
- 输出
- 游戏开始
- 游戏结束

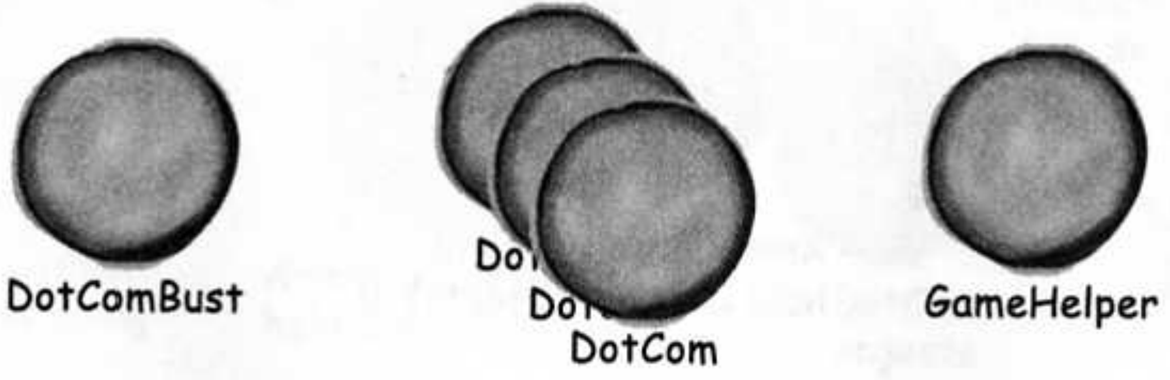
名词与动词


游戏
猜测次数
公司数目
公司
公司名字
公司位置
IOHelper
提示
游戏开始
游戏结束（显示猜测次数）
判断游戏是否结束
判断是否猜中
输入
输出

3 Classes:



5 Objects:



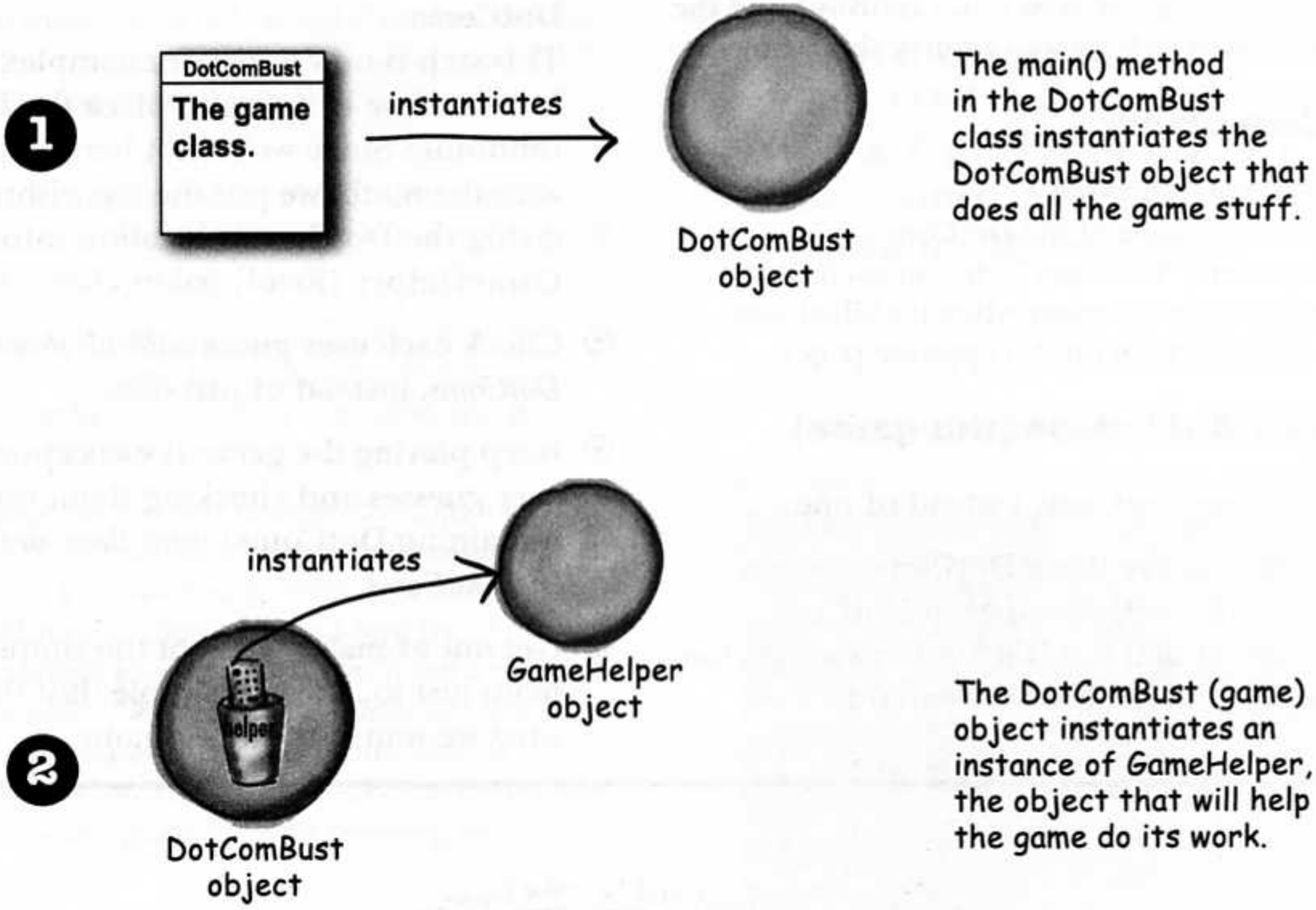
Plus 4 ArrayLists: 1 for the DotComBust and 1 for each of the 3 DotCom objects.

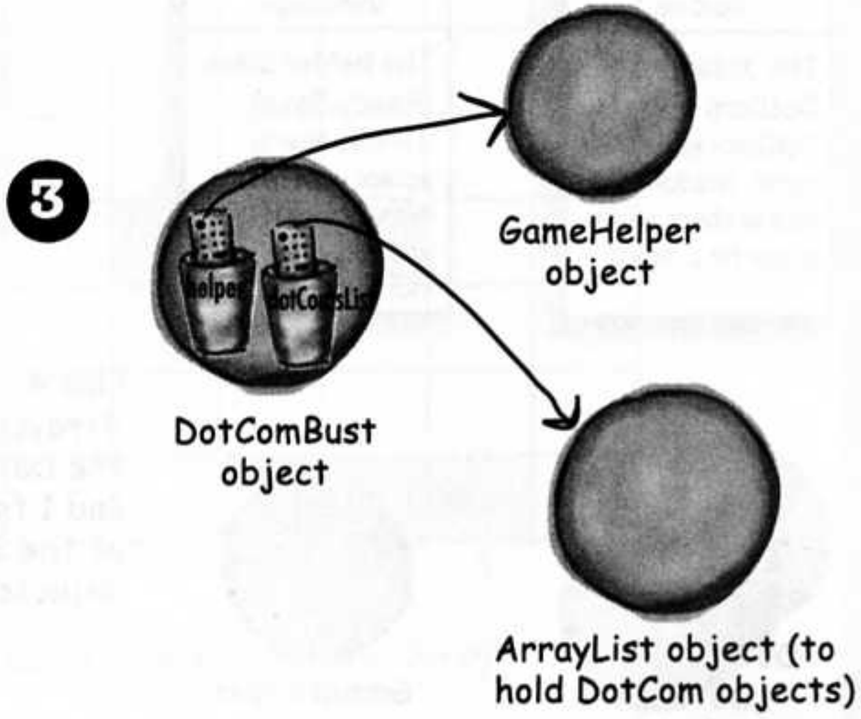
职责 - DotComBust

- 数据职责
  - DotCom集合
  - 猜测次数
- 行为职责
  - 游戏开始
  - 判断游戏结束
  - 游戏结束（显示猜测次数）
- 对象个数
  - 1个

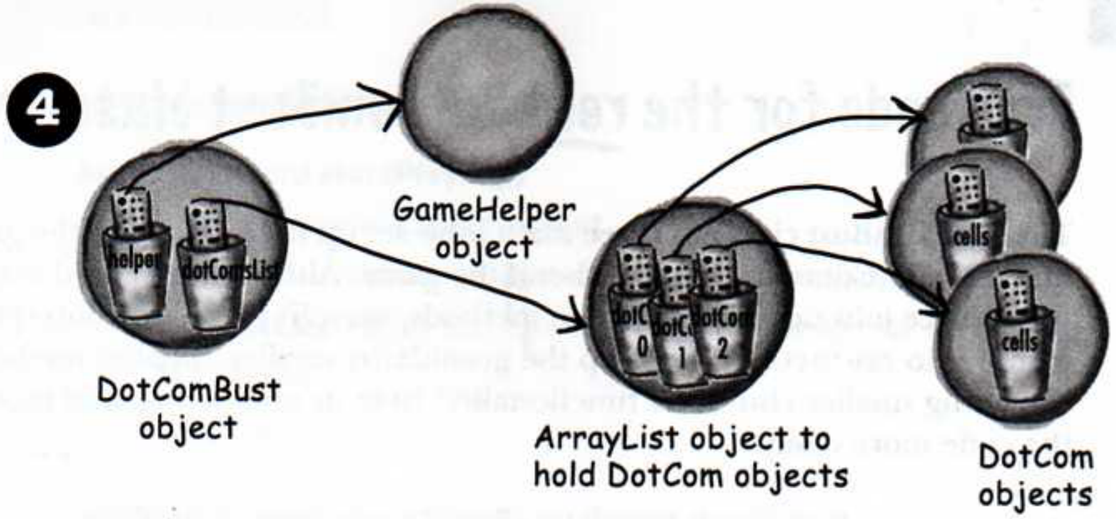
职责 - DotCom

- 数据职责
  - DotCom位置
- 行为职责
  - 判断是否猜中
- 对象个数
  - 3个


1
DotComBust
The game class.
instantiates
DotComBust object
The main() method in the DotComBust class instantiates the DotComBust object that does all the game stuff.
2
instantiates
GameHelper object
DotComBust object
The DotComBust (game) object instantiates an instance of GameHelper, the object that will help the game do its work.



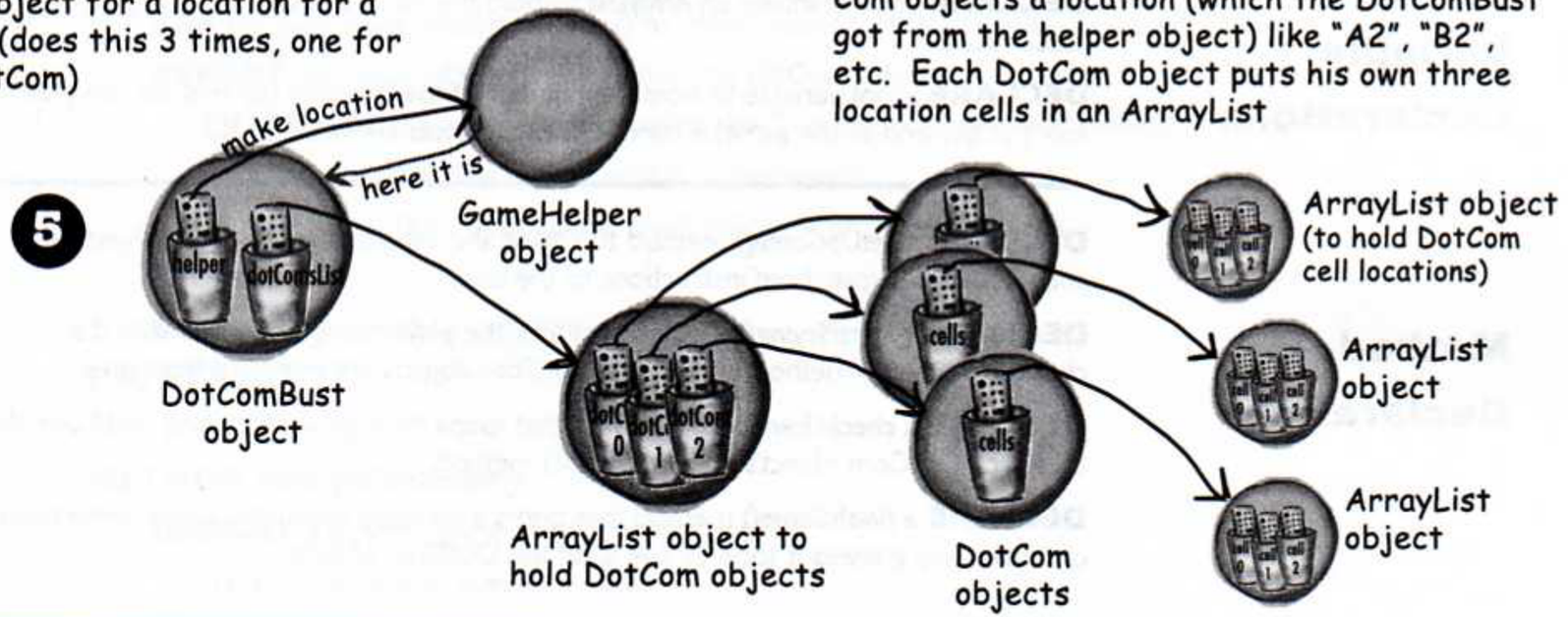
The DotComBust object instantiates an ArrayList that will hold the 3 DotCom objects.



The DotComBust object creates three DotCom objects (and puts them in the ArrayList)

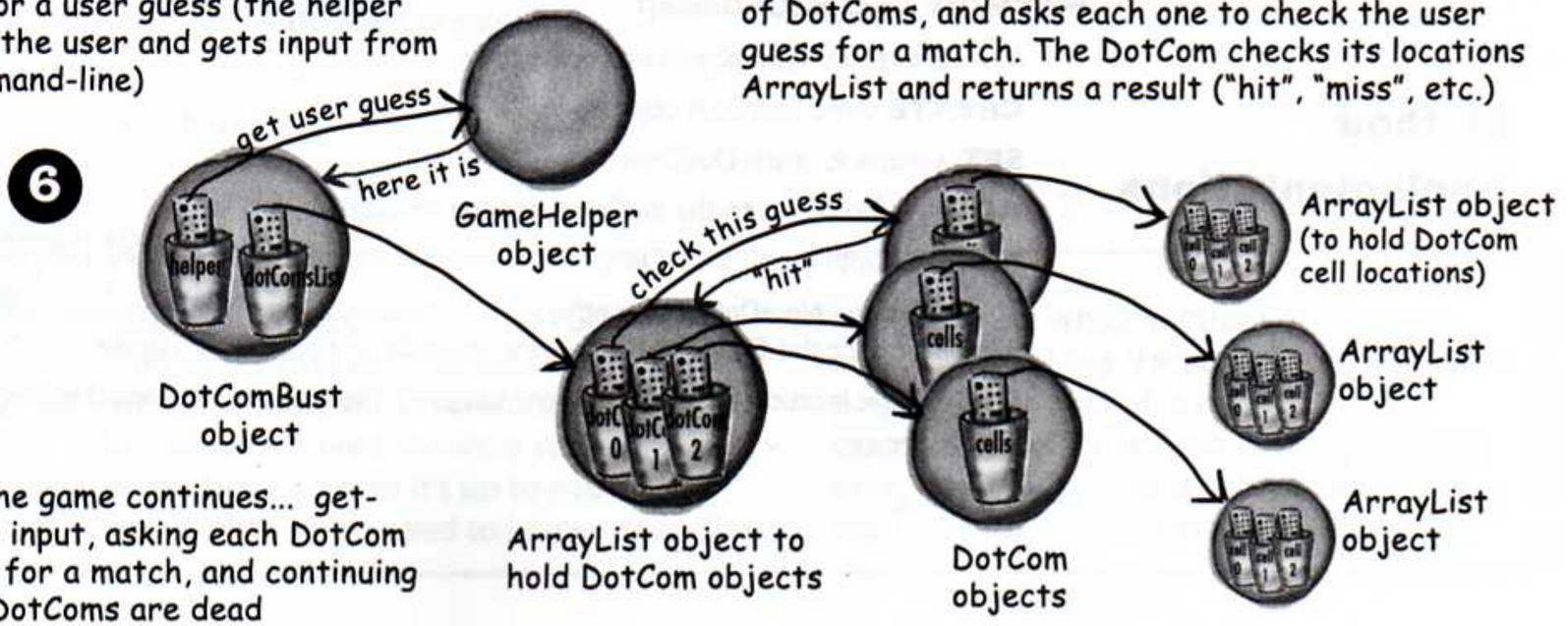
The DotComBust object asks the helper object for a location for a DotCom (does this 3 times, one for each DotCom)

The DotComBust object gives each of the DotCom objects a location (which the DotComBust got from the helper object) like "A2", "B2", etc. Each DotCom object puts his own three location cells in an ArrayList



The DotComBust object asks the helper object for a user guess (the helper prompts the user and gets input from the command-line)

The DotComBust object loops through the list of DotComs, and asks each one to check the user guess for a match. The DotCom checks its locations ArrayList and returns a result ("hit", "miss", etc.)



And so the game continues... getting user input, asking each DotCom to check for a match, and continuing until all DotComs are dead

```
import java.util.*;

public class DotComBust {
    private GameHelper helper = new GameHelper();
    private ArrayList<DotCom> dotComsList = new
ArrayList<DotCom>();
    private int numOfGuesses = 0;

    private void setUpGame() {
        DotCom one = new DotCom();
        one.setName("Pets.com");
        DotCom two = new DotCom();
        two.setName("eToys.com");
        DotCom three = new DotCom();
        three.setName("Go2.com");
        dotComsList.add(one);
        dotComsList.add(two);
        dotComsList.add(three);

        System.out.println("Your goal is to sink three dot
coms.");
        System.out.println("Pets.com, eToys.com, Go2.com");
        System.out.println("Try to sink them all in the fewest
number of guesses");

        for (DotCom dotComSet : dotComsList) {
            ArrayList<String> newLocation =
helper.placeDotCom(3);
            dotComSet.setLocationCells(newLocation);
        }
    }

    private void startPlaying() {
        while (!dotComsList.isEmpty()) {
            String userGuess = helper.getUserInput("Enter a
guess");
            checkUserGuess(userGuess);
        }
        finishGame();
    }

    private void finishGame() {
        System.out.println("All Dot Coms are dead!  Your stock
is now worthless");
        if (numOfGuesses <= 18) {
            System.out.println("It only took you " +
numOfGuesses + " guesses");
            System.out.println("You got out before your options
sank.");
        }
        else
        {
            System.out.println("Took you long enough. " +
numOfGuesses + " guesses.");
            System.out.println("Fish are dancing with your
options.");
        }
    }

    public static void main(String[] args) {
        DotComBust game = new DotComBust();
        game.setUpGame();
        game.startPlaying();
    }
}
```

DotComBust - 1

```
private void checkUserGuess(String userGuess)
{
    numOfGuesses++;
    String result = "miss";

    for (DotCom dotComToTest : dotComsList)
    {
        result = dotComToTest.checkYourself(userGuess);
        if (result.equals("hit"))
        {
            break;
        }
        if (result.equals("kill"))
        {
            dotComsList.remove(dotComToTest);
            break;
        }
    }
    System.out.println(result);
}
```

DotComBust - 2

DotCom

```
import java.util.ArrayList;

public class DotCom {
    private ArrayList<String> locationCells;

    public void setLocationCells(ArrayList<String> loc)
    {
        locationCells = loc;
    }

    public String checkYourself(String userInput)
    {
        String result = "miss";
        int index = locationCells.indexOf(userInput);
        if (index >= 0) {
            locationCells.remove(index);
            if (locationCells.isEmpty()) {
                result = "kill";
            }
            else
            {
                result = "hit";
            }
        }
        return result;
    }

    //TODO:  all the following code was added and should have been included in the book
    private String name;
    public void setName(String string) {
        name = string;
    }
}
```

GameHelper - 1

```
import java.io.*;
import java.util.*;

public class GameHelper {

    private static final String alphabet = "abcdefg";
    private int gridLength = 7;
    private int gridSize = 49;
    private int [] grid = new int[gridSize];
    private int comCount = 0;


    public String getUserInput(String prompt) {
        String inputLine = null;
        System.out.print(prompt + "   ");
        try {
            BufferedReader is = new BufferedReader(
                    new InputStreamReader(System.in));
            inputLine = is.readLine();
            if (inputLine.length() == 0 )  return null;
        } catch (IOException e) {
            System.out.println("IOException: " + e);
        }
        return inputLine.toLowerCase();
    }


…

}
```

GameHelper - 2

```
public ArrayList<String> placeDotCom(int comSize) {                    // line 19
    ArrayList<String> alphaCells = new ArrayList<String>();
    String [] alphacoords = new String [comSize];       // holds 'f6' type coords
    String temp = null;                                 // temporary String for concat
    int [] coords = new int[comSize];                   // current candidate coords
    int attempts = 0;                                   // current attempts counter
    boolean success = false;                            // flag = found a good location ?
    int location = 0;                                   // current starting location

    comCount++;                                         // nth dot com to place
    int incr = 1;                                       // set horizontal increment
    if ((comCount % 2) == 1) {                          // if odd dot com  (place vertically)
        incr = gridLength;                                // set vertical increment
    }
```

GameHelper - 3

```
while ( !success & attempts++ < 200 ) {                // main search loop  (32)
    location = (int) (Math.random() * gridSize);         // get random starting point
    //System.out.print(" try " + location);
    int x = 0;                                           // nth position in dotcom to place
    success = true;                                      // assume success
    while (success && x < comSize) {                     // look for adjacent unused spots
        if (grid[location] == 0) {                       // if not already used
            coords[x++] = location;                       // save location
            location += incr;                             // try 'next' adjacent
            if (location >= gridSize){                    // out of bounds - 'bottom'
                success = false;                            // failure
            }
            if (x>0 & (location % gridLength == 0)) {    // out of bounds - right edge
                success = false;                            // failure
            }
        } else {                                         // found already used location
            // System.out.print(" used " + location);
            success = false;                             // failure
        }
    }
}                                                      // end while
```

GameHelper - 4

```
// end while

        int x = 0;                                          // turn good location into alpha coords
        int row = 0;
        int column = 0;
        // System.out.println("\n");
        while (x < comSize) {
            grid[coords[x]] = 1;                            // mark master grid pts. as 'used'
            row = (int) (coords[x] / gridLength);           // get row value
            column = coords[x] % gridLength;                // get numeric column value
            temp = String.valueOf(alphabet.charAt(column)); // convert to alpha

            alphaCells.add(temp.concat(Integer.toString(row)));
            x++;

            // System.out.print("  coord "+x+" = " + alphaCells.get(x-1));

        }
        // System.out.println("\n");

        return alphaCells;
    }
}
```