实用Java教程

基于BlueJ的对象优先方法（第3版）

[英] David J. Barnes　Michael Kölling 著

第五章 高级行为

Java语言程序设计
课程教案

Java的类库 (class library)

- Java的类库很大，包含几千个类，每一个类又有许多方法。
- 这些类可以在所有Java程序中使用，使得开发工作变得简单。
- 但是不可能有人记住Java类库的所有细节，因此学会使用Java类库将是Java 开发中的一项重要工作。
  - 需要记住一些很重要的常用的类的名字。（比如ArrayList、String等）。
  - 学会如何查找所需的类及其细节。

注意

我们并不需要知道类的实现细节，仅仅需要知道类的使用方式!

- 接口 (interface)：类的接口描述了类的功能以及在没有类的具体实现的情况下类的使用方法。
- 实现 (inplementation)：定义一个类的完整源代码叫作这个类的实现。

如何阅读类库的文档?

Java API == Java Application Programming Interface

类的接口与实现

- Java API文档中包含的信息主要包括以下几种:
  - 类的名字。
  - 类的作用的总体描述。
  - 类的构造器和方法的列表。
  - 每一个构造器和方法的参数和返回类型。
  - 每一个构造器和方法的作用的描述。

上述这些信息构成了类的外部接口!

类的接口与实现

- Java API文档中不包含以下信息:
  - 类的私有字段（类的大多数字段都是私有的）。
  - 类的私有方法。
  - 类的源代码。

上述这些信息属于类的内部实现!

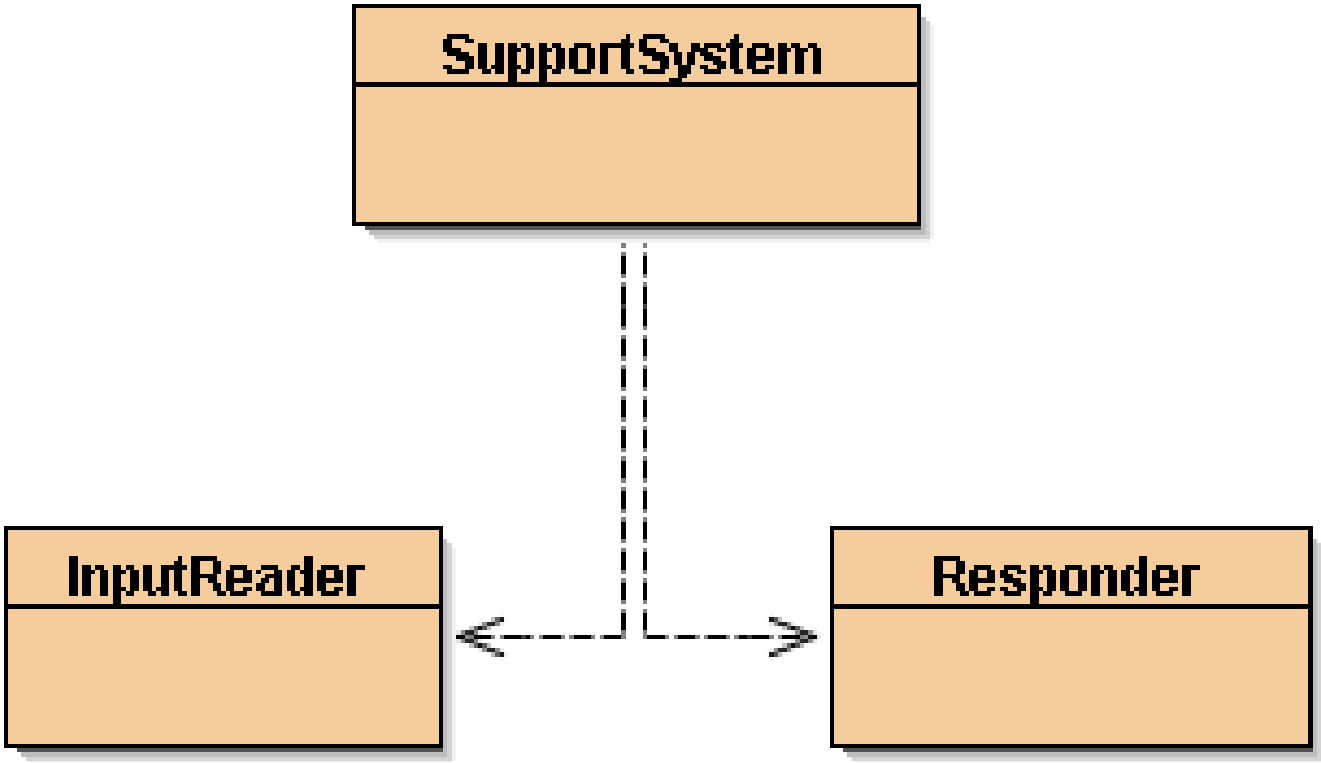
案例—TechSupport系统

- 通过TechSupport系统的开发来学习Java类库的使用。
- TechSupport系统具有如下基本特性:
  - 基于字符界面的对话窗口;
  - 系统能够根据用户提问进行相应解答;
  - 对于无法回答的提问，系统输出相应提示信息;
  - 当用户输入“bye”时，系统退出;

Demo

TechSupport (project: teck-support-complete)

对TechSupport系统的初步分析



- SupportSystem用来管理整个系统的运行。
- InputReader用来接收用户的输入信息。
- Responder用来响应用户的输入信息。

SupportSystem类中的while循环

SupportSystem.java (project:tech-support1)

```
public void start()
{
    boolean finished = false;
    printWelcome();
    while(!finished) {
        String input = reader.getInput();
        if(input.startsWith("bye")) {
            finished = true;
        } else {
            String response = responder.generateResponse()
            System.out.println(response);
        }
    }
    printGoodbye();
}
```

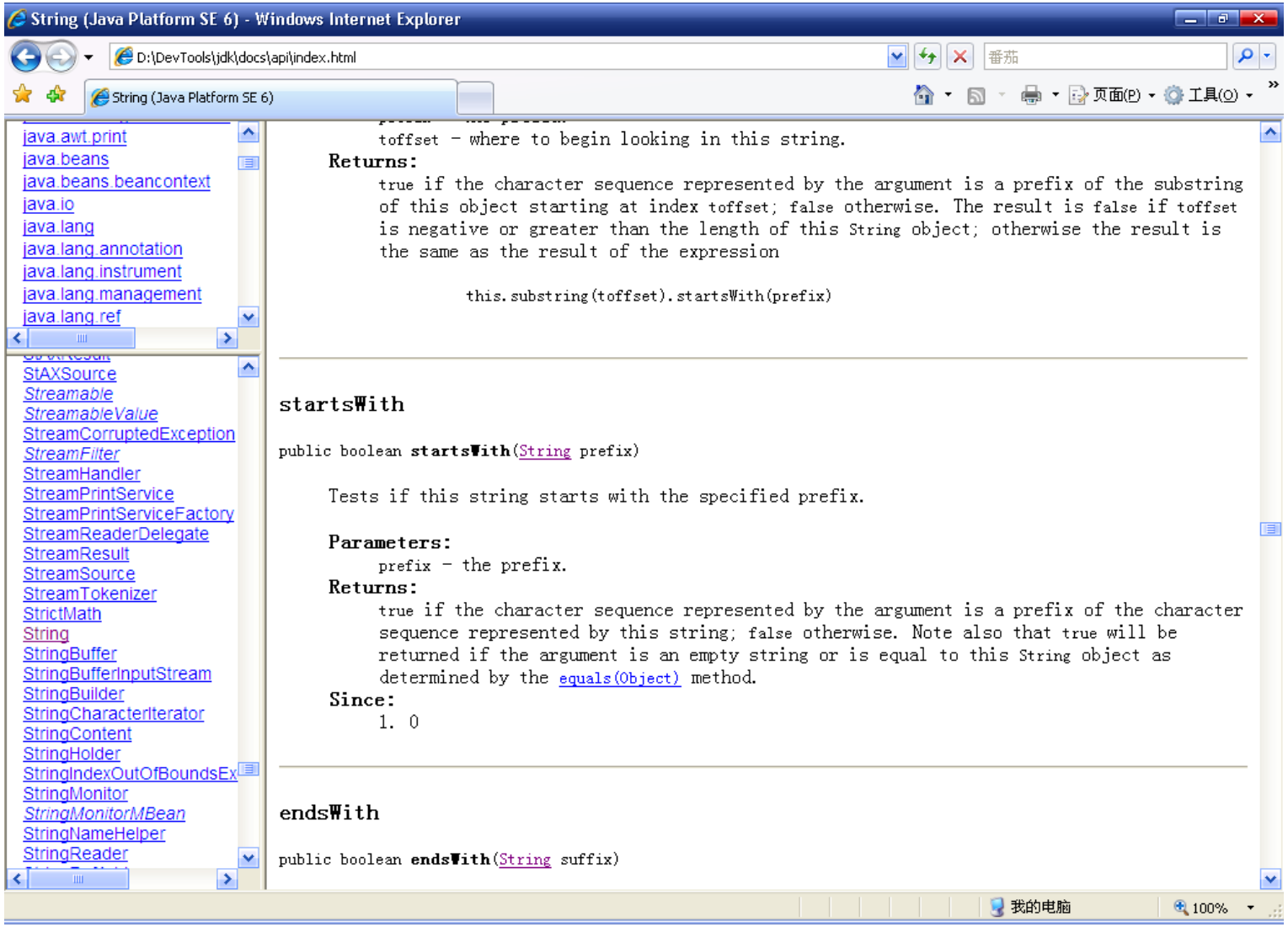
循环的结束条件

SupportSystem.java (project:tech-support1)

```
public void start()
{   ...
    while(!finished) {
        String input = reader.getInput();
        if(input.startsWith("bye")) {
            finished = true;
        }
        ...
    }
    ...
}
```

"startsWith"是什么意思？它的功能是什么？我们怎么找到相关信息？

在类库中寻找所需的信息



对TechSupport系统的一点改进

- 我们希望在输入"Bye"时系统也能够正常退出。
- 我们希望在输入多余的空格时（如：" bye "）系统也能够正常退出。

通过查找Java API，可以发现String类的trim()和toLowerCase()方法能够满足我们的需求！

SupportSystem.java

```
while(!finished) {
    String input = reader.getInput();
    input = input.trim().toLowerCase();
    if(input.startsWith("bye")) {
        finished = true;
    }
    ...
}
```

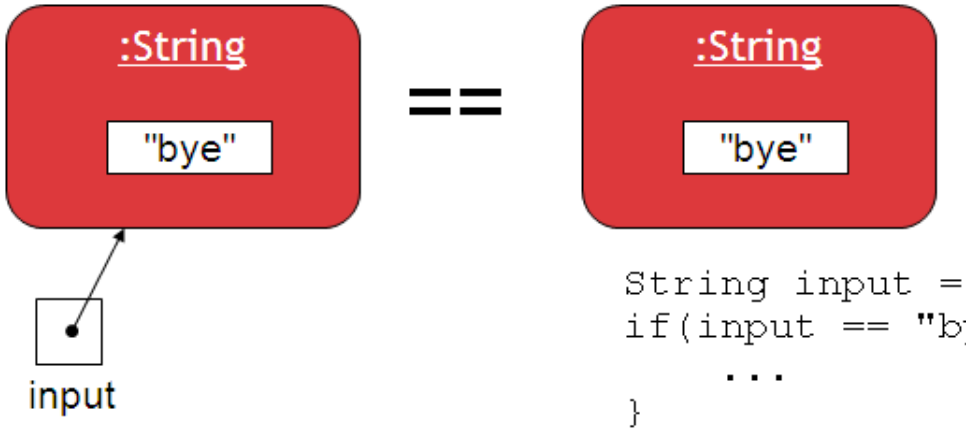
Java中的字符串

- Java中的字符串是String类型，它是java.lang包中的一个类。
- Java中的字符串都是不可变对象。

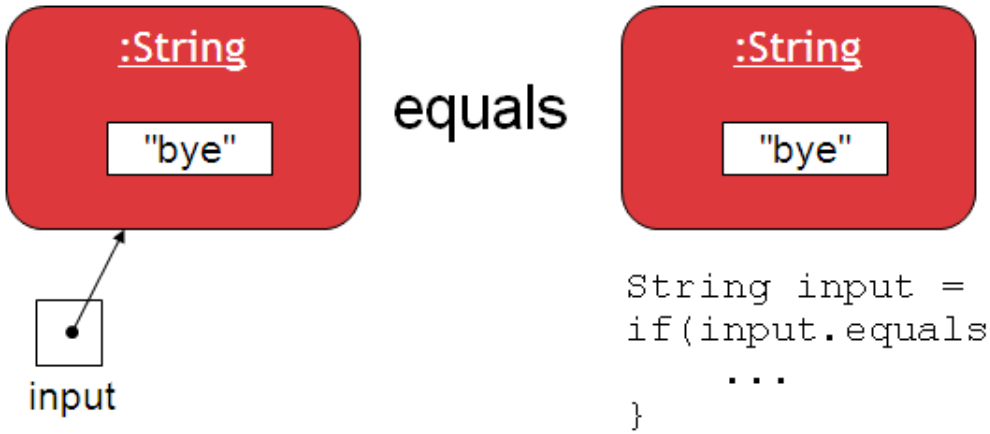
不可变对象 (immutable object)　这种类型的对象在被创建后，其内容或状态是不可以改变的。

- input = input.trim().toLowerCase();
  赋值运算符左边的input所得到的实际上是一个新字符串，而不是原来的那一个！

检查字符串是否相等



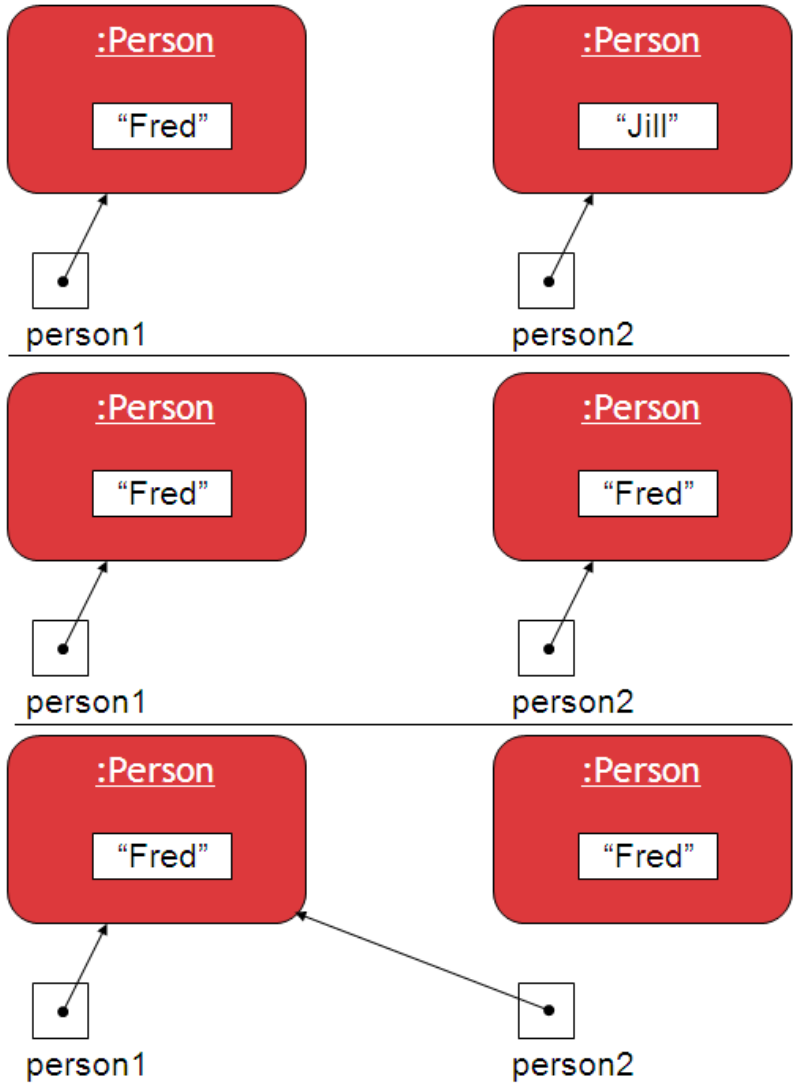

```
String input = reader.getInput();
if(input == "bye") {
    ...
}
```




```
String input = reader.getInput();
if(input.equals("bye")) {
    ...
}
```

- 相等比较运算符"=="通常用来检查运算符两边的变量是否指向同一个对象。
- String类的方法equals()用来检查两个字符串对象是否有相同的值。

小测试：判断下面几种情况的比较结果



person1 == person2

继续改进TechSupport系统

- 我们希望为TechSupport系统增加随机行为
  - 回答用户提出的问题时，系统可以从一系列待选答案中随机地挑选一个。
- 为了完成这个任务，我们需要:
  - 使用ArrayList来存储一些备选的响应字符串。
  - 系统可以产生随机数，并用该随机数作为列表的索引来得到一个响应字符串。

如何产生并使用随机数呢?

使用Random类产生随机数

- java.util包中的Random可以用来在程序中产生随机数

```
import java.util.Random;
...
Random randomGenerator = new Random();
...
int index1 = randomGenerator.nextInt();
int index2 = randomGenerator.nextInt(100);
```

在TechSupport系统中使用随机数

Responder.java (project: tech-support2)

```
import java.util.ArrayList;
import java.util.Random;
public class Responder
{
    private Random randomGenerator;
    private ArrayList<String> responses;
    public Responder()
    {
        randomGenerator = new Random();
        responses = new ArrayList<String>();
        fillResponses();
    }
    public String generateResponse()
    {
        int index = randomGenerator.nextInt(responses.size());
        return responses.get(index);
    }
    ...
```

如何使用类库中的类?

- Java类库中的类都以包 (package) 的形式进行管理。
- 要使用类库中的类，必须使用import语句将其引入程序。
  - import语句放在源文件的开头。
  - 引入后的类就可以像自己编写的类一样使用。
  - java.lang包中的类不需使用import语句引入。
  - 如果不使用import语句，则在程序中需要通过类的限定名来对其引用。

import语句的一般格式

import 包名.类名;

- import java.util.ArrayList;
- import java.util.Random;
- import java.util.*;

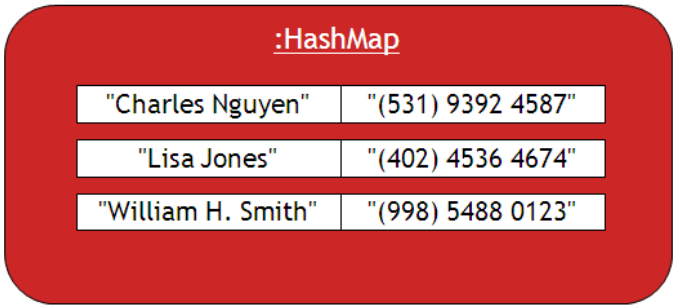
为TechSupport系统增加更复杂的行为

- 系统不应该随机回答用户的问题。
  - 回答用户提出的问题时，系统能够根据用户的输入返回相关的答案。
- 为了完成这个任务，我们需要:
  - 找出一些在典型问题中会出现的词语。
  - 将这些词语与特定的回答联系在一起。
  - 如果用户的输入中包含这些词语中的某一个，就可以产生其相关的回答。

如何将词语与特定的回答关联在一起呢?

使用映射建立关联

- 映射 (map) 是以键/值对 (key/value) 的形式保存对象的容器。通过键可以查询到其对应的值。
  - 例如：电话号码薄




```
HashMap <String, String> phoneBook =
                    new HashMap<String, String>();

phoneBook.put("Charles Nguyen", "(531) 9392 4587");
phoneBook.put("Lisa Jones", "(402) 4536 4674");
phoneBook.put("William H. Smith", "(998) 5488 0123");

String phoneNumber = phoneBook.get("Lisa Jones");
System.out.println(phoneNumber);
```

在TechSupport系统中使用映射

Responder.java (project: tech-support-complete)

```
import java.util.HashMap;
...
public class Responder
{
    private HashMap<String, String> responseMap;
    ...
    private void fillResponseMap()
    {
        responseMap.put("crash",
                        "Well, it never crashes on our system. ...");
        responseMap.put("crashes",
                        "Well, it never crashes on our system. ...");
        responseMap.put("slow",
                        "I think this has to do with your hardware. ...");
        responseMap.put("performance",
                        "Performance was quite adequate in all our tests. ...
        ...
    }
}
```

如何根据用户的输入得到相应的回答？

- 从用户的输入中得到关键字
  - 将用户输入的字符串分割为单个的单词作为关键字，并放入一个容器中
- 根据得到的关键字从列表中寻找相应的答案
  - 从关键字容器中依次取出每个关键字，并检查映射容器中是否包含以该关键字为键的键值对

如何分割字符串？关键字应该放入什么样的容器中？

分割字符串

- 可以使用String类的split()方法分割字符串

InputReader.java (project: tech-support-complete)

```
public class InputReader
{  ...
    public HashSet<String> getInput()
    {
        System.out.print("> ");     // print prompt
        String inputLine = reader.nextLine();
        input = input.trim().toLowerCase();

        String[] wordArray = inputLine.split(" ");
        ...
    }
    ...
}
```

使用集合

- 可以使用集合 (set) 来保存分割后的所有关键字

InputReader.java (project: tech-support-complete)

```
import java.util.HashSet;
...
public class InputReader
{  ...
    public HashSet<String> getInput()
    {  ...
        String[] wordArray = inputLine.split(" ");

        HashSet<String> words = new HashSet<String>();
        for(String word : wordArray) {
            words.add(word);
        }
        return words;
    }
```

集合、列表和映射的比较

- 集合 (set)、列表 (list)和映射 (map)都是可变容量的容器。
- 集合
  - 每个不同的元素只出现一次，不包含重复的元素。
  - 元素的排列没有任何顺序，不能按照加入的顺序取出元素。
  - HashSet是一种常用的集合类型。
- 列表
  - 可以包含相同的元素，但它们具有不同的索引位置。
  - 元素按照加入的顺序进行排列，可以根据索引依次访问元素。
  - ArrayList是一种常用的列表类型。
- 映射
  - 以键/值对保存元素，不能包含相同的键，但是可以有相同的值。
  - 键/值对的排列没有任何顺序，只能通过键访问其所对应的值。
  - HashMap是一种常用的映射类型。

Java中还有哪些类型的集合、列表和映射?

编写自己的类文档

- 好的类文档会使得开发和维护工作变得容易。
- 开发人员自己编写的类也应该具有像Java类库一样的文档。
- 在团队开发的环境中，开发人员可以通过查阅其他开发人员所编写的类文档来使用类。
- Java提供了专门的工具 (javadoc) 来帮助编写类文档。

类文档的组成

- 类的名字
- 关于类的功能和特性的描述。
- 版本号。
- 作者的名字。
- 每个构造器和方法的说明文档。
- ...

编写自己的类文档

构造器和方法的说明文档的组成

- 方法的名字。
- 返回的类型。
- 参数的名字和类型。
- 关于方法的功能和作用的描述。
- 关于每个参数的描述。
- 关于返回值的描述。

javadoc

SupportSystem.java (project: tech-support-complete)

```
/**
 * This class implements a technical support system.
 * It is the top level class in this project.
 * The support system communicates via text input/output
 * in the text terminal.
 *
 * This class uses an object of class InputReader to read input
 * from the user, and an object of class Responder to generate responses.
 * It contains a loop that repeatedly reads input and generates
 * output until the users wants to leave.
 *
 * @author     Michael Kolling and David J. Barnes
 * @version    1.0
 */
public class SupportSystem
{
    ...
```

javadoc

Responder.java (project: tech-support-complete)

```
/**
 * Construct a Responder
 */
public Responder() { ... }

/**
 * Generate a response from a given set of input words.
 *
 * @param words  A set of words entered by the user
 * @return       A string that should be displayed as the response
 */
public String generateResponse(HashSet<String> words) { ... }

/**
 * Enter all the known keywords and their associated responses
 * into our response map.
 */
private void fillResponseMap() { ... }
```

类中的访问修饰符

- 访问修饰符用来定义字段、构造器和方法等类成员的可见性。
- 公共访问修饰符
  - 被public所修饰的公共类成员能够从同一个类或任何其它类进行访问。
- 私有访问修饰符
  - 被private所修饰的私有类成员能够从同一个类进行访问。

除了public和private，java还提供了另外两种访问级别。详细讨论见第9章。

利用访问修饰符进行信息隐藏

- 在面向对象程序设计语言中，类的内部（实现部分）都是对其它类隐藏的。
  - 使用类的程序员不需要知道那个类的内部。
  - 使用类的程序员不允许知道那个类的内部。
- 信息隐藏保证了应用程序更好地实现模块化，降低了类之间的耦合度。
- 信息隐藏对于开发和维护大型的软件系统都是非常重要的原则。

使用private修饰类的字段来实现信息隐藏！

案例—balls工程

- 通过balls工程来进一步了解和学习本章前面介绍的知识。

Demo

balls (project: balls)

类变量

- 类也可以有字段，它们是类变量或称静态变量。
  - 类变量用static关键字修饰。
  - 不管某个类创建了多少个实例，每个类变量永远只有一个存储空间。
  - 实例变量保存在每一个对象中，每个对象都为其开辟了存储空间。

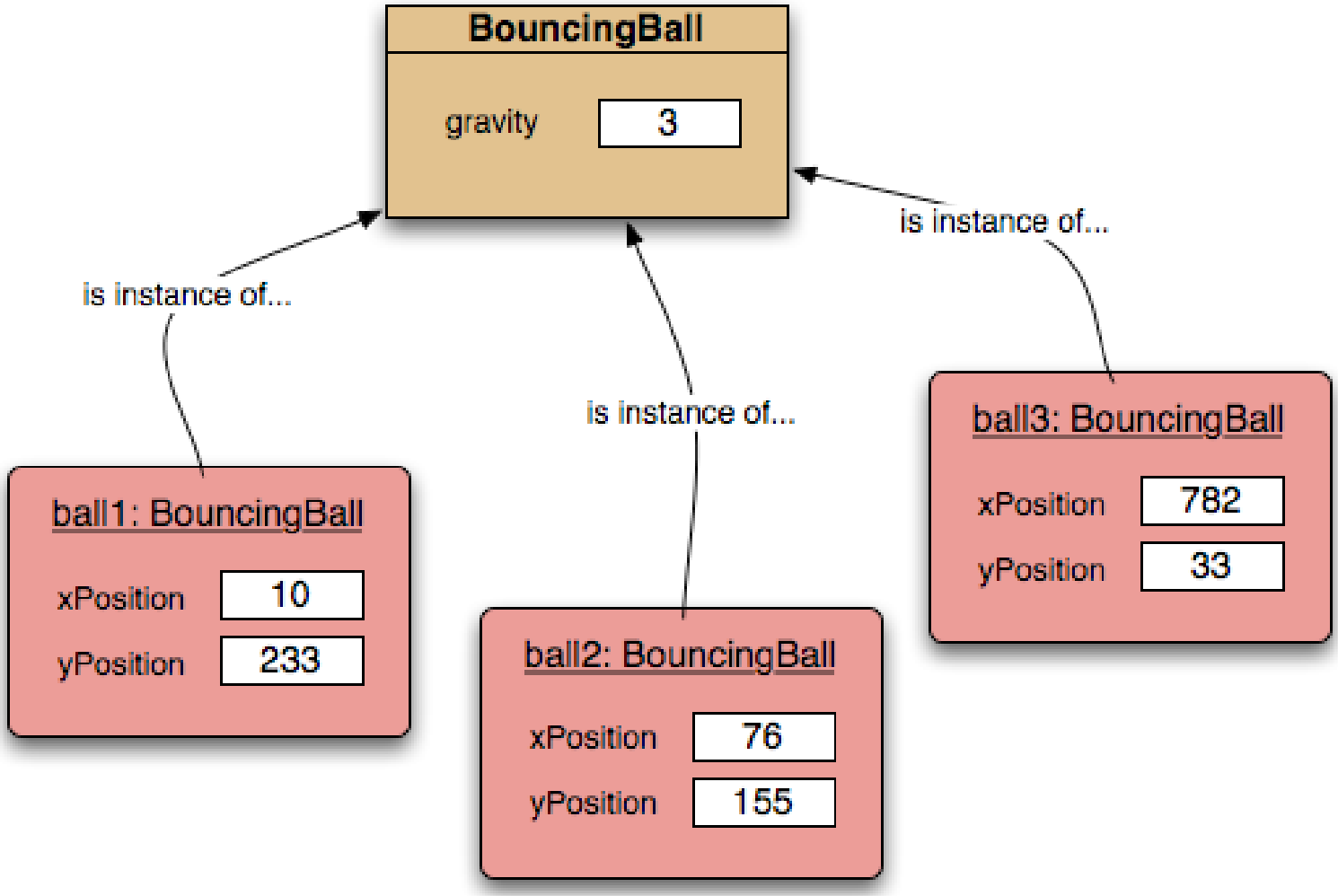
bouncing.java (project: balls)

```
public class BouncingBall
{
    private static final int GRAVITY = 3;

    private int xPosition;
    private int yPosition;

    ...
}
```

类变量


BouncingBall
gravity
3
is instance of...
is instance of...
is instance of...
ball1: BouncingBall
xPosition
10
yPosition
233
ball2: BouncingBall
xPosition
76
yPosition
155
ball3: BouncingBall
xPosition
782
yPosition
33

常量

- 常量在程序执行的过程中，其值不能改变。
  - 常量在其类型名称前用final关键字修饰。
  - 常量必须在声明的时候被初始化。
  - 常量名通常用大写字母命名。
  - 常量对一个类的所有实例都是相等的，因此类中的常量一般声明为类常量。

bouncing.java (project: balls)

```
public class BouncingBall
{
    private static final int GRAVITY = 3;

    private int xPosition;
    private int yPosition;

    ...
}
```

本章小结

- Java具有一个丰富的类库。
- 对于开发人员来说，必须熟悉Java类库的使用方法。
- 不但要能够读懂类库描述，还要会编写自己的类文档。
- 信息隐藏是开发面向对象软件系统的重要原则。
- 通过公开类的接口，隐藏类的实现，可以开发出更具模块化、更易维护的软件系统。

本章涉及到的概念

- 类库 (class library)
- 接口 (interface)
- 实现 (implementation)
- 映射 (map)
- 集合 (set)
- javadoc
- 访问修饰符 (access modifier)
- 信息隐藏 (information hiding)
- 类变量 (class variable)
- static
- 常量 (constant)
- final