高程例题

> 将PPT中直接糊脸的code分离出来，预防考察原题的情况
>
> 大家可以以此文档作为代码补全的练习题

一、Java 入门

1、补全代码，输出 DooBeeDooBeeDo

```
public class DooBee {
    public static void main(String[] args) {
        int x = 1;
        while (x < ____) {
            System.out.____("Doo");
            System.out.____("Bee");
            x = x + 1;
        }
        if (x == ____) {
            System.out.print("Do");
        }
    }
}
```

2、补全 `BeerSong` 类，实现以下功能：

- 从 99 开始递减打印啤酒瓶数的歌词。
- 当啤酒瓶数为 1 时，将 `bottles` 改为 `bottle`。
- 当啤酒瓶数为 0 时，打印结束语句。

```
package chap01;

public class BeerSong {
    public static void main(String[] args) {
        int beerNum = 99;
        String word = "bottles";

        while (beerNum > 0) {
            if (beerNum == 1) {
                word = ____; // 补全：将 "bottles" 改为 "bottle"
```

```
            }

            System.out.println(beerNum + " " + word + " of beer on the wall");
            System.out.println(beerNum + " " + word + " of beer");
            System.out.println("Take one down.");
            System.out.println("Pass it around.");
            beerNum = beerNum - 1;

            if (____) { // 补全：判断是否还有啤酒
                System.out.println(beerNum + " " + word + " of beer on the
wall");
            } else {
                System.out.println("No more bottles of beer on the wall");
            }
            System.out.println();
        }
    }
}
```

3、补全 `FreshJuice` 类和 `FreshJuiceTest` 类，实现以下功能：

- 在 `FreshJuice` 类中定义一个枚举类型 `FreshJuiceSize`，包含 `SMALL`、`MEDIUM` 和 `LARGE`。
- 在 `FreshJuiceTest` 类中创建 `FreshJuice` 对象，并设置其 `size` 属性为 `MEDIUM`，然后打印 `size` 的值。

```
class FreshJuice {
    enum FreshJuiceSize { ____, ____, ____ }
    FreshJuiceSize size;
}

public class FreshJuiceTest {
    public static void main(String[] args) {
        FreshJuice juice = new FreshJuice();
        juice.size = ____;
        System.out.println("Size: " + juice.size);
    }
}
```

二、类与对象

1、补全代码，使其输出

```
helloooo...
helloooo...
helloooo...
helloooo...
10
```

```
public class EchoTestDrive {
    public static void main(String[] args) {
        Echo e1 = new Echo();
        ______________________
        int x = 0;

        while (____) { // 补全：循环条件
            e1.hello();
            ________________________________
            if (____) { // 补全：条件判断
                e2.count = e2.count + 1;
            }
            if (____) { // 补全：条件判断
                e2.count = e2.count + e1.count;
            }
            x = x + 1;
        }
        System.out.println(e2.count);
    }
}

class Echo {
    int count = 0;

    void hello() {
        System.out.println("helloooo...");
    }
}
```

三、写一个程序的过程

1、编写一个方法 `findValue`，接受一个字符串数组和一个目标字符串作为参数，返回目标字符串在数组中的索引。如果目标字符串不存在于数组中，则返回 `-1`。

```
public class ArraySearch {
    public static int findValue(String[] array, String target) {
```

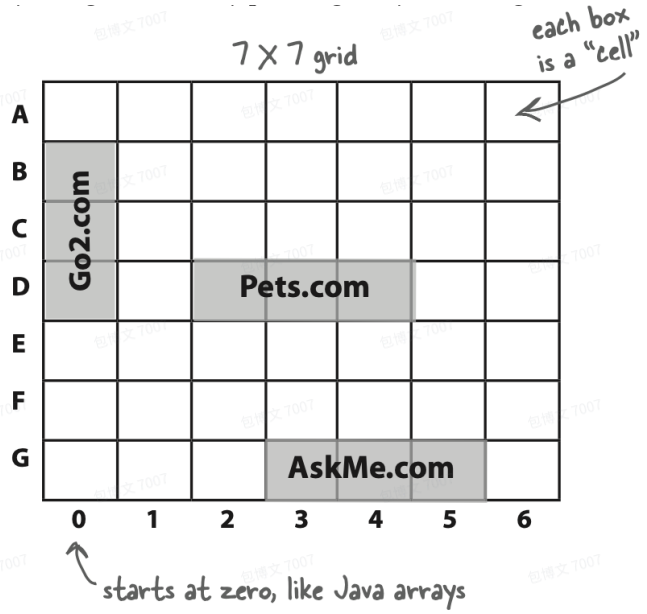
```
        // 补全代码
    }

    public static void main(String[] args) {
        String[] arrayA = {"Hello", "world", "Welcome", "to", "the", "class"};
        String target = "Welcome";
        int index = findValue(arrayA, target);
        if (index != -1) {
            System.out.println("Found at index: " + index);
        } else {
            System.out.println("Not found");
        }
    }
}
```

2、编写两个方法 `max1` 和 `max2`，分别使用不同的方式找到数组中的最大值。

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;

public class ArrayMax {
    // 方法 1：使用 for 循环
    private static void max1(double[] decMax) {
        // 补全代码
    }

    // 方法 2：使用 Collections.max
    private static void max2(Double[] decMax) {
        // 补全代码
    }

    public static void main(String[] args) {
        double[] decMax = {-2.8, -8.8, 2.3, 7.9, 4.1, -1.4, 11.3, 10.4, 8.9, 8.1, 5.8, 5.9, 7.8, 4.9, 5.7, -0.9, -0.4, 7.3, 8.3, 6.5, 9.2, 3.5, 3, 1.1, 6.5, 5.1, -1.2, -5.1, 2, 5.2, 2.1};

        max1(decMax);
        max2(decMax);
    }
}
```

3、请用**测试驱动开发**的思想实现下面游戏程序的开发

- 初始化一个 7x7 的格子
- 上面有3个.com 网站，每个网站占3格
- 用户输入"A3" 或 "C5"这样的坐标
- 计算机反馈给你"Hit"或"Miss"
- 如果3格全被打掉，返回"Sunk"
- 当清光所有 .com网站，游戏列出你的分数并结束




```
public class SimpleDotComTestDrive {

    public static void main (String[] args) {
        SimpleDotCom dot = new SimpleDotCom();
        int[] locations = {2,3,4};
        dot.setLocationCells(locations);
        String userGuess = "2";
        String result = dot.checkYourself(userGuess);
        String testResult = "failed";
        if (result.equals("hit") ) {
            testResult = "passed";
        }
        System.out.println(testResult);
    }

}
```

instantiate a SimpleDotCom object

make an int array for the location of the dot com (3 consecutive ints out of a possible 7).

invoke the setter method on the dot com.

make a fake user guess

invoke the checkYourself() method on the dot com object, and pass it the fake guess.

if the fake guess (2) gives back a "hit", it's working

Print out the test result (passed or failed)

```
public static void main(String[] args) {
    int numOfGuesses = 0;
    GameHelper helper = new GameHelper();
    SimpleDotCom theDotCom = new SimpleDotCom();
    int randomNum = (int) (Math.random() * 5);
    int[] locations = {randomNum, randomNum+1, randomNum+2};
    theDotCom.setLocationCells(locations);
    boolean isAlive = true;
    while(isAlive == true) {
        String guess = helper.getUserInput("enter a number");
        String result = theDotCom.checkYourself(guess);
        numOfGuesses++;
        if (result.equals("kill")) {
            isAlive = false;
            System.out.println("You took " + numOfGuesses + " guesses");
        } // close if
    } // close while
} // close main
```

DECLARE a variable to hold user guess count, set it to 0

make a variable to track how many guesses the user makes

MAKE a SimpleDotCom object

this is a special class we wrote that has the method for getting user input. for now, pretend it's part of Java

make the dot com object

COMPUTE a random number between 0 and 4

make a random number for the first cell, and use it to make the cell locations array

MAKE an int array with the 3 cell locations, and

give the dot com its locations (the array)

INVOKE setLocationCells on the dot com object

DECLARE a boolean isAlive

make a boolean variable to track whether the game is still alive, to use in the while loop test

WHILE the dot com is still alive

repeat while game is still alive.

GET user input

get user input String

// CHECK it

INVOKE checkYourself() on dot com

ask the dot com to check the guess; save the returned result in a String

INCREMENT numOfGuesses

increment guess count

IF result is "kill"

was it a "kill"? if so, set isAlive to false (so we won't re-enter the loop) and print user guess count

SET gameAlive to false

PRINT the number of user guesses

1. 高层设计
2. 写伪代码（prep code）
3. 基于伪代码写出对应的测试码（test code）
4. 最后迭代得到real code

四、继承与多态

1、以下是三个类的定义：

```
1   public class Doctor {
```

```
    boolean worksAtHospital;
    void treatment() {
        // 进行检查
    }
}

public class FamilyDoctor extends Doctor {
    boolean makesHouseCalls;
    void giveAdvice() {
        // 提供建议
    }
}

public class Surgeon extends Doctor {
    void treatPatient() {
        // 进行手术
    }
    void makeIncision() {
        // 进行切口（哎呀！）
    }
}
```

请回答以下问题：

1. `Surgeon` 类有多少个实例变量？
2. `FamilyDoctor` 类有多少个实例变量？
3. `Doctor`、`Surgeon` 和 `FamilyDoctor` 类分别有多少个方法？
4. `FamilyDoctor` 能否调用 `treatPatient()` 方法？
5. `FamilyDoctor` 能否调用 `makeIncision()` 方法？

2、补全代码中的空白部分，使其输出 `drift drift hoist sail`

```
public class Rowboat __________ {
    public ______ rowTheBoat() {
        System.out.print("stroke natasha");
    }
}

public class ______ {
    private int ______;
    public void ______ ( ______ ) {
        length = len;
```

```
    }
    public int getLength() {
        _______
    }
    public ______ move() {
        System.out.print("_____");
    }
}

public class TestBoats {
    ____________ main(String[] args) {
        ______ b1 = new Boat();
        Sailboat b2 = new ______();
        Rowboat b3 = new Rowboat();
        b2.setLength(32);
        b1._______( );
        b3._______( );
        _______.move();
    }
}

public class ______ ______ Boat {
    public ______ ______() {
        System.out.print("_____");
    }
}
```

3、回答相关问题：

1. 运行程序后，输出的结果是什么？请解释每行输出的含义。
2. `Acts` 类中的 `iMethod()` 方法返回的值是多少？为什么？
3. `Clowns` 类没有实现 `iMethod()` 方法，为什么它仍然可以调用 `iMethod()` 方法？

```
public class Off6 extends Clowns {
    public static void main(String[] args) {
        Nose[] i = new Nose[3];
        i[0] = new Acts();
        i[1] = new Clowns();
        i[2] = new Off6();

        for (int x = 0; x < 3; x++) {
            System.out.println(i[x].iMethod() + " " + i[x].getClass());
        }
    }
```

```
}

interface Nose {
    public int iMethod();
}

abstract class Picasso implements Nose {
    public int iMethod() {
        return 7;
    }
}

class Clowns extends Picasso {
}

class Acts extends Picasso {
    public int iMethod() {
        return 5;
    }
}
```

4、补全代码，使其通过父类的 `runReport()` 方法来设置报告

```
abstract class Report {
    void runReport() { // set-up report
    }
    void printReport() { // generic printing
    }
}

class BuzzwordsReport extends Report {
    void runReport() {
        ____________________;
        buzzwordCompliance();
        printReport();
    }
    void buzzwordCompliance() { ... }
}
```

五、正则

1、给出匹配 Email 的正则表达式

- 匹配：`user.name@example.com`，`user+name@example.co.uk`

- 不匹配：`user@.com`，`user@com`

2、给出匹配国内固话号码和手机号码的正则表达式

- 匹配：`010-12345678`，`13812345678`
- 不匹配：`123-4567890`，`0123-456789`

3、给出匹配 URL 的正则表达式（支持 `http://`，`https://`，`ftp://` 以及无前缀）

- 匹配：`http://example.com`，`https://www.example.com/path`，`ftp://ftp.example.com`
- 不匹配：`example.com`，`http://.com`

4、给出匹配 IPv4 地址的正则表达式

- 匹配：`192.168.1.1`，`10.0.0.1`
- 不匹配：`256.256.256.256`，`192.168.1`

六、重构

1、埃拉托斯特尼筛法

（1）请根据代码逻辑填空，完成埃拉托斯特尼筛法的实现。

```
public class Main {
    public static void main(String[] args) {
        if (args.length == 0) {
            System.out.println("请提供一个整数作为参数。");
            return;
        }

        int rangeOfPrimes = Integer.parseInt(args[0]);
        short i = 0, width = 11;

        for (Integer p : PrimeGenerator.generatePrimes(rangeOfPrimes)) {
            System.out.printf("%4d\t", p);
            i++;
            if (i >= width) {
                i = 0;
                System.out.println();
            }
        }
```

```
19          System.out.println();
20      }
21  }
22
23  public class PrimeGenerator {
24      private static boolean[] isCrossed;
25      private static int[] result;
26
27      public static int[] generatePrimes(int maxValue) {
28          if (maxValue < 2)
29              return new int[0];
30          else {
31              initializeArrayOfIntegers(maxValue);
32              crossOutMultiples();
33              return putUncrossedIntegersIntoResult();
34          }
35      }
36
37      private static void initializeArrayOfIntegers(int maxValue) {
38          isCrossed = new boolean[maxValue + 1];
39          for (int i = 2; i < isCrossed.length; i++)
40              isCrossed[i] = false;
41      }
42
43      private static void crossOutMultiples() {
44          int maxPrimeFactor = calcMaxPrimeFactor();
45          for (int i = 2; i <= maxPrimeFactor; i++) {
46              if (notCrossed(i))
47                  _____;
48          }
49      }
50
51      private static int calcMaxPrimeFactor() {
52          double maxPrimeFactor = Math.sqrt(isCrossed.length) + 1;
53          return (int) maxPrimeFactor;
54      }
55
56      private static void crossOutMultiplesOf(int i) {
57          for (int multiple = 2 * i; multiple < isCrossed.length; multiple += i)
58              _____;
59      }
60
61      private static boolean notCrossed(int i) {
62          return _____;
63      }
64
65      private static int[] putUncrossedIntegersIntoResult() {
```

```
66          int count = numberOfUncrossedIntegers();
67          result = new int[count];
68          for (int j = 0, i = 2; i < isCrossed.length; i++) {
69              if (notCrossed(i))
70                  _____;
71          }
72          return result;
73      }
74
75      private static int numberOfUncrossedIntegers() {
76          int count = 0;
77          for (int i = 2; i < isCrossed.length; i++) {
78              if (notCrossed(i))
79                  _____;
80          }
81          return count;
82      }
83  }
```

（2）请解释以下代码的作用：

```
1   private static int calcMaxPrimeFactor() {
2       double maxPrimeFactor = Math.sqrt(isCrossed.length) + 1;
3       return (int) maxPrimeFactor;
4   }
```

（3）请分析埃拉托斯特尼筛法的时间复杂度，并解释为什么是 `O(n log log n)`。

（4）请列举至少三种边界条件，并说明 `PrimeGenerator` 类在这些情况下的预期行为。

八、List，Set，Map

代码填空，使其能够产生如下输出：

```
1   as entered:
2   [Longs 14255, Elbert 14433, Maroon 14156, Castle 14265]
3   by name:
4   [Castle 14265, Elbert 14433, Longs 14255, Maroon 14156]
5   by height:
6   [Elbert 14433, Castle 14265, Longs 14255, Maroon 14156]
```

```
import ______;

public class SortMountains {

    LinkedList ______ mtn = new LinkedList______();

    class NameCompare _______________________________ {
        public int compare(Mountain one, Mountain two) {
            return ________________________;
        }
    }

    class HeightCompare _______________________________ {
        public int compare(Mountain one, Mountain two) {
            return ________________________; // 按高度降序排序
        }
    }

    public static void main(String[] args) {
        new SortMountains().go();
    }

    public void go() {
        mtn.add(new Mountain("Longs", 14255));
        mtn.add(new Mountain("Elbert", 14433));
        mtn.add(new Mountain("Maroon", 14156));
        mtn.add(new Mountain("Castle", 14265));

        System.out.println("as entered:\n" + mtn);

        NameCompare nc = new NameCompare();
        ______________________________;
        System.out.println("by name:\n" + mtn);

        HeightCompare hc = new HeightCompare();
        ______________________________;
        System.out.println("by height:\n" + mtn);
    }
}

class Mountain {
    ____________;
    ____________;

    _______________________________ {
        ___________________;
        ___________________;
```

```
    }

    _____________________________ {
        ___________________;
    }
}
```

九、静态类与方法

1、代码纠错

```
class Duck {
    int duckCount = 0;

    public Duck() {
        duckCount++; // 每次创建对象时，duckCount 自增
    }
}
```

2、如果class已经是final的，再标记final的方法是不是多余?

3、下面的代码中哪一个是合法的?

```
public class Foo {
    static int x;
    public void go() {
        System.out.println(x);
    }
}

public class Foo2 {
    int x;
    public static void go() {
        System.out.println(x);
    }
}

public class Foo3 {
    final int x;
    public void go() {
        System.out.println(x);
    }
}
```

```
21
22  public class Foo4 {
23      static final int x = 12;
24      public void go() {
25          System.out.println(x);
26      }
27  }
28
29  public class Foo5 {
30      static final int x = 12;
31      public void go(final int x) {
32          System.out.println(x);
33      }
34  }
35
36  public class Foo6 {
37      int x = 12;
38      public static void go(final int x) {
39          System.out.println(x);
40      }
```

4、找出字符串中的格式化符(format specifier) 的正则怎么写?

5、如何通过java.util.calendar测试到春节还有几周?

部分参考答案（PPT原话）

3.1 找到数组中具体的某个值

```
1  public static int findValue(String[] array, String target) {
2      for (int i = 0; i < array.length; i++) {
3          if (array[i].equals(target)) {
4          return i;
5          }
6      }
7      return -1;
8  }
```

- 主要关注java数组本身的方法，`.length`、`.equals`等

3.2 找到数组中的最大值

```java
// 方法 1: 使用 for 循环
private static void max1(double[] decMax) {
    double max = decMax[0];
    for (int counter = 1; counter < decMax.length; counter++) {
        if (decMax[counter] > max) {
            max = decMax[counter];
        }
    }
    System.out.println("The highest maximum for the December is: " + max);}
// 方法 2: 使用 Collections.max
private static void max2(Double[] decMax) {
    List<Double> a = new ArrayList<>(Arrays.asList(decMax));
    System.out.println("The highest maximum for the December is: " +
Collections.max(a));
}
```

4.1 多态

1. **`Surgeon` 类有多少个实例变量?**
   - 答：1 个（继承自 `Doctor` 类的 `worksAtHospital`）。
2. **`FamilyDoctor` 类有多少个实例变量?**
   - 答：2 个（继承自 `Doctor` 类的 `worksAtHospital`，以及自身的 `makesHouseCalls`）。
3. **`Doctor`、`Surgeon` 和 `FamilyDoctor` 类分别有多少个方法?**
   - `Doctor`：1 个（`treatment()`）。
   - `Surgeon`：3 个（继承的 `treatment()`，以及自身的 `treatPatient()` 和 `makeIncision()`）。
   - `FamilyDoctor`：2 个（继承的 `treatment()`，以及自身的 `giveAdvice()`）。
4. **`FamilyDoctor` 能否调用 `treatPatient()` 方法?**
   - 答：不能，`treatPatient()` 是 `Surgeon` 类的方法，`FamilyDoctor` 没有继承或实现该方法。
5. **`FamilyDoctor` 能否调用 `makeIncision()` 方法?**
   - 答：不能，`makeIncision()` 是 `Surgeon` 类的方法，`FamilyDoctor` 没有继承或实现该方法。

4.2 多态

```
public class Rowboat extends Boat {
    public void rowTheBoat() {
        System.out.print("stroke natasha");
    }
}

public class Boat {
    private int length;
    public void setLength(int len) {
        length = len;
    }
    public int getLength() {
        return length;
    }
    public void move() {
        System.out.print("drift");
    }
}

public class TestBoats {
    public static void main(String[] args) {
        Boat b1 = new Boat();
        Sailboat b2 = new Sailboat();
        Rowboat b3 = new Rowboat();
        b2.setLength(32);
        b1.move();
        b3.move();
        b2.move();
    }
}

public class Sailboat extends Boat {
    public void move() {
        System.out.print("hoist sail");
    }
}
```

4.3 接口

```
5 class Acts
7 class Clowns
7 class Off6
```

- **第一行：** `Acts` 类的 `iMethod()` 返回 `5`，类名为 `Acts`。

- **第二行**：`Clowns` 类继承 `Picasso` 类的 `iMethod()`，返回 `7`，类名为 `Clowns`。
- **第三行**：`Off6` 类继承 `Clowns` 类，因此也继承 `Picasso` 类的 `iMethod()`，返回 `7`，类名为 `Off6`。

1. `Acts` **类的** `iMethod()`：
   - 返回 `5`，因为 `Acts` 类重写了 `Picasso` 类的 `iMethod()` 方法。
2. `Clowns` **类的** `iMethod()`：
   - `Clowns` 类继承 `Picasso` 类，而 `Picasso` 类实现了 `iMethod()` 方法，因此 `Clowns` 类可以直接调用 `iMethod()` 方法。

4.4 父类

`super.runReport()`

5.1 邮箱正则

```
^ \w+ ([-+.]\w+)* @ \w+ ([-.]\w+)* \. \w+ ([-.]\w+)* $
```

5.2 电话正则

```
^(0\d{2,3}-\d{7,8}|1[3456789]\d{9})$
```

5.3 URL正则

```
^(https?|ftp):\/\/[^\s/$.?#].[^\s]*$

https://[服务器地址]:[端口号]/[文件路径][文件名]
^https:\/\/[^\s/:]+(?::\d+)?(?:\/[^\s?#]*)?(?:\?[^\s#]*)?(?:#\S*)?$
```

5.4 IPv4正则

```
^((25[0-5]|2[0-4]\d|1\d{2}|[1-9]?\d)\.){3}(25[0-5]|2[0-4]\d|1\d{2}|[1-9]?\d)$
```

6.1 素数产生程序

1. `crossOutMultiplesOf(i)`
2. `isCrossed[multiple] = true`
3. `!isCrossed[i]`
4. `result[j++] = i`
5. `count++`

这段代码用于计算筛法的最大素数因子。根据埃拉托斯特尼筛法的原理，筛选素数时只需要考虑小于等于 `sqrt(n)` 的素数的倍数。因此，`calcMaxPrimeFactor` 方法返回 `isCrossed.length` 的平方根加 1 的整数值，作为筛选的上限。

埃拉托斯特尼筛法的时间复杂度为 `O(n log log n)`，原因如下：

1. 外层循环遍历从 2 到 `sqrt(n)` 的所有数，时间复杂度为 `O(sqrt(n))`。
2. 内层循环标记每个素数的倍数，对于每个素数 `p`，需要标记 `n/p` 个倍数。
3. 根据调和级数和素数定理，总操作次数约为 `n * (1/2 + 1/3 + 1/5 + ... + 1/p)`，其中 `p` 是小于等于 `sqrt(n)` 的素数。
4. 该级数的和近似为 `log log n`，因此总时间复杂度为 `O(n log log n)`。

1. **输入为负数**：`maxValue < 2`，应返回空数组。
2. **输入为 0 或 1**：`maxValue < 2`，应返回空数组。
3. **输入为 2**：最小的素数，应返回 `[2]`。
4. **输入为较大的数（如 100）**：应返回 2 到 100 之间的所有素数。

8.1 Collection

```
import java.util.*;

public class SortMountains {

    LinkedList<Mountain> mtn = new LinkedList<Mountain>();

    class NameCompare implements Comparator<Mountain> {
        public int compare(Mountain one, Mountain two) {
            return one.name.compareTo(two.name);
        }
    }

    class HeightCompare implements Comparator<Mountain> {
        public int compare(Mountain one, Mountain two) {
```

```
            return two.height - one.height; // 按高度降序排序
        }
    }

    public static void main(String[] args) {
        new SortMountains().go();
    }

    public void go() {
        mtn.add(new Mountain("Longs", 14255));
        mtn.add(new Mountain("Elbert", 14433));
        mtn.add(new Mountain("Maroon", 14156));
        mtn.add(new Mountain("Castle", 14265));

        System.out.println("as entered:\n" + mtn);

        NameCompare nc = new NameCompare();
        Collections.sort(mtn, nc);
        System.out.println("by name:\n" + mtn);

        HeightCompare hc = new HeightCompare();
        Collections.sort(mtn, hc);
        System.out.println("by height:\n" + mtn);
    }
}

class Mountain {
    String name;
    int height;

    Mountain(String name, int height) {
        this.name = name;
        this.height = height;
    }

    @Override
    public String toString() {
        return name + " " + height;
    }
}
```

9.1 静态变量

`duckCount` 是实例变量，每个 `Duck` 对象都有自己的 `duckCount`，因此每次创建新对象时，`duckCount` 都会被初始化为 `0`，然后自增为 `1`，无法正确统计所有 `Duck` 对象的数量。

如果希望统计所有 `Duck` 对象的数量，应该将 `duckCount` 声明为静态变量。

9.2 final

final修饰的类，其属性和方法不是被final修饰的。

也就是说，final定义的类，其中的属性、方法不是final的。

可通过下面代码验证：

```
package question;

import java.lang.reflect.Field;
import java.lang.reflect.Method;
import java.lang.reflect.Modifier;

public class Question1 {

    public static void reflection(Class clazz) {

        System.out.println("------属性------");
        Field[] fields = clazz.getDeclaredFields();
        for(Field f : fields) {
            int iModifiers = f.getModifiers();
            String sModifiers = Modifier.toString(iModifiers);
            //打印所有属性的修饰符及属性名
            System.out.println(sModifiers+" "+f.getName());
        }

        System.out.println("------方法------");
        Method[] methods = clazz.getMethods();
        for(Method m: methods) {
            int iModifiers = m.getModifiers();
            String sModifiers = Modifier.toString(iModifiers);
            //打印所有方法修饰符及方法名
            System.out.println(sModifiers+" "+m.getName());
        }
    }

    public static void main(String[] args) {
        reflection(Person.class);
    }

}

final class Person {

    private String name;
    private String age;
```

```
40      private final String idCard;
41
42      Person(String idCard) {
43          this.idCard = idCard;
44      }
45
46      public static void m(){};
47      public void m1(String s){};
48      public final void m2(){};
49
50  }
```

9.3 final

Foo

- 静态变量 `x` 是合法的。
- 方法 `go()` 是实例方法，可以直接访问静态变量。
- **合法**

Foo2

- `x` 是实例变量。
- 静态方法 `go()` 中不能直接引用非静态变量 `x`（因为静态方法无法访问实例变量）。
- **不合法**（编译报错）

Foo3

- `x` 是 `final` 的实例变量，但它未被初始化（`final` 的实例变量必须在声明时或通过构造器初始化）。
- **不合法**（编译报错）

Foo4

- `x` 是 `static final` 的常量，已正确初始化。
- 方法 `go()` 是实例方法，可以访问静态变量 `x`。
- **合法**

Foo5

- 类中有一个 `static final` 的常量 `x`。
- 方法 `go(final int x)` 中，局部变量 `x` 屏蔽了类的 `x`，这是合法的。
- **合法**

Foo6

- `x` 是实例变量。
- 静态方法 `go(final int x)` 。
- **合法**

9.4 regex

```
1  import java.util.regex.Matcher;
2  import java.util.regex.Pattern;
3
4  public class FormatSpecifierFinder {
5      public static void main(String[] args) {
6          String input = "Name: %s, Age: %d, Height: %.2f, Score: %+05d, Align:
   %-10s";
7          String regex = "%[+-]?\\d*(\\.\\d+)?[bcdefgosx]";
8
9          Pattern pattern = Pattern.compile(regex);
10         Matcher matcher = pattern.matcher(input);
11
12         while (matcher.find()) {
13             System.out.println("Found format specifier: " + matcher.group());
14         }
15     }
16 }
```

- `%`：匹配百分号，表示格式化符的开始。
- `[+-]?`：可选的符号（`+` 或 `-`），用于对齐或显示正负号。
- `\d*`：可选的数字，表示宽度（如 `10` 在 `%10s` 中）。
- `(\.\d+)?`：可选的小数部分，表示精度（如 `.2` 在 `%.2f` 中）。
- `[bcdefgosx]`：匹配格式化符的类型（如 `s`、`d`、`f` 等）。

9.5 Calendar

1. **获取当前日期**：
   - 使用 `Calendar.getInstance()` 获取当前日期。
2. **设置春节日期**：
   - 使用 `Calendar.set()` 方法设置春节的阳历日期。
   - 如果春节已经过去，将年份加 1，计算下一年的春节。
3. **计算差值**：

- 使用 `getTimeInMillis()` 获取当前日期和春节日期的毫秒数。
- 计算毫秒数差值，并将其转换为天数。

4. **转换为周数**：
   - 将天数除以 7，得到周数。

```
import java.util.Calendar;

public class SpringFestivalCountdown {
    public static void main(String[] args) {
        // 获取当前日期
        Calendar today = Calendar.getInstance();

        // 设置春节日期（假设 2024 年春节是 2 月 10 日）
        Calendar springFestival = Calendar.getInstance();
        springFestival.set(Calendar.YEAR, 2024); // 设置年份
        springFestival.set(Calendar.MONTH, Calendar.FEBRUARY); // 设置月份（2 月）
        springFestival.set(Calendar.DAY_OF_MONTH, 10); // 设置日期（10 日）

        // 如果春节已经过去，设置为下一年的春节
        if (today.after(springFestival)) {
            springFestival.set(Calendar.YEAR, today.get(Calendar.YEAR) + 1);
        }

        // 计算当前日期与春节日期之间的毫秒数差值
        long diffMillis = springFestival.getTimeInMillis() - today.getTimeInMillis();

        // 将毫秒数转换为天数
        long diffDays = diffMillis / (1000 * 60 * 60 * 24);

        // 将天数转换为周数
        long diffWeeks = diffDays / 7;

        // 输出结果
        System.out.println("距离春节还有 " + diffWeeks + " 周");
    }
}
```