Java Lab4: 讲题

- 题目讲解

俞心如
51275901033@stu.ecnu.edu.cn

关于Debug：

- 仔细查看报错信息，可以问问AI助手
- 手动调试：`System.out.println()`
- IDE调试：打断点

使用String：第2题

2. 给定一个网址字符串，根据 "." 将它分解成字串. 例如 http://www.ecnu.edu.cn 分解成为 http://www, ecnu, edu, cn. (提示: 使用 split() 方法)

- 很简单，按照提示直接用split() 方法

```
public class SplitString {
    public static void main(String []args)
    {
        String []str=args[0].split(".");
        for(String temp:str) System.out.println(temp);
    }
}
```

- 跑起来之后发现跟预期想的不太一样。

使用String：第2题

```
public class SplitString {
    public static void main(String []args)
    {
        String []str=args[0].split("\\.");
        for(String temp:str) System.out.println(temp);
    }
}
```

- String.split() 方法的参数是一个正则表达式，而不是普通字符串。这意味着传入的分隔符会被当作正则表达式来处理，而不是简单的字符匹配。
- 我们希望按照普通的点号 `.` 来拆分字符串，需要对点号进行转义。在正则表达式中，转义特殊字符需要使用反斜杠 `\`，但在 Java 字符串中，反斜杠本身也需要转义，因此需要写成 `\\.`

使用String：第7题

7、 给定一个字符串，代表一个16进制数．将其转换成10进制整数，输出到标准输出．

- 我们可能会依次读入字符将16进制转为10进制
- Java的String类里面有现成的将16进制字符串转换为10进制整数的方法

```
import java.io.*;
import java.util.Scanner;
public class Htod{
public static void main(String []args)
{
    Scanner in = new Scanner(System.in);
    String hex = in.nextLine();
    long dec = Long.parseLong(hex, 16);
    System.out.println(dec);
    in.close();
    }
}
```

使用String：第11题

11. 编写程序 Print.java, 它有以下命令行选项, 根据不同的选项得到不同的运行结果.

选项	用法举例	说明
-t	`java Print -t type`	若 type=n 则输出0到9的数字, type=a 则输出a到z的字母, 默认 type=n (即不带 -t 选项执行 `java Print` 将输出数字)
-o	`java Print -o out.txt`	输出到文件out.txt. 默认输出到标准输出
-h	`java Print -h`	输出帮助信息到标准输出, 不输出其他信息

例如 `java Print -t a -o a.txt` 将输出 a 到 z 到文件 a.txt. `java Print -o b.txt` 输出数字0到9到 b.txt. `java Print -t a` , 将输出 a 到 z 到标准输出. `java Print -h` 输出的帮助信息为

```
usage: % java Print [OPTIONS]
-t type       if type=n print 0-9, if type=a print a-z. Default: type=n
-o out.txt    outputs to out.txt, Default: standard out
-h            print this help informantion
```

除了以上列出的三种选项, 如果输入其他的选项将输出错误信息 "Wrong options", 随后打印帮助信息并退出.

- 这道题类似于C语言里面读入字符串公式计算数字答案差不多。我们依次读入命令行参数分别处理

图像处理

- **数字图像**

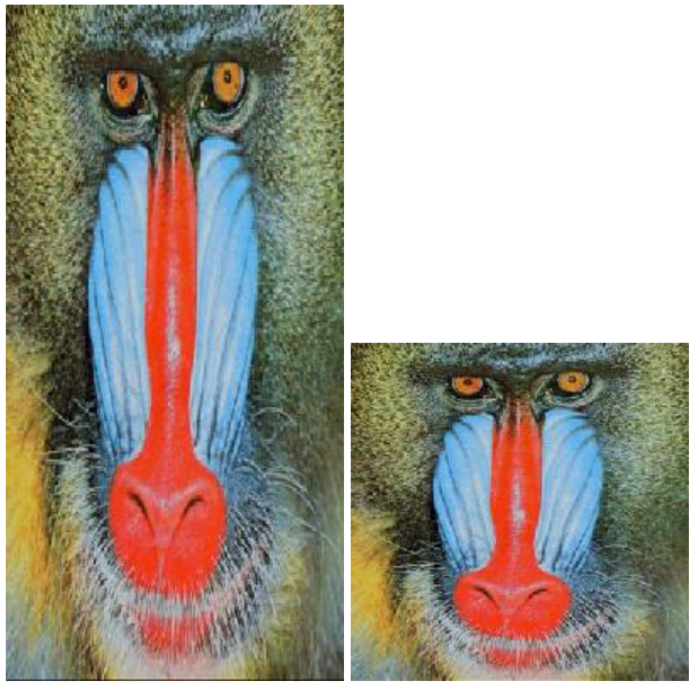
  > 一幅数字图像 (digital image) 在计算机内部被表示成像素矩阵 (pixel matrix). 每一个像素都有自己的颜色. 类比来说, 每个像素好比拼图玩具中的一块, 它们单个看起来都只是简单的色块, 但不同的块组合在一起就能构成完整的图画.

  - 所以本质上我们只是在处理一堆数字堆成的矩阵而已。

图像处理: 第3题

3. 给定一个图片文件，以及参数 `w`，`h`，将其转换成为宽为 `w`，高为 `h` 的图片

- 插值算法（最近邻插值、双线性插值、双三次插值）
- 说白了就是新图片与原来的图片之间的像素位置的映射



图像处理: 第4题

4、 给定一个图片文件，以及参数N．输出 N 张图片，其中第n张图片为从输入图片和其对应灰度图的一个渐变图．

> 1. 颜色的亮度表征了显示该颜色时需要使用多少强度的光. 直观上亮度越大, 该颜色越明亮.
>
>   严格的定义如下: $亮度 = 0.299 * r + 0.587 * g + 0.114 * b$, 其中, r, g, b分别代表该颜色的红, 绿, 蓝强度.
> 2. 图片的灰度图可以通过如下方法得到: 将图中每个像素点的颜色置为灰色, 且这些灰色的灰度值等于该颜色亮度.

- 灰色的定义已经给出
- 渐变？在灰度图和原图之间进行插值

图像处理: 第7题

7、 滤镜(`filters`)可以看作对像素矩阵的某种变换. 通过添加滤镜, 我们可以改变图片的视觉效果.

- Linear filter: 每一个像素的颜色变为周围 9 个像素点(包含它本身)颜色的平均值. 这个变换等价于将矩阵 $\begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}$ 与 9个像素点做 卷积 (Convolution) . 而这个矩阵被称为卷积矩阵 (convolutional matrix)1, 2. —— 对图像进行模糊处理, 减少图像的锐度
- Blur filter: 卷积矩阵为 $\begin{bmatrix} \frac{1}{13}, \frac{1}{13}, \frac{1}{13} \\ \frac{1}{13}, \frac{5}{13}, \frac{1}{13} \\ \frac{1}{13}, \frac{1}{13}, \frac{1}{13} \end{bmatrix}$ —— 进一步模糊图像, 比线性滤镜产生更柔和的效果
- Emboss filter: 卷积矩阵为 $\begin{bmatrix} -1, 0, 1 \\ -1, 1, 1 \\ -1, 0, 1 \end{bmatrix}$, 或者 $\begin{bmatrix} 1, 0, -1 \\ 2, 0, -2 \\ 1, 0, -1 \end{bmatrix}$ 或者 $\begin{bmatrix} -1, -1, 0 \\ -1, 1, 1 \\ 0, 1, 1 \end{bmatrix}$ —— 增强边缘的立体感
- Sharpen filter: 卷积矩阵为 $\begin{bmatrix} 0, -1, 0 \\ -1, 5, -1 \\ 0, -1, 0 \end{bmatrix}$ —— 增强图像的边缘和细节, 使图像看起来更清晰
- Oil painting filter: 给定参数 w, 将每个像素 (i, j) 的颜色替换为所有与 (i, j) *Manhattan 距离* 小于 w 的像素点中出现次数最多的颜色. —— 模拟油画的视觉效果

图像处理: 第7题

7、 滤镜(filters)可以看作对像素矩阵的某种变换. 通过添加滤镜, 我们可以改变图片的视觉效果.

```
 for (int col = 0; col < originalPicture.width(); col++) {
    for (int row = 0; row < originalPicture.height(); row++) {
        double R = 0, G = 0, B = 0;
        for (int i = -1; i <= 1; i++) {
            for (int j = -1; j <= 1; j++) {
                int x = col + i;
                int y = row + j;
                if (x >= 0 && x < originalPicture.width() && y >= 0 && y < originalPicture.height()) {
                    Color color = originalPicture.get(x, y);
                    R += embossKernel[i + 1][j + 1] * color.getRed();
                    G += embossKernel[i + 1][j + 1] * color.getGreen();
                    B += embossKernel[i + 1][j + 1] * color.getBlue();
                }
            }
        }
        int red = (int) (R > 255 ? 255 : (R < 0 ? 0 : R));
        int green = (int) (G > 255 ? 255 : (G < 0 ? 0 : G));
        int blue = (int) (B > 255 ? 255 : (B < 0 ? 0 : B));
        embossPicture.set(col, row, new Color(red, green, blue));
    }
}
```