# 实验 01：使用 Java 读写文件

## 一、实验目的

1、了解 Java I/O 库的基本组成和主要包（`java.io`，`java.nio.file`）。

2、掌握文件读写操作中常见的异常类型的处理（如 `IOException`，`FileNotFoundException`）。

3、理解“流”（Stream）的概念及其在 I/O 操作中的抽象模型。

## 二、实验学时

2 学时

## 三、实验类型

综合性

## 四、实验需求

### 1、硬件

每人配备计算机 1 台，建议优先使用个人计算机开展实验。

实验基于信息技术学院教学容器化云计算平台开展。

### 2、软件

安装 IntelliJ IDEA Community。

项目管理使用 GitLab。

### 3、网络

本地主机能够访问互联网和实验中心网络。

### 4、工具

无

# 五、实验任务

1、完成统计文本文件中每个汉字出现的次数。

2、使用 RandomAccessFile 和 FileChannel 对文件进行随机读写文本文件(从中间读写一段文字)。

3、使用 BufferedImage 比对两个同样大小图片的差异并生成一个新的对比图片框选出差异部分。

# 六、实验内容及步骤

## 1、对文本文件进行内容统计

撰写程序，读取指定的文本文件内容，并完成下面的要求。

其中文本文件自行准备，要求字数不少于 10 万字，建议下载一个 txt 格式的电子书。

**要求实现的功能：**

(1) 使用 BufferedReader 或其它工具类读取文件，可以指定文件编码（如 UTF-8）。

(2) 逐行读取文件内容。

(3) 遍历每一行的每个字符，判断是否为汉字（根据 Unicode 范围）。

(4) 使用 Map（如 HashMap）来存储每个汉字及其出现次数。

(5) 遍历结束后，输出统计结果（可以按出现次数排序后输出）。

统计结果展示格式为：本文件共有 X 字，其中汉字为 Y 字，出现频次最高的 5 个汉字分别是：A（A1 次）、B（B1 次）、C（C1 次）、D（D1 次）、E（E1 次）。

## 2、对文本文件新增内容

撰写程序，在任务 1 的文件文件中，新增指定文字“河南中医药大学智能医学工程专业 2023 级《医疗信息系统开发》课程”。

**要求实现的功能：**

(1) 从中间读取文件内容(RandomAccessFile 用 seek(), FileChannel 用 position())。

(2) 将内容写入文本文件的第 5 段之后。新增的内容为第 6 段。原文的第 6 段及之后内容，自动推后 1 段。

## 3、图片内容对比

撰写程序，对两张图片的内容进行对比，并用红色框线在两张图片中标示出不同之处。

**要求实现的功能：**

(1) 读取两张相似且大小一样图片。图片请自行准备，建议选择一张图片作为图片 1，将图片 1 复制为图片 2 后，对图片 2 进行编辑 1-2 个不同点后保存。

(2) 将每张图片划分为 20x20 的逻辑区域（总共 400 个区域）。

(3) 对于每个区域，计算两张图片对应区域的差异（可以比较像素的 RGB 值）。

(4) 设定一个阈值，如果差异超过阈值，则认为该区域有差别。

(5) 标记出有差别的区域。效果如图 1-2 所示。

1-1 图片对比实现效果

**参考代码：**

Java

```java
import javax.imageio.ImageIO;
import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

public class ImageDifferenceFinder {

    public static void main(String[] args) {
        try {
            // 加载两张图片
            BufferedImage image1 = ImageIO.read(new File("D:\\temp\\1.jpg"));
            BufferedImage image2 = ImageIO.read(new File("D:\\temp\\2.jpg"));

            // 确保图片尺寸相同
            if (image1.getWidth() != image2.getWidth() || image1.getHeight() != image2.getHeight()) {
                System.out.println("图片尺寸不一致，无法比较");
                return;
            }

            // 设置网格大小
            int gridRows = 20;
            int gridCols = 20;

            // 计算每个区域的大小
            int regionWidth = image1.getWidth() / gridCols;
            int regionHeight = image1.getHeight() / gridRows;

            // 存储有差异的区域
            List<Rectangle> diffRegions = new ArrayList<>();

            // 比较每个区域
            for (int row = 0; row < gridRows; row++) {
                for (int col = 0; col < gridCols; col++) {
                    int startX = col * regionWidth;
                    int startY = row * regionHeight;
                    int endX = Math.min(startX + regionWidth, image1.ge
```

```
tWidth());
                    int endY = Math.min(startY + regionHeight, image1.getHeight());

                    // 检查当前区域是否有差异
                    if (hasDifference(image1, image2, startX, startY, endX, endY)) {
                        diffRegions.add(new Rectangle(startX, startY, endX - startX, endY - startY));
                    }
                }
            }

            // 输出结果
            System.out.println("发现 " + diffRegions.size() + " 个有差异的区域:");
            for (int i = 0; i < diffRegions.size(); i++) {
                Rectangle rect = diffRegions.get(i);
                System.out.println("区域 " + (i + 1) + ": 位置(" + rect.x + "," + rect.y +
                        "), 大小(" + rect.width + "x" + rect.height + ")");
            }

            // 生成标记差异的图片
            BufferedImage resultImage = markDifferences(image1, image2, diffRegions);
            ImageIO.write(resultImage, "jpg", new File("比较结果.jpg"));
            System.out.println("已生成标记差异的图片: 比较结果.jpg");

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    /**
     * 检查指定区域是否有差异
     */
    private static boolean hasDifference(BufferedImage img1, BufferedImage img2,
                                         int startX, int startY, int endX, int endY) {
        // 设置差异阈值（可根据需要调整）
        double threshold = 0.1;
```

```
74          long totalDiff = 0;
75          int pixelCount = 0;
76
77          for (int y = startY; y < endY; y++) {
78              for (int x = startX; x < endX; x++) {
79                  int rgb1 = img1.getRGB(x, y);
80                  int rgb2 = img2.getRGB(x, y);
81
82                  // 计算RGB差异
83                  int r1 = (rgb1 >> 16) & 0xff;
84                  int g1 = (rgb1 >> 8) & 0xff;
85                  int b1 = rgb1 & 0xff;
86
87                  int r2 = (rgb2 >> 16) & 0xff;
88                  int g2 = (rgb2 >> 8) & 0xff;
89                  int b2 = rgb2 & 0xff;
90
91                  int diff = Math.abs(r1 - r2) + Math.abs(g1 - g2) + Mat
    h.abs(b1 - b2);
92                  totalDiff += diff;
93                  pixelCount++;
94              }
95          }
96
97          // 计算平均差异
98          double avgDiff = (double) totalDiff / (pixelCount * 3 * 255);
99          return avgDiff > threshold;
100     }
101
102     /**
103      * 生成标记差异的图片
104      */
105     private static BufferedImage markDifferences(BufferedImage img1, Bu
    fferedImage img2,
106                                                  List<Rectangle> diffRe
    gions) {
107         // 创建结果图片（将两张图片并排显示）
108         int width = img1.getWidth() * 2;
109         int height = Math.max(img1.getHeight(), img2.getHeight());
110         BufferedImage result = new BufferedImage(width, height, Buffere
    dImage.TYPE_INT_RGB);
111         Graphics2D g = result.createGraphics();
112
113         // 绘制原始图片
```

```
114         g.drawImage(img1, 0, 0, null);
115         g.drawImage(img2, img1.getWidth(), 0, null);
116 
117         // 设置绘制样式
118         g.setColor(Color.RED);
119         g.setStroke(new BasicStroke(3));
120 
121         // 标记有差异的区域
122         for (Rectangle rect : diffRegions) {
123             // 在第一张图片上标记
124             g.drawRect(rect.x, rect.y, rect.width, rect.height);
125 
126             // 在第二张图片上标记（偏移一个图片宽度）
127             g.drawRect(rect.x + img1.getWidth(), rect.y, rect.width, re
    ct.height);
128         }
129 
130         g.dispose();
131         return result;
132     }
133 }
```

# 七、实验考核

## 1、本课程实验考核方案

本课程实验考核采用【实验智能评】【实验随堂查】方式开展，根据不同的实验内容选择不同的考核方式。

【实验智能评】：实验完成后提交 GitLab，通过自动化代码评审工具进行评分。

【实验随堂查】：在实验课上通过现场演示的方式向实验指导教师进行汇报，并完成现场问答交流。

## 2、本实验考核要求

本实验考核方式：实验智能评

实验 1-3 作为本课程第 1 次实验考核。

考核要求：

（1）学生通过 GitLab 提交实验成果：{此部分说明需要提交的内容}。

（2）由 GitLab 根据成果和交流情况综合评分。