Stage3 - ImageReader Report

1933586 郑有为

文件说明

以下为 ImageReader 文件夹内容：

```
.
├── bmptest 测试图片集
├── lib
├── ImageProcessorTest.java
├── ImplementImageIO.java
├── ImplementImageProcessor.java
├── MyGrayFilter.java
├── MyRGBFilter.java
├── ImageReader.jar
├── README.md
└── Report.md
```

实现说明

ImageReaderRunner

- 参照要求给出的代码

ImplementImageIO

- **类的说明**：该类继承自 IImageIO 接口，实现了`myRead`和`myWrite`方法，其中`myRead`使用二进制流读入BMP图像并返回Image对象，`myWrite`则是将Image对象转化为BMP格式并保存到指定位置。
- **字节流读取BMP文件**：
  - 参考 BMP 的文件格式我们用14字节的数组储存BitmapFileHeader的信息，这个信息位于BMP文件的前14个字节，再用40字节的数组储存BitmapInfoHeader的信息，这个信息位于BMP文件的紧接着的40个字节，之后的字节便是像素信息。
  - 我们将从BitMapInfoHeader中读出图像的长宽、色彩模式和对齐位数，在程序中分别用变量`width, height, pixelNum, alignNum`表示，其中pixelNum可以是24（代表RGB颜色模式），可以为8（代表灰度图模式），二者在像素字节数上存在区别。

    - 在从字节流读出长度和宽度时，要记得这些数据在字节中大字端排列，我们使用 `ByteBuffer.wrap(biHeight).getInt();`来轻便的获取长宽，实现请参考ImplementImageIO的`getBiWidth`和`getBiHeight`方法。
    - 在读取像素和生Image对象上，我们首先使用一个数组记录所有像素的值，整个过程通过`read()`字节流来实现，并在读取每一行结束后需要跳过alignNum个字节`skip()`，这是由于BMP图像的对齐机制。
- **生成Image对象**：最后使用 ImageProducer 和 ToolKit来生成 Image 对象并返回。

```
ImageProducer producer = new MemoryImageSource(width, height, imageArray, 0, width);
image = Toolkit.getDefaultToolkit().createImage(producer);
```

- **写入并生成BMP文件**：
    - 创建一个 BufferedImage 对象，然后通过 Graphics2D 对象来将 Image 对象"画"到 BufferedImage 对象上。
    - 使用 File 对象创建文件路径，并用 ImageIO 的静态方法`write`将图片数据写入文件。

ImplementImageProcessor

- **类的说明**：该类继承自 IImageProcessor 接口，实现了`showChanelR`、`showChanelB`、`showChanelG`、`showGray`方法，分别产生原图像的三个通道的图像和灰度图像，这几个方法的实现是类似的。
- **生成Image对象**：我们使用 FilteredImageSource 类来创建 ImageProducer，再根据此通过 Toolkit 的 `.createImage` 生成 Image对象。
    - 使用 FilteredImageSource 类需要提供一个过滤器类，也就是我们实现的 MyRGBFilter / MyGrayFilter。

MyRGBFilter / MyGrayFilter

- **类的说明**：这两个类都继承于 RGBImageFilter类，分别处理RGB和灰度图两种情况，它们都要实现 `filterRGB` 方法（该方法提供一种模式来改变每一个色素点），并将`canFilterIndexColorModel`置为`true`
- `filterRGB` **方法实现**：
    - 在MyRGBFilter类中：通过位运算，根据颜色模式不同来返回不同的像素结果，例如R通道，就是返回`rgb & 0xffff0000`。
    - 在MyGrayFilter类中：根据灰度图的生成公式 `c = 0.299 * r + 0.587 * g + 0.114 * b`
    - 当然处理RGB三个通道的值时需要通过位运算。

ImageReaderTest

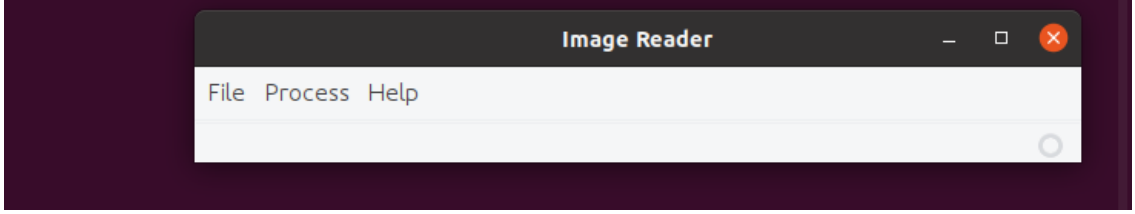
- **类的说明**：基于JUnit单元测试框架，对上述的类进行单元测试，测试内容主要为比较通过程序生成的各通道图片与目标结果图片是否一致。
- **实现思路**：
    - 在构造函数中创建 ImplementImageIO 对象和 ImplementImageProcesso 对象，读入图片`myRead`，并生成RGB三通道和灰度图，再依次调用`myWrite`把这些结果图片写到指定文件位置（bmptest/myresult文件夹）。
    - 测试方法包括：`testRed`、`testBlue`、`testGreen`和`testGray`，每一个方法的实现过程类似
        - 首先是使用ImageIO读入目标生成图片和测试生成图片（用BufferedImage类型保存）；
        - 然后对两张图片进行比较（`isEqual`），如比较一致则通过测试。

- `isEqual`方法：
  - 首先比较两张图片的长宽，若不一致则直接返回`false`。
  - 然后遍历每一个像素，使用BufferedImage提供的`getRGB(i,j)`获取像素值。再一一进行比较。

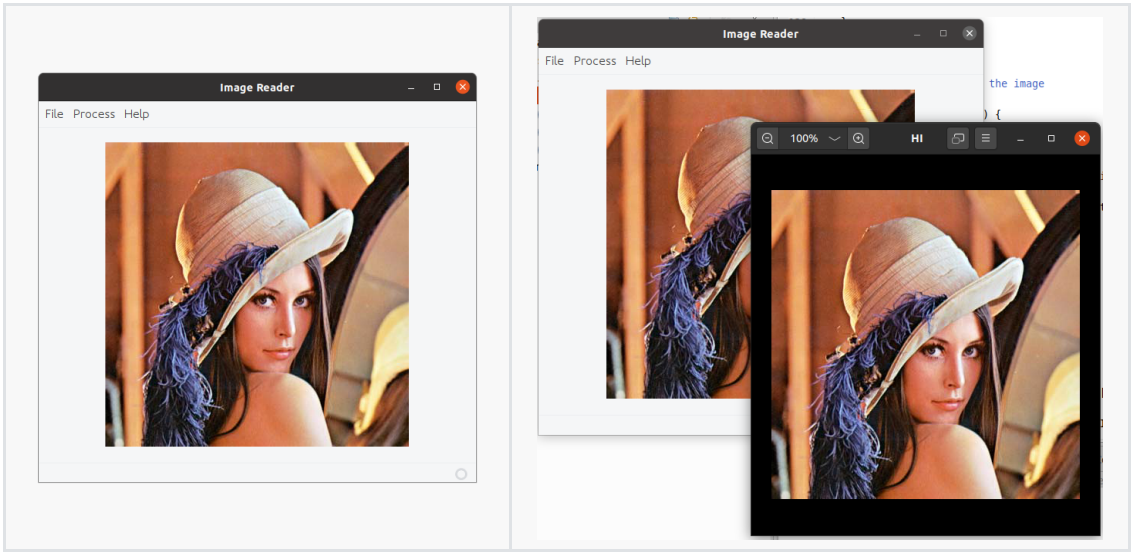
运行效果

- **使用命令行运行程序**：

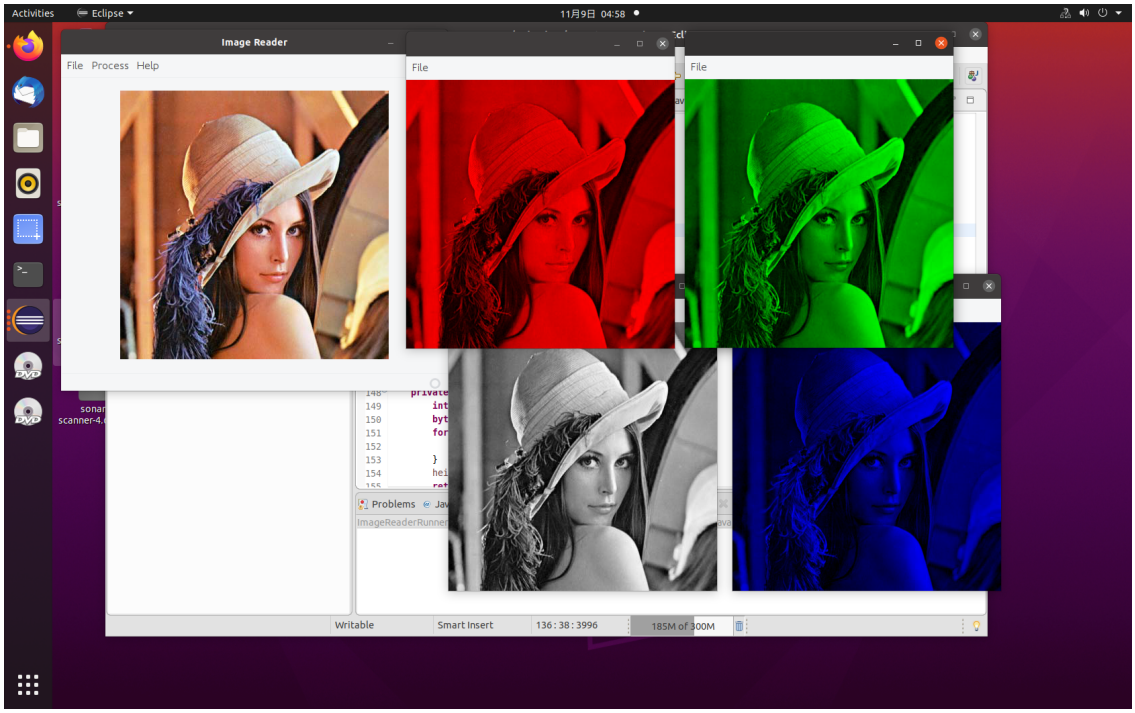
```
max@ubuntu:~/Desktop/se-training/Stage3/ImageReader$ javac -classpath .:ImageRea
der.jar ./MyGrayFilter.java ./MyRGBFilter.java ./ImplementImageIO.java ./Impleme
ntImageProcessor.java ./ImageReaderRunner.java
max@ubuntu:~/Desktop/se-training/Stage3/ImageReader$ java -classpath .:ImageRead
er.jar:./ ImageReaderRunner
Gtk-Message: 03:49:43.791: Failed to load module "canberra-gtk-module"
```



- **读入图片并保存图片**：（右图为打开保存后的图片）

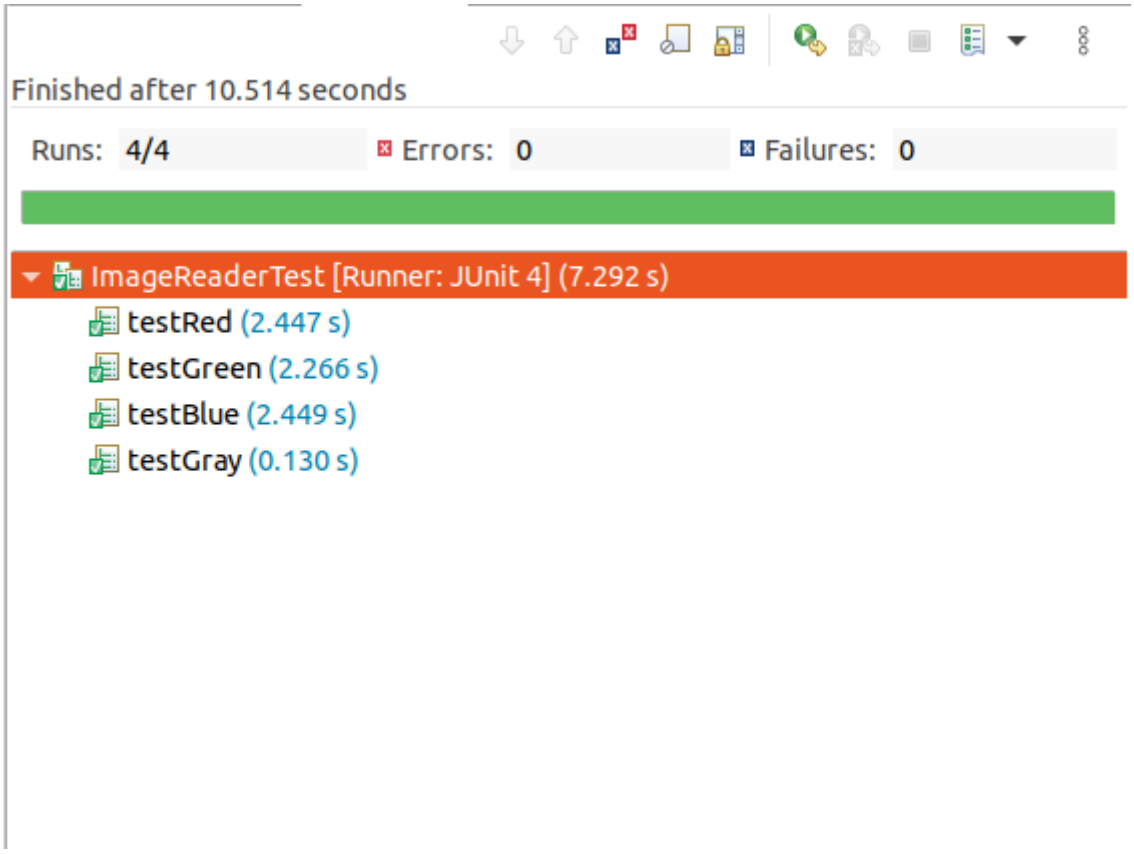


- **RGB三通道和灰度图效果**：

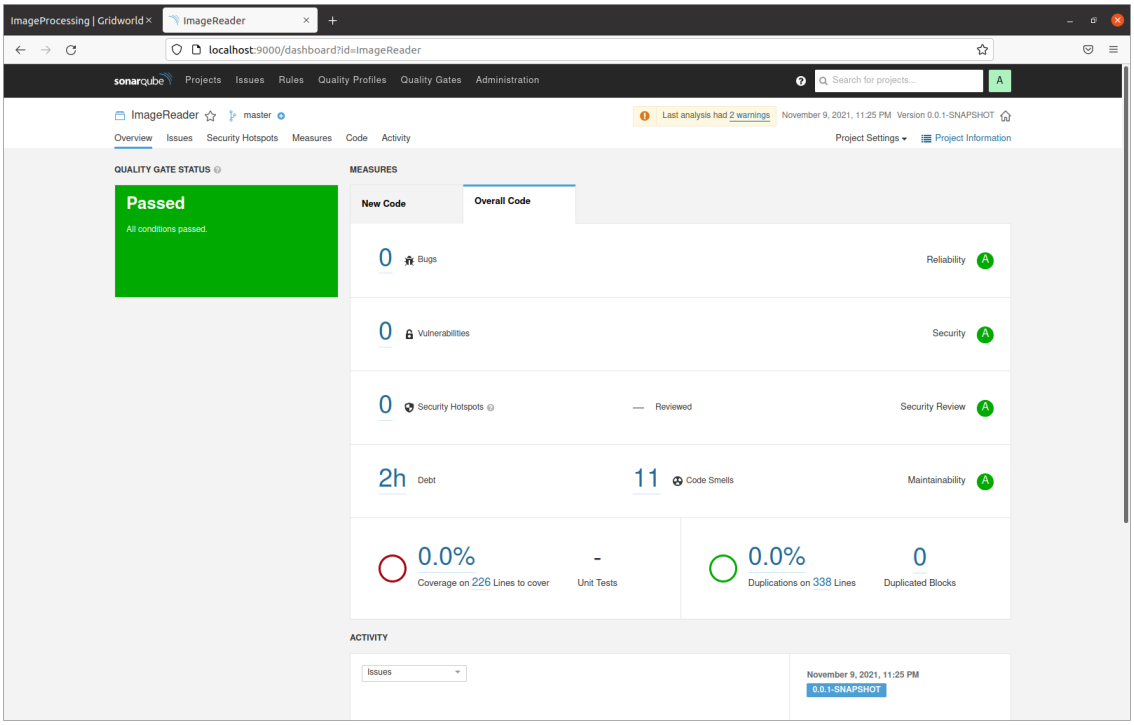


测试结果

- **JUnit 单元测试结果：Passed**



- **Sonar 代码检测结果：Passed**


ImageProcessing | Gridworld
ImageReader
localhost:9000/dashboard?id=ImageReader
sonarqube
Projects
Issues
Rules
Quality Profiles
Quality Gates
Administration
Search for projects...
ImageReader
master
Last analysis had 2 warnings
November 9, 2021, 11:25 PM
Version 0.0.1-SNAPSHOT
Overview
Issues
Security Hotspots
Measures
Code
Activity
Project Settings
Project Information
QUALITY GATE STATUS
Passed
All conditions passed.
MEASURES
New Code
Overall Code
0
Bugs
Reliability
A
0
Vulnerabilities
Security
A
0
Security Hotspots
Reviewed
Security Review
A
2h
Debt
11
Code Smells
Maintainability
A
0.0%
Coverage on 226 Lines to cover
-
Unit Tests
0.0%
Duplications on 338 Lines
0
Duplicated Blocks
ACTIVITY
Issues
November 9, 2021, 11:25 PM
0.0.1-SNAPSHOT