Chart 项目

目录

Chart 项目 1
一、 需求说明 1
二、 设计部分 2
1. 用例图 2
2. 类图 5
三、 详细设计 5
1. 数据介绍 5
2. 作图算法 5
3. 压缩处理算法 7
4. 微分处理算法 7
四、 关键代码 8
1. 检查文件 8
2. 初始化点集 9
3. 加载数据 9
4. 放大、缩小 12
5. 压缩处理 13
6. 微分处理 13
7. 作图 14
五、 测试用例 14
六、 系统总结 14

一、 需求说明

Chart 是一个做出文件的数字波形的系统，系统的输入是一个文件，系统以二进制的方式读取该文件，每隔十六位，计算一个数值，作为纵坐标，即 y 的值。而横坐标的值，则为读出的点顺序，第一个点的 x 值为 0。这样就做出了一个文件的 Chart。

上面是本系统的大致描述，下面从两个角度来说明这个系统的需求，一是功能性需求，二是非功能性需求。

功能性需求如下：

1. 做出文件的 Chart。系统能正确的以二进制的方式读取文件，并能正确的计算出 16 位二进制数的值，只是这个系统的最基本的功能。
2. 双通道。系统有两个通道，第一个通道用来显示读取文件的原本的 Chart 图像，第二个通道用来显示处理之后的数据的 Chart 图像。
3. 增大和缩小。增大，即放大原本的图像，使得某些细节更加突出；缩小，即缩小得到的图像，使得能看出整个 Chart 的轮廓、高低走势。
4. 拖动。所得到的图像，由于显示的窗口大小有限，往往不能全部显示，所以拖动功

能是必须的，为了能够使使用者看到整个数据的形式。

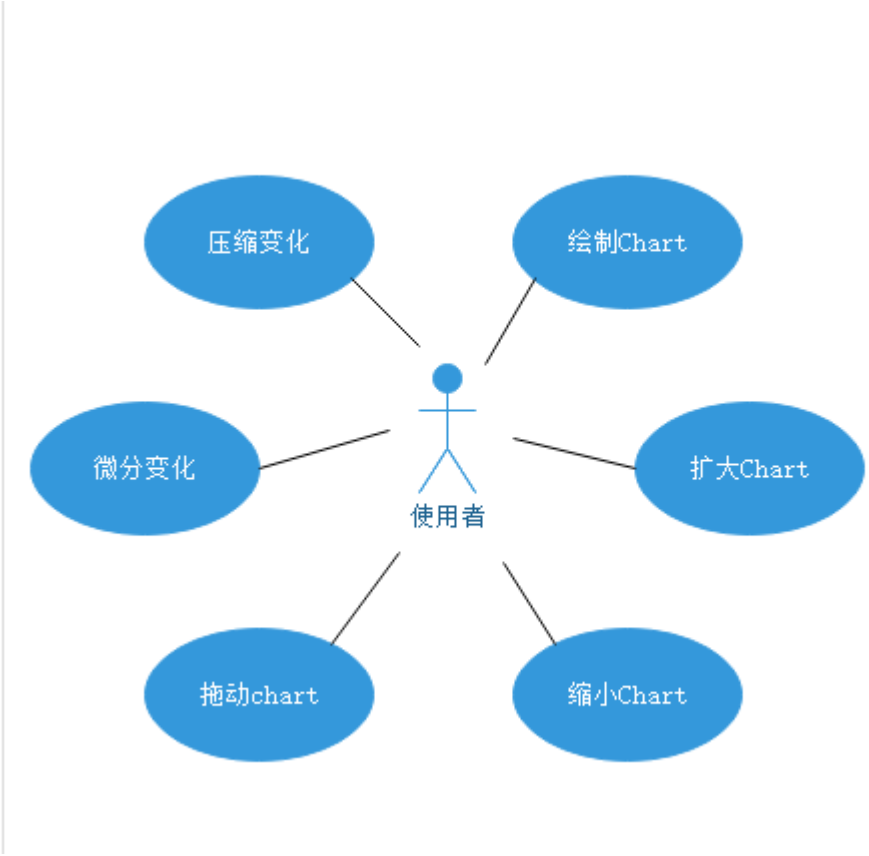
5. 压缩数据。压缩数据，实际上就是对原始得到的数据进行压缩，何为压缩？即两个点或多个点有一个相同的值。
6. 数据进行数学处理——微分。微分处理的计算公式为 f((x1+x2) / 2) = (f(x1) - f(x2)) / (x1 - x2)，即某个点的坐标为两点的斜率。

非功能性需求：

1. 性能。加载一个文件的响应时间，要做到 2s 以内，此外系统的启动的时间也必须在 3s 以内。
2. 可靠性。对输入有提示，数据有检查，防止数据异常；系统健壮性强，应该能处理系统运行过程中出现的各种异常情况，如：人为操作错误、输入非法数据、硬件设备失败等，系统应该能正确的处理，恰当的回避。
3. 环境需求。64 位的 Windows 或 Linux 操作系统，并且已经安装好了 jdk，最好 jdk 的版本大于 11.
4. 易用性。90%的用户看见这个系统之后，都能立马知道怎么使用它。
5. 可维护性。代码较好的注释，方法不超过一百行，整个类最多不超过 300 行。

二、 设计部分

1. 用例图



用例描述如下：

用例编号：001
用例名：Chart 绘制
用例描述：用户选择一个文件输入到系统中，系统根据文件内容作出 Chart 图形
参与者：系统的使用者
前置条件：
1. 用户拥有一台 OC
2. PC 上安装 java 环境
3. PC 运转正常
4. PC 上已装好此系统
后置条件：
1. 正确显示 Chart
2. 信息存入内存中
基本路径：
1. 用户打开电脑并打开系统
2. 用户点击选择文件的按钮
3. 用户选择文件
4. 系统绘制 Chart 图形
扩展点：
1. 文件过大提示
2. 文件打开过程失败提示

用例编号：002
用例名：Chart 扩大
用例描述：用户点击扩大按钮，图像扩大
前置条件：
1. 用户已经进入系统
2. 用户已经选择了文件
后置条件：
1. 原始数据保留
2. 图像显示变大
基本路径：
1. 用户点击按钮
2. 图像扩大，用户查看
扩展点：
1. 错误提示
2. 扩大倍数的间隔缩小

用例编号：003
用例名：Chart 缩小
用例描述：用户点击缩小按钮，图像缩小
前置条件：
1. 用户已经进入系统
2. 用户已经选择了文件
后置条件：

1. 原始数据保留
2. 图像显示变小

基本路径:

1. 用户点击缩小按钮
2. 图像变小，用户查看

扩展点:

1. 错误提示
2. 扩大倍数的间隔缩小

用例编号：004

用例名：拖放查看

用例描述：用户移动下侧和右侧的滚动条，查看 chart 图表

前置条件:

1. 用户已经进入系统
2. 用户已经选择了文件

后置条件:

1. 原始数据保留
2. 图像上下、左右滚动

基本路径:

1. 用户拉动滚动条或用鼠标滑轮控制
2. 图像上下左右移动

用例编号：005

用例名：微分运算

用例描述：根据已加载的文件的数据做出微分变化后的图像

前置条件:

1. 用户已经进入系统
2. 用户已经加载了文件

后置条件:

1. 原始数据不变
2. 变化后的数据改变
3. 另一通道显示变化后的图像

扩展点:

1. 数据错误恢复
2. 数据严重错误提示

用户编号：006

用例名：压缩图像

用例描述：某几个点取用相同的值来绘制新图像

前置条件:

1. 用户已经进入系统
2. 用户已经加载文件

后置条件:

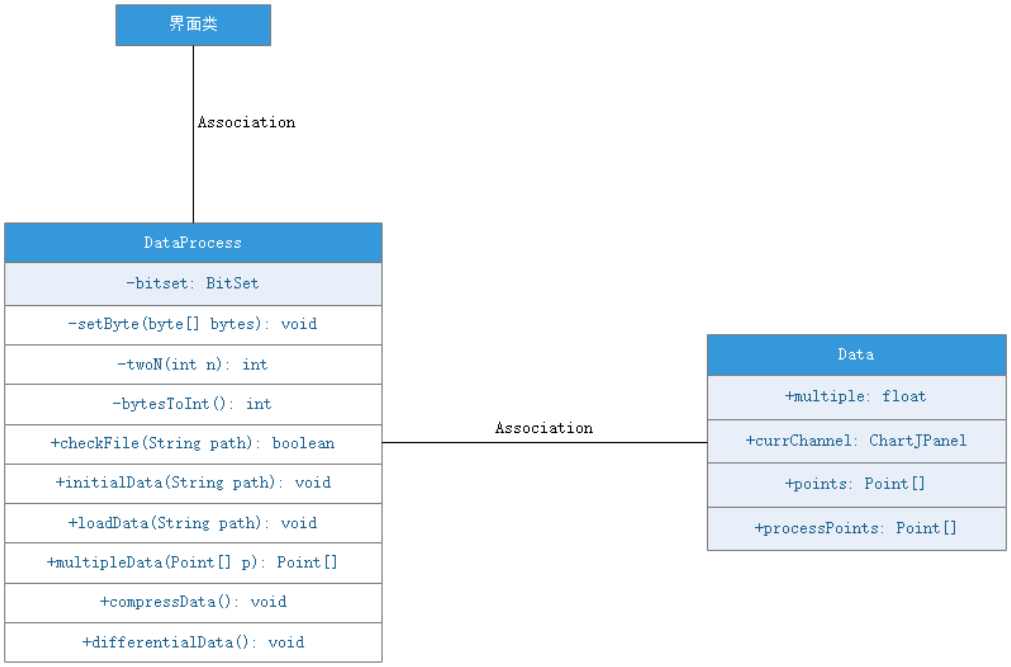
1. 原始数据改变

2. 变化后的数据改变
3. 另一通道显示变化后的图像

扩展点：

1. 数据恢复
2. 数据严重错误提示

2. 类图



（界面类省去）

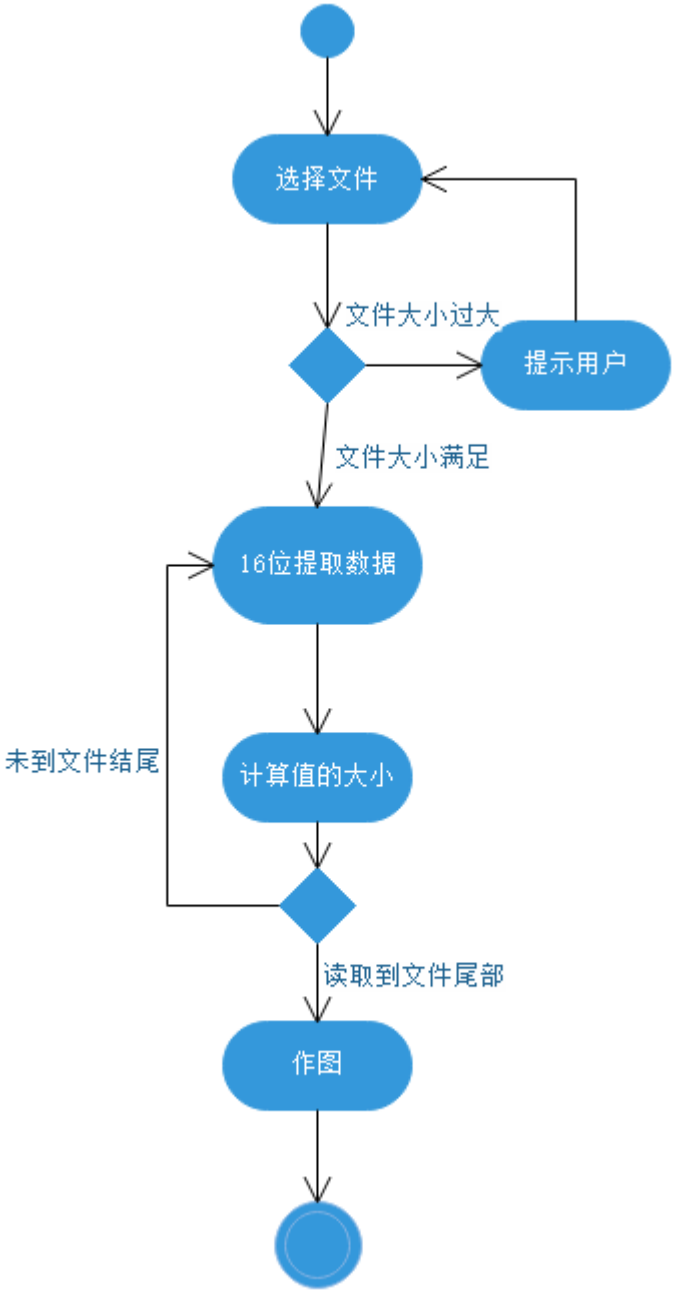
三、 详细设计

1. 数据介绍

multiple：记录此刻记录的图像的放大的倍数
currJPanel：当前选中的通道。
points：原始的数据集
processPoints：经过处理后的数据集

2. 作图算法

a) 活动图



b) 算法思路

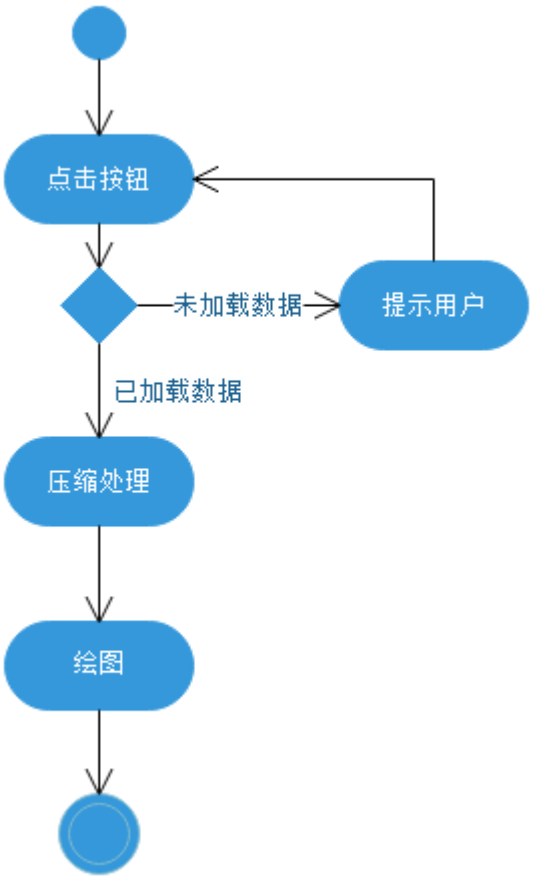
读取文件，读取两个字节的数据，将其的每一个 bit 都取出来放入一个 16bit 的 bit 数组中，之后再由这个 bit 数组计算 16 位带符号数的大小，并赋值给 point，文件读取完毕，point 也赋值完毕，再将这个 point 数组传给作图函数，做出图像。

c) 伪代码

读取文件;
循环读取 {
读取两字节;
计算数值大小并赋值给 point;
}
传入点，绘图;

3. 压缩处理算法

a) 活动图



b) 算法思路

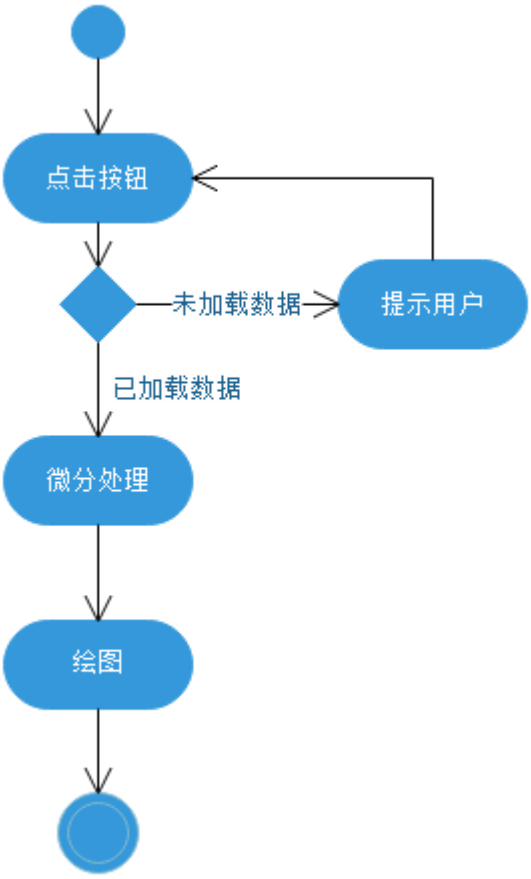
原始的数据，每三个点使用相同的数据，绘制出图像，相同的数据取平均值。

c) 伪代码

遍历 points{
计算点的 y 值和;
当计数器为 3 时{
计算求和的平均值;
将值赋给处理后的点集
}
}
重新绘制图像;

4. 微分处理算法

a) 活动图



b)　算法思路

依据公式 f((x1+x2) / 2) = (f(x1) - f(x2)) / (x1 - x2)，重新计算每个点的新值，并重新做图。

c)　伪代码

遍历原始数据集{

依据公式计算值;

}

绘制图像;

四、 关键代码

1. 检查文件

思想：读取文件，获取文件的大小，判断是否大于 1 个 G

代码：

```java
public static boolean checkFile(String path) {
    File file = new File(path);
    if (file.length() / 1024 / 1024> 1024) {
        MainWindow.dialog.setDialog("文件过大", "image/error.png");
        MainWindow.dialog.setVisible(true);
        return false;
    }
    return true;
}
```

2. 初始化点集

思想：获取文件的大小，以字节为单位，计算点的个数，并初始点值为（0,0）
代码：

```java
public static void initialData(String path) {
    File file = new File(path);

    int length = (int) (file.length() / 2);
    if (file.length() % 2 != 0) {
        length = length + 1;
    }
    Point[] points = new Point[length];

    for (int i = 0; i < length; i++) {
        points[i] = new Point(0, 0);
    }

    Data.points = points;
}
```

3. 加载数据

思想：读取两个字节的内容之后，取出每一个 bit，之后再计算大小，并赋给点
代码：

```java
public static void loadData(String path) {
    File file = new File(path);

    try {
        BufferedInputStream inputStream = new BufferedInputStream(new FileInputStream(file));
        int i = 0;
        while (true) {
            byte[] bytes = new byte[2];
            int b = inputStream.read(bytes);
            if (b == -1) {
                break;
            }
            setByte(bytes);
            Data.points[i].y = bytesToInt();
            Data.points[i].x = i;
            i++;
            //System.out.println(i);
        }
        inputStream.close();
    } catch (FileNotFoundException e) {
        MainWindow.dialog.setDialog("文件找不到", "image/error.png");
        MainWindow.dialog.setVisible(true);
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
        MainWindow.dialog.setDialog("文件读写操作错误", "image/error.png");
        MainWindow.dialog.setVisible(true);
    }
}
```

```
private static void setByte(byte[] bytes) {
    int i = 0;
    bitSet.clear();
    for (byte b: bytes) {
        if ((b & 0x01) == 0x01) {
            bitSet.set(i);
        }
        i++;

        if ((b & 0x02) == 0x02) {
            bitSet.set(i);
        }
        i++;

        if ((b & 0x04) == 0x04) {
            bitSet.set(i);
        }
        i++;

        if ((b & 0x08) == 0x08) {
            bitSet.set(i);
        }
        i++;

        if ((b & 0x10) == 0x10) {
            bitSet.set(i);
        }
        i++;

        if ((b & 0x20) == 0x20) {
            bitSet.set(i);
        }
        i++;

        if ((b & 0x40) == 0x40) {
            bitSet.set(i);
        }
        i++;

        if ((b & 0x80) == 0x80) {
            bitSet.set(i);
        }
        i++;
    }
}
```

（取出每一个 bit）

```
private static int twoN(int n) {
    if (n == 0) {
        return 1;
    }

    if (n == 1) {
        return 2;
    }

    return 2 * twoN(n - 1);
}
```

（计算 2 的 n 次方）

```
private static int bytesToInt() {
    int number = 0;
    for (int i = 0; i < 15; i++) {
        if (bitSet.get(i) == true) {
            number = number + twoN(i);
        }
    }

    if (bitSet.get(15) == true) {
        number = -number;
    }

    return number;
}
```

（将 16 位转换成整数）

4. 放大、缩小

思想：对原始点集进行扩大、缩小处理，之后返回处理后的点集
代码：

```
public static Point[] multipleData(Point[] p) {
    int length = p.length;

    Point[] points = new Point[length];

    for (int i = 0; i < length; i++) {
        points[i] = new Point(0, 0);
    }

    int i = 0;
    for (Point point: points) {
        point.x = (int) (p[i].x * 1000 * Data.multiple);
        point.y = (int) (p[i].y * Data.multiple);
        i++;
    }
    return points;
}
```

5. 压缩处理

思想：简单的波形压缩，使得波变得更紧凑，压缩量为 100，返回点集
代码：

```
public static void compressData() {
    int length = Data.points.length;

    Point[] points = new Point[length];

    for (int i = 0; i < length; i++) {
        points[i] = new Point(0, 0);
    }

    int i = 0;
    for (Point point: points) {
        point.y = Data.points[i].y;
        point.x = Data.points[i].x - 100;
        i++;
    }
    Data.processedPoints = points;
}
```

6. 微分处理

思想：公式为 f((x1+x2) / 2) = (f(x1) - f(x2)) / (x1 - x2)，计算每一个点的新值，并返回点集
代码：

```
public static void differentialData() {
    int length = Data.points.length;

    Point[] points = new Point[length - 1];

    for (int i = 0; i < length - 1; i++) {
        points[i] = new Point(0, 0);
    }

    for (int i = 0; i < points.length - 1; i++) {
        points[i].x = (Data.points[i].x + Data.points[i + 1].x) / 2;
        points[i].y = (Data.points[i].y - Data.points[i + 1].y) / (Data.points[i].x - Data.points[i + 1].x);
    }

    Data.processedPoints = points;
}
```

7. 作图

思想：遍历传入的点集，画线，并做出 x，y 轴

代码：

```
@Override
public void paintComponent(Graphics g) {
    if (points != null) {
        int length = points.length;
        g.setColor(lineColor);
        for (int i = 0; i < length - 1; i++) {
            g.drawLine(points[i].x, height + points[i].y, points[i + 1].x, height + points[i + 1].y);
        }
        g.drawLine(0, height, width, height);
    }
}
```

五、 测试用例

功能模块	测试点编号	测试点	预期结果	实际结果	是否成功
绘制图像	1	正确绘制	显示Chart	显示Chart	TRUE
	2	不加载任何文件	无响应	弹出文件不存在提示框	FALSE
	3	文件过大	提示框	出现提示框	TRUE
放大操作	4	已有数据	图像变大	图像变大	TRUE
	5	未加载数据	按钮不可点击或出现提示对话框	未响应	FALSE
缩小操作	6	已有数据	图像变小	图像变小	TRUE
	7	未加载数据	按钮不可点击或出现提示对话框	未响应	FALSE
压缩操作	8	已有数据	通道二显示图像	通道二显示图像	TRUE
	9	未加载数据	按钮不可点击或出现提示对话框	选择通道二，但无图像	FALSE
微分操作	10	已有数据	通道二显示图像	通道二显示图像	TRUE
	11	未加载数据	按钮不可点击或出现提示对话框	选择通道二，但无图像	FALSE

六、 系统总结

Chart 系统虽然是一个简单的文件转图像的系统，支持的功能也很少，此外也不怎么讲究。但是，完成这个系统的同时让我体会到了构造，是我第一次尝试从构造的角度完成的一个项目。在这个项目完成中，始终坚持一致的代码风格，良好地命名，并运用了这学期所学的知识。从详细设计、编码、验证、单元测试、集成测试和调试出现，构建了一个可工作的、

满足老师要求的系统。这些过程中，深刻体会了，如何进行防御式编程，如何降低软件的复杂度，如何实现功能的内聚性等等。收获颇多，这个项目的实现，让我切身运用所学知识，构建了一个系统，加深了对课程的学习。