重构实验

刘铭宸
软件工程 2003 班
U202010783

2022 年 5 月 31 日

目录

1 关于重构

1.1 何谓重构？

正如《重构 改善既有代码的设计》一书中所言，重构是对软件内部结构的一种调整，目的是在不改变软件可观察行为的前提下，提高其可理解性，降低其修改成本。

1.2 为何重构？

重构可以帮助改进软件的设计，使软件更容易理解，帮助找到 bug，并提高编程速度。

需求的不断变动是重构的最根本缘由，并且重构是每个开发人员都要面对的功课。代码架构最初的设计也是通过精心的设计，具备良好架构的。可是随着时间的推移、需求的剧增，必须不断的修改原有的功能、追加新的功能，还免不了有一些缺陷须要修改。为了实现变动，不可避免的要违反最初的设计构架。通过一段时间之后，软件的架构就千疮百孔了。bug 愈来愈多，愈来愈难维护，新的需求愈来愈难实现，最初的代码构架对新的需求渐渐的失去支持能力，而是成为一种制约。最后新需求的开发成本会超过开发一个新的软件的成本，这就使这个软件的生命走到了尽头。代码重构就可以最大限度的避免这样一种现象。系统发展到必定阶段后，使用重构的方式，不改变系统的外部功能，只对内部的结构进行从新的整理。经过重构，不断的调整系统的结构，使系统对于需求的变动始终具备较强的适应能力。

1.3 如何重构？——寻找代码中的坏味道

根据书中所言，首先要找到代码中的“坏味道”，在根据相应的重构手法进行重构。

2 实验内容——重构实例

2.1 神秘命名 (Mysterious Name)

```
1     public static void wordsReview(ArrayList<String>
          list, Set<Integer> set, ArrayList<Integer> fre)
          {
2          File file=new File("C:\\Users\\86182\\Desktop
               \\wordsreview.txt");
3          File file1=new File("C:\\Users\\86182\\Desktop
               \\wordsFrequency.txt");
4          try{
5              BufferedReader br=new BufferedReader(new
                   FileReader(file));
6              BufferedReader br1=new BufferedReader(new
                   FileReader(file1));
7              String str;
8              while((str=br.readLine())!=null){
9                  list.add(str);
10             }
11             String n;
12             while ((n=br1.readLine())!=null && !n.
                   equals("")){
13                 fre.add(Integer.parseInt(n));
14             }
15             Random random=new Random();
16             while(set.size()<list.size()){
17                 set.add(random.nextInt(list.size()));
18             }
19             br.close();
20             br1.close();
21         } catch (IOException e){
22             e.printStackTrace();
23         }
24    }
```

上述代码为一个用于单词复习的小程序中的一个函数, 函数功能一是从

本地的两个文件中分别读入将要复习的单词和这些单词对应的“频率”(用来衡量单词的复习状况), 并将其分别存入两个 ArrayList 类型的变量中; 二是生成随机数序列存入 Set 类型的变量中以打乱单词的出现顺序。

从这段代码中可以闻出很多种坏味道, “神秘命名”无疑是可以一眼看出的问题。首先, 函数名称 “wordsReview” 仅是将整个大程序的功能作为自己的声明, 而没有体现该函数具体做什么; 其次, 函数中定义的许多局部变量如 “br”、“br1”、“str” 等名字都没有明确的含义。

根据书中所言, 我们可以使用**变量改名 (Rename Variable)**、**改变函数声明 (Change Function Declaration)** 的方法对其进行重构。

重构后的代码

```
1   public static void wordsReadIn(ArrayList<String>
        wordList, Set<Integer> wordsOrderSet, ArrayList
        <Integer> frequencyList){
2       File file_word = new File("C:\\Users\\86182\\
            Desktop\\wordsreview.txt");
3       File file_frequency = new File("C:\\Users
            \\86182\\Desktop\\wordsFrequency.txt");
4       try{
5           BufferedReader br_word = new
                BufferedReader(new FileReader(file_word
                ));
6           BufferedReader br_frequency = new
                BufferedReader(new FileReader(
                file_frequency));
7           String str_word;
8           while((str_word = br_word.readLine()) !=
                null){
9               wordList.add(str_word);
10          }
11          String str_frequency;
12          while ((str_frequency = br_frequency.
                readLine()) != null && !str_frequency.
                equals("")){
13              frequencyList.add(Integer.parseInt(
                    str_frequency));
```

```
14              }
15              Random random = new Random();
16              while(wordsOrderSet.size()<wordList.size()
                    ){
17                  wordsOrderSet.add(random.nextInt(
                        wordList.size()));
18              }
19              br_word.close();
20              br_frequency.close();
21          } catch (IOException e){
22              e.printStackTrace();
23          }
24      }
```

2.2 过长函数 (Long Function)

```
1       public static void wordsReadIn(ArrayList<String>
            wordList, Set<Integer> wordsOrderSet, ArrayList
            <Integer> frequencyList){
2           File file_word = new File("C:\\Users\\86182\\
                Desktop\\wordsreview.txt");
3           File file_frequency = new File("C:\\Users
                \\86182\\Desktop\\wordsFrequency.txt");
4           try{
5               BufferedReader br_word = new
                    BufferedReader(new FileReader(file_word
                    ));
6               BufferedReader br_frequency = new
                    BufferedReader(new FileReader(
                    file_frequency));
7               String str_word;
8               while((str_word = br_word.readLine()) !=
                    null){
9                   wordList.add(str_word);
10              }
11              String str_frequency;
```

```
12              while ((str_frequency = br_frequency.
                    readLine()) != null && !str_frequency.
                    equals("")){
13                  frequencyList.add(Integer.parseInt(
                        str_frequency));
14              }
15              Random random = new Random();
16              while(wordsOrderSet.size()<wordList.size()
                    ){
17                  wordsOrderSet.add(random.nextInt(
                        wordList.size()));
18              }
19              br_word.close();
20              br_frequency.close();
21          } catch (IOException e){
22              e.printStackTrace();
23          }
24      }
```

承接上文，该函数还存在着“过长函数”的问题。该函数的主要功能是读入单词和对应频率，但在编程实现时为了省事就将“生成随机数作为单词出现的顺序”这一仅用两三行代码就能实现的功能也添加到了该函数中。

根据书中所言，我们可以使用**提炼函数（Extract Function）**的方法对其进行重构。

重构后的代码

```
1       public static void wordsReadIn(ArrayList<String>
            wordList, ArrayList<Integer> frequencyList){
2           File file_word = new File("C:\\Users\\86182\\
                Desktop\\wordsreview.txt");
3           File file_frequency = new File("C:\\Users
                \\86182\\Desktop\\wordsFrequency.txt");
4           try{
5               BufferedReader br_word = new
                    BufferedReader(new FileReader(file_word
                    ));
```

```
6               BufferedReader br_frequency = new
                    BufferedReader(new FileReader(
                    file_frequency));
7               String str_word;
8               while((str_word = br_word.readLine()) !=
                    null){
9                wordList.add(str_word);
10              }
11              String str_frequency;
12              while ((str_frequency = br_frequency.
                    readLine()) != null && !str_frequency.
                    equals("")){
13               frequencyList.add(Integer.parseInt(
                        str_frequency));
14              }
15              br_word.close();
16              br_frequency.close();
17          } catch (IOException e){
18              e.printStackTrace();
19          }
20      }
21
22      public static void getWordsOrder(ArrayList<String>
            wordList, Set<Integer> wordsOrderSet){
23          Random random = new Random();
24          while(wordsOrderSet.size()<wordList.size()){
25              wordsOrderSet.add(random.nextInt(wordList.
                    size()));
26          }
27      }
```

2.3 全局数据（Global Data）

```
1       public class SimpleFrame extends JFrame {
2           JLabel wordLabel = new JLabel();
3           JLabel frequencyLabel = new JLabel();
```

```
4           MeaningLabel meaningLabel;
5           ArrayList<String> wordList = new ArrayList<>()
                ;
6           Set<Integer> wordsOrderSet = new LinkedHashSet
                <>();
7           ArrayList<String> wordRandomList = new
                ArrayList<>();
8           ArrayList<Integer> frequencyList = new
                ArrayList<>();
9           Map<String,Integer> wordFrequencyMap = new
                HashMap<>();
10          ......
11      }
```

上述代码为一个界面类的成员变量，可以看到在编程时为了能够随时在程序的任何部分对其进行修改访问，将这些成员变量的作用域设置为“缺省”，即可以在同一包中的任何类中对其进行访问，所以存在着“全局数据”的问题。

根据书中所言，我们可以使用**封装变量（Encapsulate Variable）**的方法对其进行重构。

重构后的代码

```
1   public class SimpleFrame extends JFrame {
2       private JLabel wordLabel = new JLabel();
3       private JLabel frequencyLabel = new JLabel();
4       private MeaningLabel meaningLabel;
5       private ArrayList<String> wordList = new
            ArrayList<>();
6       private Set<Integer> wordsOrderSet = new
            LinkedHashSet<>();
7       private ArrayList<String> wordRandomList = new
            ArrayList<>();
8       private ArrayList<Integer> frequencyList = new
            ArrayList<>();
9       private Map<String,Integer> wordFrequencyMap =
            new HashMap<>();
```

```
10          ......
11          public JLabel getWordLabel() {
12          return wordLabel;
13      }
14
15          public void setWordLabel(JLabel wordLabel) {
16              this.wordLabel = wordLabel;
17          }
18
19          public JLabel getFrequencyLabel() {
20              return frequencyLabel;
21          }
22
23          public void setFrequencyLabel(JLabel
                frequencyLabel) {
24              this.frequencyLabel = frequencyLabel;
25          }
26
27          public MeaningLabel getMeaningLabel() {
28              return meaningLabel;
29          }
30
31          public void setMeaningLabel(MeaningLabel
                meaningLabel) {
32              this.meaningLabel = meaningLabel;
33          }
34
35          public ArrayList<String> getWordList() {
36              return wordList;
37          }
38
39          public void setWordList(ArrayList<String>
                wordList) {
40              this.wordList = wordList;
41          }
42
```

```
43          public Set<Integer> getWordsOrderSet() {
44              return wordsOrderSet;
45          }
46
47          public void setWordsOrderSet(Set<Integer>
                wordsOrderSet) {
48              this.wordsOrderSet = wordsOrderSet;
49          }
50
51          public ArrayList<String> getWordRandomList() {
52              return wordRandomList;
53          }
54
55          public void setWordRandomList(ArrayList<String
                > wordRandomList) {
56              this.wordRandomList = wordRandomList;
57          }
58
59          public ArrayList<Integer> getFrequencyList() {
60              return frequencyList;
61          }
62
63          public void setFrequencyList(ArrayList<Integer
                > frequencyList) {
64              this.frequencyList = frequencyList;
65          }
66
67          public Map<String, Integer>
                getWordFrequencyMap() {
68              return wordFrequencyMap;
69          }
70
71          public void setWordFrequencyMap(Map<String,
                Integer> wordFrequencyMap) {
72              this.wordFrequencyMap = wordFrequencyMap;
73          }
```

```
74      }
```

2.4 异曲同工的类 (Alternative Classes)

```
1       import java.awt.*;
2       import java.util.*;
3       public class SimpleLayout implements
            LayoutManager2{
4           Map<Component, MyDimension> componentMap =new
                HashMap<>();
5           @Override
6           public void addLayoutComponent(Component comp,
                Object constraints) {
7               componentMap.put(comp,(MyDimension)
                    constraints);
8           }
9
10          @Override
11          public Dimension maximumLayoutSize(Container
                target) {
12              return null;
13          }
14
15          @Override
16          public float getLayoutAlignmentX(Container
                target) {
17              return 0;
18          }
19
20          @Override
21          public float getLayoutAlignmentY(Container
                target) {
22              return 0;
23          }
24
25          @Override
```

```
26          public void invalidateLayout(Container target)
                {
27
28          }
29
30          @Override
31          public void addLayoutComponent(String name,
                Component comp) {
32
33          }
34
35          @Override
36          public void removeLayoutComponent(Component
                comp) {
37              componentMap.remove(comp);
38          }
39
40          @Override
41          public Dimension preferredLayoutSize(Container
                parent) {
42              return null;
43          }
44
45          @Override
46          public Dimension minimumLayoutSize(Container
                parent) {
47              return null;
48          }
49
50          @Override
51          public void layoutContainer(Container parent)
                {
52              int width=parent.getWidth();
53              int height=parent.getHeight();
54              Set<Component> set= componentMap.keySet();
55              for(Component i:set){
```

```
56                  i.setBounds(width* componentMap.get(i)
                        .getLeft() /400, height*
                        componentMap.get(i).getTop() /300,
                        componentMap.get(i).getWidth(),
                        componentMap.get(i).getHeight());
57              }
58          }
59      }
```

上述代码为一个自定义的布局器。SimpleLayout 类实现了 java.awt 中的 LayoutManager2 接口，但实际上仅对 addLayoutComponent()、removeLayoutComponent() 和 layoutContainer() 三个函数进行了改动，而剩余大量的其他函数都是以默认方式实现。但由于接口实现类的特性，必须对接口类中所有声明的函数进行重写，导致自定义布局器中存在着较多无用代码，且每次实现一个布局器都会重复的出现，所以存在着“异曲同工的类”的问题。

根据书中所言，我们可以使用**提炼超类（Extract Superclass）**和**函数上移（Pull Up Method）**的方法对其进行重构。

重构后的代码

```
1   import java.awt.*;
2   public abstract class LayoutAdapter implements
        LayoutManager2 {
3       @Override
4       public Dimension maximumLayoutSize(Container
            target) {
5           return null;
6       }
7
8       @Override
9       public float getLayoutAlignmentX(Container
            target) {
10          return 0;
11      }
12
13      @Override
```

```
14          public float getLayoutAlignmentY(Container
                target) {
15              return 0;
16          }
17
18          @Override
19          public void invalidateLayout(Container target)
                {
20
21          }
22
23          @Override
24          public void addLayoutComponent(String name,
                Component comp) {
25
26          }
27
28          @Override
29          public Dimension preferredLayoutSize(Container
                parent) {
30              return null;
31          }
32
33          @Override
34          public Dimension minimumLayoutSize(Container
                parent) {
35              return null;
36          }
37      }
```

```
1       import java.awt.*;
2       import java.util.*;
3       public class SimpleLayout extends LayoutAdapter {
4           Map<Component, MyDimension> componentMap =new
                HashMap<>();
5           @Override
```

```
6           public void addLayoutComponent(Component comp,
                Object constraints) {
7               componentMap.put(comp,(MyDimension)
                    constraints);
8           }
9
10          @Override
11          public void removeLayoutComponent(Component
                comp) {
12              componentMap.remove(comp);
13          }
14
15          @Override
16          public void layoutContainer(Container parent)
                {
17              int width=parent.getWidth();
18              int height=parent.getHeight();
19              Set<Component> set= componentMap.keySet();
20              for(Component i:set){
21                  i.setBounds(width* componentMap.get(i)
                        .getLeft() /400, height*
                        componentMap.get(i).getTop() /300,
                        componentMap.get(i).getWidth(),
                        componentMap.get(i).getHeight());
22              }
23          }
24      }
```

2.5 重复代码 (Duplicated Code)

```
1       rememberButton.addActionListener((e)->{
2           if(count < getWordRandomList().size()-1){
3               String s= getWordLabel().getText();
4               getWordFrequencyMap().put(s,
                    getWordFrequencyMap().get(s)-1);
5               getWordLabel().setText(getWordRandomList()
```

```
                .get(++count));
6           getFrequencyLabel().setText(
                getWordFrequencyMap().get(getWordLabel
                ().getText()).toString());
7       }
8       if(count == getWordRandomList().size()-1){
9           String s1= getWordLabel().getText();
10          getWordFrequencyMap().put(s1,
                getWordFrequencyMap().get(s1)-1);
11          count++;
12          frequencyRewriter(getWordList(),
                getWordFrequencyMap());
13      }
14  });
15
16  uncertainButton.addActionListener((e)->{
17      if(count < getWordRandomList().size()-1){
18          getWordLabel().setText(getWordRandomList()
                .get(++count));
19          getFrequencyLabel().setText(
                getWordFrequencyMap().get(getWordLabel
                ().getText()).toString());
20      }
21      if(count == getWordRandomList().size()-1){
22          count++;
23          frequencyRewriter(getWordList(),
                getWordFrequencyMap());
24      }
25  });
26
27  forgetButton.addActionListener((e)->{
28      if(count < getWordRandomList().size()-1){
29          String s= getWordLabel().getText();
30          getWordFrequencyMap().put(s,
                getWordFrequencyMap().get(s)+1);
31          setMeaningLabel(new MeaningLabel(s));
```

```
32              rememberButton.setVisible(false);
33              uncertainButton.setVisible(false);
34              forgetButton.setVisible(false);
35              preButton.setVisible(false);
36              knowButton.setVisible(true);
37              panel.add(getMeaningLabel(),new
                    MyDimension(100,100,200,90));
38              panel.add(knowButton,new MyDimension
                    (150,200,100,30));
39              getMeaningLabel().setBounds
                    (100,100,200,60);
40          }
41          if(count == getWordRandomList().size()-1){
42              String s1= getWordLabel().getText();
43              getWordFrequencyMap().put(s1,
                    getWordFrequencyMap().get(s1)+1);
44              count++;
45              frequencyRewriter(getWordList(),
                    getWordFrequencyMap());
46          }
47      });
```

上述代码为实现三个按钮 rememberButton、uncertainButton 和 forgetButton 的监听器事件的匿名内部类，目的是为了在用户点击不同按钮时实现相应的功能，并在用户复习完所有单词后将新的频率重写回频率文件。在实现“重写回文件”的功能时，三个按钮都几乎进行了同样的操作，仅有细微的不同。所以存在着“重复代码”的问题。

根据书中所言，我们可以使用**提炼函数（Extract Function）**的方法对其进行重构。

重构后的代码

```
1       rememberButton.addActionListener((e)->{
2           if(count < getWordRandomList().size()-1){
3               String s = getWordLabel().getText();
4               getWordFrequencyMap().put(s,
                    getWordFrequencyMap().get(s)-1);
```

```
5               nextWord();
6           }
7           if(count == getWordRandomList().size()-1){
8               String s1 = getWordLabel().getText();
9               getWordFrequencyMap().put(s1,
                    getWordFrequencyMap().get(s1)-1);
10              frequencyRewrite();
11          }
12      });
13
14      uncertainButton.addActionListener((e)->{
15          if(count < getWordRandomList().size()-1){
16              nextWord();
17          }
18          if(count == getWordRandomList().size()-1){
19              frequencyRewrite();
20          }
21      });
22
23      forgetButton.addActionListener((e)->{
24          if(count < getWordRandomList().size()-1){
25              String s = getWordLabel().getText();
26              getWordFrequencyMap().put(s,
                    getWordFrequencyMap().get(s)+1);
27              setMeaningLabel(new MeaningLabel(s));
28              rememberButton.setVisible(false);
29              uncertainButton.setVisible(false);
30              forgetButton.setVisible(false);
31              preButton.setVisible(false);
32              knowButton.setVisible(true);
33              panel.add(getMeaningLabel(),new
                    MyDimension(100,100,200,90));
34              panel.add(knowButton,new MyDimension
                    (150,200,100,30));
35              getMeaningLabel().setBounds
                    (100,100,200,60);
```

```
36          }
37          if(count == getWordRandomList().size()-1){
38              String s1 = getWordLabel().getText();
39              getWordFrequencyMap().put(s1,
                    getWordFrequencyMap().get(s1)+1);
40              frequencyRewrite();
41          }
42      });
43
44      public void nextWord() {
45          getWordLabel().setText(getWordRandomList().get
                (++count));
46          getFrequencyLabel().setText(
                getWordFrequencyMap().get(getWordLabel().
                getText()).toString());
47      }
48
49      public void frequencyRewrite(){
50          count++;
51          frequencyRewriter(getWordList(),
                getWordFrequencyMap());
52      }
```

2.6 数据泥团 (Data Clumps)

```
1       startButton.addActionListener((e)->{
2           startButton.setVisible(false);
3           getWordLabel().setText(getWordRandomList().get
                (++count));
4           getFrequencyLabel().setText(
                getWordFrequencyMap().get(getWordLabel().
                getText()).toString());
5           panel.add(getWordLabel(),150,60,100,30);
6           panel.add(rememberButton,150,100,100,30);
7           panel.add(uncertainButton,150,140,100,30);
8           panel.add(forgetButton,150,180,100,30);
```

```
9           panel.add(preButton,0,0,100,30);
10          panel.add(getFrequencyLabel(),150,250,100,30);
11      });
```

上述代码为实现开始按钮 startButton 监听器的匿名内部类，当用户点击该按钮时，将需要显示的组件添加到容器 panel 中去。在添加的时候，表示组件绝对位置的四个参数总是绑在一起出现，导致参数列表过长，出现了“数据泥团”的问题。

根据书中所言，我们可以使用**引入参数对象（Introduce Parameter Declaration）**和**提炼类（Extract Class）**的方法对其进行重构。

重构后的代码

```
1   public class MyDimension {
2       private int left;
3       private int top;
4       private int width;
5       private int height;
6       public MyDimension(int left,int top,int width,
            int height){
7           this.setLeft(left);
8           this.setTop(top);
9           this.setWidth(width);
10          this.setHeight(height);
11      }
12
13      public int getLeft() {
14          return left;
15      }
16
17      public void setLeft(int left) {
18          this.left = left;
19      }
20
21      public int getTop() {
22          return top;
23      }
```

```
24
25          public void setTop(int top) {
26              this.top = top;
27          }
28
29          public int getWidth() {
30              return width;
31          }
32
33          public void setWidth(int width) {
34              this.width = width;
35          }
36
37          public int getHeight() {
38              return height;
39          }
40
41          public void setHeight(int height) {
42              this.height = height;
43          }
44      }
```

```
1       startButton.addActionListener((e)->{
2           startButton.setVisible(false);
3           getWordLabel().setText(getWordRandomList().get
                (++count));
4           getFrequencyLabel().setText(
                getWordFrequencyMap().get(getWordLabel().
                getText()).toString());
5           panel.add(getWordLabel(),new MyDimension
                (150,60,100,30));
6           panel.add(rememberButton,new MyDimension
                (150,100,100,30));
7           panel.add(uncertainButton,new MyDimension
                (150,140,100,30));
8           panel.add(forgetButton,new MyDimension
```

```
            (150,180,100,30));
9           panel.add(preButton,new MyDimension
            (0,0,100,30));
10          panel.add(getFrequencyLabel(),new MyDimension
            (150,250,100,30));
11      });
```

2.7 过长参数列表 (Long Parameter List)

```
1       public static void wordsReadIn(ArrayList<String>
            wordList, ArrayList<Integer> frequencyList){
2           File file_word = new File("C:\\Users\\86182\\
            Desktop\\wordsreview.txt");
3           File file_frequency = new File("C:\\Users
            \\86182\\Desktop\\wordsFrequency.txt");
4           try{
5               BufferedReader br_word = new
                BufferedReader(new FileReader(file_word
                ));
6               BufferedReader br_frequency = new
                BufferedReader(new FileReader(
                file_frequency));
7               String str_word;
8               while((str_word = br_word.readLine()) !=
                null){
9                   wordList.add(str_word);
10              }
11              String str_frequency;
12              while ((str_frequency = br_frequency.
                readLine()) != null && !str_frequency.
                equals("")){
13                  frequencyList.add(Integer.parseInt(
                    str_frequency));
14              }
15              br_word.close();
16              br_frequency.close();
```

```
17          } catch (IOException e){
18              e.printStackTrace();
19          }
20      }
21
22      public static void getWordsOrder(ArrayList<String>
            wordList, Set<Integer> wordsOrderSet){
23          Random random = new Random();
24          while(wordsOrderSet.size()<wordList.size()){
25              wordsOrderSet.add(random.nextInt(wordList.
                    size()));
26          }
27      }
28
29      public static void frequencyRewriter(ArrayList<
            String> list ,Map<String,Integer> map){
30          File file=new File("C:\\Users\\86182\\Desktop
                \\wordsFrequency.txt");
31          try{
32              BufferedWriter bw=new BufferedWriter(new
                    FileWriter(file));
33              for(String i:list){
34                  bw.write(map.get(i).toString());
35                  bw.newLine();
36              }
37              bw.flush();
38              bw.close();
39          }catch (IOException e){
40              e.printStackTrace();
41          }
42      }
```

上述代码分别为读入单词、生成单词顺序和重写频率的函数，可以看到每个函数的参数列表都大致相同。实际上，在调用这些函数时，调用者所传入的参数就是该类中定义的成员变量，这些变量由函数自己来获得也是同样容易，出现了“过长参数列表”的问题。

根据书中所言，我们可以使用**以查询取代参数（Replace Parameter**

with Query）的方法对其进行重构。

重构后的代码

```
1   public class SimpleFrame extends JFrame {
2       ......
3       wordsReadIn();
4       getWordsOrder();
5       ......
6       public void wordsReadIn(){
7           File file_word = new File("C:\\Users
                \\86182\\Desktop\\wordsreview.txt");
8           File file_frequency = new File("C:\\Users
                \\86182\\Desktop\\wordsFrequency.txt");
9           try{
10              BufferedReader br_word = new
                    BufferedReader(new FileReader(
                    file_word));
11              BufferedReader br_frequency = new
                    BufferedReader(new FileReader(
                    file_frequency));
12              String str_word;
13              while((str_word = br_word.readLine())
                    != null){
14                  wordList.add(str_word);
15              }
16              String str_frequency;
17              while ((str_frequency = br_frequency.
                    readLine()) != null && !
                    str_frequency.equals("")){
18                  frequencyList.add(Integer.parseInt
                        (str_frequency));
19              }
20              br_word.close();
21              br_frequency.close();
22          } catch (IOException e){
23              e.printStackTrace();
```

```
24              }
25          }
26
27          public void getWordsOrder(){
28              Random random = new Random();
29              while(wordsOrderSet.size()<wordList.size()
                    ){
30                  wordsOrderSet.add(random.nextInt(
                        wordList.size()));
31              }
32          }
33
34          public void frequencyRewriter(){
35              File file=new File("C:\\Users\\86182\\
                    Desktop\\wordsFrequency.txt");
36              try{
37                  BufferedWriter bw=new BufferedWriter(
                        new FileWriter(file));
38                  for(String i:wordList){
39                      bw.write(wordFrequencyMap.get(i).
                            toString());
40                      bw.newLine();
41                  }
42                  bw.flush();
43                  bw.close();
44              }catch (IOException e){
45                  e.printStackTrace();
46              }
47          }
48      }
```

2.8 霰弹式修改 (Shotgun Surgery)

```
1       public static void frequencyWriter(int n){
2           File file1=new File("C:\\Users\\86182\\Desktop
                \\wordsFrequency.txt");
```

```
3           try {
4               BufferedWriter bw=new BufferedWriter(new
                    FileWriter(file1 ,true));
5               for(int i=0;i<n;i++) {
6                   bw.write("5");
7                   bw.newLine();
8               }
9               bw.flush();
10              bw.close();
11          }catch (IOException e){
12              e.printStackTrace();
13          }
14      }
15
16      public void wordsReadIn(){
17          File file_word = new File("C:\\Users\\86182\\
                Desktop\\wordsreview.txt");
18          File file_frequency = new File("C:\\Users
                \\86182\\Desktop\\wordsFrequency.txt");
19          try{
20              BufferedReader br_word = new
                    BufferedReader(new FileReader(file_word
                    ));
21              BufferedReader br_frequency = new
                    BufferedReader(new FileReader(
                    file_frequency));
22              String str_word;
23              while((str_word = br_word.readLine()) !=
                    null){
24                  wordList.add(str_word);
25              }
26              String str_frequency;
27              while ((str_frequency = br_frequency.
                    readLine()) != null && !str_frequency.
                    equals("")){
28                  frequencyList.add(Integer.parseInt(
```

```
                    str_frequency));
29              }
30              br_word.close();
31              br_frequency.close();
32          } catch (IOException e){
33              e.printStackTrace();
34          }
35      }
36
37      public void frequencyRewriter(){
38          File file = new File("C:\\Users\\86182\\
                Desktop\\wordsFrequency.txt");
39          try{
40              BufferedWriter bw=new BufferedWriter(new
                    FileWriter(file));
41              for(String i:wordList){
42                  //System.out.println(map.get(i));
43                  bw.write(wordFrequencyMap.get(i).
                        toString());
44                  bw.newLine();
45              }
46              bw.flush();
47              bw.close();
48          }catch (IOException e){
49              e.printStackTrace();
50          }
51      }
```

上述代码为整个程序中需要对文件进行 IO 操作的三个函数。这三个函数分散在不同的类中，当需要对文件路径进行修改时，需要分别找到这三个函数各自对其进行修改，存在着“霰弹式修改”的问题。

根据书中所言，我们可以使用**函数组合成类（Combine Functions into Class）**和**搬移函数（Move Function）**的方法对其进行重构。

重构后的代码

```
1   import java.io.*;
```

```
2  import java.util.ArrayList;
3  import java.util.Map;
4
5  public class FileIO {
6      private String wordFile = "C:\\Users\\86182\\
           Desktop\\wordsreview.txt";
7      private String frequencyFile = "C:\\Users\\86182\\
           Desktop\\wordsFrequency.txt";
8      private ArrayList<String> wordList;
9      private Map<String,Integer> wordFrequencyMap;
10     private ArrayList<Integer> frequencyList;
11     private int newWordsNumber;
12     public FileIO(ArrayList<String> wordList, Map<
           String,Integer> wordFrequencyMap, ArrayList<
           Integer> frequencyList){
13         this.setWordList(wordList);
14         this.setWordFrequencyMap(wordFrequencyMap);
15         this.setFrequencyList(frequencyList);
16     }
17
18     public FileIO(int newWordsNumber){
19         this.setNewWordsNumber(newWordsNumber);
20     }
21
22     public void frequencyWriter(){
23         File file=new File(getFrequencyFile());
24         try {
25             BufferedWriter bw=new BufferedWriter(new
                   FileWriter(file,true));
26             for(int i=0;i<newWordsNumber;i++) {
27                 bw.write("5");
28                 bw.newLine();
29             }
30             bw.flush();
31             bw.close();
32         }catch (IOException e){
```

```
33              e.printStackTrace();
34          }
35      }
36
37      public void frequencyRewriter(){
38          File file = new File(getFrequencyFile());
39          try{
40              BufferedWriter bw=new BufferedWriter(new
                    FileWriter(file));
41              for(String i: getWordList()){
42                  bw.write(getWordFrequencyMap().get(i).
                        toString());
43                  bw.newLine();
44              }
45              bw.flush();
46              bw.close();
47          }catch (IOException e){
48              e.printStackTrace();
49          }
50      }
51
52      public void wordsReadIn(){
53          File file_word = new File(getWordFile());
54          File file_frequency = new File(
                getFrequencyFile());
55          try{
56              BufferedReader br_word = new
                    BufferedReader(new FileReader(file_word
                    ));
57              BufferedReader br_frequency = new
                    BufferedReader(new FileReader(
                    file_frequency));
58              String str_word;
59              while((str_word = br_word.readLine()) !=
                    null){
60                  getWordList().add(str_word);
```

```
61              }
62              String str_frequency;
63              while ((str_frequency = br_frequency.
                    readLine()) != null && !str_frequency.
                    equals("")){
64                  getFrequencyList().add(Integer.
                        parseInt(str_frequency));
65              }
66              br_word.close();
67              br_frequency.close();
68          } catch (IOException e){
69              e.printStackTrace();
70          }
71      }
72
73      public String getWordFile() {
74          return wordFile;
75      }
76
77      public void setWordFile(String wordFile) {
78          this.wordFile = wordFile;
79      }
80
81      public String getFrequencyFile() {
82          return frequencyFile;
83      }
84
85      public void setFrequencyFile(String frequencyFile)
            {
86          this.frequencyFile = frequencyFile;
87      }
88
89      public ArrayList<String> getWordList() {
90          return wordList;
91      }
92
```

```
93      public void setWordList(ArrayList<String> wordList
            ) {
94          this.wordList = wordList;
95      }
96
97      public Map<String, Integer> getWordFrequencyMap()
            {
98          return wordFrequencyMap;
99      }
100
101     public void setWordFrequencyMap(Map<String,
            Integer> wordFrequencyMap) {
102         this.wordFrequencyMap = wordFrequencyMap;
103     }
104
105     public ArrayList<Integer> getFrequencyList() {
106         return frequencyList;
107     }
108
109     public void setFrequencyList(ArrayList<Integer>
            frequencyList) {
110         this.frequencyList = frequencyList;
111     }
112
113     public int getNewWordsNumber() {
114         return newWordsNumber;
115     }
116
117     public void setNewWordsNumber(int newWordsNumber)
            {
118         this.newWordsNumber = newWordsNumber;
119     }
120 }
```

3 参考资料

[1] 教学课件：感谢华中科技大学软件学院刘小峰老师！

[2]（美）马丁 · 福勒（Martin Fowler）著. 熊节，林从羽译. 重构：改善既有代码的设计（第二版）. 人民邮电出版社