

# Java Homework3

## part1 作业

1. 寻找JDK库中的不变类（至少3类），并进行源码分析，分析其为什么是不变的？文档说明其共性。

(1) JDK库中的不变类：String, Integer, Long

(2) 源码分析：

- String:

```
1 public final class String
2     implements java.io.Serializable, Comparable<String>, CharSequence
3 {
4     /** The value is used for character storage. */
5     private final byte[] value;
6     /** The offset is the first index of the storage that is used. */
7     private final int offset;
8     /** The count is the number of characters in the String. */
9     private final int count;
10    /** Cache the hash code for the string */
11    private int hash; // Default to 0
12    ....
13    public String(char value[]) {
14        this.value = Arrays.copyOf(value, value.length);
15    }
16    ...
17    public char[] toCharArray() {
18        // Cannot use Arrays.copyOf because of class initialization order
19        // issues
20        char result[] = new char[value.length];
21        System.arraycopy(value, 0, result, 0, value.length);
22        return result;
23    }
24    ...
25 }
```

- String类被final修饰，不可继承
- string内部所有成员都设置为私有变量
- 不存在value的setter
- 并将value和offset设置为final
- 当传入可变数组value[]时，进行copy而不是直接将value[]复制给内部变量
- 获取value时不是直接返回对象引用，而是返回对象的copy

- Integer:

```
1 public final class Integer extends Number
2     implements Comparable<Integer>, Constable, ConstantDesc {
3
4     private final int value;
5     private static final long serialVersionUID = 1360826667806852920L;
6     ...
7     public Integer(int value) {
```

```

8         this.value = value;
9     }
10    ...
11    public static Integer valueOf(int i) {
12        if (i >= IntegerCache.low && i <= IntegerCache.high)
13            return IntegerCache.cache[i + (-IntegerCache.low)];
14        return new Integer(i);
15    }
16    ...
17    static {
18        // high value may be configured by property
19        int h = 127;
20        String integerCacheHighPropValue =
21
22        sun.misc.VM.getSavedProperty("java.lang.Integer.IntegerCache.high");
23        if (integerCacheHighPropValue != null) {
24            try {
25                int i = parseInt(integerCacheHighPropValue);
26                i = Math.max(i, 127);
27                // Maximum array size is Integer.MAX_VALUE
28                h = Math.min(i, Integer.MAX_VALUE - (-low) -1);
29            } catch( NumberFormatException nfe) {
30                // If the property cannot be parsed into an int, ignore
31                it.
32            }
33        }
34        high = h;
35
36        cache = new Integer[(high - low) + 1];
37        int j = low;
38        for(int k = 0; k < cache.length; k++)
39            cache[k] = new Integer(j++);
40
41        // range [-128, 127] must be interned (JLS7 5.1.7)
42        assert IntegerCache.high >= 127;
43    }
44    ...
45 }

```

- Integer类被final修饰，不可继承
- Integer内部所有成员都设置为私有变量
- value为final，不可被修改；不存在value的setter
- 当Integer被加载时，就新建了-128到127的所有数字并存放在Integer数组cache中。当调用valueOf方法时，如果参数的值在-127到128之间，则直接从缓存中返回一个已经存在的对象。如果参数的值不在这个范围内，则new一个Integer对象返回。
- Long:

```

1 public final class Long extends Number implements Comparable<Long> {
2     private final longvalue;
3     ...
4     public static Long valueOf(long l) {
5         final int offset = 128;
6         if (l >= -128 && l <= 127) { // will cache
7             return LongCache.cache[(int)l + offset];
8         }
9         return new Long(l);
10    }
11    ...
12 }

```

- long类被final修饰，不可继承
- long内部所有成员都设置为私有变量
- longvalue为final，不可被修改；不存在longvalue的setter
- 当long被加载时，就新建了-128到127的所有数字并存放在long数组cache中。当调用valueOf方法时，如果参数的值在-127到128之间，则直接从缓存中返回一个已经存在的对象。如果参数的值不在这个范围内，则new一个long对象返回。

## 2. 对String、StringBuilder以及StringBuffer进行源代码分析：

### (1) 分析其主要数据组织及功能实现，有什么区别？

**String**：由第一题的String源码分析可知，String是不可变的对象，因此每次对String类型进行改变的时候，都会生成一个新的String对象，然后将指针指向新的String对象。

**StringBuffer**：字符串变量（线程安全）。如果要频繁对字符串内容进行修改，出于效率考虑最好使用StringBuffer。StringBuffer上的主要操作是 append 和 insert 方法，每个方法都能有效地将给定的数据转换成字符串，然后将该字符串的字符追加或插入到字符串缓冲区中。

**StringBuilder**：字符串变量（非线程安全）。在内部，StringBuilder对象被当作是一个包含字符序列的变长数组。此类提供一个与StringBuffer兼容的API，但不保证同步。该类被设计用作StringBuffer的一个简易替换，用在字符串缓冲区被单个线程使用的时候。

### 三者区别：

	String	StringBuilder	StringBuffer
Mutable	No	Yes	Yes
Thread-Safe	Yes	No	Yes
Time Efficient	No	Yes	No
Memory Efficient	No	Yes	Yes

### (2) 说明为什么这样设计，这么设计对String, StringBuilder及StringBuffer的影响？

因为String的不可变性，每次生成对象会对系统性能产生影响，特别当内存中无引用对象多了以后，JVM的GC就会开始工作，性能就会降低。

使用StringBuffer类时，每次都会对StringBuffer对象本身进行操作，而不是生成新的对象并改变对象引用。

### (3) String, StringBuilder及StringBuffer分别适合哪些场景?

- 1 - String: 如果要操作少量的数据
- 2 - StringBuilder: 单线程操作大量数据
- 3 - StringBuffer: 多线程操作大量数据, 要频繁对字符串内容进行修改

### 3. 设计不变类:

实现Vector, Matrix类, 可以进行向量、矩阵的基本运算、可以得到 (修改) Vector和Matrix中的元素, 如Vector的第k维, Matrix的第i,j位的值。

Vector类:

```
1 package immutableClass;
2
3 public class Vector {
4     private int length;
5     private double[] v;
6
7     public Vector(double[] v) { //构造函数
8         // TODO Auto-generated constructor stub
9         this.length = v.length;
10        this.v = v.clone();
11    }
12
13    public static Vector Plus(Vector a, Vector b) { //向量加法
14        if (a.length != b.length)
15            return null;
16        Vector c = new Vector(a.v);
17        for (int i = 0; i < c.length; ++i) {
18            c.v[i] = a.v[i] + b.v[i];
19        }
20        return c;
21    }
22
23    public static Vector Minus(Vector a, Vector b) { //向量减法
24        if (a.length != b.length)
25            return null;
26        Vector c = new Vector(a.v);
27        for (int i = 0; i < c.length; ++i) {
28            c.v[i] = a.v[i] - b.v[i];
29        }
30        return c;
31    }
32
33    public static double Dot(Vector a, Vector b) { //向量点乘
34        if (a.length != b.length) {
35            System.out.println("Cannot Dot Product!");
36            return 0;
37        }
38        double c = 0;
39        for (int i = 0; i < a.length; ++i) {
40            c += a.v[i] * b.v[i];
41        }
42        return c;
43    }
```

```

44
45     public Vector set(int k, double setTo) { //设置向量指定位置的值并返回修改后的向
        量
46         v[k] = setTo;
47         return this;
48     }
49
50     public double get(int k)
51     {
52         return v[k];
53     }
54
55     public int length() { //获得当前向量长度
56         return this.length;
57     }
58
59     public String toString() { //向量转换为相应的String（方便测试观察）
60         String string = "";
61         string += "{";
62         for (int i = 0; i < v.length; ++i) {
63             string += v[i];
64             if (i != v.length - 1)
65                 string += ", ";
66         }
67         string += "}";
68         return string;
69     }
70 }
71

```

### Matrix类:

```

1  package immutableClass;
2
3  public class Matrix {
4      private int row;
5      private int column;
6      private double[][] M;
7
8      public Matrix(double[][] m) { //构造函数
9          row = m.length;
10         column = m[0].length;
11         M = m.clone();
12         for (int i = 0; i < row; ++i) {
13             if (m[i].length != column) {
14                 System.out.println("The two-dimensional array is not a
        matrix! ");
15                 return;
16             }
17         }
18     }
19
20     public static Matrix Plus(Matrix a, Matrix b) { //矩阵加法
21         if ((a.row != b.row) || (a.column != b.column))
22             return null;
23         double[][] c = new double[a.row][a.column];
24         Matrix cMatrix = new Matrix(c);

```

```

25         for (int i = 0; i < cMatrix.row; ++i) {
26             for (int j = 0; j < cMatrix.column; ++j) {
27                 cMatrix.M[i][j] = a.M[i][j] + b.M[i][j];
28             }
29         }
30         return cMatrix;
31     }
32
33     public static Matrix Minus(Matrix a, Matrix b) { //矩阵减法
34         if ((a.row != b.row) || (a.column != b.column))
35             return null;
36         double[][] c = new double[a.row][a.column];
37         Matrix cMatrix = new Matrix(c);
38         for (int i = 0; i < cMatrix.row; ++i) {
39             for (int j = 0; j < cMatrix.column; ++j) {
40                 cMatrix.M[i][j] = a.M[i][j] - b.M[i][j];
41             }
42         }
43         return cMatrix;
44     }
45
46     public static Matrix Dot(Matrix a, Matrix b) { //矩阵乘法
47         if ((a.column != b.row))
48             return null;
49         double[][] c = new double[a.row][b.column];
50         Matrix cMatrix = new Matrix(c);
51         for (int i = 0; i < a.row; ++i) {
52             for (int j = 0; j < b.column; ++j) {
53                 cMatrix.M[i][j] = 0;
54                 for (int k = 0; k < b.row; ++k) {
55                     cMatrix.M[i][j] += a.M[i][k] * b.M[k][j];
56                 }
57             }
58         }
59         return cMatrix;
60     }
61
62     public Matrix set(int i, int j, double setTo) { //修改矩阵指定位置的值并返回修
改后的矩阵
63         this.M[i][j] = setTo;
64         return this;
65     }
66
67     public double get(int i, int j) { //获取矩阵指定位置的值
68         return this.M[i][j];
69     }
70
71     public int getRow() { //获取矩阵的行数
72         return this.row;
73     }
74
75     public int getColumn() { //获取矩阵的列数
76         return this.column;
77     }
78
79     public String toString() { //矩阵转换为相应String并返回（方便测试观察）
80         String string = "";
81         string += "{";

```

```

82         for (int i = 0; i < this.row; ++i) {
83             string += "{";
84             for (int j = 0; j < this.column; ++j) {
85                 string += this.M[i][j];
86                 if (j != this.column - 1)
87                     string += ", ";
88             }
89             string += "}";
90             if (i != this.row - 1)
91                 string += ",\n ";
92         }
93         string += "}";
94         return string;
95     }
96 }

```

## 实现UnmodifiableVector, UnmodifiableMatrix不可变类

### UnmodifiableVecor类:

```

1  package immutableClass;
2
3  public final class UnmodifiableVector {
4      private final int length;
5      private final double[] v;
6
7      public UnmodifiableVector(double[] v) { //构造函数
8          // TODO Auto-generated constructor stub
9          this.length = v.clone().length;
10         this.v = new double[length];
11         for (int i = 0; i < length; ++i) {
12             this.v[i] = v[i];
13         }
14     }
15
16     public static UnmodifiableVector Plus(UnmodifiableVector a,
17     UnmodifiableVector b) { //向量加法
18         if (a.length != b.length)
19             return null;
20         double[] c = new double[a.length];
21         for (int i = 0; i < c.length; ++i) {
22             c[i] = a.v[i] + b.v[i];
23         }
24         UnmodifiableVector cvector = new UnmodifiableVector(c);
25         return cvector;
26     }
27
28     public static UnmodifiableVector Minus(UnmodifiableVector a,
29     UnmodifiableVector b) { //向量减法
30         if (a.length != b.length)
31             return null;
32         double[] c = new double[a.length];
33         for (int i = 0; i < a.length; ++i) {
34             c[i] = a.v[i] - b.v[i];
35         }
36         UnmodifiableVector cvector = new UnmodifiableVector(c);
37         return cvector;
38     }
39 }

```

```

36     }
37
38     public static double Dot(UnmodifiableVector a, UnmodifiableVector b) {//
向量点乘
39         if (a.length != b.length) {
40             System.out.println("Cannot Dot Product!");
41             return 0;
42         }
43         double c = 0;
44         for (int i = 0; i < a.length; ++i) {
45             c += a.v[i] * b.v[i];
46         }
47         return c;
48     }
49
50     public UnmodifiableVector set(int k, double setTo) {//修改向量中特定的值并返
回新向量
51         UnmodifiableVector tVector = new UnmodifiableVector(this.v.clone());
52         tVector.v[k] = setTo;
53         return tVector;
54     }
55
56     public double get(int k) {//获得向量中指定点的值
57         return v.clone()[k];
58     }
59
60     public String toString() {//向量转换为String形式（方便测试观察）
61         String string = "";
62         string += "{";
63         for (int i = 0; i < v.length; ++i) {
64             string += v[i];
65             if (i != v.length - 1)
66                 string += ", ";
67         }
68         string += "}";
69         return string;
70     }
71 }

```

### UnmodifiableMatrix类:

```

1  package immutableClass;
2
3  public final class UnmodifiableMatrix {
4      private final int row;
5      private final int column;
6      private final double[][] M;
7
8      public UnmodifiableMatrix(double[][] m) {//构造函数
9          row = m.clone().length;
10         column = m.clone()[0].length;
11         this.M = new double[row][column];
12         for (int i = 0; i < row; ++i) {
13             if (m[i].length != column) {
14                 System.out.println("The two-dimensional array is not a
UnmodifiableMatrix! ");
15                 return;

```



```

16         }
17     }
18     for(int i = 0; i < row; ++i) {
19         for (int j = 0; j < column; ++j) {
20             this.M[i][j] = m[i][j];
21         }
22     }
23 }
24
25     public static UnmodifiableMatrix Plus(UnmodifiableMatrix a,
UnmodifiableMatrix b) { //矩阵加法
26         if ((a.row != b.row) || (a.column != b.column))
27             return null;
28         double[][] c = new double[a.row][a.column];
29         UnmodifiableMatrix cUnmodifiableMatrix = new UnmodifiableMatrix(c);
30         for (int i = 0; i < cUnmodifiableMatrix.row; ++i) {
31             for (int j = 0; j < cUnmodifiableMatrix.column; ++j) {
32                 cUnmodifiableMatrix.M[i][j] = a.M[i][j] + b.M[i][j];
33             }
34         }
35         return cUnmodifiableMatrix;
36     }
37
38     public static UnmodifiableMatrix Minus(UnmodifiableMatrix a,
UnmodifiableMatrix b) { //矩阵减法
39         if ((a.row != b.row) || (a.column != b.column))
40             return null;
41         double[][] c = new double[a.row][a.column];
42         UnmodifiableMatrix cUnmodifiableMatrix = new UnmodifiableMatrix(c);
43         for (int i = 0; i < cUnmodifiableMatrix.row; ++i) {
44             for (int j = 0; j < cUnmodifiableMatrix.column; ++j) {
45                 cUnmodifiableMatrix.M[i][j] = a.M[i][j] - b.M[i][j];
46             }
47         }
48         return cUnmodifiableMatrix;
49     }
50
51     public static UnmodifiableMatrix Dot(UnmodifiableMatrix a,
UnmodifiableMatrix b) { //矩阵乘法
52         if ((a.column != b.row))
53             return null;
54         double[][] c = new double[a.row][b.column];
55         UnmodifiableMatrix cUnmodifiableMatrix = new UnmodifiableMatrix(c);
56         for (int i = 0; i < a.row; ++i) {
57             for (int j = 0; j < b.column; ++j) {
58                 cUnmodifiableMatrix.M[i][j] = 0;
59                 for (int k = 0; k < b.row; ++k) {
60                     cUnmodifiableMatrix.M[i][j] += a.M[i][k] * b.M[j][k];
61                 }
62             }
63         }
64         return cUnmodifiableMatrix;
65     }
66
67     public UnmodifiableMatrix set(int i, int j, double setTo) { //修改矩阵指定
位置的值并返回修改后的矩阵
68         UnmodifiableMatrix matrix = new UnmodifiableMatrix(this.M.clone());
69         matrix.M[i][j] = setTo;

```

```

70         return matrix;
71     }
72
73     public double get(int i, int j) { //获得矩阵指定位置的值并返回
74         return this.M.clone()[i][j];
75     }
76
77     public String toString() { //矩阵转换成相对应的String并返回（方便测试观察）
78         String string = "";
79         string += "{";
80         for (int i = 0; i < this.row; ++i) {
81             string += "{";
82             for (int j = 0; j < this.column; ++j) {
83                 string += this.M[i][j];
84                 if (j != this.column - 1)
85                     string += ", ";
86             }
87             string += "}";
88             if (i != this.row - 1)
89                 string += ",\n ";
90         }
91         string += "}";
92         return string;
93     }
94 }

```

相比于前两个可变类，做出的修改有：

1. 类添加final修饰符，保证类不被继承。
2. 保证所有成员变量必须私有，并且加上final修饰
3. 不提供改变成员变量的方法，set方法新建一个向量/矩阵
4. 通过构造器初始化所有成员，进行深拷贝
5. 在get方法中，不要直接返回对象本身，而是克隆对象

实现MathUtils，含有静态方法，

- UnmodifiableVector getUnmodifiableVector (Vector v)
- UnmodifiableMatrix getUnmodifiableMatrix (Matrix m)

并进行测试说明

MathUtils类：

```

1  package immutableclass;
2
3  public class MathUtils {
4      public static UnmodifiableVector getUnmodifiableVector (Vector v)
5      {
6          double[] v = new double[v.length()];
7          for(int i = 0; i < v.length(); ++i) {
8              v[i] = v.get(i);
9          }
10         UnmodifiableVector unmodifiablevector = new UnmodifiableVector(v); //
            调用unmodifiablevector的构造函数实现转换
11         return unmodifiablevector;

```

```

12     }
13
14     public static UnmodifiableMatrix getUnmodifiableMatrix (Matrix m)
15     {
16         double[][] M = new double[m.getRow()][m.getColumn()];
17         for(int i = 0; i < m.getRow(); ++i) {
18             for(int j = 0; j < m.getColumn(); ++j) {
19                 M[i][j] = m.get(i, j);
20             }
21         }
22         UnmodifiableMatrix unmodifiableMatrix = new UnmodifiableMatrix(M);//
调用unmodifiableMatrix的构造函数实现转换
23         return unmodifiableMatrix;
24     }
25 }

```

### 测试说明:

```

1 package immutableClass;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         System.out.println("-----Vector-----");
7         double[] a = { 1, 2, 3, 4, 5 };
8         double[] b = { 1, 1, 1, 1, 1 };
9         double c = 0;
10        Vector aVector = new Vector(a);
11        Vector bVector = new Vector(b);
12        Vector cVector = Vector.Plus(aVector, bVector);
13        System.out.println();
14        System.out.println("Plus: " + cVector.toString());
15        System.out.println("aVector: " + aVector.toString());
16        System.out.println("bVector: " + bVector.toString());
17        cVector = Vector.Minus(aVector, bVector);
18        System.out.println();
19        System.out.println("Minus: " + cVector.toString());
20        System.out.println("aVector: " + aVector.toString());
21        System.out.println("bVector: " + bVector.toString());
22        c = Vector.Dot(aVector, bVector);
23        System.out.println();
24        System.out.println("Dot Product: " + c);
25        System.out.println("aVector: " + aVector.toString());
26        System.out.println("bVector: " + bVector.toString());
27        cVector.set(0, 100);
28        System.out.println();
29        System.out.println("Set cVector: " + cVector.toString());
30        System.out.println("changeTo: " + cVector.get(0));
31
32        System.out.println();
33        System.out.println("-----UnmodifiableVector-----");
34        UnmodifiableVector aUnmodifiableVector = new UnmodifiableVector(a);
35        UnmodifiableVector bunmodifiableVector = new UnmodifiableVector(b);
36        UnmodifiableVector cunmodifiableVector =
UnmodifiableVector.Plus(aUnmodifiableVector, bunmodifiableVector);

```

```

37         System.out.println();
38         System.out.println("Plus: " + cUnmodifiableVector.toString());
39         System.out.println("aUnmodifiableVector: " +
aUnmodifiableVector.toString());
40         System.out.println("bUnmodifiableVector: " +
bunmodifiableVector.toString());
41         cUnmodifiableVector = UnmodifiableVector.Minus(aUnmodifiableVector,
bunmodifiableVector);
42         System.out.println();
43         System.out.println("Minus: " + cUnmodifiableVector.toString());
44         System.out.println("aUnmodifiableVector: " +
aUnmodifiableVector.toString());
45         System.out.println("bUnmodifiableVector: " +
bunmodifiableVector.toString());
46         c = UnmodifiableVector.Dot(aUnmodifiableVector,
bunmodifiableVector);
47         System.out.println();
48         System.out.println("Dot Product: " + c);
49         System.out.println("aUnmodifiableVector: " +
aUnmodifiableVector.toString());
50         System.out.println("bunmodifiableVector: " +
bunmodifiableVector.toString());
51         cUnmodifiableVector.set(0, 100);
52         System.out.println();
53         System.out.println("Set cUnmodifiableVector: " +
cUnmodifiableVector.toString());
54         System.out.println("changeTo: " + cUnmodifiableVector.get(0));
55         UnmodifiableVector dunmodifiableVector = cUnmodifiableVector.set(0,
100);
56         System.out.println("Set cUnmodifiableVector to dunmodifiableVector:
" + dunmodifiableVector.toString());
57
58         System.out.println();
59         System.out.println("-----Matrix-----
-----");
60         double[][] aa = { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };
61         double[][] bb = { { 1, 0, 0 }, { 0, 1, 0 }, { 0, 0, 1 } };
62         Matrix aMatrix = new Matrix(aa);
63         Matrix bMatrix = new Matrix(bb);
64         Matrix cMatrix = Matrix.Plus(aMatrix, bMatrix);
65         System.out.println();
66         System.out.println("Plus:\n" + cMatrix.toString());
67         System.out.println("aMatrix:\n" + aMatrix.toString());
68         System.out.println("bMatrix:\n" + bMatrix.toString());
69         System.out.println();
70         cMatrix = Matrix.Minus(aMatrix, bMatrix);
71         System.out.println("Minus:\n" + cMatrix.toString());
72         System.out.println("aMatrix:\n" + aMatrix.toString());
73         System.out.println("bMatrix:\n" + bMatrix.toString());
74         System.out.println();
75         cMatrix = Matrix.Dot(aMatrix, bMatrix);
76         System.out.println("Dot Product:\n" + cMatrix);
77         System.out.println("aMatrix:\n" + aMatrix.toString());
78         System.out.println("bMatrix:\n" + bMatrix.toString());
79         System.out.println();
80         cMatrix.set(0, 0, 100);
81         System.out.println("Set cMatrix:\n" + cMatrix.toString());
82         System.out.println("changeTo:\n" + cMatrix.get(0, 0));

```

```

83
84     System.out.println();
85     System.out.println("-----UnmodifiableMatrix-----
-----");
86     UnmodifiableMatrix aUnmodifiableMatrix = new
UnmodifiableMatrix(aa);
87     UnmodifiableMatrix bUnmodifiableMatrix = new
UnmodifiableMatrix(bb);
88     UnmodifiableMatrix cUnmodifiableMatrix =
UnmodifiableMatrix.Plus(aUnmodifiableMatrix, bUnmodifiableMatrix);
89     System.out.println();
90     System.out.println("Plus:\n" + cUnmodifiableMatrix.toString());
91     System.out.println("aUnmodifiableMatrix:\n" +
aUnmodifiableMatrix.toString());
92     System.out.println("bUnmodifiableMatrix:\n" +
bUnmodifiableMatrix.toString());
93     System.out.println();
94     cUnmodifiableMatrix = UnmodifiableMatrix.Minus(aUnmodifiableMatrix,
bUnmodifiableMatrix);
95     System.out.println("Minus:\n" + cUnmodifiableMatrix.toString());
96     System.out.println("aUnmodifiableMatrix:\n" +
aUnmodifiableMatrix.toString());
97     System.out.println("bUnmodifiableMatrix:\n" +
bUnmodifiableMatrix.toString());
98     System.out.println();
99     cUnmodifiableMatrix = UnmodifiableMatrix.Dot(aUnmodifiableMatrix,
bUnmodifiableMatrix);
100    System.out.println("Dot Product:\n" + cUnmodifiableMatrix);
101    System.out.println("aUnmodifiableMatrix:\n" +
aUnmodifiableMatrix.toString());
102    System.out.println("bUnmodifiable Matrix:\n" +
bUnmodifiableMatrix.toString());
103    System.out.println();
104    cMatrix.set(0, 0, 100);
105    System.out.println("Set cMatrix:\n" +
cUnmodifiableMatrix.toString());
106    System.out.println("changeTo:\n" + cUnmodifiableMatrix.get(0, 0));
107    UnmodifiableMatrix dunmodifiableMatrix = cUnmodifiableMatrix.set(0,
0, 100);
108    System.out.println("Set cUnmodifiableMatrix to
dUnmodifiableMatrix:\n" + dunmodifiableMatrix.toString());
109
110    System.out.println();
111    System.out.println("-----MathUtils-----
-----");
112    System.out.println();
113    System.out.println("*****Vector*****");
114    Vector vector1 = new Vector(a);
115    Vector vector2 = new Vector(b);
116    System.out.println("vector1: " + vector1.toString());
117    System.out.println("vector2: " + vector2.toString());
118    System.out.println("change:");
119    vector1 = vector2;
120    vector2.set(2, 40);
121    System.out.println("vector1: " + vector1.toString());
122    System.out.println("vector2: " + vector2.toString());
123    System.out.println();
124    System.out.println("*****UnmodifiableVector*****");

```

```

125     UnmodifiableVector vector3 =
MathUtils.getUnmodifiableVector(vector1);
126     System.out.println("vector3: " + vector3.toString());
127     System.out.println("change:");
128     UnmodifiableVector vector4 = vector3.set(4, 80);
129     vector3 = vector3.set(1, 1000);
130     System.out.println("vector3: " + vector3.toString());
131     System.out.println("vector4: " + vector4.toString());
132
133     System.out.println();
134     System.out.println("*****Matrix*****");
135     Matrix matrix1 = new Matrix(aa);
136     Matrix matrix2 = new Matrix(bb);
137     System.out.println("matrix1:\n" + matrix1.toString());
138     System.out.println("matrix2:\n" + matrix2.toString());
139     System.out.println("change:");
140     matrix1 = matrix2;
141     matrix2.set(1, 1, 40);
142     System.out.println("matrix1:\n" + matrix1.toString());
143     System.out.println("matrix2:\n" + matrix2.toString());
144     System.out.println();
145     System.out.println("*****UnmodifiableMatrix*****");
146     UnmodifiableMatrix matrix3 =
MathUtils.getUnmodifiableMatrix(matrix1);
147     System.out.println("matrix3:\n" + matrix3.toString());
148     System.out.println("change:");
149     UnmodifiableMatrix matrix4 = matrix3.set(2, 2, 80);
150     matrix3 = matrix3.set(1,2, 1000);
151     System.out.println("matrix3:\n" + matrix3.toString());
152     System.out.println("matrix4:\n" + matrix4.toString());
153 }
154 }

```

输出结果:

```

1  -----Vector-----
2
3  Plus: {2.0, 3.0, 4.0, 5.0, 6.0}
4  avector: {1.0, 2.0, 3.0, 4.0, 5.0}
5  bvector: {1.0, 1.0, 1.0, 1.0, 1.0}
6
7  Minus: {0.0, 1.0, 2.0, 3.0, 4.0}
8  avector: {1.0, 2.0, 3.0, 4.0, 5.0}
9  bvector: {1.0, 1.0, 1.0, 1.0, 1.0}
10
11 Dot Product: 15.0
12 avector: {1.0, 2.0, 3.0, 4.0, 5.0}
13 bvector: {1.0, 1.0, 1.0, 1.0, 1.0}
14
15 Set cVector: {100.0, 1.0, 2.0, 3.0, 4.0}
16 changeTo: 100.0
17
18 -----UnmodifiableVector-----
19
20 Plus: {2.0, 3.0, 4.0, 5.0, 6.0}
21 aunmodifiableVector: {1.0, 2.0, 3.0, 4.0, 5.0}
22 bunmodifiableVector: {1.0, 1.0, 1.0, 1.0, 1.0}

```

```
23
24 Minus: {0.0, 1.0, 2.0, 3.0, 4.0}
25 aUnmodifiableVector: {1.0, 2.0, 3.0, 4.0, 5.0}
26 bUnmodifiableVector: {1.0, 1.0, 1.0, 1.0, 1.0}
27
28 Dot Product: 15.0
29 aUnmodifiableVector: {1.0, 2.0, 3.0, 4.0, 5.0}
30 bUnmodifiableVector: {1.0, 1.0, 1.0, 1.0, 1.0}
31
32 Set cUnmodifiableVector: {0.0, 1.0, 2.0, 3.0, 4.0}
33 changeTo: 0.0
34 Set cUnmodifiableVector to dUnmodifiableVector: {100.0, 1.0, 2.0, 3.0, 4.0}
35
36 -----Matrix-----
37
38 Plus:
39 {{2.0, 2.0, 3.0},
40  {4.0, 6.0, 6.0},
41  {7.0, 8.0, 10.0}}
42 aMatrix:
43 {{1.0, 2.0, 3.0},
44  {4.0, 5.0, 6.0},
45  {7.0, 8.0, 9.0}}
46 bMatrix:
47 {{1.0, 0.0, 0.0},
48  {0.0, 1.0, 0.0},
49  {0.0, 0.0, 1.0}}
50
51 Minus:
52 {{0.0, 2.0, 3.0},
53  {4.0, 4.0, 6.0},
54  {7.0, 8.0, 8.0}}
55 aMatrix:
56 {{1.0, 2.0, 3.0},
57  {4.0, 5.0, 6.0},
58  {7.0, 8.0, 9.0}}
59 bMatrix:
60 {{1.0, 0.0, 0.0},
61  {0.0, 1.0, 0.0},
62  {0.0, 0.0, 1.0}}
63
64 Dot Product:
65 {{1.0, 2.0, 3.0},
66  {4.0, 5.0, 6.0},
67  {7.0, 8.0, 9.0}}
68 aMatrix:
69 {{1.0, 2.0, 3.0},
70  {4.0, 5.0, 6.0},
71  {7.0, 8.0, 9.0}}
72 bMatrix:
73 {{1.0, 0.0, 0.0},
74  {0.0, 1.0, 0.0},
75  {0.0, 0.0, 1.0}}
76
77 Set cMatrix:
78 {{100.0, 2.0, 3.0},
79  {4.0, 5.0, 6.0},
80  {7.0, 8.0, 9.0}}
```

```

81  changeTo:
82  100.0
83
84  -----UnmodifiableMatrix-----
85
86  Plus:
87  {{2.0, 2.0, 3.0},
88   {4.0, 6.0, 6.0},
89   {7.0, 8.0, 10.0}}
90  aUnmodifiableMatrix:
91  {{1.0, 2.0, 3.0},
92   {4.0, 5.0, 6.0},
93   {7.0, 8.0, 9.0}}
94  bUnmodifiableMatrix:
95  {{1.0, 0.0, 0.0},
96   {0.0, 1.0, 0.0},
97   {0.0, 0.0, 1.0}}
98
99  Minus:
100 {{0.0, 2.0, 3.0},
101   {4.0, 4.0, 6.0},
102   {7.0, 8.0, 8.0}}
103 aUnmodifiableMatrix:
104 {{1.0, 2.0, 3.0},
105   {4.0, 5.0, 6.0},
106   {7.0, 8.0, 9.0}}
107 bUnmodifiableMatrix:
108 {{1.0, 0.0, 0.0},
109   {0.0, 1.0, 0.0},
110   {0.0, 0.0, 1.0}}
111
112 Dot Product:
113 {{1.0, 2.0, 3.0},
114   {4.0, 5.0, 6.0},
115   {7.0, 8.0, 9.0}}
116 aUnmodifiableMatrix:
117 {{1.0, 2.0, 3.0},
118   {4.0, 5.0, 6.0},
119   {7.0, 8.0, 9.0}}
120 bUnmodifiable Matrix:
121 {{1.0, 0.0, 0.0},
122   {0.0, 1.0, 0.0},
123   {0.0, 0.0, 1.0}}
124
125 Set cMatrix:
126 {{1.0, 2.0, 3.0},
127   {4.0, 5.0, 6.0},
128   {7.0, 8.0, 9.0}}
129 changeTo:
130 1.0
131 Set cUnmodifiableMatrix to dUnmodifiableMatrix:
132 {{100.0, 2.0, 3.0},
133   {4.0, 5.0, 6.0},
134   {7.0, 8.0, 9.0}}
135
136 -----MathUtils-----
137
138 *****Vector*****

```



```

139 vector1: {1.0, 2.0, 3.0, 4.0, 5.0}
140 vector2: {1.0, 1.0, 1.0, 1.0, 1.0}
141 change:
142 vector1: {1.0, 1.0, 40.0, 1.0, 1.0}
143 vector2: {1.0, 1.0, 40.0, 1.0, 1.0}
144
145 *****UnmodifiableVector*****
146 vector3: {1.0, 1.0, 40.0, 1.0, 1.0}
147 change:
148 vector3: {1.0, 1000.0, 40.0, 1.0, 1.0}
149 vector4: {1.0, 1.0, 40.0, 1.0, 80.0}
150
151 *****Matrix*****
152 matrix1:
153 {{1.0, 2.0, 3.0},
154  {4.0, 5.0, 6.0},
155  {7.0, 8.0, 9.0}}
156 matrix2:
157 {{1.0, 0.0, 0.0},
158  {0.0, 1.0, 0.0},
159  {0.0, 0.0, 1.0}}
160 change:
161 matrix1:
162 {{1.0, 0.0, 0.0},
163  {0.0, 40.0, 0.0},
164  {0.0, 0.0, 1.0}}
165 matrix2:
166 {{1.0, 0.0, 0.0},
167  {0.0, 40.0, 0.0},
168  {0.0, 0.0, 1.0}}
169
170 *****UnmodifiableMatrix*****
171 matrix3:
172 {{1.0, 0.0, 0.0},
173  {0.0, 40.0, 0.0},
174  {0.0, 0.0, 1.0}}
175 change:
176 matrix3:
177 {{1.0, 0.0, 0.0},
178  {0.0, 40.0, 1000.0},
179  {0.0, 0.0, 1.0}}
180 matrix4:
181 {{1.0, 0.0, 0.0},
182  {0.0, 40.0, 0.0},
183  {0.0, 0.0, 80.0}}

```

## part2 心得

通过本次作业，我对于Java的不变类本身有了更加深入的了解。实现不变类的时候最需要注意的是深拷贝，一开始写UnmodifiableMatrix的时候，以为直接用clone（）函数拷贝矩阵到当前Matrix就可以了，后来在测试过程中发现被赋值的矩阵和原矩阵依然指向同一片空间，修改一个会导致同时修改，和预期不符。查找资料后对于深拷贝的实现有了一定的认识，于是修改UnmodifiableMatrix的构造函数，用for循环给矩阵的每一个位置上的值赋值，避免了上述错误。

这次作业同时明白了要看清题意，因为一开始看错题目，花费了一天的时间重新实现了Java的Vector类，后来发现并不是题目要求的可以进行数值运算的Vector，不过也是通过一天的代码书写对于Java的动态数组Vector有了更加深刻的理解。