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：

```
public final class String
    implements java.io.Serializable, Comparable<String>, CharSequence
{
    /** The value is used for character storage. */
    private final byte[] value;
    /** The offset is the first index of the storage that is used. */
    private final int offset;
    /** The count is the number of characters in the String. */
    private final int count;
    /** Cache the hash code for the string */
    private int hash; // Default to 0
    ....
    public String(char value[]) {
         this.value = Arrays.copyOf(value, value.length);
     }
    ...
     public char[] toCharArray() {
     // Cannot use Arrays.copyOf because of class initialization order
issues
        char result[] = new char[value.length];
        System.arraycopy(value, 0, result, 0, value.length);
        return result;
    }
    ...
}
```

  - 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:

```
public final class Integer extends Number
        implements Comparable<Integer>, Constable, ConstantDesc {

    private final int value;
    private static final long serialVersionUID = 1360826667806852920L;
    ...
    public Integer(int value) {
```

```
        this.value = value;
    }
    ...
    public static Integer valueOf(int i) {
        if (i >= IntegerCache.low && i <= IntegerCache.high)
            return IntegerCache.cache[i + (-IntegerCache.low)];
        return new Integer(i);
    }
    ...
    static {
        // high value may be configured by property
        int h = 127;
        String integerCacheHighPropValue =

 sun.misc.VM.getSavedProperty("java.lang.Integer.IntegerCache.high");
        if (integerCacheHighPropValue != null) {
            try {
                int i = parseInt(integerCacheHighPropValue);
                i = Math.max(i, 127);
                // Maximum array size is Integer.MAX_VALUE
                h = Math.min(i, Integer.MAX_VALUE - (-low) -1);
            } catch( NumberFormatException nfe) {
                // If the property cannot be parsed into an int, ignore
it.
            }
        }
        high = h;

        cache = new Integer[(high - low) + 1];
        int j = low;
        for(int k = 0; k < cache.length; k++)
            cache[k] = new Integer(j++);

        // range [-128, 127] must be interned (JLS7 5.1.7)
        assert IntegerCache.high >= 127;
    }
    ...
}
```

- Integer类被final修饰，不可继承
- Ingeter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：

```java
public final class Long extends Number implements Comparable<Long> {
    private final longvalue;
    ...
    public static Long valueOf(long l) {
            final int offset = 128;
            if (l >= -128 && l <= 127) { // will cache
                return LongCache.cache[(int)l + offset];
            }
            return new Long(l);
    }
    ...
}
```

- 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StirngBuffer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?

```
- String：如果要操作少量的数据
- StringBuilder：单线程操作大量数据
- 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：

```java
package immutableClass;

public class Vector {
    private int length;
    private double[] v;

    public Vector(double[] v) {//构造函数
        // TODO Auto-generated constructor stub
        this.length = v.length;
        this.v = v.clone();
    }

    public static Vector Plus(Vector a, Vector b) {//向量加法
        if (a.length != b.length)
            return null;
        Vector c = new Vector(a.v);
        for (int i = 0; i < c.length; ++i) {
            c.v[i] = a.v[i] + b.v[i];
        }
        return c;
    }

    public static Vector Minus(Vector a, Vector b) {//向量减法
        if (a.length != b.length)
            return null;
        Vector c = new Vector(a.v);
        for (int i = 0; i < c.length; ++i) {
            c.v[i] = a.v[i] - b.v[i];
        }
        return c;
    }

    public static double Dot(Vector a, Vector b) {//向量点乘
        if (a.length != b.length) {
            System.out.println("Cannot Dot Product!");
            return 0;
        }
        double c = 0;
        for (int i = 0; i < a.length; ++i) {
            c += a.v[i] * b.v[i];
        }
        return c;
    }
```

```
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);
```

```
        for (int i = 0; i < cMatrix.row; ++i) {
            for (int j = 0; j < cMatrix.column; ++j) {
                cMatrix.M[i][j] = a.M[i][j] + b.M[i][j];
            }
        }
        return cMatrix;
    }

    public static Matrix Minus(Matrix a, Matrix b) {//矩阵减法
        if ((a.row != b.row) || (a.column != b.column))
            return null;
        double[][] c = new double[a.row][a.column];
        Matrix cMatrix = new Matrix(c);
        for (int i = 0; i < cMatrix.row; ++i) {
            for (int j = 0; j < cMatrix.column; ++j) {
                cMatrix.M[i][j] = a.M[i][j] - b.M[i][j];
            }
        }
        return cMatrix;
    }

    public static Matrix Dot(Matrix a, Matrix b) {//矩阵乘法
        if ((a.column != b.row))
            return null;
        double[][] c = new double[a.row][b.column];
        Matrix cMatrix = new Matrix(c);
        for (int i = 0; i < a.row; ++i) {
            for (int j = 0; j < b.column; ++j) {
                cMatrix.M[i][j] = 0;
                for (int k = 0; k < b.row; ++k) {
                    cMatrix.M[i][j] += a.M[i][k] * b.M[j][k];
                }
            }
        }
        return cMatrix;
    }

    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
        this.M[i][j] = setTo;
        return this;
    }

    public double get(int i, int j) {//获取矩阵指定位置的值
        return this.M[i][j];
    }

    public int getRow() {//获取矩阵的行数
        return this.row;
    }

    public int getColumn() {//获取矩阵的列数
        return this.column;
    }

    public String toString() {//矩阵转换为相应String并返回（方便测试观察）
        String string = "";
        string += "{";
```

```
        for (int i = 0; i < this.row; ++i) {
            string += "{";
            for (int j = 0; j < this.column; ++j) {
                string += this.M[i][j];
                if (j != this.column - 1)
                    string += ", ";
            }
            string += "}";
            if (i != this.row - 1)
                string += ",\n ";
        }
        string += "}";
        return string;
    }
}
```

实现UnmodifiableVector, UnmodifiableMatrix不可变类

UnmodifiableVecor类:

```
package immutableClass;

public final class UnmodifiableVector {
    private final int length;
    private final double[] v;

    public UnmodifiableVector(double[] v) {//构造函数
        // TODO Auto-generated constructor stub
        this.length = v.clone().length;
        this.v = new double[length];
        for (int i = 0; i < length; ++i) {
            this.v[i] = v[i];
        }
    }

    public static UnmodifiableVector Plus(UnmodifiableVector a,
UnmodifiableVector b) {//向量加法
        if (a.length != b.length)
            return null;
        double[] c = new double[a.length];
        for (int i = 0; i < c.length; ++i) {
            c[i] = a.v[i] + b.v[i];
        }
        UnmodifiableVector cVector = new UnmodifiableVector(c);
        return cVector;
    }

    public static UnmodifiableVector Minus(UnmodifiableVector a,
UnmodifiableVector b) {//向量减法
        if (a.length != b.length)
            return null;
        double[] c = new double[a.length];
        for (int i = 0; i < a.length; ++i) {
            c[i] = a.v[i] - b.v[i];
        }
        UnmodifiableVector cVector = new UnmodifiableVector(c);
        return cVector;
```

```
    }

    public static double Dot(UnmodifiableVector a, UnmodifiableVector b) {//向量点乘
        if (a.length != b.length) {
            System.out.println("Cannot Dot Product!");
            return 0;
        }
        double c = 0;
        for (int i = 0; i < a.length; ++i) {
            c += a.V[i] * b.V[i];
        }
        return c;
    }

    public UnmodifiableVector set(int k, double setTo) {//修改向量中特定的值并返回新向量
        UnmodifiableVector tVector = new UnmodifiableVector(this.V.clone());
        tVector.V[k] = setTo;
        return tVector;
    }

    public double get(int k) {//获得向量中指定点的值
        return V.clone()[k];
    }

    public String toString() {//向量转换为String形式（方便测试观察）
        String string = "";
        string += "{";
        for (int i = 0; i < V.length; ++i) {
            string += V[i];
            if (i != V.length - 1)
                string += ", ";
        }
        string += "}";
        return string;
    }
}
```

UnmodifiableMatrix类:

```
package immutableClass;

public final class UnmodifiableMatrix {
    private final int row;
    private final int column;
    private final double[][] M;

    public UnmodifiableMatrix(double[][] m) {//构造函数
        row = m.clone().length;
        column = m.clone()[0].length;
        this.M = new double[row][column];
        for (int i = 0; i < row; ++i) {
            if (m[i].length != column) {
                System.out.println("The two-dimensional array is not a UnmodifiableMatrix! ");
                return;
```

```
            }
        }
        for(int i = 0; i < row; ++i) {
            for (int j = 0; j < column; ++j) {
                this.M[i][j] = m[i][j];
            }
        }
    }

    public static UnmodifiableMatrix Plus(UnmodifiableMatrix a,
UnmodifiableMatrix b) {//矩阵加法
        if ((a.row != b.row) || (a.column != b.column))
            return null;
        double[][] c = new double[a.row][a.column];
        UnmodifiableMatrix cUnmodifiableMatrix = new UnmodifiableMatrix(c);
        for (int i = 0; i < cUnmodifiableMatrix.row; ++i) {
            for (int j = 0; j < cUnmodifiableMatrix.column; ++j) {
                cUnmodifiableMatrix.M[i][j] = a.M[i][j] + b.M[i][j];
            }
        }
        return cUnmodifiableMatrix;
    }

    public static UnmodifiableMatrix Minus(UnmodifiableMatrix a,
UnmodifiableMatrix b) {//矩阵减法
        if ((a.row != b.row) || (a.column != b.column))
            return null;
        double[][] c = new double[a.row][a.column];
        UnmodifiableMatrix cUnmodifiableMatrix = new UnmodifiableMatrix(c);
        for (int i = 0; i < cUnmodifiableMatrix.row; ++i) {
            for (int j = 0; j < cUnmodifiableMatrix.column; ++j) {
                cUnmodifiableMatrix.M[i][j] = a.M[i][j] - b.M[i][j];
            }
        }
        return cUnmodifiableMatrix;
    }

    public static UnmodifiableMatrix Dot(UnmodifiableMatrix a,
UnmodifiableMatrix b) {//矩阵乘法
        if ((a.column != b.row))
            return null;
        double[][] c = new double[a.row][b.column];
        UnmodifiableMatrix cUnmodifiableMatrix = new UnmodifiableMatrix(c);
        for (int i = 0; i < a.row; ++i) {
            for (int j = 0; j < b.column; ++j) {
                cUnmodifiableMatrix.M[i][j] = 0;
                for (int k = 0; k < b.row; ++k) {
                    cUnmodifiableMatrix.M[i][j] += a.M[i][k] * b.M[j][k];
                }
            }
        }
        return cUnmodifiableMatrix;
    }

    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
        UnmodifiableMatrix matrix = new UnmodifiableMatrix(this.M.clone());
        matrix.M[i][j] = setTo;
```

```
        return matrix;
    }

    public double get(int i, int j) {//获得矩阵指定位置的值并返回
        return this.M.clone()[i][j];
    }

    public String toString() {//矩阵转换成相对应的String并返回（方便测试观察）
        String string = "";
        string += "{";
        for (int i = 0; i < this.row; ++i) {
            string += "{";
            for (int j = 0; j < this.column; ++j) {
                string += this.M[i][j];
                if (j != this.column - 1)
                    string += ", ";
            }
            string += "}";
            if (i != this.row - 1)
                string += ",\n ";
        }
        string += "}";
        return string;
    }
}
```

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，而是克隆对象贝

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：

```
package immutableClass;

public class MathUtils {
    public static UnmodifiableVector getUnmodifiableVector (Vector v)
    {
        double[] V = new double[v.length()];
        for(int i = 0; i < v.length(); ++i) {
            V[i] = v.get(i);
        }
        UnmodifiableVector unmodifiableVector = new UnmodifiableVector(V);//
调用unmodifiableVector的构造函数实现转换
        return unmodifiableVector;
```

```
    }

    public static UnmodifiableMatrix getUnmodifiableMatrix (Matrix m)
    {
        double[][] M = new double[m.getRow()][m.getColumn()];
        for(int i = 0; i < m.getRow(); ++i) {
            for(int j = 0; j < m.getColumn(); ++j) {
                M[i][j] = m.get(i, j);
            }
        }
        UnmodifiableMatrix unmodifiableMatrix = new UnmodifiableMatrix(M);//调用unmodifiableMatrix的构造函数实现转换
        return unmodifiableMatrix;
    }
}
```

测试说明:

```
package immutableClass;

public class Main {

    public static void main(String[] args) {
        System.out.println("----------------------Vector-------------------");
        double[] a = { 1, 2, 3, 4, 5 };
        double[] b = { 1, 1, 1, 1, 1 };
        double c = 0;
        Vector aVector = new Vector(a);
        Vector bVector = new Vector(b);
        Vector cVector = Vector.Plus(aVector, bVector);
        System.out.println();
        System.out.println("Plus: " + cVector.toString());
        System.out.println("aVector: " + aVector.toString());
        System.out.println("bVector: " + bVector.toString());
        cVector = Vector.Minus(aVector, bVector);
        System.out.println();
        System.out.println("Minus: " + cVector.toString());
        System.out.println("aVector: " + aVector.toString());
        System.out.println("bVector: " + bVector.toString());
        c = Vector.Dot(aVector, bVector);
        System.out.println();
        System.out.println("Dot Product: " + c);
        System.out.println("aVector: " + aVector.toString());
        System.out.println("bVector: " + bVector.toString());
        cVector.set(0, 100);
        System.out.println();
        System.out.println("Set cVector: " + cVector.toString());
        System.out.println("changeTo: " + cVector.get(0));

        System.out.println();
        System.out.println("----------------UnmodifiableVector----------------");
        UnmodifiableVector aUnmodifiableVector = new UnmodifiableVector(a);
        UnmodifiableVector bUnmodifiableVector = new UnmodifiableVector(b);
        UnmodifiableVector cUnmodifiableVector = UnmodifiableVector.Plus(aUnmodifiableVector, bUnmodifiableVector);
```

```
        System.out.println();
        System.out.println("Plus: " + cUnmodifiableVector.toString());
        System.out.println("aUnmodifiableVector: " +
aUnmodifiableVector.toString());
        System.out.println("bUnmodifiableVector: " +
bUnmodifiableVector.toString());
        cUnmodifiableVector = UnmodifiableVector.Minus(aUnmodifiableVector,
bUnmodifiableVector);
        System.out.println();
        System.out.println("Minus: " + cUnmodifiableVector.toString());
        System.out.println("aUnmodifiableVector: " +
aUnmodifiableVector.toString());
        System.out.println("bUnmodifiableVector: " +
bUnmodifiableVector.toString());
        c = UnmodifiableVector.Dot(aUnmodifiableVector,
bUnmodifiableVector);
        System.out.println();
        System.out.println("Dot Product: " + c);
        System.out.println("aUnmodifiableVector: " +
aUnmodifiableVector.toString());
        System.out.println("bunmodifiableVector: " +
bUnmodifiableVector.toString());
        cUnmodifiableVector.set(0, 100);
        System.out.println();
        System.out.println("Set cUnmodifiableVector: " +
cUnmodifiableVector.toString());
        System.out.println("changeTo: " + cUnmodifiableVector.get(0));
        UnmodifiableVector dUnmodifiableVector = cUnmodifiableVector.set(0,
100);
        System.out.println("Set cUnmodifiableVector to dUnmodifiableVector:
" + dUnmodifiableVector.toString());

        System.out.println();
        System.out.println("--------------------Matrix---------------------
-----");
        double[][] aa = { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };
        double[][] bb = { { 1, 0, 0 }, { 0, 1, 0 }, { 0, 0, 1 } };
        Matrix aMatrix = new Matrix(aa);
        Matrix bMatrix = new Matrix(bb);
        Matrix cMatrix = Matrix.Plus(aMatrix, bMatrix);
        System.out.println();
        System.out.println("Plus:\n" + cMatrix.toString());
        System.out.println("aMatrix:\n" + aMatrix.toString());
        System.out.println("bMatrix:\n" + bMatrix.toString());
        System.out.println();
        cMatrix = Matrix.Minus(aMatrix, bMatrix);
        System.out.println("Minus:\n" + cMatrix.toString());
        System.out.println("aMatrix:\n" + aMatrix.toString());
        System.out.println("bMatrix:\n" + bMatrix.toString());
        System.out.println();
        cMatrix = Matrix.Dot(aMatrix, bMatrix);
        System.out.println("Dot Product:\n" + cMatrix);
        System.out.println("aMatrix:\n" + aMatrix.toString());
        System.out.println("bMatrix:\n" + bMatrix.toString());
        System.out.println();
        cMatrix.set(0, 0, 100);
        System.out.println("Set cMatrix:\n" + cMatrix.toString());
        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***********");
```

```
        UnmodifiableVector vector3 =
MathUtils.getUnmodifiableVector(vector1);
        System.out.println("vector3: " + vector3.toString());
        System.out.println("change:");
        UnmodifiableVector vector4 = vector3.set(4, 80);
        vector3 = vector3.set(1, 1000);
        System.out.println("vector3: " + vector3.toString());
        System.out.println("vector4: " + vector4.toString());

        System.out.println();
        System.out.println("***********Matrix************");
        Matrix matrix1 = new Matrix(aa);
        Matrix matrix2 = new Matrix(bb);
        System.out.println("matrix1:\n" + matrix1.toString());
        System.out.println("matrix2:\n" + matrix2.toString());
        System.out.println("change:");
        matrix1 = matrix2;
        matrix2.set(1, 1, 40);
        System.out.println("matrix1:\n" + matrix1.toString());
        System.out.println("matrix2:\n" + matrix2.toString());
        System.out.println();
        System.out.println("***********UnmodifiableMatrix************");
        UnmodifiableMatrix matrix3 =
MathUtils.getUnmodifiableMatrix(matrix1);
        System.out.println("matrix3:\n" + matrix3.toString());
        System.out.println("change:");
        UnmodifiableMatrix matrix4 = matrix3.set(2, 2, 80);
        matrix3 = matrix3.set(1,2, 1000);
        System.out.println("matrix3:\n" + matrix3.toString());
        System.out.println("matrix4:\n" + matrix4.toString());
    }
}
```

输出结果：

```
----------------------Vector-------------------------

Plus: {2.0, 3.0, 4.0, 5.0, 6.0}
aVector: {1.0, 2.0, 3.0, 4.0, 5.0}
bVector: {1.0, 1.0, 1.0, 1.0, 1.0}

Minus: {0.0, 1.0, 2.0, 3.0, 4.0}
aVector: {1.0, 2.0, 3.0, 4.0, 5.0}
bVector: {1.0, 1.0, 1.0, 1.0, 1.0}

Dot Product: 15.0
aVector: {1.0, 2.0, 3.0, 4.0, 5.0}
bVector: {1.0, 1.0, 1.0, 1.0, 1.0}

Set cVector: {100.0, 1.0, 2.0, 3.0, 4.0}
changeTo: 100.0

---------------UnmodifiableVector---------------------

Plus: {2.0, 3.0, 4.0, 5.0, 6.0}
aUnmodifiableVector: {1.0, 2.0, 3.0, 4.0, 5.0}
bUnmodifiableVector: {1.0, 1.0, 1.0, 1.0, 1.0}
```

```
Minus: {0.0, 1.0, 2.0, 3.0, 4.0}
aUnmodifiableVector: {1.0, 2.0, 3.0, 4.0, 5.0}
bUnmodifiableVector: {1.0, 1.0, 1.0, 1.0, 1.0}

Dot Product: 15.0
aUnmodifiableVector: {1.0, 2.0, 3.0, 4.0, 5.0}
bunmodifiableVector: {1.0, 1.0, 1.0, 1.0, 1.0}

Set cUnmodifiableVector: {0.0, 1.0, 2.0, 3.0, 4.0}
changeTo: 0.0
Set cUnmodifiableVector to dUnmodifiableVector: {100.0, 1.0, 2.0, 3.0, 4.0}

--------------------Matrix--------------------------

Plus:
{{2.0, 2.0, 3.0},
 {4.0, 6.0, 6.0},
 {7.0, 8.0, 10.0}}
aMatrix:
{{1.0, 2.0, 3.0},
 {4.0, 5.0, 6.0},
 {7.0, 8.0, 9.0}}
bMatrix:
{{1.0, 0.0, 0.0},
 {0.0, 1.0, 0.0},
 {0.0, 0.0, 1.0}}

Minus:
{{0.0, 2.0, 3.0},
 {4.0, 4.0, 6.0},
 {7.0, 8.0, 8.0}}
aMatrix:
{{1.0, 2.0, 3.0},
 {4.0, 5.0, 6.0},
 {7.0, 8.0, 9.0}}
bMatrix:
{{1.0, 0.0, 0.0},
 {0.0, 1.0, 0.0},
 {0.0, 0.0, 1.0}}

Dot Product:
{{1.0, 2.0, 3.0},
 {4.0, 5.0, 6.0},
 {7.0, 8.0, 9.0}}
aMatrix:
{{1.0, 2.0, 3.0},
 {4.0, 5.0, 6.0},
 {7.0, 8.0, 9.0}}
bMatrix:
{{1.0, 0.0, 0.0},
 {0.0, 1.0, 0.0},
 {0.0, 0.0, 1.0}}

Set cMatrix:
{{100.0, 2.0, 3.0},
 {4.0, 5.0, 6.0},
 {7.0, 8.0, 9.0}}
```

```
changeTo:
100.0

------------------UnmodifiableMatrix-----------------------

Plus:
{{2.0, 2.0, 3.0},
 {4.0, 6.0, 6.0},
 {7.0, 8.0, 10.0}}
aUnmodifiableMatrix:
{{1.0, 2.0, 3.0},
 {4.0, 5.0, 6.0},
 {7.0, 8.0, 9.0}}
bUnmodifiableMatrix:
{{1.0, 0.0, 0.0},
 {0.0, 1.0, 0.0},
 {0.0, 0.0, 1.0}}

Minus:
{{0.0, 2.0, 3.0},
 {4.0, 4.0, 6.0},
 {7.0, 8.0, 8.0}}
aUnmodifiableMatrix:
{{1.0, 2.0, 3.0},
 {4.0, 5.0, 6.0},
 {7.0, 8.0, 9.0}}
bUnmodifiableMatrix:
{{1.0, 0.0, 0.0},
 {0.0, 1.0, 0.0},
 {0.0, 0.0, 1.0}}

Dot Product:
{{1.0, 2.0, 3.0},
 {4.0, 5.0, 6.0},
 {7.0, 8.0, 9.0}}
aUnmodifiableMatrix:
{{1.0, 2.0, 3.0},
 {4.0, 5.0, 6.0},
 {7.0, 8.0, 9.0}}
bUnmodifiable  Matrix:
{{1.0, 0.0, 0.0},
 {0.0, 1.0, 0.0},
 {0.0, 0.0, 1.0}}

Set cMatrix:
{{1.0, 2.0, 3.0},
 {4.0, 5.0, 6.0},
 {7.0, 8.0, 9.0}}
changeTo:
1.0
Set cUnmodifiableMatrix to dUnmodifiableMatrix:
{{100.0, 2.0, 3.0},
 {4.0, 5.0, 6.0},
 {7.0, 8.0, 9.0}}

------------------MathUtils-----------------------

**********Vector*************
```

```
vector1: {1.0, 2.0, 3.0, 4.0, 5.0}
vector2: {1.0, 1.0, 1.0, 1.0, 1.0}
change:
vector1: {1.0, 1.0, 40.0, 1.0, 1.0}
vector2: {1.0, 1.0, 40.0, 1.0, 1.0}

***********UnmodifiableVector************
vector3: {1.0, 1.0, 40.0, 1.0, 1.0}
change:
vector3: {1.0, 1000.0, 40.0, 1.0, 1.0}
vector4: {1.0, 1.0, 40.0, 1.0, 80.0}

***********Matrix************
matrix1:
{{1.0, 2.0, 3.0},
 {4.0, 5.0, 6.0},
 {7.0, 8.0, 9.0}}
matrix2:
{{1.0, 0.0, 0.0},
 {0.0, 1.0, 0.0},
 {0.0, 0.0, 1.0}}
change:
matrix1:
{{1.0, 0.0, 0.0},
 {0.0, 40.0, 0.0},
 {0.0, 0.0, 1.0}}
matrix2:
{{1.0, 0.0, 0.0},
 {0.0, 40.0, 0.0},
 {0.0, 0.0, 1.0}}

***********UnmodifiableMatrix************
matrix3:
{{1.0, 0.0, 0.0},
 {0.0, 40.0, 0.0},
 {0.0, 0.0, 1.0}}
change:
matrix3:
{{1.0, 0.0, 0.0},
 {0.0, 40.0, 1000.0},
 {0.0, 0.0, 1.0}}
matrix4:
{{1.0, 0.0, 0.0},
 {0.0, 40.0, 0.0},
 {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。