Java - Homework3

1. JDK库中的不变类

String, Integer, Double都是不变类

它们的关键数据都是private final的, 对数据的修改都会创建一个新的对象

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

主要数据组织:

Strings:

```java
@Stable
private final byte[] value;
private final byte coder;
private int hash;
private static final long serialVersionUID = -6849794470754667710L;
static final boolean COMPACT_STRINGS;
static { COMPACT_STRINGS = true; }
private static final ObjectStreamField[] serialPersistentFields = new ObjectStreamField[0];
```

StringBuilder:

```java
static final long serialVersionUID = 4383685877147921099L;
```

StringBuffer:

```java
private transient String toStringCache;
static final long serialVersionUID = 3388685877147921107L;
```

StringBuffer和StringBuilder都继承了类AbstractStringBuilder, 因此它们都有数据:

```java
byte[] value;
byte coder;
int count;
private static final byte[] EMPTYVALUE = new byte[0];
```

功能实现:

String所有会改变数据的操作都睡新建一个对象, 然后返回新的对象或是让引用指向新的对象

而StringBuilder和StringBuffer会对数据原地修改

这样设计使String成为不变类, 使用更加方便, 安全, 而StringBuilder和StringBuffer可以在频繁修改数据时发挥其高效性, 使程序效率更高, 运行更快.

StringBuffer的操作前都有Synchronized, 让StringBuffer所有的操作都是同步的, 保证了线程安全.

```java
String s1 = "Welcome to Java";
String s2 = new String("Welcome to Java");
String s3 = "Welcome to Java";
System.out.println("s1 == s2 is " + (s1 == s2));
System.out.println("s1 == s3 is " + (s1 == s3));
```

`s1`创建时, 在常量池中新建一个String对象然后引用它, `s2`创建时, 在堆中新建一个String对象然后引用它, `s3`创建时, 因为常量池中已经有`“Welcome to Java”`的String对象, 所以直接引用它.

既然`==`是对所引用对象是否相同的比较, `s1==s2`返回false, `s1==s3`返回true.

3. 设计不变类

Vector:

```java
public class myVector {
    private double elems[];
//    public int capacity;
//    public int size;

    public myVector(double[] x){
        this.elems = x.clone();
    }
    public myVector(int dim){
        this.elems = new double[dim];
    }
    // return the dimension of this vector
    public int dim(){
        return this.elems.length;
    }
    public myVector copy(){
        return new myVector(this.elems.clone());
    }
    // set the value of some element by index
    public void setElemAt(int dim, double x){
        elems[dim] = x;
    }
    // return the result of a scalar multiplication on this vector
    public myVector numMul(int k){
        myVector res = this.copy();
        for(int i=0; i<elems.length; i++)
            res.elems[i] *= k;
        return res;
    }
    // return the sum of two vectors, new object
    public myVector add(myVector x){
        if(x.dim()!=this.dim())
            return null;
        myVector res = x.copy();
        for(int i=0; i<elems.length; i++)
            res.elems[i] += this.elems[i];
        return res;
    }
    // return the inner product of two vectors, new object
    public double dotMul(myVector x){
        if(x.dim()!=this.dim())
            System.err.println("Two vectors in inner production should have the same dimension.");
        double res = 0;
        for(int i=0; i<this.dim()&&i<x.dim(); i++)
            res += this.elems[i] * x.elems[i];
        return res;
    }
    // return a element by index
    public double elemAt(int index){
        return elems[index];
    }
    // like "equals" in String
    public boolean equals(myVector x){
        if(this.dim() != x.dim()) return false;
        for(int i=0; i<this.dim(); i++)
            if(this.elems[i] != x.elems[i]) return false;
        return true;
    }
    @Override // transform the vector into a string, serve for print
    public String toString(){
        String s = "[";
        for(int i=0; i<this.elems.length; i++)
            s += (i==0?"":"  ") + this.elems[i];
        s += "]";
        return s;
    }

    static public void main(String[] args){
        double[] a1 = {1,2,3,4,5};
        double[] a2 = {2,3,4,5,6};
        myVector v1 = new myVector(a1);
        myVector v2 = new myVector(a2);

        myVector v3 = v1.numMul(2);
        myVector v4 = v1.add(v2);
        double v5 = v1.dotMul(v2);

        System.out.println("v1: " + v1);
        System.out.println("v2: " + v2);
        System.out.println("v1.dim: " + v1.dim());
        System.out.println("v1.elemAt(0)" + v1.elemAt(0));
        v1.setElemAt(0,12);
        System.out.println("v1 after set(0,12): " + v1);
        v1.setElemAt(0,1);
        System.out.println("v1 after set(0,1): " + v1);
        System.out.println("2*v1: " + v1.numMul(2));
        System.out.println("v1+v2: " + v1.add(v2));
        System.out.println("v1·v2: " + v1.dotMul(v2));
    }
}
```

```
// test results
v1: [1.0  2.0  3.0  4.0  5.0]
v2: [2.0  3.0  4.0  5.0  6.0]
v1.dim: 5
v1.elemAt(0)1.0
v1 after set(0,12): [12.0  2.0  3.0  4.0  5.0]
v1 after set(0,1): [1.0  2.0  3.0  4.0  5.0]
2*v1: [2.0  4.0  6.0  8.0  10.0]
v1+v2: [3.0  5.0  7.0  9.0  11.0]
v1·v2: 70.0
```

Matrix:

```
public class myMatrix {
    private double[][] elems;

    public myMatrix(int m, int n){
        elems = new double[m][n];
    }
    public myMatrix(double[][] x){
        for(int i=0; i<x.length; i++){
            if(x[i].length != x[0].length) {
                System.err.println("Each row in a matrix should have the same length.");
                return;
            }
        }
        this.elems = x.clone();
    }
    public myMatrix copy(){
        double[][] aa = new double[this.rows()][this.cols()];
        myMatrix res = new myMatrix(aa);
        return res;
    }
    // return the number of rows of the matrix
    public int rows(){
        return elems.length;
    }
```

```java
    // return the number of cols of the matrix
    public int cols(){
        if(elems.length<=0) return 0;
        return elems[0].length;
    }
    // set the value of some element by index
    public void setElemAt(int i, int j, double x){
        this.elems[i][j] = x;
    }
    // return a element by index
    public double elemAt(int i, int j){
        return this.elems[i][j];
    }
    // return the result of a scalar multiplication on this matrix
    public myMatrix numMul(int k){
        myMatrix res = new myMatrix(this.rows(), this.cols());
        for(int i=0; i<this.rows(); i++)
            for(int j=0; j<this.cols(); j++)
                res.elems[i][j] = this.elems[i][j] * k;
        return res;
    }
    // return the sum of two matrices, new object
    public myMatrix add(myMatrix x){
        if(this.rows()!=x.rows() || this.cols()!=x.cols())
            return null;
        myMatrix res = new myMatrix(this.rows(), this.cols());
        for(int i=0; i<res.rows(); i++)
            for(int j=0; j<res.cols(); j++)
                res.elems[i][j] = this.elems[i][j] + x.elems[i][j];
        return res;
    }
    // return the product of two matrices, new object
    // require the rows of the left equals the cols of the right
    public myMatrix mul(myMatrix x){
        if(this.cols()!=x.rows())
            return null;
        myMatrix res = new myMatrix(this.rows(), x.cols());
        for(int i=0; i<res.rows(); i++)
            for(int j=0; j<res.cols(); j++)
                for(int k=0; k<this.cols(); k++)
                    res.elems[i][j] += this.elems[i][k] * x.elems[k][j];
        return res;
    }
    // transpose of a matrix, return a new object
    public myMatrix transpose(){
        myMatrix res = new myMatrix(this.cols(), this.rows());
        for(int i=0; i<res.rows(); i++)
            for(int j=0; j<res.cols(); j++)
                res.elems[i][j] = this.elems[j][i];
        return res;
    }
    // just like the "equals" in String
    public boolean equals(myMatrix x){
        if(this.rows()!=x.rows() || this.cols()!=x.cols()) return false;
        for(int i=0; i<this.rows(); i++)
            for(int j=0; j<this.cols(); j++)
                if(this.elems[i][j]!=x.elems[i][j]) return false;
        return true;
    }
    @Override // transform the matrix into a string, serve for print
    public String toString(){
        String s = "[";
        for(int i=0; i<this.rows(); i++){
            s += (i==0?"":" ") + "[";
            for(int j=0; j<this.cols(); j++)
                s += (j==0?"":"  ") + this.elems[i][j];
            s += (i==this.rows()-1 ? "]" : "]\n");
        }
        s += "]\n";
        return s;
    }

    // test cases
    public static void main(String[] args){
        double[][] a1 = {
                {1,2},
                {3,4},
                {5,6},
        };
        double[][] a2 = {
                {2,3},
                {4,5},
                {6,7},
        };
        double[][] a3 = {
                {1,2,3},
                {4,5,6}
        };
        myMatrix m1 = new myMatrix(a1);
        myMatrix m2 = new myMatrix(a2);
        myMatrix m3 = new myMatrix(a3);

        myMatrix m4 = m1.numMul(2);
        myMatrix m5 = m1.add(m2);
        myMatrix m6 = m1.mul(m3);
        myMatrix m7 = m2.transpose();

        System.out.println("m1:\n" + m1);
        System.out.println("m2:\n" + m2);
        System.out.println("m3:\n" + m3);
        System.out.println("m1.rows: " + m1.rows());
        System.out.println("m1.cols: " + m1.cols());
        System.out.println("m1[0][0]: " + m1.elemAt(0,0));
        m1.setElemAt(0,0,0);
        System.out.println("m1 after set(0,0,0):\n" + m1);
        m1.setElemAt(0,0,1);
        System.out.println("m1 after set(0,0,1):\n" + m1);

        System.out.println("2*m1:\n" + m4);
        System.out.println("m1+m2:\n" + m5);
        System.out.println("m1·m3:\n" + m6);
        System.out.println("m2.transpose():\n" + m7);
    }
```

```
}
```

```
// test results:
m1:
[[1.0  2.0]
 [3.0  4.0]
 [5.0  6.0]]

m2:
[[2.0  3.0]
 [4.0  5.0]
 [6.0  7.0]]

m3:
[[1.0  2.0  3.0]
 [4.0  5.0  6.0]]

m1.rows: 3
m1.cols: 2
m1[0][0]: 1.0
m1 after set(0,0,0):
[[0.0  2.0]
 [3.0  4.0]
 [5.0  6.0]]

m1 after set(0,0,1):
[[1.0  2.0]
 [3.0  4.0]
 [5.0  6.0]]

2*m1:
[[2.0  4.0]
 [6.0  8.0]
 [10.0  12.0]]

m1+m2:
[[3.0  5.0]
 [7.0  9.0]
 [11.0  13.0]]

m1·m3:
[[9.0  12.0  15.0]
 [19.0  26.0  33.0]
 [29.0  40.0  51.0]]

m2.transpose():
[[2.0  4.0  6.0]
 [3.0  5.0  7.0]]
```

UnmodifiableVector:

```java
public class UnmodifiableVector {
    private final double[] elms;

    public UnmodifiableVector(double[] _elms) {
        this.elms = _elms.clone();
    }
    // return the dimension of this vector
    public int dim(){
        return this.elms.length;
    }
    // get some element by index
    public double elemAt(int index){
        return this.elms[index];
    }
    // return a new vector after some position being set
    public UnmodifiableVector setElemAt(int index, double x){
        double[] a = this.elms.clone();
        a[index] = x;
        return new UnmodifiableVector(a);
    }
    // return a number product with another vector
    public UnmodifiableVector numMul(int k){
        double[] a = this.elms.clone();
        for(int i=0; i<a.length; i++) a[i] *= k;
        return new UnmodifiableVector(a);
    }
    // return the sum with another vector
    public UnmodifiableVector add(UnmodifiableVector x){
        if(x.dim()!=this.dim()){
            System.err.println("Two vectors in addition should have the same dimension.");
            return null;
        }
        double[] a = this.elms.clone();
        for(int i=0; i<a.length; i++)
            a[i] += x.elms[i];
        return new UnmodifiableVector(a);
    }
    // return the inner product with another vector
    public double dotMul(UnmodifiableVector x){
        if(x.dim()!=this.dim())
            System.err.println("Two vectors in inner production should have the same dimension.");
        double res = 0;
        for(int i=0; i<this.dim()&&i<x.dim(); i++)
            res += this.elms[i] * x.elms[i];
        return res;
    }
    // like "equals" in String
    public boolean equals(UnmodifiableVector x){
        if(this.dim() != x.dim()) return false;
        for(int i=0; i<this.dim(); i++)
            if(this.elms[i] != x.elms[i]) return false;
        return true;
    }
    @Override // transform the vector into a string, serve for print
    public String toString(){
        String s = "[";
        for(int i=0; i<this.elms.length; i++)
            s += (i==0?"":"  ") + this.elms[i];
        s += "]";
        return s;
    }

    // test cases
    public static void main(String[] args){
        double[] a1 = {1,2,3,4,5};
        double[] a2 = {2,3,4,5,6};
        UnmodifiableVector v1 = new UnmodifiableVector(a1);
        UnmodifiableVector v2 = new UnmodifiableVector(a2);

        System.out.println("v1.dim: " + v1.dim());
        System.out.println("v1[0]: " + v1.elemAt(0));
        System.out.println("v1.setElemAt(0,12): " + v1.setElemAt(0,12));
        System.out.println("v1 after v1.setElemAt(0,12): " + v1);
        System.out.println("2*v1: " + v1.numMul(2));
        System.out.println("v1+v2: " + v1.add(v2));
        System.out.println("v1·v2: " + v1.dotMul(v2));
        System.out.println("v1: " + v1);
    }
}
```

```
// test results:
v1.dim: 5
v1[0]: 1.0
v1.setElemAt(0,12): [12.0  2.0  3.0  4.0  5.0]
v1 after v1.setElemAt(0,12): [1.0  2.0  3.0  4.0  5.0]
2*v1: [2.0  4.0  6.0  8.0  10.0]
v1+v2: [3.0  5.0  7.0  9.0  11.0]
v1·v2: 70.0
v1: [1.0  2.0  3.0  4.0  5.0]
```

UnmodifiableMatrix:

```java
public class UnmodifiableMatrix {
    private final double[][] elms;

    public UnmodifiableMatrix(double[][] _elms){
        this.elms = new double[_elms.length][];
        for(int i=0; i<this.elms.length; i++)
            this.elms[i] = _elms[i].clone();
    }
    // return the number of rows of the matrix
    public int rows(){
        return this.elms.length;
    }
    // return the number of cols of the matrix
    public int cols(){
        if(this.elms.length<=0) return 0;
        return this.elms[0].length;
    }
    // get some element by index
    public double elemAt(int i, int j){
        return this.elms[i][j];
    }
    // return a new matrix after some position being set
    public UnmodifiableMatrix setElemAt(int i, int j, double x){
        double[][] a = new double[this.rows()][];
        for(int k=0; k<a.length; k++){
            a[k] = this.elms[k].clone();
        }
        a[i][j] = x;
        return new UnmodifiableMatrix(a);
    }
    // return a multiple of the matrix
    public UnmodifiableMatrix numMul(int k){
        double[][] a = new double[this.rows()][];
        for(int i=0; i<a.length; i++)
            a[i] = this.elms[i].clone();
        for(int i=0; i<a.length; i++)
            for(int j=0; j<a[0].length; j++)
                a[i][j] *= k;
        return new UnmodifiableMatrix(a);
    }
    // return the sum with another matrix
    public UnmodifiableMatrix add(UnmodifiableMatrix x){
        if(this.rows()!=x.rows() || this.cols()!=x.cols()){
            System.err.println("Two matrices in addition should have the same shape.");
            return null;
        }
        double[][] a = new double[this.rows()][];
        for(int k=0; k<a.length; k++)
            a[k] = this.elms[k].clone();
        for(int i=0; i<a.length; i++)
            for(int j=0; j<a[0].length; j++)
                a[i][j] += x.elms[i][j];
        return new UnmodifiableMatrix(a);
    }
    // return the product with another matrix multiplied on its right
    public UnmodifiableMatrix mul(UnmodifiableMatrix x){
        if(this.cols()!=x.rows()){
            System.err.println("The cols of the left and the rows of the right");
            return null;
        }
        double[][] a = new double[this.rows()][x.cols()];
        for(int i=0; i<a.length; i++)
            for(int j=0; j<a[0].length; j++)
                for(int k=0; k<this.cols(); k++)
                    a[i][j] += this.elms[i][k]*x.elms[k][j];
        return new UnmodifiableMatrix(a);
    }
    // return the transposition of the matrix
    public UnmodifiableMatrix transpose(){
        double[][] a = new double[this.cols()][this.rows()];
        for(int i=0; i<this.cols(); i++)
            for(int j=0; j<this.rows(); j++)
                a[i][j] = this.elms[j][i];
        return new UnmodifiableMatrix(a);
    }
    // just like the "equals" in String
    public boolean equals(UnmodifiableMatrix x){
        if(this.rows()!=x.rows() || this.cols()!=x.cols()) return false;
        for(int i=0; i<this.rows(); i++)
            for(int j=0; j<this.cols(); j++)
                if(this.elms[i][j]!=x.elms[i][j]) return false;
        return true;
    }
    @Override // transform the matrix into a string, serve for print
    public String toString(){
        String s = "[";
        for(int i=0; i<this.rows(); i++){
            s += (i==0?"":" ") + "[";
            for(int j=0; j<this.cols(); j++)
                s += (j==0?"":"  ") + this.elms[i][j];
            s += (i==this.rows()-1 ? "]" : "]\n");
        }
        s += "]\n";
        return s;
    }

    // test cases
    public static void main(String[] args){
        double[][] a1 = {
                {1,2,3},
                {4,5,6}
        };
        double[][] a2 = {
                {2,3,4},
                {5,6,7}
        };
        double[][] a3 = {
                {1,2},
                {3,4},
                {5,6}
        };
        UnmodifiableMatrix m1 = new UnmodifiableMatrix(a1);
        UnmodifiableMatrix m2 = new UnmodifiableMatrix(a2);
        UnmodifiableMatrix m3 = new UnmodifiableMatrix(a3);

        System.out.println("m1.rows(): " + m1.rows());
        System.out.println("m1.cols(): " + m1.cols());
        System.out.println("m1[0][0]: " + m1.elemAt(0,0));
        System.out.println("m1.setElemAt(0,0,12):\n" + m1.setElemAt(0,0,12));
        System.out.println("m1 after m1.setElemAt(0,0,12):\n" + m1);

        System.out.println("2*m1:\n" + m1.numMul(2));
        System.out.println("m1+m2:\n" + m1.add(m2));
        System.out.println("m1·m3:\n" + m1.mul(m3));
        System.out.println("m1.transpose():\n" + m1.transpose());
        System.out.println("m1:\n" + m1);
    }
}
```

```
// test results
m1.rows(): 2
```

```
m1.cols(): 3
m1[0][0]: 1.0
m1.setElemAt(0,0,12):
[[12.0  2.0  3.0]
 [4.0  5.0  6.0]]

m1 after m1.setElemAt(0,0,12):
[[1.0  2.0  3.0]
 [4.0  5.0  6.0]]

2*m1:
[[2.0  4.0  6.0]
 [8.0  10.0  12.0]]

m1+m2:
[[3.0  5.0  7.0]
 [9.0  11.0  13.0]]

m1·m3:
[[22.0  28.0]
 [49.0  64.0]]

m1.transpose():
[[1.0  4.0]
 [2.0  5.0]
 [3.0  6.0]]

m1:
[[1.0  2.0  3.0]
 [4.0  5.0  6.0]]
```

MathUtils:

```java
public class MathUtils {
    // convert a vector into unmodifiable vector
    public static UnmodifiableVector
getUnmodifiableVector(myVector v){
        double[] a = new double[v.dim()];
        for(int i=0; i<a.length; i++)
            a[i] = v.elemAt(i);
        return new UnmodifiableVector(a);
    }
    // convert a matrix into unmodifiable matrix
    public static UnmodifiableMatrix getUnmodifiableMatrix(myMatrix m){
        double[][] a = new double[m.rows()][m.cols()];
        for(int i=0; i<m.rows(); i++)
            for(int j=0; j<m.cols(); j++)
                a[i][j] = m.elemAt(i,j);
        return new UnmodifiableMatrix(a);
    }
    // test
    public static void main(String args[]){
        double[] a1 = {0,1,2,3,4,5,6,7,8,9};
        myVector v1 = new myVector(a1);
        UnmodifiableVector uv1 = getUnmodifiableVector(v1);
        System.out.println("vector conversion finished:\n" + uv1);

        double[][] a2 = {
                {1,2,3},
                {4,5,6}
        };
        myMatrix m1 = new myMatrix(a2);
        UnmodifiableMatrix um1 = getUnmodifiableMatrix(m1);
        System.out.println("matrix conversion finished:\n" + um1);
    }
}
```

```
// test results
vector conversion finished:
[0.0   1.0   2.0   3.0   4.0   5.0   6.0   7.0   8.0   9.0]
matrix conversion finished:
[[1.0   2.0   3.0]
 [4.0   5.0   6.0]]
```