

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:

```
1  @Stable
2  private final byte[] value;
3  private final byte coder;
4  private int hash;
5  private static final long serialVersionUID =
    -6849794470754667710L;
6  static final boolean COMPACT_STRINGS;
7  static { COMPACT_STRINGS = true; }
8  private static final ObjectStreamField[]
    serialPersistentFields = new ObjectStreamField[0];
```

StringBuilder:

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

StringBuffer:

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

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

```
1 byte[] value;
2 byte coder;
3 int count;
4 private static final byte[] EMPTYVALUE = new byte[0];
```

功能实现:

String所有会改变数据的操作都新建一个对象, 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.

```
1 String s1 = "welcome to Java";
2 String s2 = new String("welcome to Java");
3 String s3 = "welcome to Java";
4 System.out.println("s1 == s2 is " + (s1 == s2));
5 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:

```
1 public class myVector {
2     private double elems[];
3     // public int capacity;
4     // public int size;
5
6     public myVector(double[] x){
```

```

7         this.elems = x.clone();
8     }
9     public myVector(int dim){
10         this.elems = new double[dim];
11     }
12     // return the dimension of this vector
13     public int dim(){
14         return this.elems.length;
15     }
16     public myVector copy(){
17         return new myVector(this.elems.clone());
18     }
19     // set the value of some element by index
20     public void setElemAt(int dim, double x){
21         elems[dim] = x;
22     }
23     // return the result of a scalar multiplication on
this vector
24     public myVector numMul(int k){
25         myVector res = this.copy();
26         for(int i=0; i<elems.length; i++){
27             res.elems[i] *= k;
28         }
29         return res;
30     }
31     // return the sum of two vectors, new object
32     public myVector add(myVector x){
33         if(x.dim()!=this.dim())
34             return null;
35         myVector res = x.copy();
36         for(int i=0; i<elems.length; i++){
37             res.elems[i] += this.elems[i];
38         }
39         return res;
40     }
41     // return the inner product of two vectors, new
object
42     public double dotMul(myVector x){
43         if(x.dim()!=this.dim())
44             System.err.println("Two vectors in inner
production should have the same dimension.");
45         double res = 0;
46         for(int i=0; i<this.dim()&& i<x.dim(); i++){
47             res += this.elems[i] * x.elems[i];
48         }
49         return res;
50     }

```

```

47     }
48     // return a element by index
49     public double elemAt(int index){
50         return elems[index];
51     }
52     // like "equals" in String
53     public boolean equals(myVector x){
54         if(this.dim() != x.dim()) return false;
55         for(int i=0; i<this.dim(); i++)
56             if(this.elems[i] != x.elems[i]) return false;
57         return true;
58     }
59     @Override // transform the vector into a string,
serve for print
60     public String toString(){
61         String s = "[";
62         for(int i=0; i<this.elems.length; i++)
63             s += (i==0?"":" ") + this.elems[i];
64         s += "]";
65         return s;
66     }
67
68     static public void main(String[] args){
69         double[] a1 = {1,2,3,4,5};
70         double[] a2 = {2,3,4,5,6};
71         myVector v1 = new myVector(a1);
72         myVector v2 = new myVector(a2);
73
74         myVector v3 = v1.numMul(2);
75         myVector v4 = v1.add(v2);
76         double v5 = v1.dotMul(v2);
77
78         System.out.println("v1: " + v1);
79         System.out.println("v2: " + v2);
80         System.out.println("v1.dim: " + v1.dim());
81         System.out.println("v1.elemAt(0)" +
v1.elemAt(0));
82         v1.setElemAt(0,12);
83         System.out.println("v1 after set(0,12): " + v1);
84         v1.setElemAt(0,1);
85         System.out.println("v1 after set(0,1): " + v1);
86         System.out.println("2*v1: " + v1.numMul(2));
87         System.out.println("v1+v2: " + v1.add(v2));

```

```

88         System.out.println("v1.v2: " + v1.dotMul(v2));
89     }
90 }

```

```

1 // test results
2 v1: [1.0  2.0  3.0  4.0  5.0]
3 v2: [2.0  3.0  4.0  5.0  6.0]
4 v1.dim: 5
5 v1.elementAt(0)1.0
6 v1 after set(0,12): [12.0  2.0  3.0  4.0  5.0]
7 v1 after set(0,1): [1.0  2.0  3.0  4.0  5.0]
8 2*v1: [2.0  4.0  6.0  8.0  10.0]
9 v1+v2: [3.0  5.0  7.0  9.0  11.0]
10 v1.v2: 70.0

```

Matrix:

```

1 public class myMatrix {
2     private double[][] elems;
3
4     public myMatrix(int m, int n){
5         elems = new double[m][n];
6     }
7     public myMatrix(double[][] x){
8         for(int i=0; i<x.length; i++){
9             if(x[i].length != x[0].length) {
10                 System.err.println("Each row in a matrix
should have the same length.");
11                 return;
12             }
13         }
14         this.elems = x.clone();
15     }
16     public myMatrix copy(){
17         double[][] aa = new double[this.rows()]
[this.cols()];
18         myMatrix res = new myMatrix(aa);
19         return res;
20     }
21     // return the number of rows of the matrix
22     public int rows(){
23         return elems.length;
24     }

```

```

25 // return the number of cols of the matrix
26 public int cols(){
27     if(elems.length<=0) return 0;
28     return elems[0].length;
29 }
30 // set the value of some element by index
31 public void setElemAt(int i, int j, double x){
32     this.elems[i][j] = x;
33 }
34 // return a element by index
35 public double elemAt(int i, int j){
36     return this.elems[i][j];
37 }
38 // return the result of a scalar multiplication on
this matrix
39 public myMatrix numMul(int k){
40     myMatrix res = new myMatrix(this.rows(),
this.cols());
41     for(int i=0; i<this.rows(); i++)
42         for(int j=0; j<this.cols(); j++)
43             res.elems[i][j] = this.elems[i][j] * k;
44     return res;
45 }
46 // return the sum of two matrices, new object
47 public myMatrix add(myMatrix x){
48     if(this.rows()!=x.rows() ||
this.cols()!=x.cols())
49         return null;
50     myMatrix res = new myMatrix(this.rows(),
this.cols());
51     for(int i=0; i<res.rows(); i++)
52         for(int j=0; j<res.cols(); j++)
53             res.elems[i][j] = this.elems[i][j] +
x.elems[i][j];
54     return res;
55 }
56 // return the product of two matrices, new object
57 // require the rows of the left equals the cols of
the right
58 public myMatrix mul(myMatrix x){
59     if(this.cols()!=x.rows())
60         return null;

```

```

61         myMatrix res = new myMatrix(this.rows(),
x.cols());
62         for(int i=0; i<res.rows(); i++)
63             for(int j=0; j<res.cols(); j++)
64                 for(int k=0; k<this.cols(); k++)
65                     res.elems[i][j] += this.elems[i][k]
* x.elems[k][j];
66         return res;
67     }
68     // transpose of a matrix, return a new object
69     public myMatrix transpose(){
70         myMatrix res = new myMatrix(this.cols(),
this.rows());
71         for(int i=0; i<res.rows(); i++)
72             for(int j=0; j<res.cols(); j++)
73                 res.elems[i][j] = this.elems[j][i];
74         return res;
75     }
76     // just like the "equals" in String
77     public boolean equals(myMatrix x){
78         if(this.rows()!=x.rows() ||
this.cols()!=x.cols()) return false;
79         for(int i=0; i<this.rows(); i++)
80             for(int j=0; j<this.cols(); j++)
81                 if(this.elems[i][j]!=x.elems[i][j])
return false;
82         return true;
83     }
84     @Override // transform the matrix into a string,
serve for print
85     public String toString(){
86         String s = "[";
87         for(int i=0; i<this.rows(); i++){
88             s += (i==0?"":" ") + "[";
89             for(int j=0; j<this.cols(); j++)
90                 s += (j==0?"":" ") + this.elems[i][j];
91             s += (i==this.rows()-1 ? "]" : "]\n");
92         }
93         s += "]\n";
94         return s;
95     }
96
97     // test cases

```

```

98     public static void main(String[] args){
99         double[][] a1 = {
100             {1,2},
101             {3,4},
102             {5,6},
103         };
104         double[][] a2 = {
105             {2,3},
106             {4,5},
107             {6,7},
108         };
109         double[][] a3 = {
110             {1,2,3},
111             {4,5,6}
112         };
113         myMatrix m1 = new myMatrix(a1);
114         myMatrix m2 = new myMatrix(a2);
115         myMatrix m3 = new myMatrix(a3);
116
117         myMatrix m4 = m1.numMul(2);
118         myMatrix m5 = m1.add(m2);
119         myMatrix m6 = m1.mul(m3);
120         myMatrix m7 = m2.transpose();
121
122         System.out.println("m1:\n" + m1);
123         System.out.println("m2:\n" + m2);
124         System.out.println("m3:\n" + m3);
125         System.out.println("m1.rows: " + m1.rows());
126         System.out.println("m1.cols: " + m1.cols());
127         System.out.println("m1[0][0]: " +
m1.elementAt(0,0));
128         m1.setElemAt(0,0,0);
129         System.out.println("m1 after set(0,0,0):\n" +
m1);
130         m1.setElemAt(0,0,1);
131         System.out.println("m1 after set(0,0,1):\n" +
m1);
132
133         System.out.println("2*m1:\n" + m4);
134         System.out.println("m1+m2:\n" + m5);
135         System.out.println("m1.m3:\n" + m6);
136         System.out.println("m2.transpose():\n" + m7);
137     }

```



```
1 // test results:
2 m1:
3 [[1.0  2.0]
4  [3.0  4.0]
5  [5.0  6.0]]
6
7 m2:
8 [[2.0  3.0]
9  [4.0  5.0]
10 [6.0  7.0]]
11
12 m3:
13 [[1.0  2.0  3.0]
14  [4.0  5.0  6.0]]
15
16 m1.rows: 3
17 m1.cols: 2
18 m1[0][0]: 1.0
19 m1 after set(0,0,0):
20 [[0.0  2.0]
21  [3.0  4.0]
22  [5.0  6.0]]
23
24 m1 after set(0,0,1):
25 [[1.0  2.0]
26  [3.0  4.0]
27  [5.0  6.0]]
28
29 2*m1:
30 [[2.0  4.0]
31  [6.0  8.0]
32  [10.0 12.0]]
33
34 m1+m2:
35 [[3.0  5.0]
36  [7.0  9.0]
37  [11.0 13.0]]
38
39 m1.m3:
40 [[9.0 12.0 15.0]
41  [19.0 26.0 33.0]]
```

```
42 [29.0  40.0  51.0]]
43
44 m2.transpose():
45 [[2.0  4.0  6.0]
46  [3.0  5.0  7.0]]
```

UnmodifiableVector:

```
1 public class UnmodifiableVector {
2     private final double[] elms;
3
4     public UnmodifiableVector(double[] _elms) {
5         this.elms = _elms.clone();
6     }
7     // return the dimension of this vector
8     public int dim(){
9         return this.elms.length;
10    }
11    // get some element by index
12    public double elemAt(int index){
13        return this.elms[index];
14    }
15    // return a new vector after some position being set
16    public UnmodifiableVector setElemAt(int index, double
17    x){
18        double[] a = this.elms.clone();
19        a[index] = x;
20        return new UnmodifiableVector(a);
21    }
22    // return a number product with another vector
23    public UnmodifiableVector numMul(int k){
24        double[] a = this.elms.clone();
25        for(int i=0; i<a.length; i++) a[i] *= k;
26        return new UnmodifiableVector(a);
27    }
28    // return the sum with another vector
29    public UnmodifiableVector add(UnmodifiableVector x){
30        if(x.dim()!=this.dim()){
31            System.err.println("Two vectors in addition
32            should have the same dimension.");
33            return null;
34        }
35        double[] a = this.elms.clone();
```

```

34         for(int i=0; i<a.length; i++)
35             a[i] += x.elms[i];
36         return new UnmodifiableVector(a);
37     }
38     // return the inner product with another vector
39     public double dotMul(UnmodifiableVector x){
40         if(x.dim()!=this.dim())
41             System.err.println("Two vectors in inner
production should have the same dimension.");
42         double res = 0;
43         for(int i=0; i<this.dim()&& i<x.dim(); i++)
44             res += this.elms[i] * x.elms[i];
45         return res;
46     }
47     // like "equals" in String
48     public boolean equals(UnmodifiableVector x){
49         if(this.dim() != x.dim()) return false;
50         for(int i=0; i<this.dim(); i++)
51             if(this.elms[i] != x.elms[i]) return false;
52         return true;
53     }
54     @Override // transform the vector into a string,
serve for print
55     public String toString(){
56         String s = "[";
57         for(int i=0; i<this.elms.length; i++)
58             s += (i==0?"":", ") + this.elms[i];
59         s += "]";
60         return s;
61     }
62
63     // test cases
64     public static void main(String[] args){
65         double[] a1 = {1,2,3,4,5};
66         double[] a2 = {2,3,4,5,6};
67         UnmodifiableVector v1 = new
UnmodifiableVector(a1);
68         UnmodifiableVector v2 = new
UnmodifiableVector(a2);
69
70         System.out.println("v1.dim: " + v1.dim());
71         System.out.println("v1[0]: " + v1.elementAt(0));

```

```

72         System.out.println("v1.setElemAt(0,12): " +
    v1.setElemAt(0,12));
73         System.out.println("v1 after v1.setElemAt(0,12):
    " + v1);
74         System.out.println("2*v1: " + v1.numMul(2));
75         System.out.println("v1+v2: " + v1.add(v2));
76         System.out.println("v1.v2: " + v1.dotMul(v2));
77         System.out.println("v1: " + v1);
78     }
79 }

```

```

1 // test results:
2 v1.dim: 5
3 v1[0]: 1.0
4 v1.setElemAt(0,12): [12.0  2.0  3.0  4.0  5.0]
5 v1 after v1.setElemAt(0,12): [1.0  2.0  3.0  4.0  5.0]
6 2*v1: [2.0  4.0  6.0  8.0  10.0]
7 v1+v2: [3.0  5.0  7.0  9.0  11.0]
8 v1.v2: 70.0
9 v1: [1.0  2.0  3.0  4.0  5.0]

```

UnmodifiableMatrix:

```

1 public class UnmodifiableMatrix {
2     private final double[][] elms;
3
4     public UnmodifiableMatrix(double[][] _elms){
5         this.elms = new double[_elms.length][];
6         for(int i=0; i<this.elms.length; i++)
7             this.elms[i] = _elms[i].clone();
8     }
9     // return the number of rows of the matrix
10    public int rows(){
11        return this.elms.length;
12    }
13    // return the number of cols of the matrix
14    public int cols(){
15        if(this.elms.length<=0) return 0;
16        return this.elms[0].length;
17    }
18    // get some element by index
19    public double elemAt(int i, int j){
20        return this.elms[i][j];

```

```

21     }
22     // return a new matrix after some position being set
23     public UnmodifiableMatrix setElemAt(int i, int j,
double x){
24         double[][] a = new double[this.rows()][];
25         for(int k=0; k<a.length; k++){
26             a[k] = this.elms[k].clone();
27         }
28         a[i][j] = x;
29         return new UnmodifiableMatrix(a);
30     }
31     // return a multiple of the matrix
32     public UnmodifiableMatrix numMul(int k){
33         double[][] a = new double[this.rows()][];
34         for(int i=0; i<a.length; i++)
35             a[i] = this.elms[i].clone();
36         for(int i=0; i<a.length; i++)
37             for(int j=0; j<a[0].length; j++)
38                 a[i][j] *= k;
39         return new UnmodifiableMatrix(a);
40     }
41     // return the sum with another matrix
42     public UnmodifiableMatrix add(UnmodifiableMatrix x){
43         if(this.rows()!=x.rows() ||
this.cols()!=x.cols()){
44             System.err.println("Two matrices in addition
should have the same shape.");
45             return null;
46         }
47         double[][] a = new double[this.rows()][];
48         for(int k=0; k<a.length; k++)
49             a[k] = this.elms[k].clone();
50         for(int i=0; i<a.length; i++)
51             for(int j=0; j<a[0].length; j++)
52                 a[i][j] += x.elms[i][j];
53         return new UnmodifiableMatrix(a);
54     }
55     // return the product with another matrix multiplied
on its right
56     public UnmodifiableMatrix mul(UnmodifiableMatrix x){
57         if(this.cols()!=x.rows()){
58             System.err.println("The cols of the left and
the rows of the right");

```

```

59         return null;
60     }
61     double[][] a = new double[this.rows()]
[x.cols()];
62     for(int i=0; i<a.length; i++)
63         for(int j=0; j<a[0].length; j++)
64             for(int k=0; k<this.cols(); k++)
65                 a[i][j] += this.elms[i][k]*x.elms[k]
[j];
66     return new UnmodifiableMatrix(a);
67 }
68 // return the transposition of the matrix
69 public UnmodifiableMatrix transpose(){
70     double[][] a = new double[this.cols()]
[this.rows()];
71     for(int i=0; i<this.cols(); i++)
72         for(int j=0; j<this.rows(); j++)
73             a[i][j] = this.elms[j][i];
74     return new UnmodifiableMatrix(a);
75 }
76 // just like the "equals" in String
77 public boolean equals(UnmodifiableMatrix x){
78     if(this.rows()!=x.rows() ||
this.cols()!=x.cols()) return false;
79     for(int i=0; i<this.rows(); i++)
80         for(int j=0; j<this.cols(); j++)
81             if(this.elms[i][j]!=x.elms[i][j]) return
false;
82     return true;
83 }
84 @Override // transform the matrix into a string,
serve for print
85 public String toString(){
86     String s = "[";
87     for(int i=0; i<this.rows(); i++){
88         s += (i==0?"":" ") + "[";
89         for(int j=0; j<this.cols(); j++)
90             s += (j==0?"":" ") + this.elms[i][j];
91         s += (i==this.rows()-1 ? "]" : "]\n");
92     }
93     s += "]\n";
94     return s;
95 }

```

```

96
97     // test cases
98     public static void main(String[] args){
99         double[][] a1 = {
100             {1,2,3},
101             {4,5,6}
102         };
103         double[][] a2 = {
104             {2,3,4},
105             {5,6,7}
106         };
107         double[][] a3 = {
108             {1,2},
109             {3,4},
110             {5,6}
111         };
112         UnmodifiableMatrix m1 = new
UnmodifiableMatrix(a1);
113         UnmodifiableMatrix m2 = new
UnmodifiableMatrix(a2);
114         UnmodifiableMatrix m3 = new
UnmodifiableMatrix(a3);
115
116         System.out.println("m1.rows(): " + m1.rows());
117         System.out.println("m1.cols(): " + m1.cols());
118         System.out.println("m1[0][0]: " +
m1.elemAt(0,0));
119         System.out.println("m1.setElemAt(0,0,12):\n" +
m1.setElemAt(0,0,12));
120         System.out.println("m1 after
m1.setElemAt(0,0,12):\n" + m1);
121
122         System.out.println("2*m1:\n" + m1.numMul(2));
123         System.out.println("m1+m2:\n" + m1.add(m2));
124         System.out.println("m1.m3:\n" + m1.mul(m3));
125         System.out.println("m1.transpose():\n" +
m1.transpose());
126         System.out.println("m1:\n" + m1);
127     }
128 }

```

```

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

```

```

3  m1.cols(): 3
4  m1[0][0]: 1.0
5  m1.setElemAt(0,0,12):
6  [[12.0  2.0  3.0]
7   [4.0   5.0  6.0]]
8
9  m1 after m1.setElemAt(0,0,12):
10 [[1.0  2.0  3.0]
11    [4.0  5.0  6.0]]
12
13 2*m1:
14 [[2.0  4.0  6.0]
15    [8.0 10.0 12.0]]
16
17 m1+m2:
18 [[3.0  5.0  7.0]
19    [9.0 11.0 13.0]]
20
21 m1.m3:
22 [[22.0 28.0]
23    [49.0 64.0]]
24
25 m1.transpose():
26 [[1.0  4.0]
27    [2.0  5.0]
28    [3.0  6.0]]
29
30 m1:
31 [[1.0  2.0  3.0]
32    [4.0  5.0  6.0]]

```

MathUtils:

```

1  public class MathUtils {
2      // convert a vector into unmodifiable vector
3      public static UnmodifiableVector
getUnmodifiableVector(myVector v){
4          double[] a = new double[v.dim()];
5          for(int i=0; i<a.length; i++)
6              a[i] = v.elemAt(i);
7          return new UnmodifiableVector(a);
8      }
9      // convert a matrix into unmodifiable matrix

```



```

10     public static UnmodifiableMatrix
getUnmodifiableMatrix(myMatrix m){
11         double[][] a = new double[m.rows()][m.cols()];
12         for(int i=0; i<m.rows(); i++)
13             for(int j=0; j<m.cols(); j++)
14                 a[i][j] = m.elemAt(i,j);
15         return new UnmodifiableMatrix(a);
16     }
17     // test
18     public static void main(String args[]){
19         double[] a1 = {0,1,2,3,4,5,6,7,8,9};
20         myVector v1 = new myVector(a1);
21         UnmodifiableVector uv1 =
getUnmodifiableVector(v1);
22         System.out.println("vector conversion
finished:\n" + uv1);
23
24         double[][] a2 = {
25             {1,2,3},
26             {4,5,6}
27         };
28         myMatrix m1 = new myMatrix(a2);
29         UnmodifiableMatrix um1 =
getUnmodifiableMatrix(m1);
30         System.out.println("matrix conversion
finished:\n" + um1);
31     }
32 }

```

```

1 // test results
2 vector conversion finished:
3 [0.0  1.0  2.0  3.0  4.0  5.0  6.0  7.0  8.0  9.0]
4 matrix conversion finished:
5 [[1.0  2.0  3.0]
6  [4.0  5.0  6.0]]

```