Дисциплина: *СД и ООП* **Тема:** *Вектор*
Специалност: *КН 1 курс, 3 и 4 групи* **Дата**: *31.03.2010 г.*

Задача 1: Интерфейсът **BigInteger** представя операции на *голямо цяло десетично число без знак*:

```
public interface BigInteger {
        public int length(); //Връща дължината на текущото число
        public BigInteger add(BigInteger arg); //Събира текущото число с числото arg
        public BigInteger mult(BigInteger arg); //Умножава текущото число по числото arg
        public int compareTo(BigInteger arg); //Сравнява текущото число с числото arg
        public boolean equals(Object obj); //Проверява дали текущото число и числото arg съвпадат
        public Object clone(); //Създава копие на текущото число
        public String toString(); //Преобразува текущото число в низ
}
```

Класът **VectorBigInteger** реализира интерфейса **BigInteger**, като цифрите на голямо цяло десетично число $N = a_n\ base^n + a_{n-1}\ base^{n-1} + \ldots + a_1\ base + a_0$, $0 \le a_i < base$, се съхраняват във вектор $digits = (a_0,\ldots,a_n)$ и $base = 10^k$, $k \ge 1$.

При реализацията са използвани родовият вектор **java.util.Vector<E>** и интерфейсът **java.lang.Comparable<T>** - виж http://java.sun.com/javase/6/docs/api/.

```
import java.util.*;

public class VectorBigInteger implements BigInteger, Comparable<VectorBigInteger > {
        //Data
        private Vector<Integer> digits;       //digits = (a0,...,an)
        public static int base = 10;          //base = 10^k, k = 1, 2, ...

        //Constructors
        public  VectorBigInteger() { this("0"); } //N=0
        public  VectorBigInteger(BigInteger arg) { this(arg.toString()); } //N=arg
        public  VectorBigInteger(String arg) {…}  //arg="an...a0" => N=an…a0

        //Private method
        private static int howLong(int n) {
                int d = 0;
                do {
                        d++;
                        n /= 10;
                }while(n != 0);
                return d;
        }

        //BigInteger methods implementation
        public int length() { return digits.size() * (howLong(base)-1); }

        public BigInteger add(BigInteger arg) {
                VectorBigInteger Int1 = this;
                VectorBigInteger Int2 = new VectorBigInteger(arg.toString());
                Vector<Integer> left = Int1.digits;
                Vector<Integer> right = Int2.digits;

                int max = Math.max(left.size(),right.size());
                int min = Math.min(left.size(),right.size());

                VectorBigInteger w = new VectorBigInteger();
                Vector<Integer> result = w.digits = new Vector<Integer>();

                int d, c = 0;
                for(int i = 0; i < min; i++) {
```

```
                d = left.get(i) + right.get(i) + c;
                result.add(result.size(),d % base);
                c = d / base;
        }

        Vector<Integer> temp;
        if(max == left.size()) temp = left;
        else temp = right;
        for(int i = min; i < temp.size(); i++) {
                d = temp.get(i) + c;
                result.add(result.size(),d % base);
                c = d / base;
        }
        if(c != 0) result.add(result.size(),c);

        return w;
}

 public BigInteger mult(BigInteger arg) {
        VectorBigInteger Int1 = this;
        VectorBigInteger Int2 = new VectorBigInteger(arg.toString());
        Vector<Integer> left = Int1.digits;
        Vector<Integer> right = Int2.digits;

        VectorBigInteger w = new VectorBigInteger();
        Vector<Integer> result = w.digits = new Vector<Integer>();;

        for(int i = 0; i < left.size() + right.size()-1; i++)
                result.add(0,0);

        int d, c;
        for(int i = 0; i < left.size(); i++) {
                d = left.get(i);
                for(int j = 0; j < right.size(); j++) {
                        c = result.get(i+j) + d * right.get(j);
                        result.set(i+j,c);
                }
        }

        c = 0;
        for(int i = 0; i < result.size(); i++) {
                d = result.get(i) + c;
                result.set(i,d % base);
                c = d / base;
        }

        if(c != 0) result.add(result.size(),c);

        return w;
   }

   public int compareTo(BigInteger arg) {
        VectorBigInteger left = this;
        VectorBigInteger right = new VectorBigInteger(arg.toString());
        return left.compareTo(right);
   }

   //Comparable method implementation
   public int compareTo(VectorBigInteger arg) {
        String left = this.toString();
        String right = arg.toString();
        int diff = Math.abs(left.length() - right.length());
        if(left.length() < right.length())
                for(int i = 0; i <= diff; i++)
                        left = "0" + left;
```

```
            else if(right.length() < left.length())
                for(int i = 0; i <= diff; i++)
                    right = "0" + right;
            return left.compareTo(right);
        }

        //Override methods - inherited from class Object
        public boolean equals(Object obj){ System.out.println("Not implemented!"); return true; }
        public Object clone(){ System.out.println("Not implemented!"); return null; }

        public String toString() {
            String result = "";     //digits = (a0,...,an) => result="an...a0"
            int k = howLong(base)-1;
            for(int i = 0; i < digits.size(); i++){
                String w = (digits.get(i)).toString();
                while(k > w.length())
                    w = 0  + w;
                result = w + result;
            }
            return result;
        }

        //Constants
        public static final VectorBigInteger ZERO = new VectorBigInteger("0");
        public static final VectorBigInteger ONE = new VectorBigInteger("1");
        public static final VectorBigInteger TEN = new VectorBigInteger("10");
}
```

Задача 2: Да се реализира родов вектор **VectorSorter<E>** с възможност за сортиране на елементите.

```
import java.util.*;

public class VectorSorter<E> extends Vector<E> {
    //Methods inherited from class Vector<E>

    //New method
    public void sort() {
        boolean flag=true;
        while(flag) {
            flag=false;
            for(int i = 0; i < super.size() - 1; i++) {
                E e1 = super.get(i);
                E e2 = super.get(i+1);
                if(((Comparable)e1).compareTo(e2) > 0) {
                    flag = true;
                    super.set(i,e2);
                    super.set(i+1,e1);
                }
            }
        }
    }
}
```

Задача 3: Да се реализира нареден родов вектор **SortedVector<E>**.

```
import java.util.*;

public class SortedVector<E> extends Vector<E> {
    //Methods inherited from class Vector<E>

    //New method
    public void addInSortedVector(E x) {
        if(super.isEmpty() || ((Comparable)x).compareTo(super.get(0)) <= 0) super.add(0,x);
```

```
                else {
                        int i;
                        for(i = 0; i < super.size(); i++) {
                                if(((Comparable)x).compareTo(super.get(i)) <= 0) {
                                        super.add(i,x);
                                        break;
                                }
                        }
                        if(i == super.size()) super.add(x);
                }
        }
}
```

Задачи за упражнение:

1.Да се реализират клас **VectorStack** с елементи от тип **Object** без ограничение за броя на елментите и клас **FancyVectorStack**, съгласно спецификацията:

```
public class VectorStack implements Stack {
        //Представяне на стек чрез вектор

        //Конструктори
        public VectorStack() {…} //Създава празен стек
        public VectorStack(int n) {…} //Създава празен стек с начален капацитет n
        public VectorStack(Object[] arg) {…} //Създава стек от елементите на масива arg, като първият
                                //елемент на масива е на дъното на стека, а последният елемент – на върха
        //Реализация на методите на интерфейса Stack
        //Други методи
}

public class FancyVectorStack extends VectorStack implements FancyCollection {
        //Конструктори
        public FancyVectorStack() {…} //Създава празен стек
        public FancyVectorStack(int n) {…} //Създава празен стек с капацитет n
        public FancyVectorStack(FancyCollection arg) {…} //Създава стек от елементите на  колекцията arg
        public FancyVectorStack(Object[] arg) {…} //Създава стек от елементите на масива arg, като първият
                                //елемент на масива е на дъното на стека, а последният елемент – на върха
        //Наследяване на методите на класа VectorStack
        //Реализация на методите на интерфейса FancyCollection
        //Предефиниране на методите, наследени от клас Object
}
```

2.Да се реализират клас **VectorDeque** с елементи от тип **Object** без ограничение за броя на елментите и клас **FancyVectorDeque**, съгласно спецификацията:

```
public class VectorDeque implements Deque {
        //Представяне на дек чрез вектор

        //Конструктори
        public VectorDeque() {…} //Създава празен дек
        public VectorDeque(int n) {…} //Създава празен дек с начален капацитет n
        public VectorDeque(Object[] arg) {…} //Създава дек от елементите на масива arg, като първият
                                //елемент на масива е в началото на дека, а последният елемент – накрая
        //Реализация на методите на интерфейса Deque
        //Други методи
}

public class FancyVectorDeque extends VectorDeque implements FancyCollection {
        //Конструктори
        public FancyVectorDeque() {…} //Създава празен дек
        public FancyVectorDeque(FancyCollection arg) {…} //Създава дек от елементите на  колекцията arg
        public FancyVectorDeque(Object[] arg) {…} //Създава дек от елементите на масива arg, като първият
                                //елемент на масива е в началото на дека, а последният елемент - накрая
        //Наследяване на методите на интерфейса Deque
```

```
        //Реализация на методите на интерфейса FancyCollection
        //Предефиниране на методите, наследени от клас Object
}
```

Упътване: При реализацията може да се използва собствен клас **Vector** или родовия вектор **java.util.Vector<E>**. Във втория случай, декларирането на променлива-вектор с елементи от тип **Object**, изглежда по следния начин:

java.util.Vector *ИмеНаПроменлива;*

3.Дадени са родовите интерфейси:

```
public interface GDeque<E> {
        public  boolean isEmpty();
        public void addFront(E x);
        public void addRear(E x);
        public E removeFront();
        public E removeRear();
        public E getFirst();
        public E getLast();
}

public interface GFancyCollection<E> {
        public  boolean isEmpty();
        public int size();
        public void clear();
        public boolean contains(E arg);
        public Object[] toArray();
        public Object clone();
        public boolean equals(Object arg);
        public String toString();
}
```

Да се реализират родов клас **GVectorDeque<E>** и родов клас **GFancyVectorDeque<E>**, съгласно спецификацията:

```
import java.util.Vector;

public class GVectorDeque<E> implements GDeque<E> {
        //Представяне на дек
        protected Vector<E> elements;

        //Конструктори
        public GVectorDeque() {…} //Създава празен дек
        public GVectorDeque(int n) {…} //Създава празен дек с начален капацитет n
        public GVectorDeque(Object[] arg) {…} //Създава дек от елементите на масива arg, като първият
                                        //елемент на масива е в началото на дека, а последният елемент – накрая
        //Реализация на методите на интерфейса GDeque
}

public class GFancyVectorDeque<E> extends GVectorDeque<E> implements GFancyCollection<E>, Cloneable  {
        //Конструктори
        public GFancyVectorDeque() {…} //Създава празен дек
        public GFancyVectorDeque(GFancyCollection<E> arg){…}//Създава дек от елементите на  колекцията arg
        public GFancyVectorDeque(Object[] arg) {…} //Създава дек от елементите на масива arg

        //Наследяване на методите на интерфейса GDeque
        //Реализация на методите на интерфейса GFancyCollection
        //Предефиниране на методите, наследени от клас Object
}
```