Дисциплина: *СД и ООП* **Тема:** *Итератор*
Специалност: *КН 1 курс, 3 и 4 групи* **Седмица**: *13.04 -15.04.2009 г.*

Интерфейсът **Iterator** представя операции над обекти за обхождане на елементите на колекции, наречени *итератори*. Всеки итератор за дадена колекция "посещава" еднократно всеки един от нейните елементи, съгласно дадено правило за обхождане.

```
public interface Iterator {
        public Object next();
        //Връща поредния непосетен елемент, ако има такъв. В противен случа активира изключение
        //NoSuchElementException
        public boolean hasNext();
        //Проверява дали има още непосетени елементи
        public void remove();
        //Изключва елемента, който последно е върнат от метода next. Ако методът next не е още извикван
        //или вече е извикан методът remove, се активира изключение IllegalStateException. Ако методът
        //remove не е реализиран,  се активира изключение UnsupportedOperationException
        public void reset();
        //Подготвя текущия итератор за повторно обхождане
}
```

Следващата таблица представя различни начини за обхождане на масив, списък, дърво и таблица, както и класовете, които представят съответните итератори.

Структура	Начин на обхождане	Клас
Масив	От елемент с даден индекс	ArrayIterator
Списък	От първия към последния елмент	LinkedListIterator DoublyLinkedListIterator
Списък	От последния към първия елемент	LinkedListReverseIterator DoublyLinkedListReverseIterator
Списък	От най-малкия елмент към най-големия или обратно (за списъци с елементи от тип Comparable)	LinkedListSortedIterator DoublyLinkedListSortedIterator
Дърво	Смесено обхождане	BSTreeInorderIterator
Дърво	Възходящо обхождане	BSTreePostorderIterator
Дърво	Низходящо обхождане	BSTreePreorderIterator
Дърво	По нива	BSTreeBreadthFirstIterator
Таблица	На ключовете	ListTableKeysIterator TreeTableKeysIterator HashTableKeysIterator
Таблица	На стойностите	ListTableValuesIterator TreeTableValuesIterator HashTableValuesIterator
Таблица	На ключовете - от най-малкия към най-големия или обратно (за таблици с ключове от тип Comparable)	ListTableKeysSortedIterator TreeTableKeysSortedIterator HashTableKeysSortedIterator

Задача 1: Да се реализира клас **ArrayIterator<E>** за обхождане на елементите на масив:

```
import java.util.NoSuchElementException;
public class ArrayIterator<E> implements GIterator<E> {
        //Data
        private Object data[];      //Elements
        private int head;           //First element
        private int count;          //Number of elements
        private int current;        //Current element
        private int remaining;      //Number of remaining elements

        //Constructors
        public ArrayIterator(Object[] arg) { this(arg,0,arg.length); }
        public ArrayIterator(Object arg[],int first,int size) {
                data = arg;
```

```
                head = first;
                count = size;
                reset();
        }

        //GIterator methods implementation
        public boolean hasNext() { return remaining > 0; }

        @SuppressWarnings("unchecked")
        public E next() {
                if(hasNext()){
                        Object temp = data[current];
                        current = (current+1)%data.length;
                        remaining--;
                        return (E)temp;
                }
                throw new NoSuchElementException();
        }

        public void remove() { throw new UnsupportedOperationException(); }
        public void reset() {
                current = head;
                remaining = count;
        }
}
```

Задача 2: Да се реализира итератор за обхождане на елементите на опашка от първия към последния - в класа **GArrayQueue<E>** се добавя метод

public GIterator<E> iterator() { return new ArrayIterator<E>(array,begin,number); }

Тестване:

```
class TestArrayIterator {
        public static void main(String[] args) {
                Object[] array = { new Integer(1), new Integer(2), new Integer(3) };
                GIterator<Integer> it = new ArrayIterator<Integer>(array);
                System.out.println("Testing ArrayIterator...");
                System.out.println("     Printing array elements from index 0...");
                while(it.hasNext())
                        System.out.print(it.next() + "   ");

                it = new ArrayIterator<Integer>(array,1,3);
                System.out.println("\n     Printing array elements from index 1...");
                while(it.hasNext())
                        System.out.print(it.next() + "   ");

                it = new ArrayIterator<Integer>(array,2,3);
                System.out.println("\n     Printing array elements from index 2...");
                while(it.hasNext())
                        System.out.print(it.next() + "   ");

                System.out.println("\nTesting GArrayQueue iterator...");
                GArrayQueue<Integer> q = new GArrayQueue<Integer>(10);
                q.add(30); q.add(20); q.add(10);
                it = q.iterator();
                System.out.println("     Printing queue elements...");
                while(it.hasNext())
                        System.out.print(it.next() + "   ");
                System.out.println();
        }
}
```