Дисциплина: *СД и ООП* **Тема:** *Опашка*
Специалност: *КН 1 курс, 3 и 4 групи* **Седмица**: *06.04 -12.04.2009 г.*

Задача 1: Използване на родов интерфейс **java.util.Queue<E>** и родов клас **java.util.concurrent.ArrayBlockingQueue <E>** - виж http://java.sun.com/javase/6/docs/api/:

```
public interface Queue<E> extends Collection<E> {
        public boolean    add(E e);
        //Inserts the specified element into this queue if it is possible to do so
        //immediately without violating capacity restrictions, returning true upon
        //success and throwing an IllegalStateException if no space is currently available.
        public E element();
        //Retrieves, but does not remove, the head of this queue.
        //Throws: NoSuchElementException if this queue is empty.
        public E remove();
        //Retrieves and removes the head of this queue.
        //Throws: NoSuchElementException if this queue is empty.
        public boolean offer(E e);
        //Inserts the specified element into this queue if it is possible to do so
        //immediately without violating capacity restrictions.
        //Returns: true if the element was added to this queue, else false
        public E peek();
        //Retrieves, but does not remove, the head of this queue, or returns null if this queue is empty.
        public E poll();
        //Retrieves and removes the head of this queue, or returns null if this queue is empty.
}
```

Интерфейсът **MultipleQueue<E>** представя операции *на колекция от опашки*:

```
import java.util.Queue;

public interface MultipleQueue<E> {
        public boolean isEmpty();
        //Проверява дали текущата колекция е празна
        public boolean add(E object, int whichComponent);
        //Включва x в опашка с номер whichComponent ∈ [1; numberOfComponents()] на текущата колекция
        public E remove(int whichComponent);
        //Изключва елемент от опашка с номер whichComponent ∈ [1; numberOfComponents()] на
        текущата //колекция и го връща
        public E element(int whichComponent);
        //Връща елемент от опашка с номер whichComponent ∈ [1; numberOfComponents()] на
        текущата //колекция
        public Queue<E> getQueue(int whichComponent);
        //Връща опашка с номер whichComponent ∈ [1; numberOfComponents()] на текущата колекция
        public int size();
        //Връща броя на елементите в текущата колекция
        public int numberOfComponents();
        //Връща броя на опашките в текущата колекция
}
```

Да се реализира клас **ArrayMultipleQueue<E>**, съгласно спецификацията:

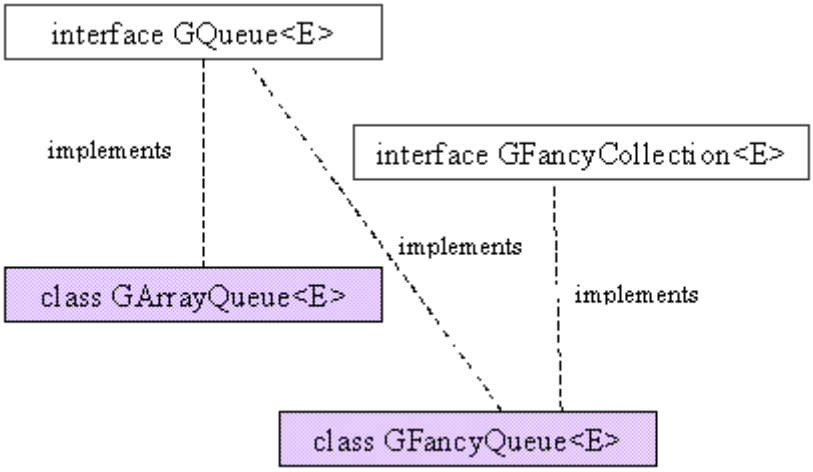
```
import java.util.Queue;
import java.util.concurrent.ArrayBlockingQueue;
import java.util.NoSuchElementException;

public class ArrayMultipleQueue<E> implements MultipleQueue<E> {
        //Представяне на колекция от опашки
        private Object[] array;

        //Конструктор
        public ArrayMultipleQueue(int n) {…} //Създава празна колекция от n опашки (обекти на клас
```

```
                                                    //ArrayBlockingQueue<E>)
        //Реализация на методите на интерфейса MultipleQueue
}
```

Задача 2: Дадена е йерархията от интерфейси и класове:



Интерфейсът **GQueue<E>** представя операции на *опашка:*

```
public interface GQueue<E> {
        public boolean     add(E e);
        //Inserts the specified element into this queue if it is possible to do so
        //immediately without violating capacity restrictions, returning true upon
        //success and throwing an IllegalStateException if no space is currently available.
        public E element();
        //Retrieves, but does not remove, the head of this queue.
        //Throws: NoSuchElementException if this queue is empty.
        public E remove();
        //Retrieves and removes the head of this queue.
        //Throws: NoSuchElementException if this queue is empty.
        public boolean offer(E e);
        //Inserts the specified element into this queue if it is possible to do so
        //immediately without violating capacity restrictions.
        //Returns: true if the element was added to this queue, else false
        public E peek();
        //Retrieves, but does not remove, the head of this queue, or returns null if this queue is empty.
        public E poll();
        //Retrieves and removes the head of this queue, or returns null if this queue is empty.
        public boolean isEmpty();
        public Object[] toArray();
        public Object clone();
}
```

Използва се интерфейсът **java.lang.Cloneable** - виж http://java.sun.com/javase/6/docs/api/.

1.Да се реализира клас **GArrayQueue<E>**, съгласно спецификацията:

```
import java.util.NoSuchElementException;

public class GArrayQueue<E> implements GQueue<E>, Cloneable {
        //Представяне на опашка чрез масив
        protected Object[] array;   //Кръгов буфер за елементите
        protected int begin;        //Индекс на първия елемент
        protected int end;          //Индекс на последния елемент
        protected int number;       //Брой на елементите

        //Конструктори
        public GArrayQueue() {…} //Създава празна опашка
        public GArrayQueue(int n) {…} //Създава празна опашка  с капацитет n
        public GArrayQueue(Object[] arg) {…} //Създава опашка  от елементите на масива arg, като първият
```

```
            //елемент на масива е в началото на опашката, а последният елемент – накрая на опашката
        //Реализация на методите на интерфейса GQueue
        //Други методи
        public boolean isFull() {…}
        public int capacity() {…}
        //Предефиниране на метод, наследен от клас Object
        public Object clone() {…}
}
```

2.Да се реализира клас **GFancyQueue<E>**, съгласно спецификацията:

```
import java.util.NoSuchElementException;

public class GFancyQueue<E> implements GQueue<E>, GFancyCollection<E>, Cloneable {
        //Представяне
        private GQueue<E> elements;

        //Конструктори
        public GFancyQueue(GQueue<E> arg) {…} //Създава колекция-опашка, съвпадаща с опашката arg

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

Задача за упражнение:

1.Като се използва **java.util.concurrent.ArrayBlockingQueue<E>**, да се реализира:

1.1. Клас **IntegersPrinter2** за генериране на случаен брой от случайни цели числа и извеждане на числата в реда на получаването им

1.2. Клас **TextPrinter2** за въвеждане на текст и извеждане на първите n (случайно число) реда от него

2.Като се използва **ArrayMultipleQueue<E>**, да се реализира:

2.1. Клас **TextPrinter3** за въвеждане на последователност от думи и тяхното извеждане в азбучен ред на първите им букви. Думите, започващи с една и съща буква, се извеждат в реда на въвеждането им

2.2. Клас **IntegersPrinter3** за въвеждане на последователност от цели положителни числа и тяхното извеждане в намаляващ ред на дължините им. Числата с една и съща дължина се извеждат в реда на въвеждането им

3.Да се реализира клас **GVectorQueue<E>**, съгласно спецификацията:

```
import java.util.NoSuchElementException;

public class GVectorQueue<E> implements GQueue<E>, Cloneable {
        //Представяне на опашка чрез масив
        private Vector<E> elements;

        //Конструктори
        public GVectorQueue() {…} //Създава празна опашка
        public GVectorQueue(int n) {…} //Създава празна опашка  с капацитет n
        public GVectorQueue(Object[] arg) {…} //Създава опашка  от елементите на масива arg, като първият
                //елемент на масива е в началото на опашката, а последният елемент – накрая на опашката
        //Реализация на методите на интерфейса Gqueue
        //Други методи
        public int capacity() {…}
        //Предефиниране на метод, наследен от клас Object
        public Object clone() {…}
}
```