112 學年度　第一學期　　　　　課號：I4390

進階程式開發技術

程式作業報告

程式作業 **04**

姓名：李信均

座號：I4B18

(1) Create a **generic class**, called **genQueue**, to represent a queue. A queue has three instance variables: the ***size***, the ***rear***, and a generic array ***elements***. The elements stored in the queue are of the same generic type ***E***. The generic array ***elements*** is used to store the elements in the queue. The value of the instance variable ***size*** is the maximum number of elements that can be stored in the generic array ***elements***. The queue is implemented such that the first element of the queue is ***elements*[0]**. The value of instance variable ***rear*** is the index of the last element in the queue. A queue has two operations, the ***enqueue*** and ***dequeue*** operations. The ***enqueue*** operation place an element after the last element of the queue. The ***dequeue*** operation removes the first element ***elements*[0]** of the queue and moves the remaining elements one position forward in the array ***elements*** such that the new first element of the queue will be ***elements*[0]**. **(40 分)**

- Provide a constructor with an integer parameter ***s*** to initialize the queue as an empty queue that can store ***s*** elements.
- Provide a ***enqueue*** method to place an element after the last element of the queue
- Provide a ***dequeue*** method to remove the first element of the queue
- Design ***enqueueAll*** method to enqueue all elements from an array into the queue. The method must use the ***enqueue*** method to enqueue element into the queue. This method should have two parameters, one parameter represents the queue, and the other one represents an array.
  - The queue's type of the ***enqueueAll*** method must be must be bounded wildcards, either upper bounded or lower bounded, of generic type ***Number***. Choose the corrected one such that your

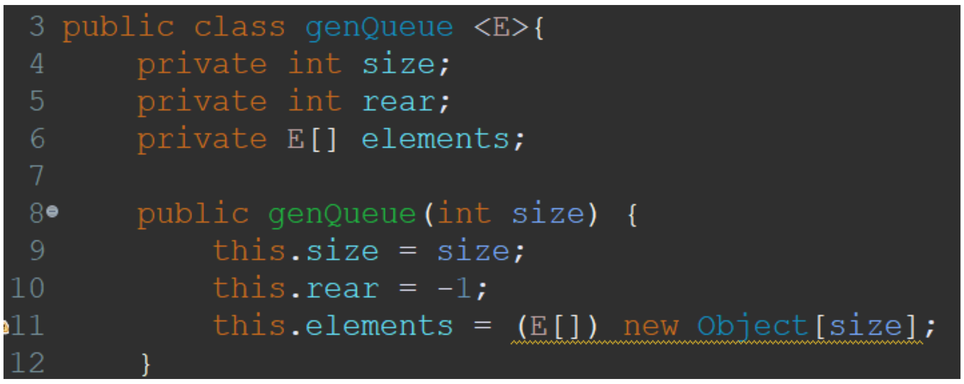
- Design ***dequeueAll*** method to dequeue all elements from a queue. The methods must use the ***dequeue*** method to dequeue element from the queue. This method should have a parameter represents the queue.
  - The queue's type of the ***dequeueAll*** method must be bounded wildcards, either upper bounded or lower bounded, of generic type ***E***. Choose the corrected one such that your program can run.

Write a test application to demonstrate the usage of the **genQueue** generic class. Test your program to enqueue and dequeue elements of type **Integer** and **Double**.

Remark: To create a generic array, you must create an array of ***Object*** type and cast the array type into the generic type. For example, assume that **elements** is an array of generic type E, the following statement create a generic array.

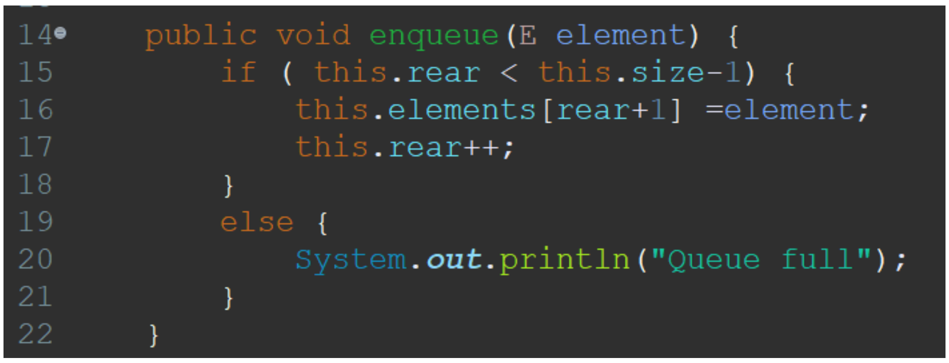
elements = (E[]) new Object[*size*] ;

建構子和 member



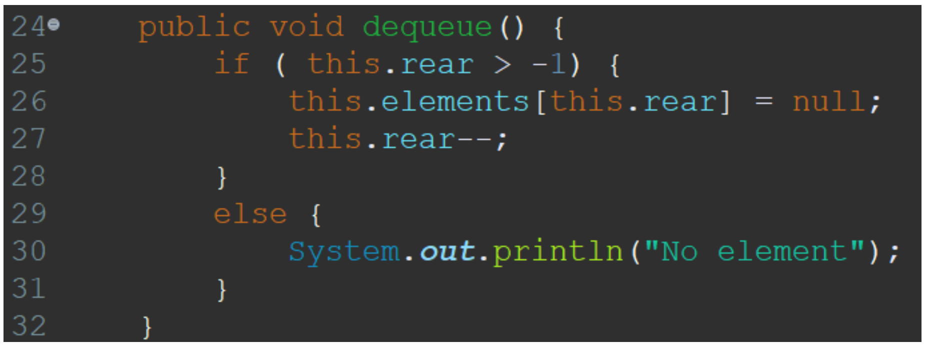
```
public class genQueue <E>{
    private int size;
    private int rear;
    private E[] elements;

    public genQueue(int size) {
        this.size = size;
        this.rear = -1;
        this.elements = (E[]) new Object[size];
    }
```

Enqueue，當被加入時 rear 往後移



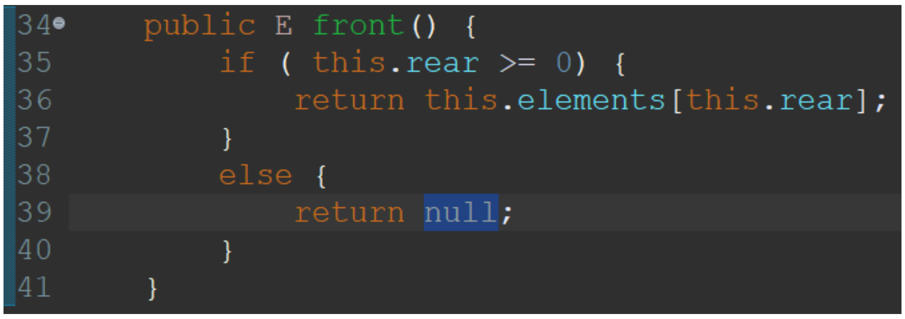
```java
    public void enqueue(E element) {
        if ( this.rear < this.size-1) {
            this.elements[rear+1] =element;
            this.rear++;
        }
        else {
            System.out.println("Queue full");
        }
    }
```

dequeue，當被刪除時 rear 往前移



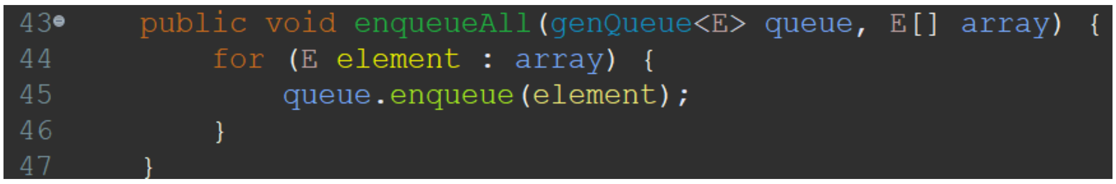
```java
    public void dequeue() {
        if ( this.rear > -1) {
            this.elements[this.rear] = null;
            this.rear--;
        }
        else {
            System.out.println("No element");
        }
    }
```

Front，取得目前 queue 最前面的值



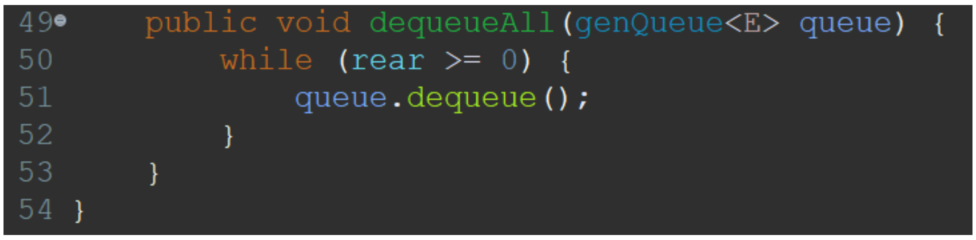
```java
    public E front() {
        if ( this.rear >= 0) {
            return this.elements[this.rear];
        }
        else {
            return null;
        }
    }
```

enqueueAll，加入陣列



```java
    public void enqueueAll(genQueue<E> queue, E[] array) {
        for (E element : array) {
            queue.enqueue(element);
        }
    }
```

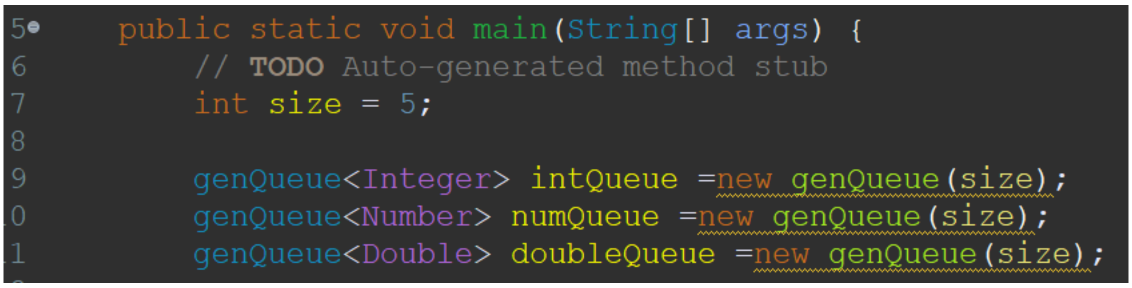
dequeueAll，移除全部



```java
    public void dequeueAll(genQueue<E> queue) {
        while (rear >= 0) {
            queue.dequeue();
        }
    }
}
```

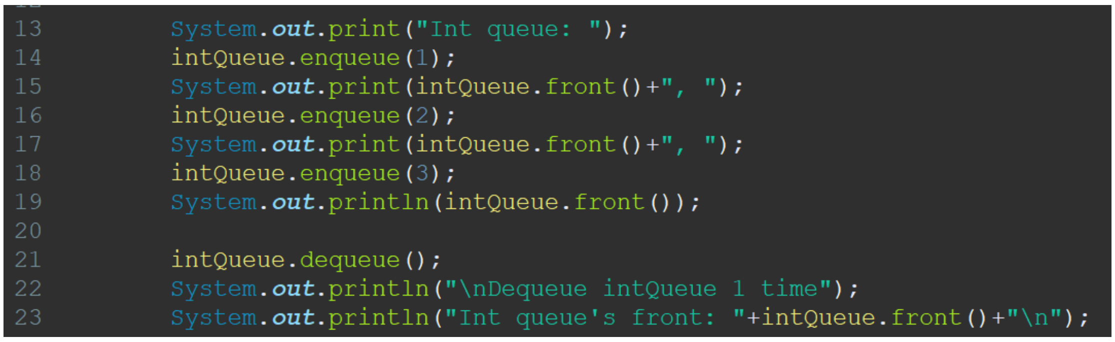
Test program：

創建三個 queue，分別為 int、number、double

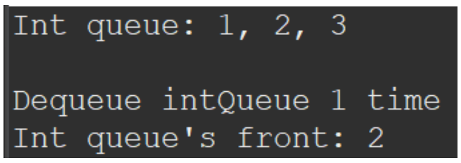


```java
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int size = 5;

        genQueue<Integer> intQueue =new genQueue(size);
        genQueue<Number> numQueue =new genQueue(size);
        genQueue<Double> doubleQueue =new genQueue(size);
```

測試 int queue，依序加入 1、2、3，印出後刪除第一個

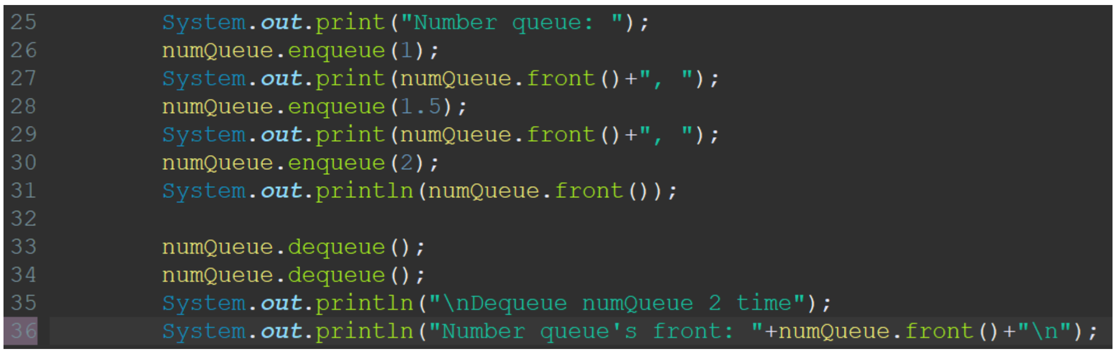


```java
        System.out.print("Int queue: ");
        intQueue.enqueue(1);
        System.out.print(intQueue.front()+", ");
        intQueue.enqueue(2);
        System.out.print(intQueue.front()+", ");
        intQueue.enqueue(3);
        System.out.println(intQueue.front());

        intQueue.dequeue();
        System.out.println("\nDequeue intQueue 1 time");
        System.out.println("Int queue's front: "+intQueue.front()+"\n");
```

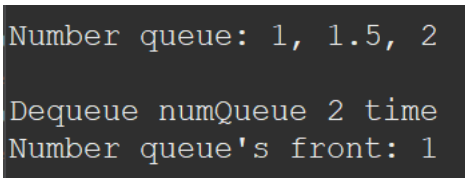


```
Int queue: 1, 2, 3

Dequeue intQueue 1 time
Int queue's front: 2
```

測試 num queue，依序加入 1、1.5、2，印出後刪除前兩個

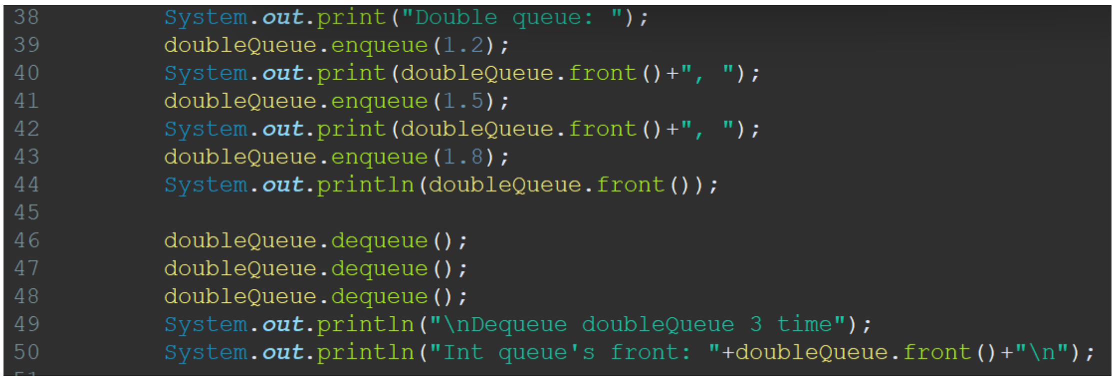


```java
        System.out.print("Number queue: ");
        numQueue.enqueue(1);
        System.out.print(numQueue.front()+", ");
        numQueue.enqueue(1.5);
        System.out.print(numQueue.front()+", ");
        numQueue.enqueue(2);
        System.out.println(numQueue.front());

        numQueue.dequeue();
        numQueue.dequeue();
        System.out.println("\nDequeue numQueue 2 time");
        System.out.println("Number queue's front: "+numQueue.front()+"\n");
```

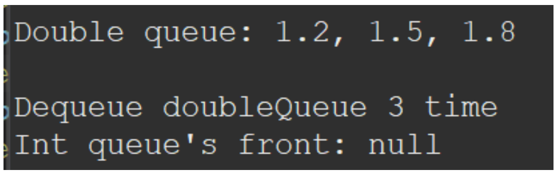


```
Number queue: 1, 1.5, 2

Dequeue numQueue 2 time
Number queue's front: 1
```

測試 double queue，依序加入 1.2、1.5、1.8，印出後刪除前三個

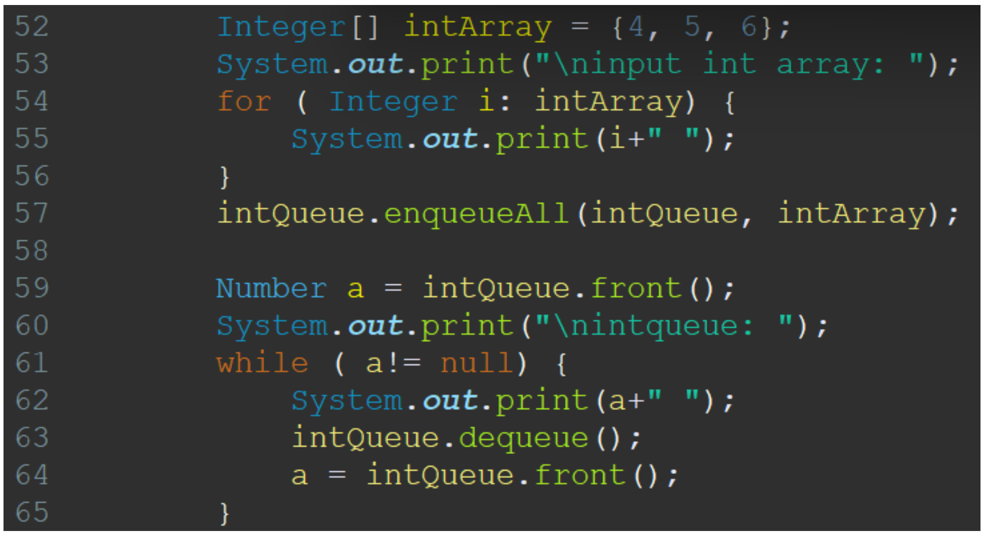


```java
        System.out.print("Double queue: ");
        doubleQueue.enqueue(1.2);
        System.out.print(doubleQueue.front()+", ");
        doubleQueue.enqueue(1.5);
        System.out.print(doubleQueue.front()+", ");
        doubleQueue.enqueue(1.8);
        System.out.println(doubleQueue.front());

        doubleQueue.dequeue();
        doubleQueue.dequeue();
        doubleQueue.dequeue();
        System.out.println("\nDequeue doubleQueue 3 time");
        System.out.println("Int queue's front: "+doubleQueue.front()+"\n");
```

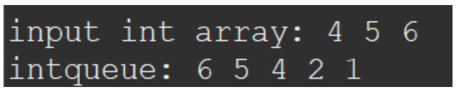


```
Double queue: 1.2, 1.5, 1.8

Dequeue doubleQueue 3 time
Int queue's front: null
```

測試 enqueueAll，宣告陣列裡面有 4、5、6 並加入 queue，因為要先移除最前面的項目才能取後面的值，所以需要依序移除並取得，2、1 為前面還沒移除的數

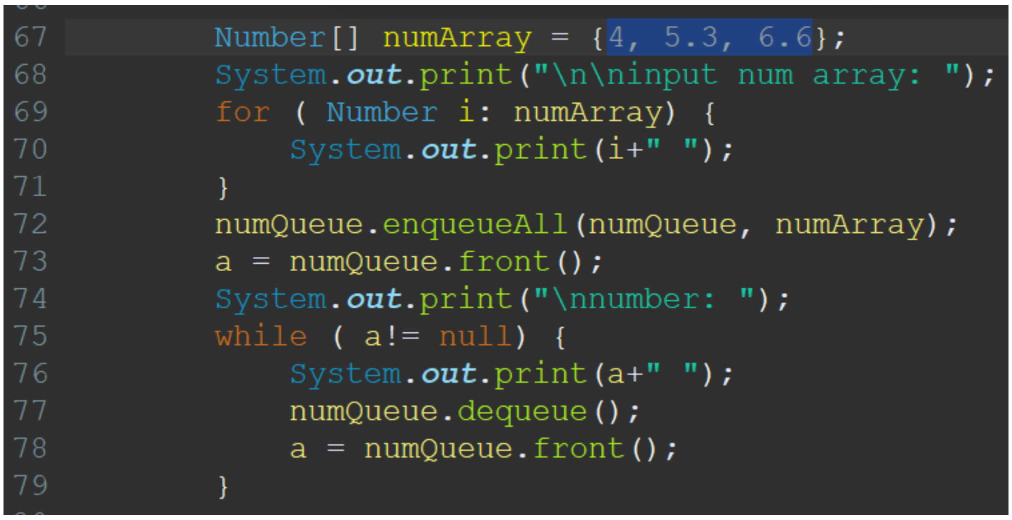


```java
        Integer[] intArray = {4, 5, 6};
        System.out.print("\ninput int array: ");
        for ( Integer i: intArray) {
            System.out.print(i+" ");
        }
        intQueue.enqueueAll(intQueue, intArray);

        Number a = intQueue.front();
        System.out.print("\nintqueue: ");
        while ( a!= null) {
            System.out.print(a+" ");
            intQueue.dequeue();
            a = intQueue.front();
        }
```

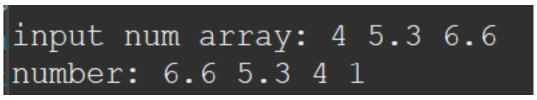


```
input int array: 4 5 6
intqueue: 6 5 4 2 1
```

宣告陣列裡面有 4, 5.3, 6.6 並加入 queue，1 為前面還沒移除的數

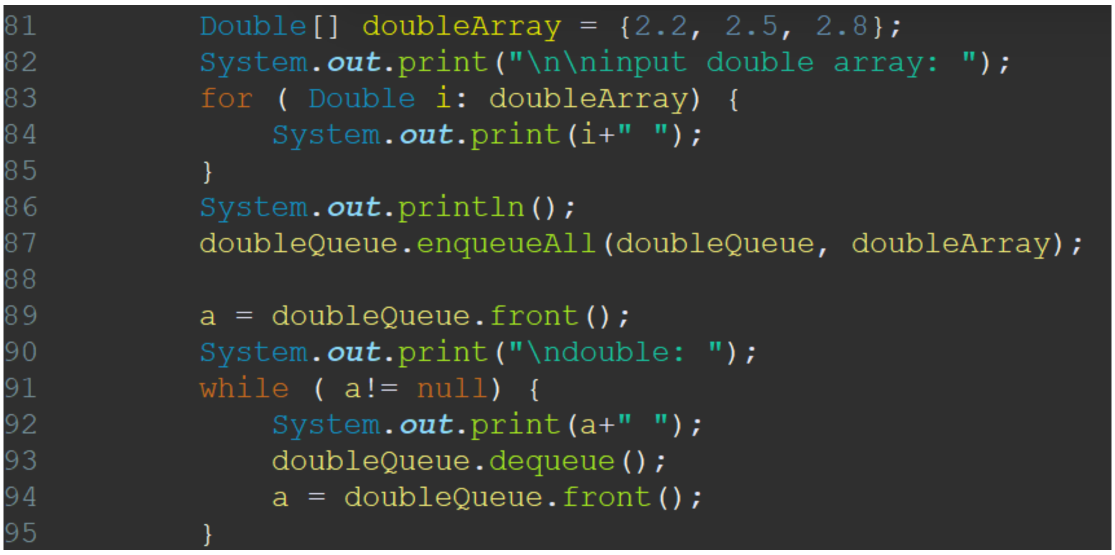


```java
        Number[] numArray = {4, 5.3, 6.6};
        System.out.print("\n\ninput num array: ");
        for ( Number i: numArray) {
            System.out.print(i+" ");
        }
        numQueue.enqueueAll(numQueue, numArray);
        a = numQueue.front();
        System.out.print("\nnumber: ");
        while ( a!= null) {
            System.out.print(a+" ");
            numQueue.dequeue();
            a = numQueue.front();
        }
```

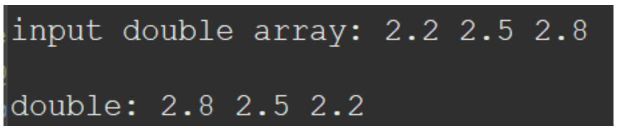


```
input num array: 4 5.3 6.6
number: 6.6 5.3 4 1
```

宣告陣列裡面有 4, 5.3, 6.6 並加入 queue

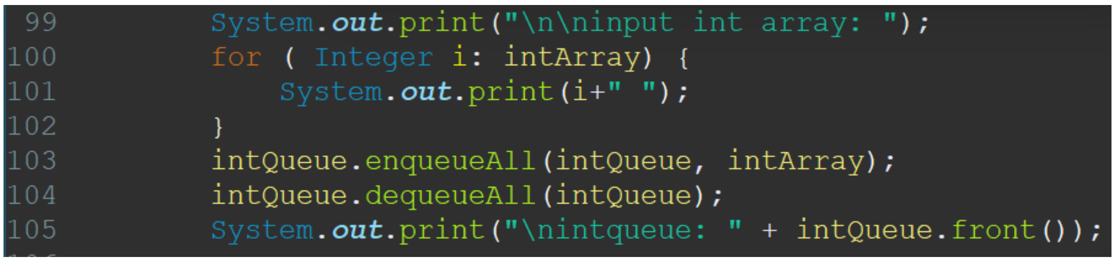


```java
        Double[] doubleArray = {2.2, 2.5, 2.8};
        System.out.print("\n\ninput double array: ");
        for ( Double i: doubleArray) {
            System.out.print(i+" ");
        }
        System.out.println();
        doubleQueue.enqueueAll(doubleQueue, doubleArray);

        a = doubleQueue.front();
        System.out.print("\ndouble: ");
        while ( a!= null) {
            System.out.print(a+" ");
            doubleQueue.dequeue();
            a = doubleQueue.front();
        }
```

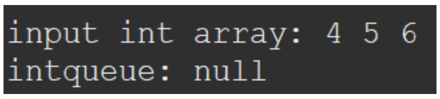


```
input double array: 2.2 2.5 2.8

double: 2.8 2.5 2.2
```

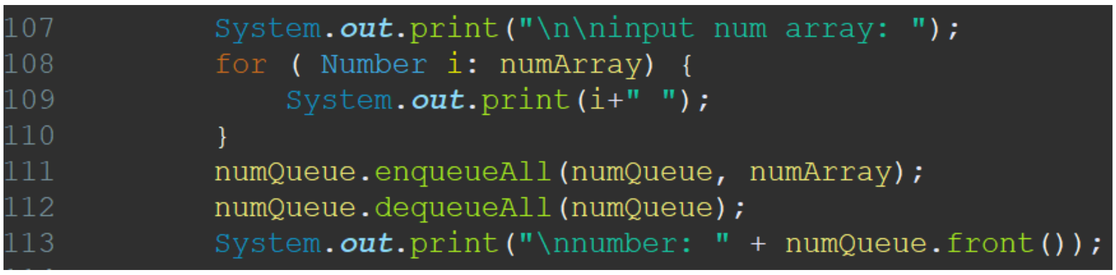
測試 dequeueAll，都沿用以上的 array



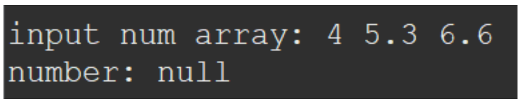
```java
        System.out.print("\n\ninput int array: ");
        for ( Integer i: intArray) {
            System.out.print(i+" ");
        }
        intQueue.enqueueAll(intQueue, intArray);
        intQueue.dequeueAll(intQueue);
        System.out.print("\nintqueue: " + intQueue.front());
```



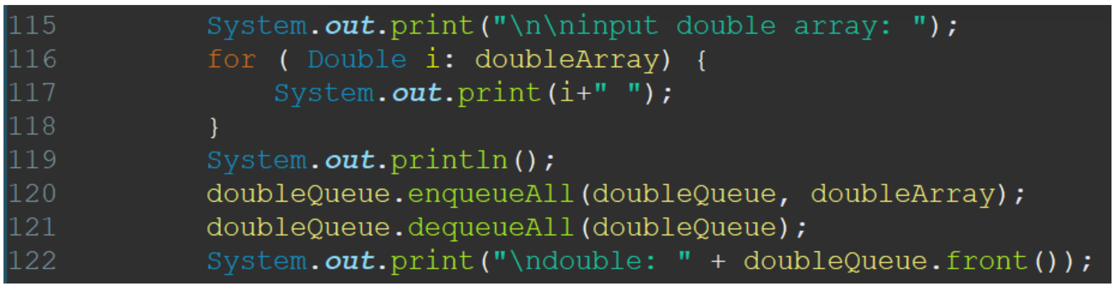
```
input int array: 4 5 6
intqueue: null
```



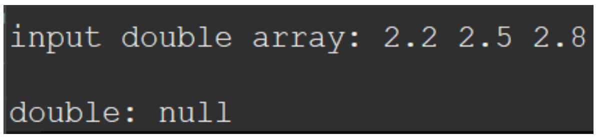
```java
        System.out.print("\n\ninput num array: ");
        for ( Number i: numArray) {
            System.out.print(i+" ");
        }
        numQueue.enqueueAll(numQueue, numArray);
        numQueue.dequeueAll(numQueue);
        System.out.print("\nnumber: " + numQueue.front());
```



```
input num array: 4 5.3 6.6
number: null
```

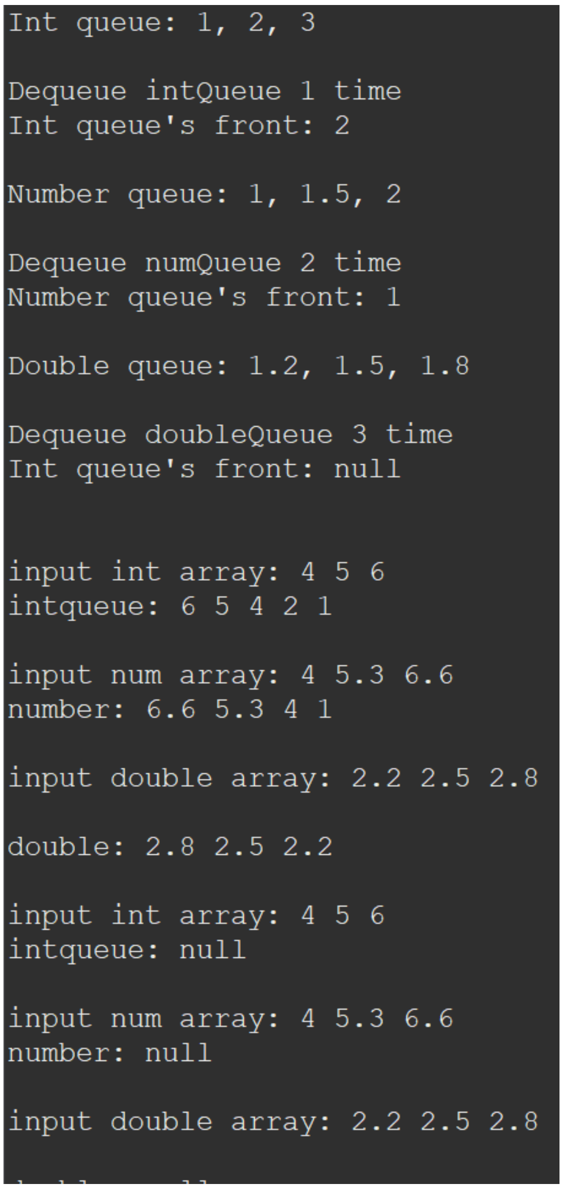


```java
        System.out.print("\n\ninput double array: ");
        for ( Double i: doubleArray) {
            System.out.print(i+" ");
        }
        System.out.println();
        doubleQueue.enqueueAll(doubleQueue, doubleArray);
        doubleQueue.dequeueAll(doubleQueue);
        System.out.print("\ndouble: " + doubleQueue.front());
```

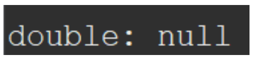


```
input double array: 2.2 2.5 2.8

double: null
```

完整輸出：



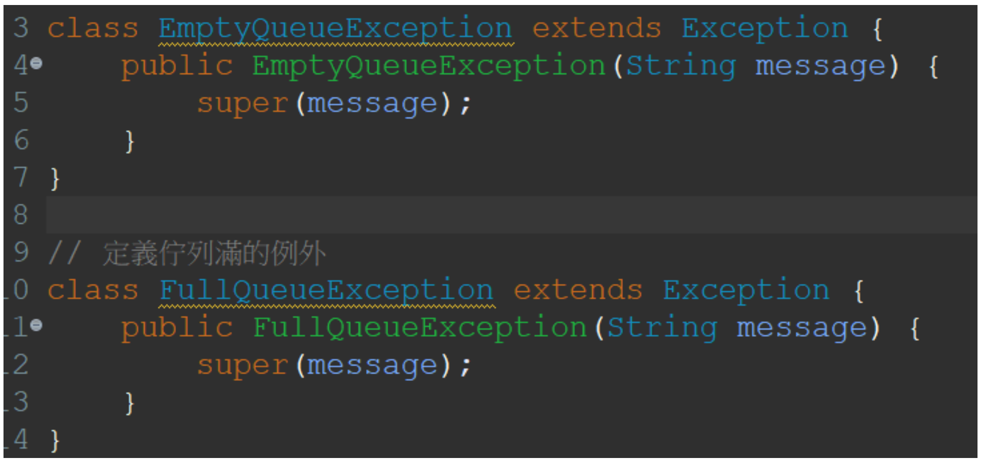
```
Int queue: 1, 2, 3

Dequeue intQueue 1 time
Int queue's front: 2

Number queue: 1, 1.5, 2

Dequeue numQueue 2 time
Number queue's front: 1

Double queue: 1.2, 1.5, 1.8

Dequeue doubleQueue 3 time
Int queue's front: null


input int array: 4 5 6
intqueue: 6 5 4 2 1

input num array: 4 5.3 6.6
number: 6.6 5.3 4 1

input double array: 2.2 2.5 2.8

double: 2.8 2.5 2.2

input int array: 4 5 6
intqueue: null

input num array: 4 5.3 6.6
number: null

input double array: 2.2 2.5 2.8

```



```
double: null
```

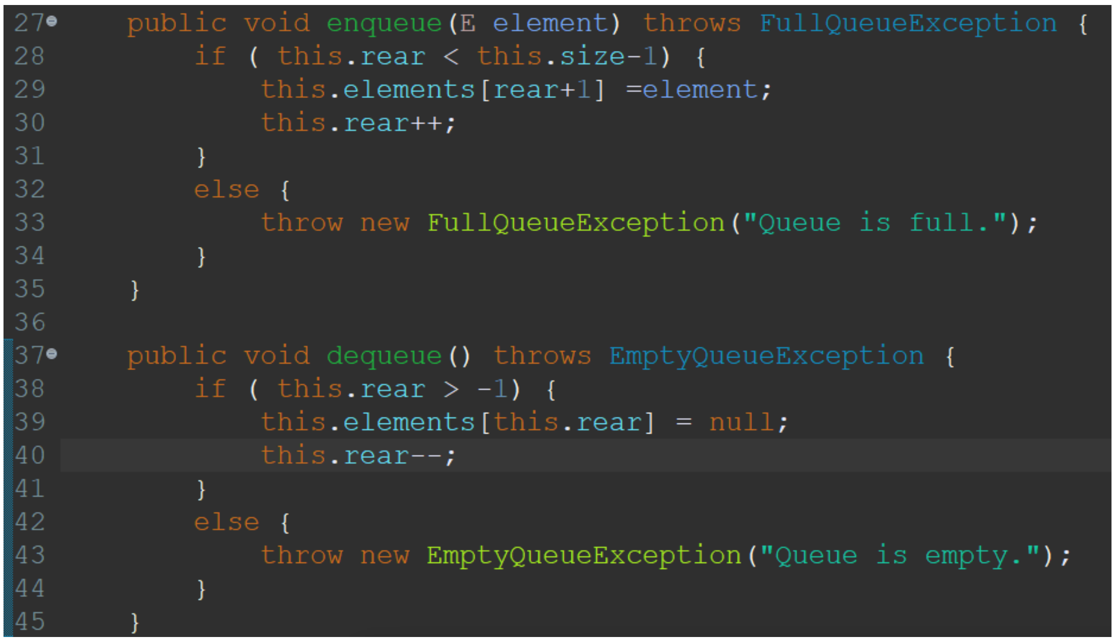
(2) Design **exception classes** to catch the empty and full of the queue. Modify the program in problem (1) to check those exceptions. Write a test application to show the use of the exceptions. In your test application, demonstrate the handling the exceptions by using **throw, throws**, **catch clause**, and calling the **getStackTrace()** and **getMessage()**. **(30 分)**

新增兩個 class 分別為空的和滿的 exept



```java
class EmptyQueueException extends Exception {
    public EmptyQueueException(String message) {
        super(message);
    }
}

// 定義佇列滿的例外
class FullQueueException extends Exception {
    public FullQueueException(String message) {
        super(message);
    }
}
```

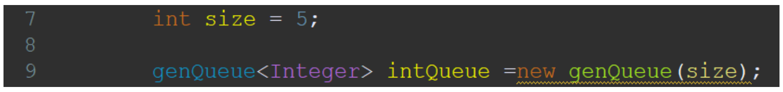
讓 enqueue 跟 dequeue 都 throw exeption



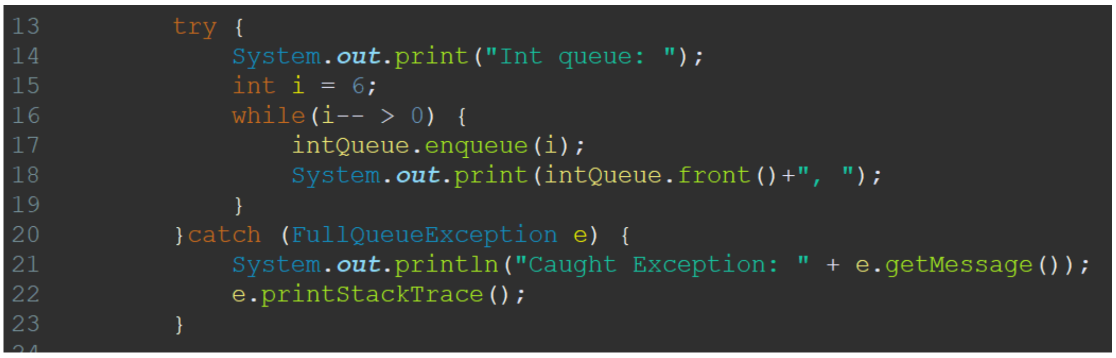
```java
    public void enqueue(E element) throws FullQueueException {
        if ( this.rear < this.size-1) {
            this.elements[rear+1] =element;
            this.rear++;
        }
        else {
            throw new FullQueueException("Queue is full.");
        }
    }

    public void dequeue() throws EmptyQueueException {
        if ( this.rear > -1) {
            this.elements[this.rear] = null;
            this.rear--;
        }
        else {
            throw new EmptyQueueException("Queue is empty.");
        }
    }
```

Test

宣告大小為五 queue，接著填入 6 個數，分別為 6 開始的遞減數字

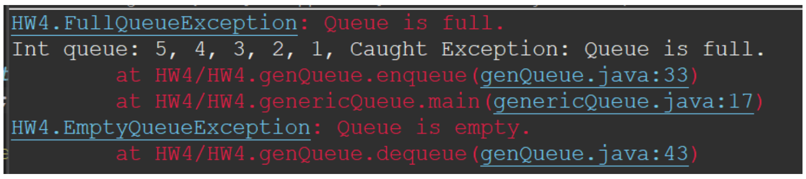


```java
        int size = 5;

        genQueue<Integer> intQueue =new genQueue(size);
```



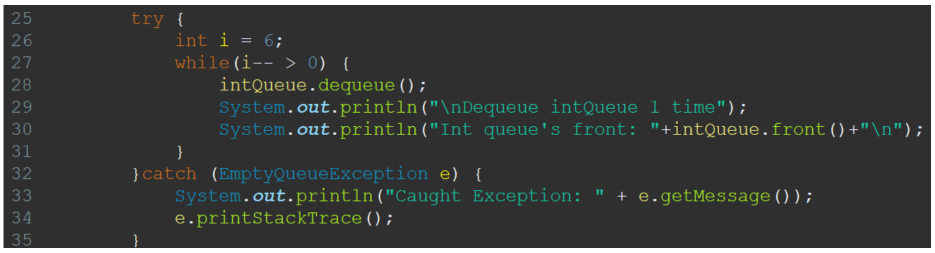
```java
        try {
            System.out.print("Int queue:  ");
            int i = 6;
            while(i-- > 0) {
                intQueue.enqueue(i);
                System.out.print(intQueue.front()+", ");
            }
        }catch (FullQueueException e) {
            System.out.println("Caught Exception: " + e.getMessage());
            e.printStackTrace();
        }
```

拋出 exception

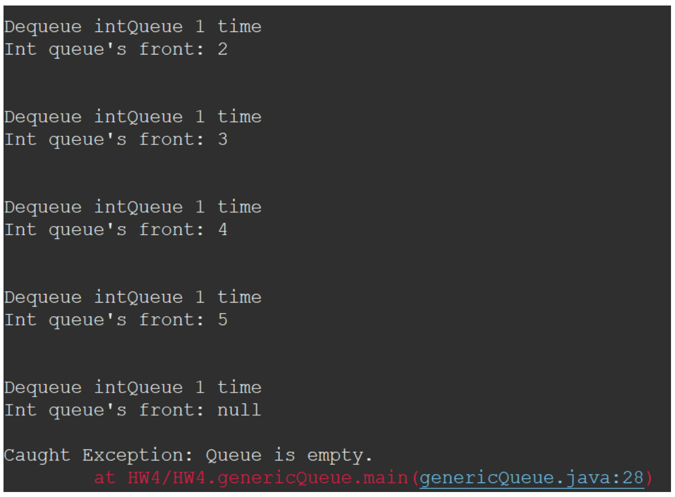


```
HW4.FullQueueException: Queue is full.
Int queue: 5, 4, 3, 2, 1, Caught Exception: Queue is full.
	at HW4/HW4.genQueue.enqueue(genQueue.java:33)
	at HW4/HW4.genericQueue.main(genericQueue.java:17)
HW4.EmptyQueueException: Queue is empty.
	at HW4/HW4.genQueue.dequeue(genQueue.java:43)
```

移除 6 個數保證會有 exept



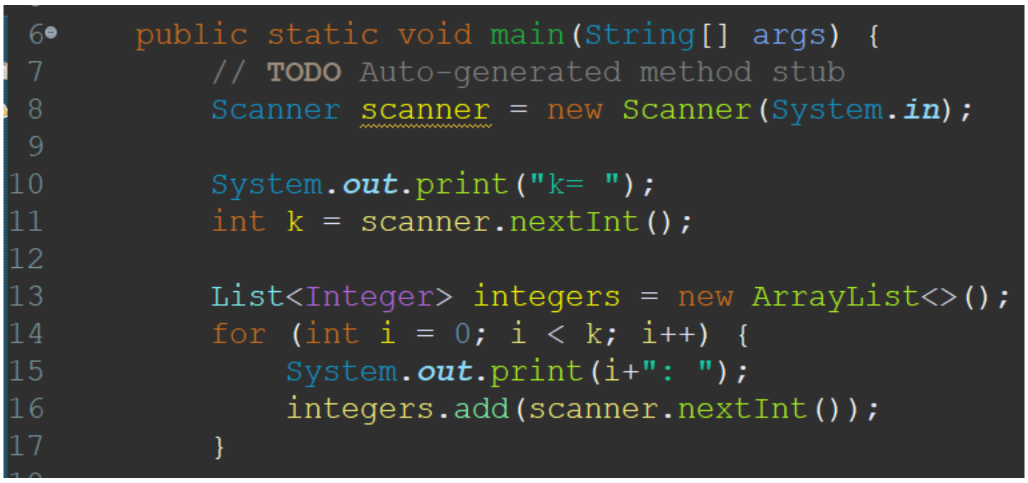
```java
        try {
            int i = 6;
            while(i-- > 0) {
                intQueue.dequeue();
                System.out.println("\nDequeue intQueue 1 time");
                System.out.println("Int queue's front: "+intQueue.front()+"\n");
            }
        }catch (EmptyQueueException e) {
            System.out.println("Caught Exception: " + e.getMessage());
            e.printStackTrace();
        }
```



```
Dequeue intQueue 1 time
Int queue's front: 2


Dequeue intQueue 1 time
Int queue's front: 3


Dequeue intQueue 1 time
Int queue's front: 4


Dequeue intQueue 1 time
Int queue's front: 5


Dequeue intQueue 1 time
Int queue's front: null

Caught Exception: Queue is empty.
	at HW4/HW4.genericQueue.main(genericQueue.java:28)
```

(3) Write a program that reads ***k*** integers and finds the ones that has the most occurrences. Your program should first input value of ***k*** and then input ***k*** integers. Your program output the integers that has the most occurrences. For example, if you entered 20 30 -40 30 -15 20, the number 20 and 30 both occurred most often with 2 occurrences. Your program should output 20 and 30 as the most occurrences integers. Your program must represent the integer and its occurrences as an **MAP** object (integer, occurrences). Your program must design by using the classes that implement the **MAP** interface, and other classes that implement **Collection** interface when necessary. **(30 分)**

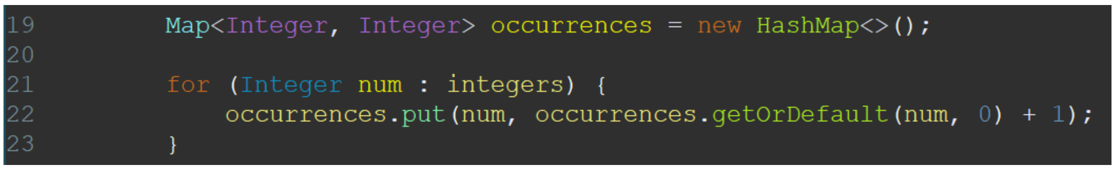
Input k 個數



```java
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner scanner = new Scanner(System.in);

        System.out.print("k= ");
        int k = scanner.nextInt();

        List<Integer> integers = new ArrayList<>();
        for (int i = 0; i < k; i++) {
            System.out.print(i+": ");
            integers.add(scanner.nextInt());
        }
```

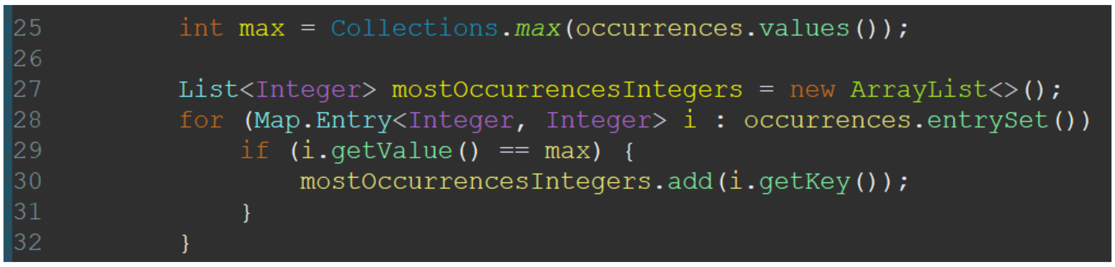
將 input 的數放進 map，這裡有用到 getOrDefault，假如 map 已存在

該數字則出現次數加一，如果沒出現過該數字則設為 0



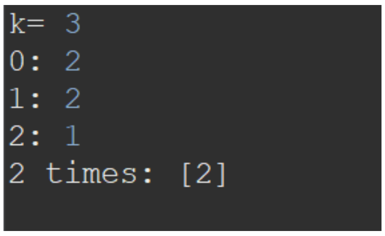
```java
        Map<Integer, Integer> occurrences = new HashMap<>();

        for (Integer num : integers) {
            occurrences.put(num, occurrences.getOrDefault(num, 0) + 1);
        }
```

使用 Collections 找出 max，並比對是哪個數字

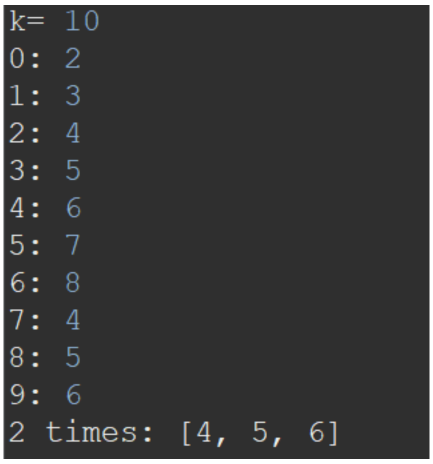


```java
        int max = Collections.max(occurrences.values());

        List<Integer> mostOccurrencesIntegers = new ArrayList<>();
        for (Map.Entry<Integer, Integer> i : occurrences.entrySet())
            if (i.getValue() == max) {
                mostOccurrencesIntegers.add(i.getKey());
            }
        }
```

結果：



```
k= 3
0: 2
1: 2
2: 1
2 times: [2]
```



```
k= 10
0: 2
1: 3
2: 4
3: 5
4: 6
5: 7
6: 8
7: 4
8: 5
9: 6
2 times: [4, 5, 6]
```