Java Interview Handbook — Batch 6

Generated: 2025-09-13 02:24:57Z (UTC)

Java Theory & Cheatsheet

```
JAVA THEORY & CHEATSHEET
------------------------
- Types: int, long, double, boolean, char, String
- Collections: List, Set, Map; ArrayList, LinkedList, HashSet, HashMap; PriorityQueue
- Streams/Lambdas; Comparator; Optional
- Concurrency: Thread, synchronized, ExecutorService
- DSA: arrays, linked lists, stacks/queues, heaps, trees, graphs
- Testing: JUnit 5 @Test, Assertions.*
```

050. MatrixMultiply

Statement: Mirror of the Python problem; Java implementation.
Explanation: Brute-force then optimized approach; include complexity.

```
package problems;


public class Problem050MatrixMultiply {
    public static int[][] multiply(int[][] A, int[][] B){
        int m=A.length, n=A[0].length, p=B[0].length;
        int[][] C=new int[m][p];
        for(int i=0;i<m;i++) for(int k=0;k<n;k++) if(A[i][k]!=0)
            for(int j=0;j<p;j++) C[i][j]+=A[i][k]*B[k][j];
        return C;
    }
}
```

```
package tests;

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
import problems.Problem050MatrixMultiply;


public class TestProblem050MatrixMultiply {
    @Test void t(){
        int[][] A={{1,2},{3,4}}, B={{5,6},{7,8}};
        int[][] R = Problem050MatrixMultiply.multiply(A,B);
        assertArrayEquals(new int[]{19,22}, R[0]);
        assertArrayEquals(new int[]{43,50}, R[1]);
    }
}
```

050. TopologicalSort

Statement: Mirror of the Python problem; Java implementation.
Explanation: Brute-force then optimized approach; include complexity.

```
package problems;


import java.util.*;
public class Problem050TopologicalSort {
    public static java.util.List<Integer> topo(int n, int[][] edges){
        java.util.List<java.util.List<Integer>> g=new ArrayList<>(); for(int i=0;i<n;i++) g.add(new ArrayList<>());
        int[] indeg=new int[n]; for(int[] e: edges){ g.get(e[0]).add(e[1]); indeg[e[1]]++; }
        java.util.ArrayDeque<Integer> q=new java.util.ArrayDeque<>(); for(int i=0;i<n;i++) if(indeg[i]==0) q.add(i);
        java.util.List<Integer> res=new java.util.ArrayList<>(); while(!q.isEmpty()){ int u=q.poll(); res.add(u); for(i
        return res;
    }
}
```

```
package tests;

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
import problems.Problem050TopologicalSort;


public class TestProblem050TopologicalSort {
    @Test void t(){ assertEquals(4, Problem050TopologicalSort.topo(4, new int[][]{ {0,1}, {0,2}, {1,3}, {2,3} }).size()
}
```

051. SlidingWindowMax

Statement: Mirror of the Python problem; Java implementation.
Explanation: Brute-force then optimized approach; include complexity.

```
package problems;


public class Problem051SlidingWindowMax {
    public static int[] maxSliding(int[] a,int k){
        int n=a.length; if(n==0||k==0) return new int[0];
        int[] res=new int[n-k+1]; java.util.Deque<Integer> dq=new java.util.ArrayDeque<>();
        for(int i=0;i<n;i++){ while(!dq.isEmpty() && dq.peekFirst()<=i-k) dq.pollFirst(); while(!dq.isEmpty() && a[dq.p
        return res;
    }
}
```

```
package tests;

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
import problems.Problem051SlidingWindowMax;


public class TestProblem051SlidingWindowMax {
    @Test void t(){ assertArrayEquals(new int[]{3,3,5,5,6,7}, Problem051SlidingWindowMax.maxSliding(new int[]{1,3,-1,-3
}
```

051. TransposeMatrix

Statement: Mirror of the Python problem; Java implementation.
Explanation: Brute-force then optimized approach; include complexity.

```
package problems;


public class Problem051TransposeMatrix {
    public static int[][] transpose(int[][] M){
        int m=M.length, n=M[0].length;
        int[][] T=new int[n][m];
        for(int i=0;i<m;i++) for(int j=0;j<n;j++) T[j][i]=M[i][j];
        return T;
    }
}
```

```
package tests;

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
import problems.Problem051TransposeMatrix;


public class TestProblem051TransposeMatrix {
    @Test void t(){
        int[][] M={{1,2,3},{4,5,6}};
        int[][] T=Problem051TransposeMatrix.transpose(M);
        assertArrayEquals(new int[]{1,4}, T[0]);
        assertArrayEquals(new int[]{2,5}, T[1]);
        assertArrayEquals(new int[]{3,6}, T[2]);
    }
}
```

053. SpiralMatrix

Statement: Mirror of the Python problem; Java implementation.
Explanation: Brute-force then optimized approach; include complexity.

```
package problems;


import java.util.*;
public class Problem053SpiralMatrix {
    public static java.util.List<Integer> spiral(int[][] m){
        List<Integer> res=new ArrayList<>();
        if(m.length==0) return res;
        int top=0, bot=m.length-1, left=0, right=m[0].length-1;
        while(top<=bot && left<=right){
            for(int j=left;j<=right;j++) res.add(m[top][j]); top++;
            for(int i=top;i<=bot;i++) res.add(m[i][right]); right--;
            if(top<=bot) for(int j=right;j>=left;j--) res.add(m[bot][j]); bot--;
            if(left<=right) for(int i=bot;i>=top;i--) res.add(m[i][left]); left++;
        }
        return res;
    }
}
```

```
package tests;

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
import problems.Problem053SpiralMatrix;


import java.util.*;
public class TestProblem053SpiralMatrix {
    @Test void t(){
        int[][] m={{1,2,3},{4,5,6},{7,8,9}};
        assertEquals(List.of(1,2,3,6,9,8,7,4,5), Problem053SpiralMatrix.spiral(m));
    }
}
```

055. MedianTwoSortedArraysSmall

Statement: Mirror of the Python problem; Java implementation.
Explanation: Brute-force then optimized approach; include complexity.

```
package problems;


public class Problem055MedianTwoSortedArraysSmall {
    public static double median(int[] A, int[] B){
        int n=A.length+B.length;
        int i=0,j=0,prev=0,cur=0;
        for(int k=0;k<=n/2;k++){
            prev=cur;
            if(i<A.length && (j>=B.length || A[i]<=B[j])) cur=A[i++];
            else cur=B[j++];
        }
        if(n%2==1) return cur;
        return (prev+cur)/2.0;
    }
}
```

```
package tests;

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
import problems.Problem055MedianTwoSortedArraysSmall;


public class TestProblem055MedianTwoSortedArraysSmall {
    @Test void t(){
        assertEquals(2.0, Problem055MedianTwoSortedArraysSmall.median(new int[]{1,3}, new int[]{2}), 1e-9);
        assertEquals(2.5, Problem055MedianTwoSortedArraysSmall.median(new int[]{1,2}, new int[]{3,4}), 1e-9);
    }
}
```

060. MaxProfitKTransactions

Statement: Mirror of the Python problem; Java implementation.
Explanation: Brute-force then optimized approach; include complexity.

```
package problems;


public class Problem060MaxProfitKTransactions {
    public static int maxProfit(int k, int[] prices){
        int n=prices.length; if(n==0||k==0) return 0;
        if(k>=n/2){ // unlimited
            int prof=0; for(int i=1;i<n;i++) if(prices[i]>prices[i-1]) prof+=prices[i]-prices[i-1]; return prof;
        }
        int[] buy=new int[k+1], sell=new int[k+1];
        for(int i=0;i<=k;i++){ buy[i]=Integer.MIN_VALUE/4; sell[i]=0; }
        for(int p: prices){
            for(int t=1;t<=k;t++){
                buy[t]=Math.max(buy[t], sell[t-1]-p);
                sell[t]=Math.max(sell[t], buy[t]+p);
            }
        }
        return sell[k];
    }
}
```

```
package tests;

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
import problems.Problem060MaxProfitKTransactions;


public class TestProblem060MaxProfitKTransactions {
    @Test void t(){
        assertEquals(7, Problem060MaxProfitKTransactions.maxProfit(2, new int[]{3,2,6,5,0,3}));
    }
}
```

061. HeapSort

Statement: Mirror of the Python problem; Java implementation.
Explanation: Brute-force then optimized approach; include complexity.

```
package problems;


public class Problem061HeapSort {
    public static void sort(int[] a){
        int n=a.length;
        for(int i=n/2-1;i>=0;i--) heapify(a,n,i);
        for(int i=n-1;i>=0;i--){
            int t=a[0]; a[0]=a[i]; a[i]=t;
            heapify(a,i,0);
        }
    }
    static void heapify(int[] a,int n,int i){
        int largest=i, l=2*i+1, r=2*i+2;
        if(l<n && a[l]>a[largest]) largest=l;
        if(r<n && a[r]>a[largest]) largest=r;
        if(largest!=i){ int t=a[i]; a[i]=a[largest]; a[largest]=t; heapify(a,n,largest); }
    }
}
```

```
package tests;

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
import problems.Problem061HeapSort;


public class TestProblem061HeapSort {
    @Test void t(){
        int[] a={4,1,3,2}; Problem061HeapSort.sort(a);
        assertArrayEquals(new int[]{1,2,3,4}, a);
    }
}
```

063. CountingSortSimple

Statement: Mirror of the Python problem; Java implementation.
Explanation: Brute-force then optimized approach; include complexity.

```
package problems;


public class Problem063CountingSortSimple {
    public static int[] sort(int[] a, int maxVal){
        int[] count=new int[maxVal+1];
        for(int x: a) count[x]++;
        int idx=0;
        for(int v=0; v<=maxVal; v++) while(count[v]-- > 0) a[idx++]=v;
        return a;
    }
}
```

```
package tests;

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
import problems.Problem063CountingSortSimple;


public class TestProblem063CountingSortSimple {
    @Test void t(){
        int[] a={3,1,2,1,0}; assertArrayEquals(new int[]{0,1,1,2,3}, Problem063CountingSortSimple.sort(a,3));
    }
}
```

064. RadixSortSimple

Statement: Mirror of the Python problem; Java implementation.
Explanation: Brute-force then optimized approach; include complexity.

```
package problems;


public class Problem064RadixSortSimple {
    public static int[] sort(int[] a){
        int max=0; for(int x: a) if(x>max) max=x;
        int exp=1; int n=a.length;
        int[] out=new int[n];
        while(max/exp>0){
            int[] cnt=new int[10];
            for(int i=0;i<n;i++) cnt[(a[i]/exp)%10]++;
            for(int i=1;i<10;i++) cnt[i]+=cnt[i-1];
            for(int i=n-1;i>=0;i--) out[--cnt[(a[i]/exp)%10]] = a[i];
            System.arraycopy(out,0,a,0,n);
            exp*=10;
        }
        return a;
    }
}
```

```
package tests;

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
import problems.Problem064RadixSortSimple;


public class TestProblem064RadixSortSimple {
    @Test void t(){
        int[] a={170,45,75,90,802,24,2,66}; assertArrayEquals(new int[]{2,24,45,66,75,90,170,802}, Problem064RadixSortS
    }
}
```