Java Interview Handbook — Batch 7

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.*
```

094. MinWindowSubstring

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

```
package problems;


public class Problem094MinWindowSubstring {
    public static String minWindow(String s, String t){
        if(t.length()==0) return "";
        int[] need=new int[128]; int required=0;
        for(char c: t.toCharArray()){ if(need[c]==0) required++; need[c]++; }
        int[] have=new int[128]; int formed=0;
        int l=0, bestLen=Integer.MAX_VALUE, bestL=0;
        for(int r=0;r<s.length();r++){
            char c=s.charAt(r); have[c]++;
            if(have[c]==need[c] && need[c]>0) formed++;
            while(formed==required){
                if(r-l+1<bestLen){bestLen=r-l+1; bestL=l;}
                char lc=s.charAt(l); have[lc]--; if(have[lc]<need[lc] && need[lc]>0) formed--; l++;
            }
        }
        return bestLen==Integer.MAX_VALUE? "": s.substring(bestL,bestL+bestLen);
    }
}
```

```
package tests;

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
import problems.Problem094MinWindowSubstring;


public class TestProblem094MinWindowSubstring {
    @Test void t(){ assertEquals("BANC", Problem094MinWindowSubstring.minWindow("ADOBECODEBANC","ABC")); }
}
```

095. MaxRectangleHistogram

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

```
package problems;


import java.util.*;
public class Problem095MaxRectangleHistogram {
    public static int largestRectangleArea(int[] heights){
        int n=heights.length, max=0;
        java.util.Deque<Integer> st=new java.util.ArrayDeque<>();
        for(int i=0;i<=n;i++){
            int h = (i==n)? 0: heights[i];
            while(!st.isEmpty() && h < heights[st.peek()]){
                int height=heights[st.pop()];
                int left = st.isEmpty()? -1: st.peek();
                int width = i - left - 1;
                max = Math.max(max, height*width);
            }
            st.push(i);
        }
        return max;
    }
}
```

```
package tests;

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
import problems.Problem095MaxRectangleHistogram;


public class TestProblem095MaxRectangleHistogram {
    @Test void t(){ assertEquals(10, Problem095MaxRectangleHistogram.largestRectangleArea(new int[]{2,1,5,6,2,3})); }
}
```

098. UrlValidation

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

```
package problems;


import java.util.regex.*;
public class Problem098UrlValidation {
    private static final Pattern P = Pattern.compile("^(https?://)?([\w.-]+)(:[0-9]+)?(/.*)?$");
    public static boolean isValid(String url){ return url!=null && P.matcher(url).matches(); }
}
```

```
package tests;

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
import problems.Problem098UrlValidation;


public class TestProblem098UrlValidation {
    @Test void t(){ assertTrue(Problem098UrlValidation.isValid("https://example.com")); assertFalse(Problem098UrlValida
}
```

099. EmailValidationRegex

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

```
package problems;


import java.util.regex.*;
public class Problem099EmailValidationRegex {
    private static final Pattern P = Pattern.compile("^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$");
    public static boolean isValid(String email){ return email!=null && P.matcher(email).matches(); }
}
```

```
package tests;

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
import problems.Problem099EmailValidationRegex;


public class TestProblem099EmailValidationRegex {
    @Test void t(){ assertTrue(Problem099EmailValidationRegex.isValid("a@b.com")); assertFalse(Problem099EmailValidatio
}
```

101. FindKthSmallestBst

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

```
package problems;


public class Problem101FindKthSmallestBst {
    public static class Node{int v; Node l,r; Node(int v){this.v=v;}}
    public static Integer kthSmallest(Node root, int k){
        java.util.Deque<Node> st=new java.util.ArrayDeque<>();
        Node cur=root; int count=0;
        while(cur!=null || !st.isEmpty()){
            while(cur!=null){ st.push(cur); cur=cur.l; }
            cur=st.pop();
            if(++count==k) return cur.v;
            cur=cur.r;
        }
        return null;
    }
}
```

```
package tests;

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
import problems.Problem101FindKthSmallestBst;


public class TestProblem101FindKthSmallestBst {
    @Test void t(){
        var r=new Problem101FindKthSmallestBst.Node(2); r.l=new Problem101FindKthSmallestBst.Node(1); r.r=new Problem10
        assertEquals(2, Problem101FindKthSmallestBst.kthSmallest(r,2));
    }
}
```

102. CountSetBits

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

```
package problems;


public class Problem102CountSetBits {
    public static int countBits(int x){
        int c=0; while(x!=0){ x&=x-1; c++; } return c;
    }
}
```

```
package tests;

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
import problems.Problem102CountSetBits;


public class TestProblem102CountSetBits {
    @Test void t(){ assertEquals(3, Problem102CountSetBits.countBits(0b1011)); }
}
```

103. HammingDistance

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

```
package problems;


public class Problem103HammingDistance {
    public static int hamming(int a, int b){
        int x=a^b, c=0; while(x!=0){ x&=x-1; c++; } return c;
    }
}
```

```
package tests;

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
import problems.Problem103HammingDistance;


public class TestProblem103HammingDistance {
    @Test void t(){ assertEquals(2, Problem103HammingDistance.hamming(1,4)); }
}
```

104. GrayCode

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

```
package problems;


import java.util.*;
public class Problem104GrayCode {
    public static java.util.List<Integer> grayCode(int n){
        List<Integer> res=new ArrayList<>(); res.add(0);
        for(int i=0;i<n;i++){
            int add = 1<<i;
            for(int j=res.size()-1;j>=0;j--) res.add(res.get(j)+add);
        }
        return res;
    }
}
```

```
package tests;

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
import problems.Problem104GrayCode;


import java.util.*;
public class TestProblem104GrayCode {
    @Test void t(){ assertEquals(List.of(0,1,3,2), Problem104GrayCode.grayCode(2)); }
}
```

105. PowerSetIterative

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

```
package problems;


import java.util.*;
public class Problem105PowerSetIterative {
    public static java.util.List<java.util.List<Integer>> powerSet(int[] nums){
        List<List<Integer>> res=new ArrayList<>(); res.add(new ArrayList<>());
        for(int x: nums){
            int sz=res.size();
            for(int i=0;i<sz;i++){ List<Integer> n=new ArrayList<>(res.get(i)); n.add(x); res.add(n); }
        }
        return res;
    }
}
```

```
package tests;

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
import problems.Problem105PowerSetIterative;


import java.util.*;
public class TestProblem105PowerSetIterative {
    @Test void t(){ assertEquals(8, Problem105PowerSetIterative.powerSet(new int[]{1,2,3}).size()); }
}
```