Java: Solutions to requested problems

1. Duplicates characters in a given string

Approach: Use a LinkedHashMap to preserve order and count occurrences.

```
import java.util.*;
public class Duplicates {
  public static void main(String[] args) {
    String s = "programming";
    Map<Character,Integer> freq = new LinkedHashMap<>();
    for(char c: s.toCharArray()) {
      freq.put(c, freq.getOrDefault(c,0)+1);
    }
    freq.forEach((k,v)->{ if(v>1) System.out.println(k + " -> " + v); });
  }
}
```

2. Print common characters in two strings

Approach: Count frequencies for both strings and print intersection (min count).

```
import java.util.*;
public class CommonChars {
  public static void main(String[] args) {
    String a = "hello", b = "world";
    int[] fa = new int[256], fb = new int[256];
    for(char c: a.toCharArray()) fa[c]++;
    for(char c: b.toCharArray()) fb[c]++;
    for(int i=0;i<256;i++) {
      int m = Math.min(fa[i], fb[i]);
      for(int k=0;k<m;k++) System.out.print((char)i);
    }
    System.out.println();
  }
}
```

3. Program to replace spaces with -

Approach: Use String.replace or build manually.

```
public class ReplaceSpaces {
  public static void main(String[] args) {
    String s = "replace spaces here";
    System.out.println(s.replace(' ', '-'));
  }
}
```

4. Remove spaces and reverse a string

Approach: Remove spaces then reverse using StringBuilder.

```
public class RemoveSpacesReverse {
  public static void main(String[] args) {
    String s = "a b c d";
    String noSpaces = s.replaceAll("\s+", "");
    String rev = new StringBuilder(noSpaces).reverse().toString();
    System.out.println(rev);
  }
}
```

5. Implement bubble sort

Approach: Classic nested-loop swap with early exit optimization.

```
public class BubbleSort {
  public static void bubbleSort(int[] a) {
    int n = a.length;
    for(int i=0;i<n-1;i++) {
      boolean swapped = false;
      for(int j=0;j<n-1-i;j++) {
        if(a[j] > a[j+1]) { int t = a[j]; a[j]=a[j+1]; a[j+1]=t; swapped=true; }
      }
      if(!swapped) break;
    }
  }
  public static void main(String[] args) {
    int[] arr = {5,2,8,3,1};
    bubbleSort(arr);
    for(int v: arr) System.out.print(v+" ");
  }
}
```

6. Circular linked list (single-node insertion & traversal)

Approach: Simple node class with next pointing to head for circular behavior.

```
class CNode { int val; CNode next; CNode(int v){val=v;} }
public class CircularList {
  private CNode head;
  public void add(int v) {
    CNode node = new CNode(v);
    if(head==null) { head = node; node.next = head; return; }
    CNode cur = head;
    while(cur.next != head) cur = cur.next;
    cur.next = node; node.next = head;
  }
  public void traverse(int steps) {
    if(head==null) return;
    CNode cur = head;
    for(int i=0;i<steps;i++) {
      System.out.print(cur.val + " ");
      cur = cur.next;
    }
  }
  public static void main(String[] args) {
    CircularList cl = new CircularList();
    cl.add(1); cl.add(2); cl.add(3);
    cl.traverse(10); // loops around
  }
}
```

7. Implement ArrayList without using built-in collections

Approach: Backing array with resizing.

```
public class SimpleArrayList<E> {
  private Object[] data;
  private int size;
  public SimpleArrayList() { data = new Object[10]; }
  public void add(E e) {
    if(size==data.length) resize();
    data[size++] = e;
  }
  @SuppressWarnings("unchecked")
  public E get(int idx) { if(idx<0||idx>=size) throw new IndexOutOfBoundsException(); return (E)dat
  public int size() { return size; }
  private void resize() { Object[] n = new Object[data.length*2]; System.arraycopy(data,0,n,0,data.
  public static void main(String[] args) {
    SimpleArrayList<Integer> a = new SimpleArrayList<>();
    a.add(1); a.add(2); System.out.println(a.get(1));
  }
}
```

8. Program to find frequency of even and odd numbers in given matrix

Approach: Iterate matrix and count.

```
public class EvenOddMatrix {
  public static void main(String[] args) {
    int[][] m = {{1,2,3},{4,5,6}};
    int even=0, odd=0;
    for(int[] row: m) for(int v: row) { if(v%2==0) even++; else odd++; }
    System.out.println("Even="+even+" Odd="+odd);
  }
}
```

9. If a string is palindrome

Approach: Normalize then compare with reverse.

```
public class Palindrome {
  public static boolean isPalindrome(String s) {
    String t = s.replaceAll("\s+","" ).toLowerCase();
    return t.equals(new StringBuilder(t).reverse().toString());
  }
  public static void main(String[] args) {
    System.out.println(isPalindrome("A man a plan a canal Panama"));
  }
}
```

10. Two strings are anagrams

Approach: Sort or count characters. Using count for O(n).

```
import java.util.*;
public class Anagram {
  public static boolean isAnagram(String a, String b) {
    a = a.replaceAll("\s+","" ).toLowerCase();
    b = b.replaceAll("\s+","" ).toLowerCase();
    if(a.length()!=b.length()) return false;
    int[] cnt = new int[256];
    for(char c: a.toCharArray()) cnt[c]++;
    for(char c: b.toCharArray()) if(--cnt[c]<0) return false;
    return true;
  }
  public static void main(String[] args){ System.out.println(isAnagram("listen","silent")); }
}
```

11. Find missing number in array

Approach: Use sum formula or XOR. Sum formula (watch overflow) — use long.

```
public class MissingNumber {
  public static int missing(int[] a, int n) { // numbers 1..n, one missing
    long total = (long)n*(n+1)/2;
    long sum=0; for(int v:a) sum+=v;
    return (int)(total - sum);
  }
  public static void main(String[] args){
    int[] a = {1,2,4,5}; System.out.println(missing(a,5)); // prints 3
  }
}
```

12. Implement stack and queue using arrays

Approach: Simple array-backed implementations.

```
// Stack using array
public class ArrayStack {
  private int[] a; private int top = -1;
  public ArrayStack(int cap){ a = new int[cap]; }
  public void push(int v){ if(top==a.length-1) throw new RuntimeException("Full"); a[++top]=v; }
  public int pop(){ if(top==-1) throw new RuntimeException("Empty"); return a[top--]; }
}
// Queue using circular array
public class ArrayQueue {
  private int[] a; private int head=0, tail=0, size=0;
  public ArrayQueue(int cap){ a=new int[cap]; }
  public void offer(int v){ if(size==a.length) throw new RuntimeException("Full"); a[tail]=v; tail=
  public int poll(){ if(size==0) throw new RuntimeException("Empty"); int v=a[head]; head=(head+1)%
}
```

13. Count vowels and consonants in a string

Approach: Iterate and check letters.

```
public class VowelConsonant {
  public static void main(String[] args) {
    String s = "Hello World".toLowerCase();
    int v=0,c=0;
    for(char ch: s.toCharArray()) {
      if(!Character.isLetter(ch)) continue;
      if("aeiou".indexOf(ch)>=0) v++; else c++;
    }
    System.out.println("Vowels="+v+" Consonants="+c);
  }
}
```

14. Wrapper class in Java (explanation)

Wrapper classes are object representations of Java primitive types, e.g., Integer for int, Double for double, etc. They allow primitives to be used where objects are required (generics, collections). Since Java 5 autoboxing/unboxing automatically converts between primitives and wrappers. Common useful methods: Integer.parseInt(String), Double.valueOf(), Integer.compare(), etc.

```
Example:
Integer x = 10; // autoboxing
int y = x; // unboxing
```