Java Interview Programs: Basic to Advanced (With Code)

Prepared collection of Java programs covering syntax, control flow, patterns, arrays & strings, OOP, exception s & files, collections, algorithms, multithreading, Java 8 features, and mini system design demos. Each sectio n includes concise, runnable code snippets (single-file examples).

Hello World program

```
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

Sum of two numbers

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int a = sc.nextInt(), b = sc.nextInt();
        System.out.println(a + b);
        sc.close();
    }
}
```

Check even or odd

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        int n = new java.util.Scanner(System.in).nextInt();
        System.out.println(n % 2 == 0 ? "Even" : "Odd");
    }
}
```

Largest of two numbers

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int a = sc.nextInt(), b = sc.nextInt();
        System.out.println(a > b ? a : b);
        sc.close();
    }
}
```

Largest of three numbers

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int a = sc.nextInt(), b = sc.nextInt(), c = sc.nextInt();
        int max = a;
        if(b > max) max = b;
        if(c > max) max = c;
        System.out.println(max);
        sc.close();
    }
}
```

Positive, Negative, or Zero check

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        int n = new java.util.Scanner(System.in).nextInt();
        if(n > 0) System.out.println("Positive");
        else if(n < 0) System.out.println("Negative");
        else System.out.println("Zero");
    }
}
```

Leap year check

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        int y = new java.util.Scanner(System.in).nextInt();
        boolean leap = (y % 400 == 0) || (y % 4 == 0 && y % 100 != 0);
        System.out.println(leap ? "Leap Year" : "Not Leap Year");
    }
}
```

Swap two numbers (with and without third variable)

```
public class Main {
    public static void main(String[] args) {
        int a = 10, b = 20;
        // With third variable
        int temp = a; a = b; b = temp;
        System.out.println(a + " " + b);
        // Without third variable (XOR)
        a = 10; b = 20;
        a ^= b; b ^= a; a ^= b;
        System.out.println(a + " " + b);
    }
}
```

Factorial of a number (iterative)

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        int n = new java.util.Scanner(System.in).nextInt();
        long fact = 1;
        for(int i=2;i<=n;i++) fact *= i;
        System.out.println(fact);
    }
}
```

Fibonacci series (first N)

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        int n = new java.util.Scanner(System.in).nextInt();
        long a=0,b=1;
        for(int i=0;i<n;i++){
            System.out.print(a + (i<n-1?" ":""));
            long c=a+b; a=b; b=c;
        }
    }
}
```

Reverse a number

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        int n = new java.util.Scanner(System.in).nextInt();
        int rev = 0;
        while(n!=0){ rev = rev*10 + n%10; n/=10; }
        System.out.println(rev);
    }
}
```

Palindrome number check

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        int n = new java.util.Scanner(System.in).nextInt();
        int t = n, rev = 0;
        while(n!=0){ rev = rev*10 + n%10; n/=10; }
        System.out.println(t==rev ? "Palindrome" : "Not Palindrome");
```

```
    }
}
```

Armstrong number check (3-digit)

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        int n = new java.util.Scanner(System.in).nextInt();
        int t = n, sum=0;
        while(t!=0){ int d=t%10; sum += d*d*d; t/=10; }
        System.out.println(sum==n ? "Armstrong" : "Not Armstrong");
    }
}
```

Prime number check

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        int n = new java.util.Scanner(System.in).nextInt();
        if(n<2){ System.out.println("Not Prime"); return; }
        for(int i=2;i*i<=n;i++) if(n%i==0){ System.out.println("Not Prime"); return; }
        System.out.println("Prime");
    }
}
```

Print all prime numbers between 1 to N

```
import java.util.Scanner;
public class Main {
    static boolean isPrime(int n){
        if(n<2) return false;
        for(int i=2;i*i<=n;i++) if(n%i==0) return false;
        return true;
    }
    public static void main(String[] args) {
        int n = new java.util.Scanner(System.in).nextInt();
        for(int i=2;i<=n;i++) if(isPrime(i)) System.out.print(i + " ");
    }
}
```

Table of a number (multiplication table)

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        int n = new java.util.Scanner(System.in).nextInt();
        for(int i=1;i<=10;i++) System.out.println(n + " x " + i + " = " + (n*i));
    }
}
```

Sum of digits of a number

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        int n = new java.util.Scanner(System.in).nextInt();
        int sum=0; while(n!=0){ sum+=n%10; n/=10; }
        System.out.println(sum);
    }
}
```

Print numbers 1 to 100 without loop (using recursion)

```
public class Main {
    static void print(int i){
        if(i>100) return;
        System.out.println(i);
        print(i+1);
    }
    public static void main(String[] args) { print(1); }
}
```

Square pattern (****)

```
public class Main {
    public static void main(String[] args) {
        int n=4;
        for(int i=0;i<n;i++){
            for(int j=0;j<n;j++) System.out.print("*");
            System.out.println();
        }
    }
}
```

Right triangle pattern

```
public class Main {
    public static void main(String[] args) {
        int n=5;
        for(int i=1;i<=n;i++){
            for(int j=1;j<=i;j++) System.out.print("*");
            System.out.println();
        }
    }
}
```

Inverted triangle pattern

```
public class Main {
    public static void main(String[] args) {
        int n=5;
        for(int i=n;i>=1;i--){
            for(int j=1;j<=i;j++) System.out.print("*");
            System.out.println();
        }
    }
}
```

Pyramid pattern

```
public class Main {
    public static void main(String[] args) {
        int n=5;
        for(int i=1;i<=n;i++){
            for(int s=1;s<=n-i;s++) System.out.print(" ");
            for(int j=1;j<=2*i-1;j++) System.out.print("*");
            System.out.println();
        }
    }
}
```

Diamond pattern

```
public class Main {
    public static void main(String[] args) {
        int n=5;
        for(int i=1;i<=n;i++){
            for(int s=1;s<=n-i;s++) System.out.print(" ");
            for(int j=1;j<=2*i-1;j++) System.out.print("*");
            System.out.println();
        }
        for(int i=n-1;i>=1;i--){
            for(int s=1;s<=n-i;s++) System.out.print(" ");
            for(int j=1;j<=2*i-1;j++) System.out.print("*");
            System.out.println();
        }
    }
}
```

Number pattern (1, 12, 123, ...)

```
public class Main {
    public static void main(String[] args) {
        int n=5;
        for(int i=1;i<=n;i++){
```

```
            for(int j=1;j<=i;j++) System.out.print(j);
            System.out.println();
        }
    }
}
```

Alphabet pattern (A, AB, ABC, ...)

```
public class Main {
    public static void main(String[] args) {
        int n=5;
        for(int i=1;i<=n;i++){
            for(char c='A'; c<'A'+i; c++) System.out.print(c);
            System.out.println();
        }
    }
}
```

Find largest and smallest element in an array

```
public class Main {
    public static void main(String[] args) {
        int[] a = {5, 2, 9, -1, 7};
        int min=a[0], max=a[0];
        for(int v: a){ if(v<min) min=v; if(v>max) max=v; }
        System.out.println("Min="+min+", Max="+max);
    }
}
```

Reverse an array

```
import java.util.Arrays;
public class Main {
    public static void main(String[] args) {
        int[] a = {1,2,3,4,5};
        for(int i=0,j=a.length-1;i<j;i++,j--){
            int t=a[i]; a[i]=a[j]; a[j]=t;
        }
        System.out.println(Arrays.toString(a));
    }
}
```

Sort an array (ascending/descending)

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Integer[] a = {5,1,4,2,3};
        Arrays.sort(a); // ascending
        System.out.println(java.util.Arrays.toString(a));
        Arrays.sort(a, java.util.Collections.reverseOrder()); // descending
        System.out.println(java.util.Arrays.toString(a));
    }
}
```

Find duplicate elements in an array

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        int[] a = {1,2,3,1,2,4,5};
        java.util.Set<Integer> seen = new java.util.HashSet<>();
        java.util.Set<Integer> dup = new java.util.HashSet<>();
        for(int v: a) if(!seen.add(v)) dup.add(v);
        System.out.println(dup);
    }
}
```

Find missing number in array (1 to N)

```
public class Main {
    public static void main(String[] args) {
        int[] a = {1,2,4,5};
        int n = 5;
```

```
        int sum = n*(n+1)/2;
        int s = 0; for(int v: a) s+=v;
        System.out.println(sum - s);
    }
}
```

Find second largest element in an array

```
public class Main {
    public static void main(String[] args) {
        int[] a = {5,1,9,2,9,7};
        Integer first=null, second=null;
        for(int v: a){
            if(first==null || v>first){ second=first; first=v; }
            else if(v!=first && (second==null || v>second)) second=v;
        }
        System.out.println(second);
    }
}
```

Matrix addition, subtraction, multiplication

```
public class Main {
    public static void main(String[] args) {
        int[][] A={{1,2},{3,4}};
        int[][] B={{5,6},{7,8}};
        int n=2;
        int[][] add=new int[n][n], sub=new int[n][n], mul=new int[n][n];
        for(int i=0;i<n;i++) for(int j=0;j<n;j++){
            add[i][j]=A[i][j]+B[i][j];
            sub[i][j]=A[i][j]-B[i][j];
        }
        for(int i=0;i<n;i++) for(int j=0;j<n;j++)
            for(int k=0;k<n;k++) mul[i][j]+=A[i][k]*B[k][j];
        // Print one result as example
        for(int[] r: mul){
            for(int v: r) System.out.print(v + " ");
            System.out.println();
        }
    }
}
```

Transpose of a matrix

```
public class Main {
    public static void main(String[] args) {
        int[][] A={{1,2,3},{4,5,6}};
        int r=A.length, c=A[0].length;
        int[][] T=new int[c][r];
        for(int i=0;i<r;i++) for(int j=0;j<c;j++) T[j][i]=A[i][j];
        for(int[] row: T){ for(int v: row) System.out.print(v + " "); System.out.println(); }
    }
}
```

String reversal

```
public class Main {
    public static void main(String[] args) {
        String s = "hello";
        StringBuilder sb = new StringBuilder(s);
        System.out.println(sb.reverse().toString());
    }
}
```

Check if a string is a palindrome

```
public class Main {
    public static void main(String[] args) {
        String s="madam";
        String r=new StringBuilder(s).reverse().toString();
        System.out.println(s.equals(r) ? "Palindrome" : "Not Palindrome");
    }
}
```

Count vowels and consonants in a string

```
public class Main {
    public static void main(String[] args) {
        String s="Interview Prep";
        s=s.toLowerCase();
        int v=0,c=0;
        for(char ch: s.toCharArray()){
            if(ch>='a' && ch<='z'){
                if("aeiou".indexOf(ch)>=0) v++; else c++;
            }
        }
        System.out.println("Vowels="+v+", Consonants="+c);
    }
}
```

Remove duplicate characters from string

```
import java.util.LinkedHashSet;
public class Main {
    public static void main(String[] args) {
        String s="programming";
        java.util.Set<Character> set=new LinkedHashSet<>();
        StringBuilder sb=new StringBuilder();
        for(char ch: s.toCharArray()) if(set.add(ch)) sb.append(ch);
        System.out.println(sb.toString());
    }
}
```

Find first non-repeated character in string

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        String s="swiss";
        Map<Character,Integer> map=new LinkedHashMap<>();
        for(char ch: s.toCharArray()) map.put(ch, map.getOrDefault(ch,0)+1);
        for(Map.Entry<Character,Integer> e: map.entrySet())
            if(e.getValue()==1){ System.out.println(e.getKey()); return; }
        System.out.println("None");
    }
}
```

Check anagram strings

```
import java.util.Arrays;
public class Main {
    public static void main(String[] args) {
        String a="listen", b="silent";
        char[] x=a.replaceAll("\\s","").toLowerCase().toCharArray();
        char[] y=b.replaceAll("\\s","").toLowerCase().toCharArray();
        Arrays.sort(x); Arrays.sort(y);
        System.out.println(java.util.Arrays.equals(x,y) ? "Anagram" : "Not Anagram");
    }
}
```

Find substring in a string (without contains())

```
public class Main {
    static boolean contains(String s, String sub){
        for(int i=0;i+sub.length()<=s.length();i++){
            int j=0;
            while(j<sub.length() && s.charAt(i+j)==sub.charAt(j)) j++;
            if(j==sub.length()) return true;
        }
        return false;
    }
    public static void main(String[] args) {
        System.out.println(contains("helloworld","world"));
    }
}
```

Class and Object demo

```
class Person {
    String name; int age;
    void introduce(){ System.out.println(name + ", " + age); }
}
public class Main {
    public static void main(String[] args) {
        Person p=new Person(); p.name="Alice"; p.age=25; p.introduce();
    }
}
```

Constructor demo (default, parameterized, copy)

```
class Box {
    int w,h;
    Box(){ this.w=1; this.h=1; }
    Box(int w,int h){ this.w=w; this.h=h; }
    Box(Box b){ this.w=b.w; this.h=b.h; }
    int area(){ return w*h; }
}
public class Main {
    public static void main(String[] args) {
        Box a=new Box(), b=new Box(2,3), c=new Box(b);
        System.out.println(a.area()+" "+b.area()+" "+c.area());
    }
}
```

Method overloading & overriding

```
class Calc {
    int add(int a,int b){ return a+b; }
    int add(int a,int b,int c){ return a+b+c; } // overloading
}
class AdvCalc extends Calc {
    @Override int add(int a,int b){ return super.add(a,b)+1; } // overriding
}
public class Main {
    public static void main(String[] args) {
        Calc c=new Calc(); AdvCalc ac=new AdvCalc();
        System.out.println(c.add(1,2));
        System.out.println(ac.add(1,2));
    }
}
```

Inheritance example (single, multilevel, hierarchical)

```
class A { void f(){ System.out.println("A"); } }
class B extends A { void g(){ System.out.println("B"); } } // single
class C extends B { void h(){ System.out.println("C"); } } // multilevel
class D extends A { void i(){ System.out.println("D"); } } // hierarchical
public class Main {
    public static void main(String[] args) {
        new C().f(); new C().g(); new C().h();
        new D().f(); new D().i();
    }
}
```

Abstract class program

```
abstract class Shape { abstract double area(); }
class Circle extends Shape {
    double r; Circle(double r){ this.r=r; }
    double area(){ return Math.PI*r*r; }
}
public class Main {
    public static void main(String[] args) {
        Shape s = new Circle(2.0);
        System.out.println(s.area());
    }
}
```

Interface example

```
interface Drawable { void draw(); }
```

```
class Square implements Drawable {
    public void draw(){ System.out.println("Drawing square"); }
}
public class Main {
    public static void main(String[] args) {
        Drawable d = new Square(); d.draw();
    }
}
```

Encapsulation (getters & setters)

```
class Employee {
    private int id; private String name;
    public int getId(){ return id; }
    public void setId(int id){ this.id = id; }
    public String getName(){ return name; }
    public void setName(String name){ this.name = name; }
}
public class Main {
    public static void main(String[] args) {
        Employee e=new Employee(); e.setId(1); e.setName("Bob");
        System.out.println(e.getId()+": "+e.getName());
    }
}
```

Polymorphism demo

```
class Animal { void sound(){ System.out.println("some sound"); } }
class Dog extends Animal { void sound(){ System.out.println("bark"); } }
class Cat extends Animal { void sound(){ System.out.println("meow"); } }
public class Main {
    public static void main(String[] args) {
        Animal a=new Dog(); a.sound();
        a=new Cat(); a.sound();
    }
}
```

Static keyword demo

```
public class Main {
    static int count=0;
    public Main(){ count++; }
    public static void main(String[] args) {
        new Main(); new Main();
        System.out.println(Main.count);
    }
}
```

this and super keyword usage

```
class Base {
    int x=10;
    Base(int x){ this.x = x; }
}
class Derived extends Base {
    int x=20;
    Derived(){ super(5); System.out.println(this.x + ", " + super.x); }
}
public class Main { public static void main(String[] args){ new Derived(); } }
```

Try–catch–finally example

```
public class Main {
    public static void main(String[] args) {
        try{
            int x = 10/0;
        }catch(ArithmeticException e){
            System.out.println("Cannot divide by zero");
        }finally{
            System.out.println("Finally always runs");
        }
    }
}
```

Throw and throws usage

```
public class Main {
    static int div(int a,int b) throws ArithmeticException {
        if(b==0) throw new ArithmeticException("b cannot be zero");
        return a/b;
    }
    public static void main(String[] args) {
        System.out.println(div(10,2));
    }
}
```

Custom exception program

```
class InvalidAgeException extends Exception {
    InvalidAgeException(String msg){ super(msg); }
}
public class Main {
    static void vote(int age) throws InvalidAgeException {
        if(age<18) throw new InvalidAgeException("Underage");
    }
    public static void main(String[] args) {
        try{ vote(16); }catch(Exception e){ System.out.println(e.getMessage()); }
    }
}
```

Read from a file

```
import java.io.*;
public class Main {
    public static void main(String[] args) throws Exception {
        try(BufferedReader br = new BufferedReader(new FileReader("input.txt"))){
            String line; while((line=br.readLine())!=null) System.out.println(line);
        }
    }
}
```

Write to a file

```
import java.io.*;
public class Main {
    public static void main(String[] args) throws Exception {
        try(BufferedWriter bw = new BufferedWriter(new FileWriter("output.txt"))){
            bw.write("Hello file");
        }
    }
}
```

Count words in a file

```
import java.io.*;
public class Main {
    public static void main(String[] args) throws Exception {
        int words=0;
        try(BufferedReader br = new BufferedReader(new FileReader("input.txt"))){
            String line;
            while((line=br.readLine())!=null){
                String[] parts=line.trim().split("\\s+");
                if(parts.length==1 && parts[0].isEmpty()) continue;
                words += parts.length;
            }
        }
        System.out.println(words);
    }
}
```

Copy content of one file to another

```
import java.io.*;
public class Main {
    public static void main(String[] args) throws Exception {
        try(BufferedInputStream in = new BufferedInputStream(new FileInputStream("in.txt"));
            BufferedOutputStream out = new BufferedOutputStream(new FileOutputStream("out.txt"))){
```

```
            byte[] buf = new byte[4096]; int len;
            while((len=in.read(buf))!=-1) out.write(buf,0,len);
        }
    }
}
```

ArrayList demo (add, remove, iterate)

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        List<String> list=new ArrayList<>();
        list.add("a"); list.add("b"); list.remove("a");
        for(String s: list) System.out.println(s);
    }
}
```

LinkedList demo

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        LinkedList<Integer> q=new LinkedList<>();
        q.add(1); q.addFirst(0); q.addLast(2);
        System.out.println(q);
    }
}
```

HashSet & TreeSet demo

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Set<Integer> hs=new HashSet<>(); hs.add(3); hs.add(1); hs.add(3);
        System.out.println(hs);
        Set<Integer> ts=new TreeSet<>(hs);
        System.out.println(ts);
    }
}
```

HashMap & TreeMap demo

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Map<String,Integer> map=new HashMap<>();
        map.put("a",1); map.put("b",2);
        System.out.println(map);
        Map<String,Integer> tmap=new TreeMap<>(map);
        System.out.println(tmap);
    }
}
```

Iterate through HashMap (entrySet, keySet, values)

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Map<String,Integer> map=new HashMap<>();
        map.put("x",10); map.put("y",20);
        for(Map.Entry<String,Integer> e: map.entrySet()) System.out.println(e.getKey()+"="+e.getValue(
        for(String k: map.keySet()) System.out.println(k);
        for(Integer v: map.values()) System.out.println(v);
    }
}
```

Sort elements using Collections.sort()

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        List<Integer> list=Arrays.asList(5,3,1,4,2);
        Collections.sort(list);
        System.out.println(list);
```

```
        Collections.sort(list, Collections.reverseOrder());
        System.out.println(list);
    }
}
```

Convert Array to List and vice versa

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        String[] a = {"a","b","c"};
        List<String> list = new ArrayList<>(Arrays.asList(a));
        String[] back = list.toArray(new String[0]);
        System.out.println(list + " | " + java.util.Arrays.toString(back));
    }
}
```

Frequency of elements using HashMap

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        int[] a={1,2,2,3,3,3};
        Map<Integer,Integer> freq=new HashMap<>();
        for(int v: a) freq.put(v, freq.getOrDefault(v,0)+1);
        System.out.println(freq);
    }
}
```

Remove duplicates from list using Set

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        List<Integer> list = new ArrayList<>(Arrays.asList(1,2,2,3,3,4));
        List<Integer> unique = new ArrayList<>(new LinkedHashSet<>(list));
        System.out.println(unique);
    }
}
```

Reverse words in a sentence

```
public class Main {
    public static void main(String[] args) {
        String s="Java is fun";
        String[] parts=s.split("\\s+");
        StringBuilder sb=new StringBuilder();
        for(int i=parts.length-1;i>=0;i--) sb.append(parts[i]).append(i==0?"":" ");
        System.out.println(sb.toString());
    }
}
```

Check balanced parentheses using Stack

```
import java.util.*;
public class Main {
    static boolean balanced(String s){
        Map<Character,Character> m=new HashMap<>();
        m.put(')', '('); m.put(']', '['); m.put('}', '{');
        Deque<Character> st=new ArrayDeque<>();
        for(char ch: s.toCharArray()){
            if(m.containsValue(ch)) st.push(ch);
            else if(m.containsKey(ch)) if(st.isEmpty() || st.pop()!=m.get(ch)) return false;
        }
        return st.isEmpty();
    }
    public static void main(String[] args) { System.out.println(balanced("{[()]}")); }
}
```

Find factorial using recursion

```
public class Main {
    static long fact(int n){ return n<=1 ? 1 : n*fact(n-1); }
    public static void main(String[] args) { System.out.println(fact(5)); }
```

```
}
```

Find nth Fibonacci number using recursion

```
public class Main {
    static long fib(int n){ return n<=1 ? n : fib(n-1)+fib(n-2); }
    public static void main(String[] args) { System.out.println(fib(10)); }
}
```

Binary search implementation

```
public class Main {
    static int bs(int[] a,int key){
        int l=0,r=a.length-1;
        while(l<=r){
            int m=l+(r-l)/2;
            if(a[m]==key) return m;
            if(a[m]<key) l=m+1; else r=m-1;
        }
        return -1;
    }
    public static void main(String[] args) {
        int[] a={1,3,5,7,9};
        System.out.println(bs(a,7));
    }
}
```

Linear search implementation

```
public class Main {
    static int ls(int[] a,int key){
        for(int i=0;i<a.length;i++) if(a[i]==key) return i;
        return -1;
    }
    public static void main(String[] args) {
        int[] a={4,2,7,1};
        System.out.println(ls(a,7));
    }
}
```

Bubble sort, Selection sort, Insertion sort

```
import java.util.Arrays;
public class Main {
    static void bubble(int[] a){
        for(int i=0;i<a.length-1;i++)
            for(int j=0;j<a.length-1-i;j++)
                if(a[j]>a[j+1]){ int t=a[j]; a[j]=a[j+1]; a[j+1]=t; }
    }
    static void selection(int[] a){
        for(int i=0;i<a.length-1;i++){
            int min=i;
            for(int j=i+1;j<a.length;j++) if(a[j]<a[min]) min=j;
            int t=a[i]; a[i]=a[min]; a[min]=t;
        }
    }
    static void insertion(int[] a){
        for(int i=1;i<a.length;i++){
            int key=a[i], j=i-1;
            while(j>=0 && a[j]>key){ a[j+1]=a[j]; j--; }
            a[j+1]=key;
        }
    }
    public static void main(String[] args) {
        int[] a={5,1,4,2,8};
        bubble(a.clone()); selection(a.clone()); insertion(a.clone());
        System.out.println(Arrays.toString(a));
    }
}
```

Quick sort, Merge sort

```
import java.util.Arrays;
```

```
public class Main {
    static void quick(int[] a,int l,int r){
        if(l>=r) return;
        int i=l,j=r,p=a[l+(r-l)/2];
        while(i<=j){
            while(a[i]<p) i++;
            while(a[j]>p) j--;
            if(i<=j){ int t=a[i]; a[i]=a[j]; a[j]=t; i++; j--; }
        }
        if(l<j) quick(a,l,j);
        if(i<r) quick(a,i,r);
    }
    static void mergeSort(int[] a,int l,int r){
        if(l>=r) return;
        int m=(l+r)/2;
        mergeSort(a,l,m); mergeSort(a,m+1,r);
        int[] tmp=new int[r-l+1]; int i=l,j=m+1,k=0;
        while(i<=m && j<=r) tmp[k++]= a[i]<=a[j]? a[i++]:a[j++];
        while(i<=m) tmp[k++]=a[i++];
        while(j<=r) tmp[k++]=a[j++];
        System.arraycopy(tmp,0,a,l,tmp.length);
    }
    public static void main(String[] args) {
        int[] a={5,2,9,1,5,6};
        quick(a,0,a.length-1);
        mergeSort(a,0,a.length-1);
        System.out.println(Arrays.toString(a));
    }
}
```

Count occurrences of each word in a string

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        String s="to be or not to be";
        Map<String,Integer> map=new LinkedHashMap<>();
        for(String w: s.trim().toLowerCase().split("\\s+"))
            map.put(w, map.getOrDefault(w,0)+1);
        System.out.println(map);
    }
}
```

Find longest substring without repeating characters

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        String s="abcabcbb";
        int l=0, best=0; Map<Character,Integer> pos=new HashMap<>();
        for(int r=0;r<s.length();r++){
            char c=s.charAt(r);
            if(pos.containsKey(c) && pos.get(c)>=l) l=pos.get(c)+1;
            pos.put(c,r); best=Math.max(best, r-l+1);
        }
        System.out.println(best);
    }
}
```

LRU Cache implementation using LinkedHashMap

```
import java.util.*;
class LRU<K,V> extends LinkedHashMap<K,V>{
    private final int cap;
    LRU(int cap){ super(cap, 0.75f, true); this.cap=cap; }
    protected boolean removeEldestEntry(Map.Entry<K,V> e){ return size()>cap; }
}
public class Main {
    public static void main(String[] args) {
        LRU<Integer,Integer> cache=new LRU<>(2);
```

```
        cache.put(1,1); cache.put(2,2); cache.get(1); cache.put(3,3);
        System.out.println(cache.keySet());
    }
}
```

Producer–Consumer problem using Threads

```
import java.util.concurrent.ArrayBlockingQueue;
public class Main {
    public static void main(String[] args) throws Exception {
        ArrayBlockingQueue<Integer> q = new ArrayBlockingQueue<>(2);
        Thread producer = new Thread(() -> {
            try{
                for(int i=1;i<=5;i++){ q.put(i); System.out.println("Produced "+i); }
            }catch(Exception ignored){}
        });
        Thread consumer = new Thread(() -> {
            try{
                for(int i=1;i<=5;i++){ System.out.println("Consumed "+q.take()); }
            }catch(Exception ignored){}
        });
        producer.start(); consumer.start();
        producer.join(); consumer.join();
    }
}
```

Deadlock example in Java

```
public class Main {
    public static void main(String[] args) throws Exception {
        final Object A=new Object(); final Object B=new Object();
        Thread t1=new Thread(()->{ synchronized(A){ try{Thread.sleep(100);}catch(Exception e){} synchro
){} } });
        Thread t2=new Thread(()->{ synchronized(B){ try{Thread.sleep(100);}catch(Exception e){} synchro
){} } });
        t1.start(); t2.start(); t1.join(); t2.join();
    }
}
```

Singleton design pattern implementation

```
class Singleton {
    private static volatile Singleton instance;
    private Singleton(){}
    public static Singleton getInstance(){
        if(instance==null){
            synchronized(Singleton.class){
                if(instance==null) instance=new Singleton();
            }
        }
        return instance;
    }
}
public class Main { public static void main(String[] args){ System.out.println(Singleton.getInstance()
```

Create thread using Thread class

```
public class Main extends Thread {
    public void run(){ System.out.println("Hello from Thread"); }
    public static void main(String[] args){ new Main().start(); }
}
```

Create thread using Runnable interface

```
public class Main {
    public static void main(String[] args){
        Runnable r = () -> System.out.println("Hello from Runnable");
        new Thread(r).start();
    }
}
```

Synchronization example

```
class Counter {
```

```
    int c=0;
    synchronized void inc(){ c++; }
}
public class Main {
    public static void main(String[] args) throws Exception {
        Counter ctr=new Counter();
        Thread t1=new Thread(()->{ for(int i=0;i<1000;i++) ctr.inc(); });
        Thread t2=new Thread(()->{ for(int i=0;i<1000;i++) ctr.inc(); });
        t1.start(); t2.start(); t1.join(); t2.join();
        System.out.println(ctr.c);
    }
}
```

Inter-thread communication (wait(), notify())

```
class Shared {
    private int data; private boolean ready=false;
    synchronized void produce(int v) throws InterruptedException {
        while(ready) wait();
        data=v; ready=true; notify();
    }
    synchronized int consume() throws InterruptedException {
        while(!ready) wait();
        ready=false; notify(); return data;
    }
}
public class Main {
    public static void main(String[] args) throws Exception {
        Shared s=new Shared();
        Thread p=new Thread(()->{ try{ s.produce(42); }catch(Exception e){} });
        Thread c=new Thread(()->{ try{ System.out.println(s.consume()); }catch(Exception e){} });
        p.start(); c.start(); p.join(); c.join();
    }
}
```

Thread pool example (ExecutorService)

```
import java.util.concurrent.*;
public class Main {
    public static void main(String[] args) throws Exception {
        ExecutorService pool=Executors.newFixedThreadPool(2);
        for(int i=1;i<=4;i++){
            final int id=i;
            pool.submit(()-> System.out.println("Task "+id+" by "+Thread.currentThread().getName()));
        }
        pool.shutdown();
    }
}
```

Lambda expressions example

```
interface Op { int run(int a,int b); }
public class Main {
    public static void main(String[] args) {
        Op add=(a,b)->a+b;
        System.out.println(add.run(2,3));
    }
}
```

Functional interfaces (Predicate, Function, Consumer)

```
import java.util.function.*;
public class Main {
    public static void main(String[] args) {
        Predicate<Integer> isEven = x -> x%2==0;
        Function<Integer,Integer> square = x -> x*x;
        Consumer<Integer> print = System.out::println;
        if(isEven.test(4)) print.accept(square.apply(4));
    }
}
```

Streams API (filter, map, reduce, collect)

```
import java.util.*;
import java.util.stream.*;
public class Main {
    public static void main(String[] args) {
        List<Integer> nums=Arrays.asList(1,2,3,4,5);
        int sumSquaresOfEven = nums.stream().filter(x->x%2==0).map(x->x*x).reduce(0,Integer::sum);
        List<Integer> doubled = nums.stream().map(x->x*2).collect(Collectors.toList());
        System.out.println(sumSquaresOfEven + " " + doubled);
    }
}
```

Method references

```
import java.util.*;
public class Main {
    static void print(Integer x){ System.out.println(x); }
    public static void main(String[] args) {
        List<Integer> list=Arrays.asList(1,2,3);
        list.forEach(Main::print);
    }
}
```

Optional class usage

```
import java.util.Optional;
public class Main {
    public static void main(String[] args) {
        Optional<String> o = Optional.ofNullable(null);
        System.out.println(o.orElse("default"));
    }
}
```

Library Management System (mini demo)

```
import java.util.*;
class Book { String id,title; boolean issued; Book(String id,String t){this.id=id;this.title=t;} }
class Library {
    Map<String,Book> books=new HashMap<>();
    void add(Book b){ books.put(b.id,b); }
    boolean issue(String id){ Book b=books.get(id); if(b!=null && !b.issued){ b.issued=true; return tru
turn false; }
    void list(){ books.values().forEach(b->System.out.println(b.id+":"+b.title+":"+b.issued)); }
}
public class Main {
    public static void main(String[] args) {
        Library lib=new Library(); lib.add(new Book("1","Java Basics")); lib.add(new Book("2","DSA"));
        lib.issue("1"); lib.list();
    }
}
```

Banking System (deposit, withdraw, balance)

```
class Account {
    private int balance;
    Account(int bal){ this.balance=bal; }
    synchronized void deposit(int amt){ balance+=amt; }
    synchronized boolean withdraw(int amt){ if(amt<=balance){ balance-=amt; return true; } return false
    synchronized int balance(){ return balance; }
}
public class Main {
    public static void main(String[] args) {
        Account acc=new Account(1000);
        acc.deposit(500); acc.withdraw(200);
        System.out.println(acc.balance());
    }
}
```

Employee Management System (mini)

```
import java.util.*;
class Employee { int id; String name; Employee(int id,String n){this.id=id;this.name=n;} }
class EMS {
    Map<Integer,Employee> map=new HashMap<>();
```

```
    void add(Employee e){ map.put(e.id,e); }
    Employee get(int id){ return map.get(id); }
}
public class Main { public static void main(String[] args){ EMS s=new EMS(); s.add(new Employee(1,"Ana"
tem.out.println(s.get(1).name); } }
```

Student Management System (mini)

```
import java.util.*;
class Student { int id; String name; Student(int id,String n){this.id=id;this.name=n;} }
class SMS {
    Map<Integer,Student> map=new HashMap<>();
    void add(Student s){ map.put(s.id,s); }
    Student get(int id){ return map.get(id); }
}
public class Main { public static void main(String[] args){ SMS s=new SMS(); s.add(new Student(1,"Max"
em.out.println(s.get(1).name); } }
```