Solutions to Java Practice Questions

1. ArrayList

Q1: Cities

```java
import java.util.*;

class Q1 {
    public static void main(String[] args) {
        ArrayList<String> cities = new ArrayList<>();
        cities.add("Paris");
        cities.add("London");
        cities.add("New York");

        System.out.println(cities); // [Paris, London, New York]
    }
}
```

Q2: Remove evens

```java
import java.util.*;

class Q2 {
    public static void main(String[] args) {
        ArrayList<Integer> numbers = new ArrayList<>(Arrays.asList(1,2,3,4,5));
        numbers.removeIf(n -> n % 2 == 0);
        System.out.println(numbers); // [1, 3, 5]
    }
}
```

Q3: Largest number

```java
import java.util.*;

class Q3 {
    public static void main(String[] args) {
        ArrayList<Integer> nums = new ArrayList<>(Arrays.asList(10, 25, 5, 30));
        int max = Collections.max(nums);
        System.out.println(max); // 30
    }
}
```

Q4: Reverse list manually

```java
import java.util.*;
```

```java
class Q4 {
    public static void main(String[] args) {
        ArrayList<String> list = new ArrayList<>(Arrays.asList("A","B","C","D"));
        ArrayList<String> reversed = new ArrayList<>();
        for (int i = list.size() - 1; i >= 0; i--) {
            reversed.add(list.get(i));
        }
        System.out.println(reversed); // [D, C, B, A]
    }
}
```

Q5: User search

```java
import java.util.*;

class User {
    int id; String name;
    User(int id, String name) { this.id = id; this.name = name; }
}

class Q5 {
    public static void main(String[] args) {
        ArrayList<User> users = new ArrayList<>();
        users.add(new User(1,"Alice"));
        users.add(new User(2,"Bob"));

        User found = null;
        for (User u : users) {
            if (u.id == 2) found = u;
        }
        System.out.println(found != null ? found.name : "Not Found"); // Bob
    }
}
```

---

2. Map

Q1: Fruits

```java
import java.util.*;

class M1 {
    public static void main(String[] args) {
        Map<String, Integer> fruits = new HashMap<>();
        fruits.put("Apple", 100);
        fruits.put("Banana", 50);
        fruits.put("Mango", 120);
        System.out.println(fruits);
    }
}
```

Q2: Student lookup

```
import java.util.*;

class M2 {
    public static void main(String[] args) {
        Map<Integer, String> students = new HashMap<>();
        students.put(101,"Alice");
        students.put(102,"Bob");

        System.out.println(students.get(101)); // Alice
    }
}
```

Q3: Character frequency

```
import java.util.*;

class M3 {
    public static void main(String[] args) {
        String text = "banana";
        Map<Character,Integer> freq = new HashMap<>();
        for (char c : text.toCharArray()) {
            freq.put(c, freq.getOrDefault(c,0)+1);
        }
        System.out.println(freq); // {a=3, b=1, n=2}
    }
}
```

Q4: LinkedHashMap order

```
import java.util.*;

class M4 {
    public static void main(String[] args) {
        Map<Integer,String> employees = new LinkedHashMap<>();
        employees.put(1,"Alice");
        employees.put(2,"Bob");
        employees.put(3,"Charlie");
        System.out.println(employees); // Maintains order
    }
}
```

Q5: TreeMap descending

```
import java.util.*;

class M5 {
    public static void main(String[] args) {
        TreeMap<Integer,String> rollMap = new TreeMap<>(Collections.reverseOrder());
        rollMap.put(101,"Alice");
        rollMap.put(103,"Charlie");
        rollMap.put(102,"Bob");
        System.out.println(rollMap); // {103=Charlie, 102=Bob, 101=Alice}
    }
}
```

3. Lambda & Functional Interfaces

Q1: Sort list

```
import java.util.*;

class L1 {
    public static void main(String[] args) {
        List<Integer> nums = Arrays.asList(5,3,8,1);
        nums.sort((a,b) -> a-b);
        System.out.println(nums); // [1,3,5,8]
    }
}
```

Q2: Predicate filter

```
import java.util.*;
import java.util.function.Predicate;

class L2 {
    public static void main(String[] args) {
        List<Integer> nums = Arrays.asList(5,12,18,7);
        Predicate<Integer> greaterThan10 = n -> n > 10;
        nums.stream().filter(greaterThan10).forEach(System.out::println);
        // 12, 18
    }
}
```

Q3: Function string length

```
import java.util.function.Function;

class L3 {
    public static void main(String[] args) {
        Function<String,Integer> lengthFn = s -> s.length();
        System.out.println(lengthFn.apply("Spring")); // 6
    }
}
```

Q4: Custom functional interface

```
@FunctionalInterface
interface MathOperation {
    int operate(int a, int b);
}

class L4 {
    public static void main(String[] args) {
        MathOperation add = (a,b) -> a+b;
        MathOperation mul = (a,b) -> a*b;

        System.out.println(add.operate(2,3)); // 5
        System.out.println(mul.operate(2,3)); // 6
    }
}
```

Q5: Consumer

```
import java.util.*;
import java.util.function.Consumer;

class L5 {
    public static void main(String[] args) {
        List<String> words = Arrays.asList("Java","Spring","Boot");
        Consumer<String> printer = w -> System.out.println(w+" ("+w.length()+")");
        words.forEach(printer);
    }
}
```

---

4. Stream API

Q1: Double numbers

```
import java.util.*;
import java.util.stream.*;

class S1 {
    public static void main(String[] args) {
        List<Integer> nums = Arrays.asList(2,4,6,8);
        List<Integer> doubled = nums.stream().map(n -> n*2).toList();
```

```java
        System.out.println(doubled); // [4,8,12,16]
    }
}
```

Q2: Filter names

```java
import java.util.*;
import java.util.stream.*;

class S2 {
    public static void main(String[] args) {
        List<String> names = Arrays.asList("Alice","Bob","Andrew","Tom");
        List<String> result = names.stream().filter(n -> n.startsWith("A")).toList();
        System.out.println(result); // [Alice, Andrew]
    }
}
```

Q3: Sum of odd numbers

```java
import java.util.*;
import java.util.stream.*;

class S3 {
    public static void main(String[] args) {
        List<Integer> nums = Arrays.asList(1,2,3,4,5);
        int sum = nums.stream().filter(n -> n%2!=0).reduce(0,(a,b)->a+b);
        System.out.println(sum); // 9
    }
}
```

Q4: Group by department

```java
import java.util.*;
import java.util.stream.*;

class Employee {
    String name, dept;
    Employee(String n,String d){name=n;dept=d;}
}

class S4 {
    public static void main(String[] args) {
        List<Employee> emps = Arrays.asList(
            new Employee("Alice","IT"),
            new Employee("Bob","HR"),
            new Employee("Charlie","IT")
        );

        Map<String,List<Employee>> grouped =
            emps.stream().collect(Collectors.groupingBy(e -> e.dept));

        grouped.forEach((dept,list) -> {
            System.out.println(dept+": "+list.stream().map(e->e.name).toList());
        });
    }
}
```

Q5: Flatten list of lists

```
import java.util.*;
import java.util.stream.*;

class S5 {
    public static void main(String[] args) {
        List<List<Integer>> nested = Arrays.asList(
            Arrays.asList(1,2),
            Arrays.asList(3,4),
            Arrays.asList(5)
        );

        List<Integer> flat = nested.stream()
                                   .flatMap(List::stream)
                                   .toList();

        System.out.println(flat); // [1,2,3,4,5]
    }
}
```

5. Utility Classes

Q1: Optional with Map

```
import java.util.*;

class U1 {
    public static void main(String[] args) {
        Map<Integer,String> map = new HashMap<>();
        map.put(1,"Hello");

        String result = Optional.ofNullable(map.get(2)).orElse("Unknown");
        System.out.println(result); // Unknown
    }
}
```

Q2: Print today's date

```java
import java.time.*;
import java.time.format.DateTimeFormatter;

class U2 {
    public static void main(String[] args) {
        LocalDate today = LocalDate.now();
        System.out.println(today.format(DateTimeFormatter.ofPattern("dd-MM-yyyy")));
    }
}
```

Q3: Parse date

```java
import java.time.*;

class U3 {
    public static void main(String[] args) {
        LocalDate d = LocalDate.parse("2025-09-24");
        System.out.println(d.getYear()); // 2025
    }
}
```

Q4: Reverse string with StringBuilder

```java
class U4 {
    public static void main(String[] args) {
        StringBuilder sb = new StringBuilder("SpringBoot");
        System.out.println(sb.reverse()); // tooBgnirpS
    }
}
```

Q5: Optional with method

```java
import java.util.*;

class User {
    String name;
    User(String n){name=n;}
}

class U5 {
    static User findUserById(int id){
        return id==1 ? new User("Alice") : null;
    }
    public static void main(String[] args) {
        Optional<User> u = Optional.ofNullable(findUserById(2));
        System.out.println(u.map(user -> user.name).orElse("Not Found")); // Not Found
    }
}
```

🔥 Use Case: Orders Grouped by Customer

```java
import java.util.*;
import java.util.stream.*;

class Order {
    int orderId;
    String customer;
    int amount;
    Order(int id,String c,int amt){orderId=id;customer=c;amount=amt;}
}

class UseCase {
    public static void main(String[] args) {
        List<Order> orders = Arrays.asList(
            new Order(1,"Alice",100),
            new Order(2,"Bob",200),
            new Order(3,"Alice",150),
            new Order(4,"Charlie",150),
            new Order(5,"Bob",200)
        );

        Map<String,Integer> totals = orders.stream()
            .collect(Collectors.groupingBy(
                o -> o.customer,
                Collectors.summingInt(o -> o.amount)
            ));

        totals.forEach((cust,total) -> System.out.println(cust+" → "+total));
        // Alice → 250
        // Bob → 400
        // Charlie → 150
    }
}
```