MODULE: Java Programming

ASSIGNMENT 1 – Java Basics & Logic Building

Learning Objectives

This assignment will help learners:

- Practice loop constructs
- Work with 1D and 2D arrays
- Enforce input validation
- Apply algorithmic thinking
- Understand date computation logic
- Produce clean, modular Java code

General Instructions

1. Write each question in a separate `.java` file.
2. Use **iterative logic only** (no recursion).
3. Use meaningful variable names and proper indentation.
4. Display clear prompts and readable output.
5. Validate user inputs wherever necessary.
6. Use helper methods for clarity.
7. Test each program with at least **three different inputs**.
8. Avoid magic numbers—use constants where needed.

Estimated Time & Difficulty

Question	Time	Difficulty
Q1	20–25 min	Beginner
Q2	20–30 min	Beginner
Q3	10–15 min	Beginner
Q4	20–30 min	Beginner → Intermediate
Q5	45–60 min	Intermediate → Advanced

🧪 Evaluation Rubric

Criteria	Weight
Correctness	50%

Criteria	Weight
Code Quality	20%
Input Validation	10%
Modularity (methods)	10%
Output Format	10%

Q1. Fibonacci Series & Average Calculation

Requirements

1. Generate the first **20 Fibonacci numbers** using iterative logic.
2. Store all numbers in an array.
3. Print the numbers in one line.
4. Compute and print the average.

Expected Output

```
The first 20 Fibonacci numbers are:
1 1 2 3 5 8 ... 6765
Average: 885.5
```

Common Mistakes to Avoid

- Incorrect starting values
- Not storing values in array
- Overflow—use `long` if necessary

Q2. Grades Average Calculator with Validation

Requirements

1. Prompt user for number of students.
2. Collect grades into an `int[]`.
3. Each grade must be between **0 and 100**.
4. If invalid, print:

```
Invalid grade. Try again.
```

5. Compute and print the average.

Sample Interaction

```
Enter number of students: 3
Enter grade for student 1: 55
Enter grade for student 2: 108
Invalid grade. Try again.
Enter grade for student 2: 56
Enter grade for student 3: 57

Average: 56.0
```

Corner Cases

- Number of students = 0 → show message and exit

- Reject negative grades
- Reject grades > 100

Q3. Array Copy Method

Task

Write the following method:

```
public static int[] copyOf(int[] array)
```

Requirements:

- Return a **new independent array** with the same values.
- Do **not** return the same reference.
- Demonstrate independence by modifying the copied array.

Corner Cases

- Empty array → return new empty array
- Single-element array → return new array

Q4. 2D Array – Pattern Triangle (Pascal-Style)

Requirements

Using a 2D array, print the following pattern:

```
1
1 2
1 2 3
1 2 3 4
...
1 2 3 4 5 6 7 8
```

Notes

- This is **not** mathematical Pascal's triangle.
- Must use nested loops and a 2D `int[][]` array.

Common Mistakes

- Misaligned spacing
- Trailing spaces
- Incorrect array initialization

Q5. Next Date Calculator

Requirements

1. Input **day, month, year** from the user.
2. Validate the date.
3. Compute the **next day's date** manually.
4. Display both dates.

Sample Output

```
Enter day: 31
Enter month: 12
Enter year: 2022

Today: 31/12/2022
Next Date: 1/1/2023
```

Validations Required

- Correct days for each month
- Leap year logic:

```
(year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)
```

- Month-end transitions
- Year-end transition (31 Dec → next year)

Common Mistakes

- Incorrect leap year calculation
- Wrong month lengths
- Not handling December rollover

Optional (Not Required)

Use Java 8 `LocalDate` to compare outputs.

BONUS CHALLENGES (Optional)

Bonus 1: Fibonacci Performance Check (No Recursion)

Measure performance using:

- Basic iterative loop
- Iterative loop with `ArrayList<Long>`

Bonus 2: Grade Distribution Histogram

Print ranges:

```
0–20: **
21–40: ****
41–60: ***
61–80: *
81–100: **
```

Bonus 3: Triangle Printing Using StringBuilder

Improve output efficiency.

Bonus 4: Next-Date Comparison Using LocalDate

Compare manual logic with API results.

Reflection Questions

1. Why is input validation critical in real-world applications?

2. How is deep copy different from shallow copy?
3. Why is date calculation prone to errors in software?
4. What loop-related mistakes did you avoid consciously?
5. Which question required the most logical thinking and why?