USER STORY IMPLEMENTATION - CORE JAVA

1. Write a Java program that prints all prime numbers between 1 and 100 using nested loops and efficient control statements.
2. Randomly generate a number between 1 and 100.Let the user guess, and after each guess, tell them whether the guess was too low, too high, or correct.Use `do-while` loops and control flow constructs properly.
3. Copy Elements from One Array to Another-Write a Java program to copy all elements of one array into another array.
4. Remove Duplicate Elements-Write a Java method that removes duplicate elements from an integer array (ignore order).
5. Sort an Array in Ascending and Descending Order-Write two methods: one to sort an array in ascending order and another to sort it in descending order (without using built-in sorting methods).
6. Left Rotate the Elements of an Array-Write a Java program to left rotate the elements of an array by a given number of positions.
7. Find the Second Largest Element-Write a Java program to find the second largest element in an array without sorting the array.
8. Find LCM of Two Numbers-Write a Java program to find the Least Common Multiple (LCM) of two integers using a loop.
9. Check if a Number is a Perfect Number-Write a Java program to check whether a given number is a perfect number (i.e., the sum of its divisors equals the number itself, e.g., 6 → 1+2+3 = 6).
10. Find the Type of Triangle-Write a Java program to determine whether a triangle is Equilateral, Isosceles, or Scalene based on the lengths of its sides.
11. Electricity Bill Calculator-Write a Java program to calculate electricity bill:

    First 100 units → ₹1.5 per unit

    Next 200 units → ₹2.5 per unit

    Above 300 units → ₹4 per unit

    Add a surcharge of ₹50 if total bill exceeds ₹500.

12. Implement a program that swaps two variables without using a third variable or arithmetic operators (+, -, *, /).
13. Write a Java program to determine whether two numbers are equal using only logical operators.
14. Implement exponentiation (power function) manually without using `Math.pow()`, using only loops and `*` operator.
15. Simulate the ternary operator (`condition ? true : false`) manually using only `if-else` statements.

16. Without using relational operators (`<`, `>`, `==`), write a Java program to find which of two numbers is greater.
17. Write a Java program that simulates a vending machine with nested switch-cases.
18. Implement a number guessing game with three difficulty levels (easy, medium, hard) that control the number of attempts.
19. Create a mini bank management system with options to deposit, withdraw, transfer, check balance using switch-case and loops.
20. Write a grade analyzer that takes multiple students' marks and classifies grades using nested `if-else`.
21. Implement a program to find and print all Armstrong numbers between 1 and 10000.
22. Write a program to reverse each word of a sentence individually without using built-in functions.
23. Implement a program to find all prime numbers between 1 and N without using functions.
24. Find the GCD (Greatest Common Divisor) of two numbers without using recursion.
25. Create overloaded methods for finding the maximum of two, three, and four numbers.
26. Write a recursive method to compute the Fibonacci sequence up to N terms.
27. Implement a method to validate a strong password (must contain uppercase, lowercase, digit, special character).
28. Write a method that accepts an array of integers and returns the median.
29. Create a method to simulate a simple library system: borrow, return, and view book availability.
30. Implement a method that finds the longest palindromic substring in a given string.
31. Write a method to calculate compound interest recursively.
32. Implement a method that checks if two strings are anagrams of each other.
33. Design a `BankAccount` class with methods for deposit, withdraw, transfer, and account balance check.