CORE JAVA PROGRAMS

List of Programming Challenges Solved:
1. Create a class to output “good morning” using a text editor and check output.
2. Create a new Project in Intelij Idea and output “subscribe” on the console.
3. Show the patterns
4. Show the patterns using single print statement
5. Create a program to input name of the person and respond with ”Welcome NAME to KG Coding”
6. Create a program to add two numbers.
7. Create a program to swap two numbers.
8. Create a program that takes two numbers and shows result of all arithmetic operators (+,-,*,/,%).
9. Create a program to calculate product of two floating points numbers.
10. Create a program to calculate Perimeter of a rectangle. Perimeter of rectangle ABCD = A+B+C+D
11. Create a program to calculate the Area of a Triangle. Area of triangle = ½*B*H
12. Create a program to calculate simple interest. Simple Interest = (P x T x R)/100
13. Create a program to calculate Compound interest. Compound Interest = P(1 + R/100)t
14. Create a program to convert Fahrenheit to Celsius °C = (°F - 32) × 5/9
15. Create a program that determines if a number is positive, negative, or zero.
16. Create a program that determines if a number is odd or even.
17. Create a program that determines the greatest of the three numbers.
18. Create a program that determines if a given year is a leap year (considering conditions like divisible by 4 but not 100, unless also divisible by 400).
19. Create a program that calculates grades based on marks A -> above 90% B -> above 75% C -> above 60% D -> above 30% F -> below 30%
20. Create a program that categorize a person into different age groups Child -> below 13 Teen -> below 20 Adult -> below 60 Senior-> above 60
21. Create a program that shows bitwise AND of two numbers.
22. Create a program that shows bitwise OR of two numbers.
23. Create a program that shows bitwise XOR of two numbers.
24. Create a program that shows bitwise compliment of a number.
25. Create a program that shows use of left shift operator.
26. Create a program that shows use of right shift operator.
27. Write a program to check if a given number is even or odd using bitwise operators.
28. Develop a program that prints the multiplication table for a given number.
29. Create a program to sum all odd numbers from 1 to a specified number N.
30. Write a function that calculates the factorial of a given number.
31. Create a program that computes the sum of the digits of an integer.
32. Create a program to find the Least Common Multiple (LCM) of two numbers.
33. Create a program to find the Greatest Common Divisor (GCD) of two integers.

34. Create a program to check whether a given number is prime.
35. Create a program to reverse the digits of a number.
36. Create a program to print the Fibonacci series up to a certain number.
37. Create a program to check if a number is an Armstrong number.
38. Create a program to verify if a number is a palindrome.
39. Create a program that print patterns using loops.

PART ---2---FROM 40 BEGINS

40. Create a program to find the sum and average of all elements in an array.
41. Create a program to find number of occurrences of an element in an array.
42. Create a program to find the maximum and minimum element in an array.
43. Create a program to check if the given array is sorted.
44. Create a program to return a new array deleting a specific element.
45. Create a program to reverse an array.
46. Create a program to check is the array is palindrome or not.
47. Create a program to merge two sorted arrays.
48. Create a program to search an element in a 2-D array.
49. Create a program to do sum and average of all elements in a 2-D array.
50. Create a program to find the sum of two diagonal elements.
51. Create a Book class for a library system. Instance variables: title, author, isbn. Static variable: totalBooks, a counter for the total number of book instances. Instance methods: borrowBook(), returnBook(). Static method: getTotalBooks(), to get the total number of books in the library.
52. Design a Course class. Instance variables: courseName, enrolledStudents. Static variable: maxCapacity, the maximum number of students for any course. Instance methods: enrollStudent(String studentName), unenrollStudent(String studentName). Static method: setMaxCapacity(int capacity), to set the maximum capacity for courses.
53. Create a program to find the minimum of two numbers.
54. Create a program to find if the given number is even or odd.
55. Create a program to calculate the absolute value of a given integer.
56. Create a program to Based on a student's score, categorize as "High", "Moderate", or "Low" using the ternary operator (e.g., High for scores > 80, Moderate for 50-80, Low for < 50).
57. Create a program to print the month of the year based on a number (1-12) input by the user.
58. Create a program to create a simple calculator that uses a switch statement to perform basic arithmetic operations like addition, subtraction, multiplication, and division.
59. Create a program using do-while to find password checker until a valid password is entered.
60. Create a program using do-while to implement a number guessing game.
61. Create a program using for loop multiplication table for a number.
62. Create a program using for to display if a number is prime or not.
63. Create a program using for-each to find the maximum value in an integer array.
64. Create a program using for-each to the occurrences of a specific element in an array.
65. Create a program using break to read inputs from the user in a loop and break the loop if a specific keyword (like "exit") is entered.
66. Create a program using continue to sum all positive numbers entered by the user; skip any negative numbers.
67. Create a program using continue to print only even numbers using continue for odd numbers.
68. Create a program using recursion to display the Fibonacci series upto a certain number.
69. Create a program using recursion to check if a string is a palindrome using recursion.

70. Define a Student class with fields like name and age, and use toString to print student details.
71. Concatenate and Convert: Take two strings, concatenate them, and convert the result to uppercase.
72. Calculate the area and circumference of a circle for a given radius using Math.PI
73. Simulate a dice roll using Math.random() and display the outcome (1 to 6).
74. Create a number guessing game where the program selects a random number, and the user has to guess it.
75. Take an array of words and concatenate them into a single string using StringBuilder.
76. Create an object with final fields and a constructor to initialize them.
77. Create a simple application with at least two packages: com.example.geometry and com.example.utils. In the geometry package, define classes like Circle and Rectangle. In the utils package, create a Calculator class that can compute areas of these shapes.
78. Define a BankAccount class with private attributes like accountNumber, accountHolderName, and balance. Provide public methods to deposit and withdraw money, ensuring that these methods don't allow illegal operations like withdrawing more money than the current balance.
79. Define a class Employee with private attributes (like name, age, and salary), public methods to get and set these attributes, and a package-private method to displayEmployeeDetails. Create another class in the same package to test access to the displayEmployeeDetails method.--

PART 2 --THE END

80. Start with a base class LibraryItem that includes common attributes like itemID, title, and author, and methods like checkout() and returnItem(). Create subclasses such as Book, Magazine, and DVD, each inheriting from LibraryItem. Add unique attributes to each subclass, like ISBN for Book, issueNumber for Magazine, and duration for DVD.
81. Create a class Person with attributes name and age. Override equals() to compare Person objects based on their attributes. Override hashCode() consistent with the definition of equals().
82. Create a class ArrayOperations with a static nested class Statistics. Statistics could have methods like mean(), median(), which operate on an array.
83. Create an abstract class Shape with an abstract method calculateArea(). Implement two subclasses: Circle and Square. Each subclass should have relevant attributes (like radius for Circle, side for Square) and their own implementation of the calculateArea() method.
84. Create an interface Flyable with an abstract method fly(). Create an abstract class Bird that implements Flyable. Implement a subclass Eagle that extends Bird. Provide an implementation for the fly() method.
85. In a class Calculator, create multiple add() methods that overload each other and can sum two integers, three integers, or two doubles. Demonstrate how each can be called with different numbers of parameters.
86. Define a base class Vehicle with a method service() and a subclass Car that overrides service(). In Car's service(), provide a specific implementation that calls super.service() as well, to show how overriding works.
87. Arithmetic Exception Handling Write a program that asks the user to enter two integers and then divides the first by the second. The program should handle any arithmetic exceptions that may occur (like division by zero) and display an appropriate message. Key Points: - Use Scanner

to read user input. - Implement a try-catch block to handle ArithmeticException. - Display a user-friendly message if division by zero occurs.
88. File Not Found Exception Handling. Write a program to read a filename from the user and display its content. The program should handle the situation where the file does not exist. Key Points: - Use Scanner to read the filename from the user. - Use FileReader to read the file content. - Implement a try-catch block to handle FileNotFoundException. - Display a message informing the user if the file is not found.
89. Write a method concatenate Strings that takes variable arguments of String type and concatenates them into a single string.
90. Write a program that sorts a list of String objects in descending order using a custom Comparator.
91. Use the Collections class to count the frequency of a particular element in an ArrayList.
92. Write a method that swaps two elements in an ArrayList, given their indices.
93. Create a program that reverses the elements of a List and prints the reversed list.
94. Create a PriorityQueue of a custom class Student with attributes name and grade. Use a comparator to order by grade.
95. Write a program that takes a string and returns the number of unique characters using a Set.
96. Create an enum called Day that represents the days of the week. Write a program that prints out all the days of the week from this enum.
97. Enhance the Day enum by adding an attribute that indicates whether it is a weekday or weekend. Add a method in the enum that returns whether it's a weekday or weekend, and write a program to print out each day along with its type.
98. Create a Map where the keys are country names (as String) and the values are their capitals (also String). Populate the map with at least five countries and their capitals. Write a program that prompts the user to enter a country name and then displays the corresponding capital, if it exists in the map.
99. Write a program that creates two threads. Each thread should print "Hello from Thread X", where X is the number of the thread (1 or 2), ten times, then terminate.
100. Write a program that starts a thread and prints its state after each significant event (creation, starting, and termination). Use Thread.sleep() to simulate long-running tasks and Thread.getState() to print the thread's state.
101. Create three threads. Ensure that the second thread starts only after the first thread ends and the third thread starts only after the second thread ends using the join method. Each thread should print its start and end along with its name.
102. Simulate a traffic signal using threads. Create three threads representing three signals: RED, YELLOW, and GREEN. Each signal should be on for a certain time, then switch to the next signal in order. Use sleep for timing and synchronize to make sure only one signal is active at a time.
103. Write a program that creates a single-threaded executor service. Define and submit a simple Runnable task that prints numbers from 1 to 10. After submission, shut down the executor.
104. Create a fixed thread pool with a specified number of threads using Executors.newFixedThreadPool(int). Submit multiple tasks to this executor, where each task should print the current thread's name and sleep for a random time between 1 and 5 seconds. Finally, shut down the executor and handle proper termination using awaitTermination.
105. Write a program that uses an executor service to execute multiple Callable tasks. Each task should calculate and return the factorial of a number provided to it. Use Future objects to receive

the results of the calculations. After all tasks are submitted, retrieve the results from the futures, print them, and ensure the executor service is shut down correctly.

106. Write a lambda expression that takes two integers and returns their multiplication. Then, apply this lambda to a pair of numbers.

107. Convert an array of strings into a stream. Then, use the stream to print each string to the console.

108. Given a list of strings, use stream operations to filter out strings that have length of 10 or more and then concatenate the remaining strings.

109. Given a list of integers, use stream operations to filter odd numbers and print them.

110. Create your own functional interface with a single abstract method that accepts an integer and returns a boolean. Implement it using a lambda that checks if the number is prime.

111. Write two versions of a program that calculates the factorial of a number: one using structural (procedural) programming, and the other using functional programming.

112. Write a function that accepts a string and returns an Optional<String>. If the string is empty or null, return an empty Optional, otherwise, return an Optional containing the uppercase version of the string.

113. Given an array of integers, create a stream, use the distinct operation to remove duplicates, and collect the result into a new list.

114. Create a list of employees with name and salary fields. Write a comparator that sorts the employees by salary. Then, use this comparator to sort your list using the sort stream operation.

115. Create a list of strings representing numbers ("1", "2", ...). Convert each string to an integer, then again calculating squares of each number using the map operation and sum up the resulting integers.