OOPS:

- Write a class called Car that encapsulates the properties and behaviors of a car. The class should have private instance variables for the car's make, model, and year, and public methods for getting and setting those variables. The class should also have a method called startEngine() that prints a message indicating that the engine has started.

- Create a subclass of Car called ElectricCar that adds an instance variable for the car's battery range and a method called chargeBattery() that prints a message indicating that the battery is being charged.

- Write a program that demonstrates polymorphism by creating an array of Car objects that contains both Car and ElectricCar objects. Use a loop to call the startEngine() method on each object in the array, and observe how the correct method is called for each object.

- write a program to divide two number and print the remainder and quotient, without using the '/' or '*' operator and only using '+' and '-' operators.

Exception Handling

Objective: Write a method that processes user input and handles exceptions appropriately.
Details: Implement a method processInput() that reads a numerical input from the user and calculates the reciprocal. Handle exceptions related to invalid input (e.g., non-numerical input, division by zero).
Functions to Implement:
processInput(): Prompts user for a number, calculates the reciprocal, and handles any possible input errors or exceptions.

Final keyword:

- Write a program that demonstrates the use of the final keyword with variables, methods, and classes. Create a class with a final variable, a final method, and a final class, and observe how the final keyword affects each one.

Abstraction, interface basics:

- Write an abstract class called Animal that defines a method called makeSound() that is meant to be overridden by concrete subclasses. Create a subclass called Dog that overrides the makeSound() method to print a message indicating that the dog is barking.

Primitive and Reference Variable:

- Write a program that demonstrates Primitive and Reference Variable by passing an integer and an array to a method, and modifying them in the method. Print the integer and the array in main , after they come out of the method. Observe how the integer is passed by value, but the array is passed by reference.

- Write a program to demonstrate the use of a enum by creating an enum called DaysOfWeek and write code logic to print "Holiday" or "Not Holiday" based on the day of the week.

- Write a program to find all the trailing zeroes in the factorial of a number

- Write a program where u create a utility method inside a "utility" package to find the length of a given integer. The method should be accessible without creating an object for the class.

Object Class:

- create two classes user and secondUser with same variables like name, age, etc. Create getters, setters, all argument constructors and initialize the objects of these two classes by creating objects for them. override the toString, hashcode and equals methods for only the user class. try to show the behaviour of the object with and without the overriden methods, by printing them using toString, and comparing them using equals()

String Handling

Objective: Implement a string processor that performs various manipulations and searches on strings.
Details: Create a class StringProcessor that provides methods to format, split, and search within strings.
Functions to Implement:
reverseString(String str): Returns the reverse of the given string.
countOccurrences(String text, String sub): Counts how many times sub appears in text.
splitAndCapitalize(String str): Splits the string by spaces and capitalizes each word.

Finding All Anagrams in a String

Objective: Given a string s and a non-empty string p, find all the start indices of p's anagrams in s.
Details: The goal is to develop a function that, given a string s and a non-empty string p, returns all the starting indices of the substrings in s that are anagrams of p. An anagram of a string is another string that contains the same characters, only the order of characters can be different. For example, "abc" is an anagram of "bca".
Functions to Implement:
List<Integer> findAnagrams(String s, String p): Returns a list of starting indices of the anagrams of p in s.