Please do all the below assignment by considering the below

* In the object oriented way.
* It should be like generalized api (application interface)
* Create two separate packages. One for the api and other for test classes.
* Write a separate function to implement it.
* Write a separate test class to test it.
* The function should have proper method signature.
* Avoid writing SOP statements in these functions.
* Should able to extend the functionality easily without modifying the existing logic.
* Follow proper naming conventions for package, class, method variable.
* Do not use generalized names for any identifier.
* Consider code reusability, memory and performance.

**ASSIGNMENTS**

General programming logic

1. Write a function to add two numbers.
2. Write a function to get biggest between 2 numbers.
3. Write a function to find whether the give number is even or not
4. Write a function to get grade of 3 subject marks.
   1. Grade is based on average of 3 subject marks
      1. Average>90 => A+
      2. Average<90 and average>=80 => A
      3. Average<80 and average>=70 => A-
      4. Average<70 and average>=60 => B+
      5. Average<60 and average>=50 => B
      6. Average<50 => FAIL
5. Write a function to print first 10 natural numbers
6. Write a function to print first 10 natural numbers in reverse order
7. Write a function to print first 10 even natural numbers
8. Write a function to print numbers between the range ( say 10 to 100 OR 100 to 200 etc.,)
9. Write a function to print mathematical table of given number
10. Write a function to find whether the given number is prime or not.
11. Write a program to print prime numbers between 2 to 100
12. Write a function to get sum of individual digits of given number
    1. i/p: 123 => output : 6 ( i.e., 1+2+3)
13. Write a function to get lucky number of given number
    1. Lucky number => sum of individual digits of given number
    2. If the sum is more than 9, again do sum of individual digits of this sum.
       1. i/p : 12345 => 6 ( i.e., 1+2+3+4+5 = 15, 1+5 = 6)
14. Write a function to get reverse of given number
15. Write a function to find whether the given number is palindrome or not
16. Write a function to find factorial of given number
17. Write a program to find ncr. Hint : ncr = ( n! / ( (n-r)! \* r!)
18. Write a function to find sum of individual digits of given number
19. Write a function to get reverse of given number
20. Write a function to find whether the given number is palindrome or not.
21. Write a function to convert given digit into the word format.
    1. If i/p is 1 -> output : One
    2. If i/p is 2 -> output : Two
22. Write a function to convert given number into word format.
    1. If i/p is 123 -> output : One Hundred Twenty Three

Printing \* or number in specific format

1. Write a program to print the \*s in the following formats

\*

\* \*

\* \* \*

\* \* \* \*



\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \*

\* \* \*

\* \*

\*

* 1. 1

1 2

1 2 3

1 2 3 4

|  |
| --- |
|  |

**String Operations**

1. Write a function to find the given two strings are equals or not.
2. Write a function to reverse the string
3. Write a function to find whether the given string is palindrome or not.
4. Write a function to count number characters in the given string.
5. Write a function to count number words in the given string
6. Write a function to count number of lower characters in the given string
7. Write a function to simulate trim function. (i.e., remove white spaces in the beginning and ending of the string if any)
8. Write a function to remove extra white spaces in the given sentence( If more than once white spaces there, remove it.)
   1. Ex input: Hello how are you?
   2. Output: Hello how are you?
9. Write a function to count how many number of times a particular character appears/present in the given string.

Ex: given string = “hello word”

Given character = ‘l’

Output: 2

1. Write a program to count how many number of times each character appears/present in the given string.

Ex: given string = “hello word”

Output:

h.....1

e.....1

l......2

o.....2

w....1

r.....1

d...1

Note: Do it after learning HashMap.

Arrays

Write a function to get biggest number in the given array.

Write a function to get index /position of biggest number in the given aray

Write a function to get sum of all the numbers in the given array

Write a function to search whether the given number is present or not in the given array (linear search and binary search)

Write a function to sort the given array (bubble sort/selection sort/ quick sort/ merge sort etc.,)

Write a function to merge two given arrays.

Write a function to insert an element in the array in given position

Write a function to delete an element from given position in the array.

Write a function delete given element from given array.

OOP

1. Understanding Generalization(Inheritance) and Composition(Creating the object)
   1. Create a class Employee,
      1. add id, name and salary properties. Generate getter/setter methods
      2. Test it.
         1. Write EmployeeTest class.
         2. Create instance of Employee in main method and try to display the employee details.
            1. Write generalized display method which display any employee instance.
            2. OR
            3. Override toString method in Employee class and display instance itself.(instead of displaying individual properties)
      3. Add a parameterized constructor in the Employee class (with mandatory properties). Test the same in EmployeeTest class.
   2. Create two objects of Employee class. Compare these objects whether two are equal or not.
      1. Compare by giving same values for all the properties.
      2. Compare by giving different value for all/few properties.
      3. Note: understand equals and hashCode methods
   3. Create ContractEmployee class which extends Employee class. Add ‘duration’ property and setter/getter methods
      1. Test the same in EmployeeTest class.
   4. Create Address class
      1. Add street, city, pincode properties and generate setter/getter methods
      2. Add Address object in Employee Class and generate setter/getter methods
      3. Test the same in EmployeeTest by creating
         1. Two employee instances with different details(id, name, salary and address)
         2. Display both employee details with address.
2. Collections
   1. ArrayList
      1. Write a program to add few integers in the array list and display the same (by writing display method)
         1. Display using iterator.
         2. Display using enhanced for loop.
         3. Write a search method to find whether given number is present or not.
         4. Write a get method to get the number from a particular position. (one line)
      2. Call sort method - Collectins.sort(arrayList) and call the display method again.
      3. Do the above for String objects.
      4. Do the above for Employee objects.
         1. Sort all employees based on salary.
         2. Sort all employees based on name
         3. Understand comparator usage
3. Employee Management

* Employee management application
* We should able to do CRUD operations (Create, Retrieve, Update, Delete operations)
* Create new employee.
* Retrieve employee details by providing emp id.
* Update the employee details.
* Delete employee by providing emp id.
* Get all employees.
* Similar for the Department – all CRUD operations.
* Add employee to particular department.
* Remove employee from particular department.
* Get all employees from particular department.
* Get all employees whose salary > some amount.
* Get all employees whose salary in-between some range.
* Get all employees whose salary in-between some range from particular department.

Step

* Create separate project. Project name like EmployeeManagement.
* Create 3 separate packages
  + com.cts.bean ( or com.cts.model)
  + com.cts.service
  + com.cts.service.impl
* In model package
  + Create Employee class (id, name and salary) and generate setter/getter methods.
  + Create Department class (id, name, List<Employee> employees) and generate setter/getter methods.
* In service package
  + Create EmployeeService interface
  + Create DepartmentService interface
  + Add proper method signatures for all the CRUD operations.
  + In EmployeeService interface
    - Public Boolean addEmployee(Employee emp);
    - Public Employee getEmployee(int id);
    - Public List<Employee> getAllEmployees();
    - Public List<Employee> getAllEmployees(int salary);
    - Public List<Employee> getAllEmployees(int minSalary, int maxSalary);
    - Public Boolean deleteEmployee(int id);
    - Public Boolean updateEmployee(Employee emp)
  + In DepartmentService interface
    - Public Boolean addDepartment(Department dept);
    - Public Department Department (int id);
    - Public List< Department > getAllDepartments();
    - Public Boolean deleteDepartment(int id); //cascade on delete?
    - Public Boolean updateDepartment(Department dept);
    - Public Boolean addEmployeeToDepartment(int empID, int deptID);
    - Public Boolean deleteEmployeeFromDepartment(int empID, int deptID);
    - Public List<Employee> getAllEmployees(int deptID);
* In service.impl package
  + Create two classes
    - Crete EmployeeServiceImpl class which implements EmployeeService class
    - Create DepartmentServiceImpl class which implements Department class
    - Implement all the methods.
* Implementation EmployeeServiceImpl
  + Declare employee array list
    - List<Employee> employeeList;
  + Initialize employeeList in EmployeeServiceImpl() constructor
    - employeeList = new ArrayList<Employee>();
  + Implementing CRUD operations
    - addEmployee
      * Public Boolean addEmployee(Employee emp)
      * {
      * employeeList.add(emp);
      * return true;
      * }
    - getEmployee
      * public Employee getEmployee(int id)
      * {
      * //searching logic – linear search
      * }
    - getAllEmployees
      * public List<Employee> getAllEmployees()
      * {
      * return employeeList;
      * }
    - deleteEmployee
      * public boolean deleteEmployee(int id)
      * {
      * //check employee exist or not
      * //if does not exist, return false
      * // if exist, delete and return true
      * }
    - UpdateEmployee
      * Public Boolean updateEmployee(Employee emp)
      * {
      * //check employee exist or not
      * //if employee does not exist, we cannot update, return false
      * //if the employee exist, get from collections, and update it.
      * }
* DepartmentServiceImpl
  + Do similar as EmployeeServiceImpl