GUJARAT UNIVERSITY

5 Year integrated M. Sc. (Computer Science)

Semester: V

Java Programming (Practical Assignment - 4)

Date :- 14-10-2025 and Submission Date : 7-11-2025

Q1. Character Frequency Counter (using `charAt()` and `length()`)

Concepts: class, methods, charAt(), loops
Problem:
Create a class `CharFrequency` with methods to count the frequency of each character (case-insensitive) in a string without using arrays or collections.
Print each unique character and its count.

Hint: use nested loops and `charAt()` inside a class method.

Q2. String Comparator (using `equals()` and `equalsIgnoreCase()`)

Concepts: class, constructor, object comparison
Problem:
Design a class `StringCompare` with two string attributes initialized using a constructor.
Write methods:

`compareExact()` → compares with `equals()`

`compareIgnoreCase()` → compares with `equalsIgnoreCase()`

Print appropriate messages with the result.

Add-on: Count the number of differing characters if they're not equal.

Q3. Dictionary Sorter (using `compareTo()`)

Concepts: class, array of objects, sorting logic
Problem:
Define a class `DictionarySorter` that accepts multiple words through a constructor.
Implement a method `sortWords()` that sorts them alphabetically using `compareTo()`.
Display:

The sorted list

The smallest and largest word

Q4. Palindrome Substrings Finder (using `substring()`)

Concepts: class, methods, nested loops, string manipulation
Problem:
Create a class `PalindromeSubstrings` that:

Accepts a string through constructor

Finds and prints all substrings that are palindromes

Prints the total count of palindrome substrings

(Hint: use nested loops + substring())

Q5. Word Occurrence Finder (using `indexOf()` and `lastIndexOf()`)

Concepts: class, encapsulation, indexOf logic
Problem:
Create a class `SubstringFinder` with a method `findOccurrences(String main, String sub)`
→ Prints all starting and ending indices of every occurrence of `sub` in `main`.

Handle overlapping substrings as well.

Q6. Email Validator (using `matches()` and regex)

Concepts: class, constructor, regex validation
Problem:
Design a class `EmailValidator` which takes an email as input and checks if it's valid using the `matches()` method.
If invalid, print the reason (missing `@`, invalid domain, etc.).

(Hint: Use regex and if conditions for descriptive output.)

Q7. Anagram Checker (using `toCharArray()` and sorting)

Concepts: class, methods, arrays, logic
Problem:
Create a class `AnagramCheck` with two string members.
Write a method `isAnagram()` that checks if both strings are anagrams of each other (case-insensitive).
Use `toCharArray()`, sort, and compare.

Q8. Filename Filter (using `startsWith()` and `endsWith()`)

Concepts: class, arrays, filtering
Problem:
Define a class `FileFilter` which stores an array of filenames.
Implement a method `filterFiles()` that prints only those starting with `"temp"` and ending with `".txt"`.

Q9. String Cleaner (using `replaceAll()` with regex)

Concepts: class, data sanitization, regex
Problem:

Create a class `StringCleaner` with a method `cleanText(String data)` that removes:

Digits

Punctuation marks

Multiple spaces

Use regex with `replaceAll()` and print the cleaned text.

Q10. Performance Test (String vs StringBuilder – Immutability)

Concepts: class, performance comparison, immutability
Problem:
Make a class `PerformanceTest` that:

Concatenates 10,000 words using `String`

Repeats using `StringBuilder`

Prints time taken by each
Demonstrate how immutability affects speed.

Q11. Password Strength Analyzer (using multiple String methods)

Concepts: class design, encapsulation, condition checks
Problem:
Create a class `PasswordAnalyzer` with:

A constructor to take password input

Method `analyze()` that checks:

Minimum 8 characters (`length()`)

Contains upper & lower case (`matches()`)

Contains digits and special chars (`matches("[0-9]")`, `matches("[^a-zA-Z0-9]")`)
Print strength level: Weak / Moderate / Strong

Q12. Unique Word Extractor (using `split()` and `equalsIgnoreCase()`)

Concepts: class, string splitting, duplication removal
Problem:
Build a class `UniqueWordExtractor` that:

Splits a sentence into words using `split()`

Removes duplicates (case-insensitive)

Prints the unique words in order of appearance

Q13. Sentence Pattern Reverser

Concepts: class, multiple String functions, logic
Problem:
Write a class `SentencePattern` that:

Takes a full sentence

Reverses each word individually (not the sentence order)

Removes extra spaces

Capitalizes the first letter of each word

(Uses: split(), trim(), substring(), toUpperCase(), toLowerCase())