**Velammal College of Engineering and Technology, Madurai**

**Autonomous**

**Department of Computer Science and Engineering**

**21CS205 Object oriented programming lab**

**Exercise No 7**

String Handling

Team 1

1. Write an application that prompts the user for a password that contains at least two uppercase letters, at least two lowercase letters, and at least two digits. Continuously reprompt the user until a valid password is entered. After each entry, display a message indicating whether the user was successful or the reason the user was not successful. Save the file as ValidatePassword.java.
2. Write a Java program that performs various text analysis operations on the input text. The program should be able to:

* Calculate the total number of characters in the text (excluding whitespace).
* Calculate the total number of words in the text. Words are separated by spaces.
* Determine the number of sentences in the text. Sentences are terminated by periods ('.'), question marks ('?'), or exclamation marks ('!').
* Identify the longest word in the text.
* Determine the average word length in the text (in characters).
* Find and print the most frequent word in the text.

Example Input:

This is a sample text. It contains several words. What's the average word length?

Example Output:

Total characters (excluding whitespace): 52

Total words: 12

Total sentences: 3

Longest word: "contains"

Average word length: 4.33

Most frequent word: "is" (appears 2 times)

Team 2

1. Write a Java program to find the second most frequent character in a given string.

Sample Output

The given string is: successes

The second most frequent char in the string is: c

1. The program should check whether an email address follows the basic rules of a valid email format.

The rules for a valid email address are as follows:

1. The email address must contain a single "@" symbol.
2. The "@" symbol must not be the first or last character in the email address.
3. The email address must have at least one character before and after the "@" symbol.
4. The email address must contain a dot (".") after the "@" symbol, indicating the domain.
5. The domain must have at least one character before and after the dot.
6. The domain can only contain letters (A-Z, a-z), digits (0-9), hyphens ("-"), and underscores ("\_").

Your program should do the following:

1. Prompt the user to enter an email address.
2. Check if the entered email address conforms to the rules mentioned above.
3. Provide feedback to the user regarding the validity of the email address. The feedback should be one of the following:
   * "Valid" if the email address follows all the rules.
   * "Invalid" if the email address violates any of the rules.

Example Output:

Please enter an email address: john.doe@example.com

Email address is valid.

Please enter an email address: user@example.

Email address is invalid.

Team 3

1. Three-letter acronyms are common in the business world. For example, in Java you use the IDE (Integrated Development Environment) in the JDK (Java Development Kit) to write programs used by the JVM (Java Virtual Machine) that you might send over a LAN (local area network). Programmers even use the acronym TLA to stand for three-letter acronym. Write a program that allows a user to enter three words, and display the appropriate three-letter acronym in all uppercase letters. If the user enters more than three words, ignore the extra words. Save the file as ThreeLetterAcronym.java.
2. Write a Java program that counts the frequency of each word in a given text document. Your program should analyze the text, ignore punctuation marks and case, and provide a count for each unique word.

The program should do the following:

1. Prompt the user to enter a text document.
2. Analyze the input text and count the frequency of each unique word (ignoring punctuation marks and case).
3. Display a list of unique words and their corresponding frequencies in alphabetical order.

Enter a text document: This is a sample text. Is this the same text? No, it's not.

Word Frequency:

a: 1

is: 2

it's: 1

no: 1

not: 1

same: 1

text: 2

the: 1

this: 2

Team 4

1. Write an application that accepts a word from a user and converts it to Pig Latin. If a word starts with a consonant, the Pig Latin version removes all consonants from the beginning of the word and places them at the end, followed by ay. For example, cricket becomes icketcray. If a word starts with a vowel, the Pig Latin version is the original word with ay added to the end. For example, apple becomes appleay. If y is the first letter in a word, it is treated as a consonant; otherwise, it is treated as a vowel. For example, young becomes oungyay, but system becomes ystemsay. For this program, assume that the user will enter only a single word consisting of all lowercase letters. Save the file as PigLatin.java.
2. The program should analyze the passwords and provide feedback on their strength based on certain criteria.

The password strength criteria are as follows:

1. The password must be at least 8 characters long.
2. The password must contain at least one uppercase letter (A-Z).
3. The password must contain at least one lowercase letter (a-z).
4. The password must contain at least one digit (0-9).
5. The password may contain special characters (e.g., @, #, $, %), but it is not required.

The program should do the following:

1. Prompt the user to enter a password.
2. Check if the password meets each of the above criteria.
3. Provide feedback to the user regarding the strength of their password. The feedback should be one of the following:
   * "Weak" if the password is less than 8 characters long or does not contain at least one uppercase letter, one lowercase letter, and one digit.
   * "Moderate" if the password meets the length requirement and contains at least one uppercase letter, one lowercase letter, and one digit, but does not contain special characters.
   * "Strong" if the password meets all the criteria, including containing special characters.

The program should repeat the process until the user enters a strong password. Once a strong password is entered, the program should display a success message.

Example Output:

Please enter a password: myPass123

Password strength: Weak

Please enter a password: StrongP@ss

Password strength: Strong

Password is strong. You may proceed.

Team 5

1. Write a program that inserts parentheses, a space, and a dash into a string of 10 user-entered numbers to format it as a phone number. For example, 5153458912 becomes (515) 345-8912. If the user does not enter exactly 10 digits, display an error message. Continue to accept user input until the user enters 999. Save the file as PhoneNumberFormat.java.
2. A palindromic substring is a substring that reads the same forwards as it does backward.

Your program should do the following:

1. Prompt the user to enter a string.
2. Analyze the input string and find all palindromic substrings within it.
3. Count the total number of palindromic substrings.
4. Display the palindromic substrings found and their count to the user.

Example Input:

Enter a string: racecarabcdeedcba

Example Output:

Palindromic substrings found:

1. racecar

2. cdeedc

3. abcdeedcba

Total palindromic substrings: 3

Team 6

1. Write an application that prompts a user for a full name and street address and constructs an ID from the user’s initials and numeric part of the address. For example, the user William Henry Harrison who lives at 34 Elm would have an ID of WHH34, whereas user Addison Mitchell who lives at 1778 Monroe would have an ID of AM1778. Save the file as ConstructID.java
2. Write a Java program that allows a user to encrypt and decrypt text messages using a simple encryption algorithm. The program should be able to take a message, encrypt it, and then decrypt it back to the original message.

Here's the algorithm:

1. For encryption, each character in the original message is shifted forward in the alphabet by a fixed number of positions (e.g., 3 positions). For example, 'A' becomes 'D', 'B' becomes 'E', and 'Z' becomes 'C'. Punctuation and spaces should remain unchanged.
2. For decryption, the encrypted message is shifted backward by the same fixed number of positions to reveal the original message.

Your program should do the following:

1. Prompt the user to enter a message to encrypt.
2. Encrypt the message using the specified algorithm.
3. Display the encrypted message.
4. Decrypt the encrypted message to reveal the original message.
5. Display the original message.

Example Output:

Enter a message to encrypt: Hello, World!

Encrypted message: Khoor, Zruog!

Decrypted message: Hello, World!

Team 7

1. Write a program called **CaesarCode** to cipher the Caesar's code. The program shall prompt user for a plaintext string consisting of mix-case letters only; compute the ciphertext; and print the ciphertext in uppercase. For example,

Enter a plaintext string: **Testing**

The ciphertext string is: WHVWLQJ

Caesar's Code is one of the simplest encryption techniques. Each letter in the plaintext is replaced by a letter some fixed number of position (n) down the alphabet cyclically. For ex. consider n=3. That is, 'A' is replaced by 'D', 'B' by 'E', 'C' by 'F', ..., 'X' by 'A', ..., 'Z' by 'C'.

1. Write a java program that reads a string from inputs containing first name, last name and computes an e-mail address with first 3 letters of the first name, first 4 letters of last name, ‘.’ separator and domain. Display the outputs by invoking objects.

Team 8

1. Write a program called **Bin2Dec** to convert an input binary string into its equivalent decimal number.

Sample Output

Enter a Binary string: **1011**

The equivalent decimal number for binary "1011" is: 11

Enter a Binary string: **1234**

error: invalid binary string "1234"

1. Write a Java program to divide a string in n equal parts.

Sample output

The given string is: abcdefghijklmnopqrstuvwxy

The string divided into 5 parts and they are: abcde fghij klmno pqrst uvwxy