Name – Aryan Singh

Reg no. – RA2411030010159

Section Y1

Week 2 lab problems

Q1 Write a program to find and replace all occurrences of a substring in a text without using the replace() method

PROGRAM

package javabootcamp.WEEK2;

public class OccurencesOfSubstringWithoutReplace {

    public static String findAndReplace(String text, String oldSub, String newSub) {

        StringBuilder result = new StringBuilder();

        int i = 0;

        while (i < text.length()) {

            if (i + oldSub.length() <= text.length() &&

                text.substring(i, i + oldSub.length()).equals(oldSub)) {

                result.append(newSub);

                i += oldSub.length();

            } else {

                result.append(text.charAt(i));

                i++;

            }

        }

        return result.toString();

    }

    public static void main(String[] args) {

        String text = "the cat sat on the mat with another cat";

        String oldSub = "cat";

        String newSub = "dog";

        System.out.println("Original Text: " + text);

        System.out.println("Modified Text: " + findAndReplace(text, oldSub, newSub));

    }

}

OUTPUT



Q2 Write a program to convert text between different cases (uppercase, lowercase, title case) using ASCII values without using built-in case conversion methods

PROGRAM

package javabootcamp.WEEK2;

import java.util.Scanner;

public class CaseConverter {

    public static String toUpperCase(String str) {

        String result = "";

        for (int i = 0; i < str.length(); i++) {

            char ch = str.charAt(i);

            if (ch >= 'a' && ch <= 'z') {

                ch = (char)(ch - 32);

            }

            result += ch;

        }

        return result;

    }

    public static String toLowerCase(String str) {

        String result = "";

        for (int i = 0; i < str.length(); i++) {

            char ch = str.charAt(i);

            if (ch >= 'A' && ch <= 'Z') {

                ch = (char)(ch + 32);

            }

            result += ch;

        }

        return result;

    }

    public static String toTitleCase(String str) {

        String result = "";

        boolean newWord = true;

        for (int i = 0; i < str.length(); i++) {

            char ch = str.charAt(i);

            if (ch == ' ') {

                result += ch;

                newWord = true;

            } else {

                if (newWord) {

                    if (ch >= 'a' && ch <= 'z') {

                        ch = (char)(ch - 32);

                    }

                    newWord = false;

                } else {

                    if (ch >= 'A' && ch <= 'Z') {

                        ch = (char)(ch + 32);

                    }

                }

                result += ch;

            }

        }

        return result;

    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter text: ");

        String text = sc.nextLine();

        System.out.println("Uppercase: " + toUpperCase(text));

        System.out.println("Lowercase: " + toLowerCase(text));

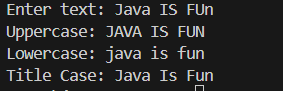
        System.out.println("Title Case: " + toTitleCase(text));

        sc.close();

    }

}

OUTPUT



Q3 Write a program to analyze and compare the performance of String concatenation vs StringBuilder vs StringBuffer for building large strings

PROGRAM

package javabootcamp.WEEK2;

public class StringPerformanceTest {

    public static void main(String[] args) {

        int iterations = 100000;

        long start, end;

        start = System.currentTimeMillis();

        String s = "";

        for (int i = 0; i < iterations; i++) {

            s += "a";

        }

        end = System.currentTimeMillis();

        System.out.println("String time: " + (end - start) + " ms");

        start = System.currentTimeMillis();

        StringBuilder sb = new StringBuilder();

        for (int i = 0; i < iterations; i++) {

            sb.append("a");

        }

        end = System.currentTimeMillis();

        System.out.println("StringBuilder time: " + (end - start) + " ms");

        start = System.currentTimeMillis();

        StringBuffer sbf = new StringBuffer();

        for (int i = 0; i < iterations; i++) {

            sbf.append("a");

        }

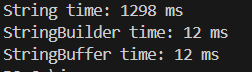
        end = System.currentTimeMillis();

        System.out.println("StringBuffer time: " + (end - start) + " ms");

    }

}

OUTPUT



Q4 Write a program to create a simple encryption and decryption system using ASCII character shifting (Caesar Cipher implementation)

PROGRAM

package javabootcamp.WEEK2;

import java.util.Scanner;

public class CaesarCipher {

    public static String encrypt(String text, int shift) {

        String result = "";

        for (int i = 0; i < text.length(); i++) {

            char ch = text.charAt(i);

            ch = (char) (ch + shift);

            result += ch;

        }

        return result;

    }

    public static String decrypt(String text, int shift) {

        String result = "";

        for (int i = 0; i < text.length(); i++) {

            char ch = text.charAt(i);

            ch = (char) (ch - shift);

            result += ch;

        }

        return result;

    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter text: ");

        String text = sc.nextLine();

        System.out.print("Enter shift value: ");

        int shift = sc.nextInt();

        String encrypted = encrypt(text, shift);

        System.out.println("Encrypted text: " + encrypted);

        String decrypted = decrypt(encrypted, shift);

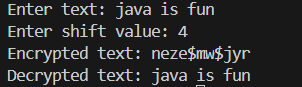
        System.out.println("Decrypted text: " + decrypted);

        sc.close();

    }

}

OUTPUT



Q5 Write a program to extract and analyze different parts of an email address using substring() and indexOf() methods

OUTPUT

package javabootcamp.WEEK2;

import java.util.Scanner;

public class EmailAnalyzer {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter email address: ");

        String email = sc.nextLine();

        int atIndex = email.indexOf('@');

        int dotIndex = email.lastIndexOf('.');

        if (atIndex == -1 || dotIndex == -1 || atIndex > dotIndex) {

            System.out.println("Invalid email format");

        } else {

            String username = email.substring(0, atIndex);

            String domain = email.substring(atIndex + 1, dotIndex);

            String extension = email.substring(dotIndex + 1);

            System.out.println("Full Email: " + email);

            System.out.println("Username: " + username);

            System.out.println("Domain: " + domain);

            System.out.println("Extension: " + extension);

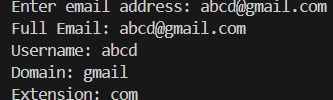
        }

        sc.close();

    }

}

OUTPUT



Q6 Write a program to create a text formatter that justifies text to a specified width using StringBuilder for efficient string manipulation

PROGRAM

package javabootcamp.WEEK2;

import java.util.\*;

public class TextJustifier {

    public static List<String> justifyText(String[] words, int width) {

        List<String> result = new ArrayList<>();

        List<String> line = new ArrayList<>();

        int lineLength = 0;

        for (String word : words) {

            if (lineLength + word.length() + line.size() > width) {

                result.add(justifyLine(line, lineLength, width));

                line.clear();

                lineLength = 0;

            }

            line.add(word);

            lineLength += word.length();

        }

        StringBuilder lastLine = new StringBuilder();

        for (int i = 0; i < line.size(); i++) {

            if (i > 0) lastLine.append(" ");

            lastLine.append(line.get(i));

        }

        while (lastLine.length() < width) lastLine.append(" ");

        result.add(lastLine.toString());

        return result;

    }

    private static String justifyLine(List<String> line, int lineLength, int width) {

        if (line.size() == 1) {

            StringBuilder sb = new StringBuilder(line.get(0));

            while (sb.length() < width) sb.append(" ");

            return sb.toString();

        }

        int totalSpaces = width - lineLength;

        int spaceSlots = line.size() - 1;

        int evenSpace = totalSpaces / spaceSlots;

        int extraSpace = totalSpaces % spaceSlots;

        StringBuilder sb = new StringBuilder();

        for (int i = 0; i < line.size(); i++) {

            sb.append(line.get(i));

            if (i < spaceSlots) {

                for (int s = 0; s < evenSpace; s++) sb.append(" ");

                if (extraSpace > 0) {

                    sb.append(" ");

                    extraSpace--;

                }

            }

        }

        return sb.toString();

    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter text:");

        String text = sc.nextLine();

        System.out.print("Enter line width: ");

        int width = sc.nextInt();

        String[] words = text.split("\\s+");

        List<String> justified = justifyText(words, width);

        System.out.println("\nJustified Text:");

        for (String line : justified) {

            System.out.println(line);

        }

        sc.close();

    }

}

OUTPUT

