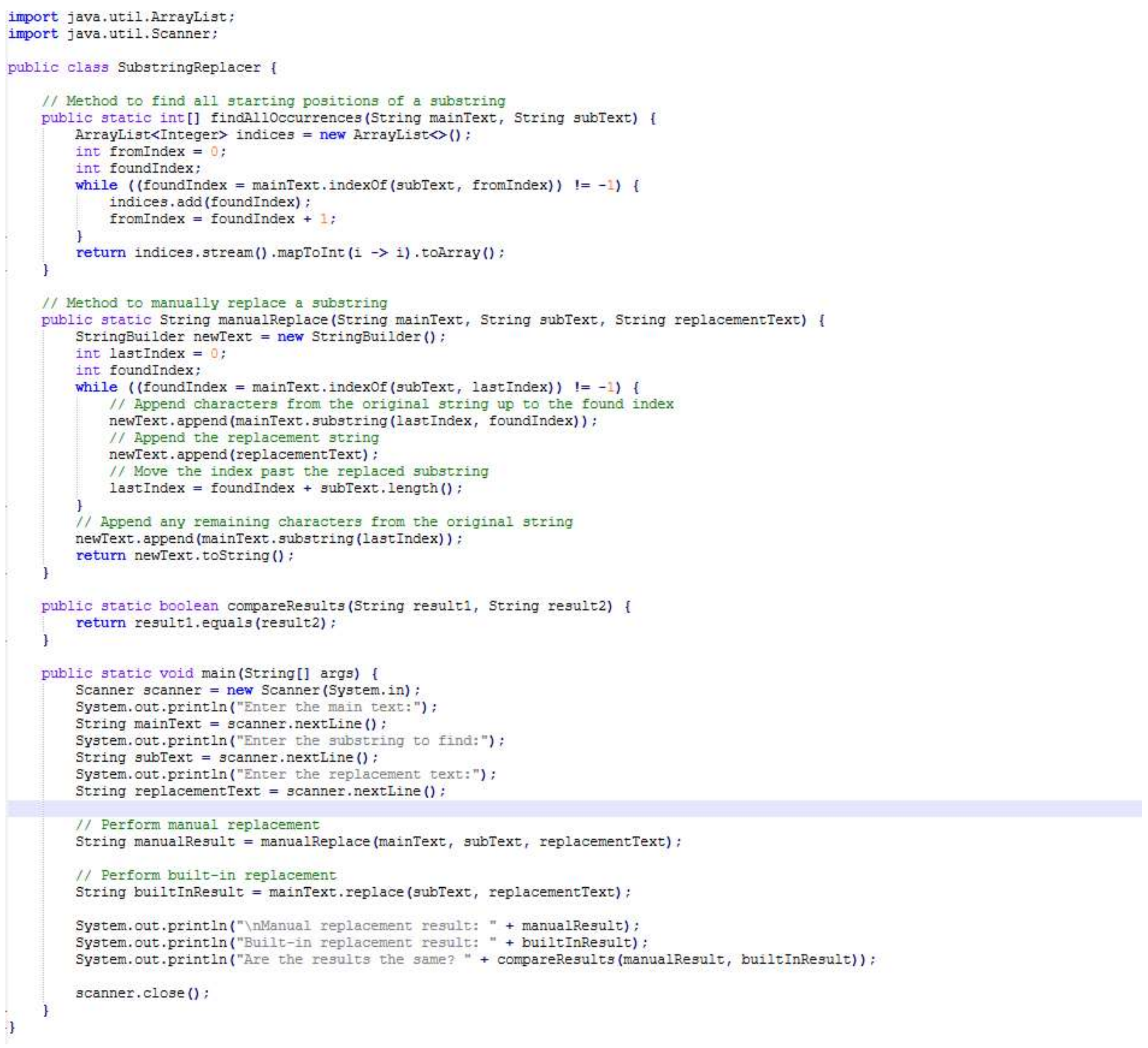
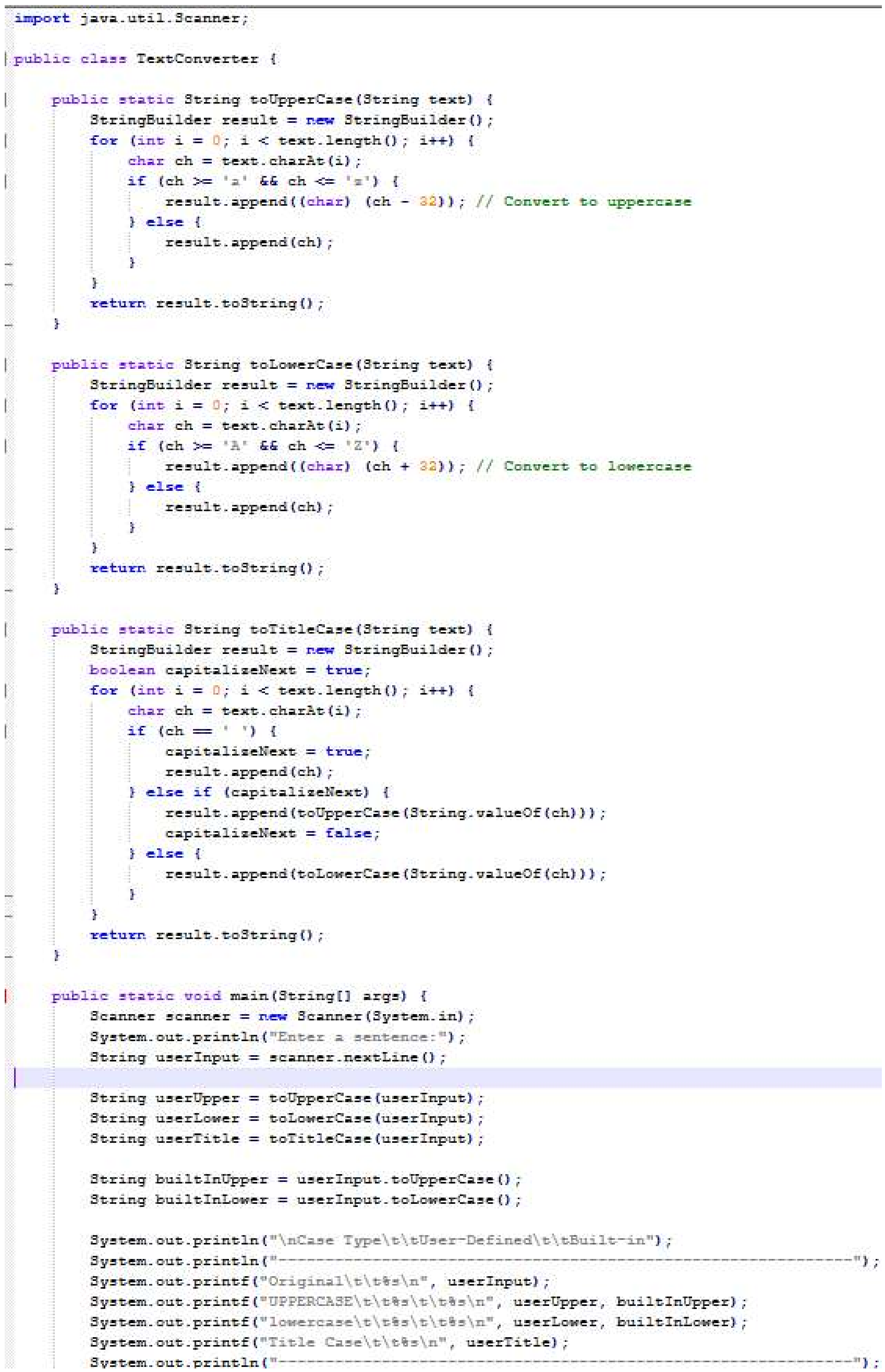
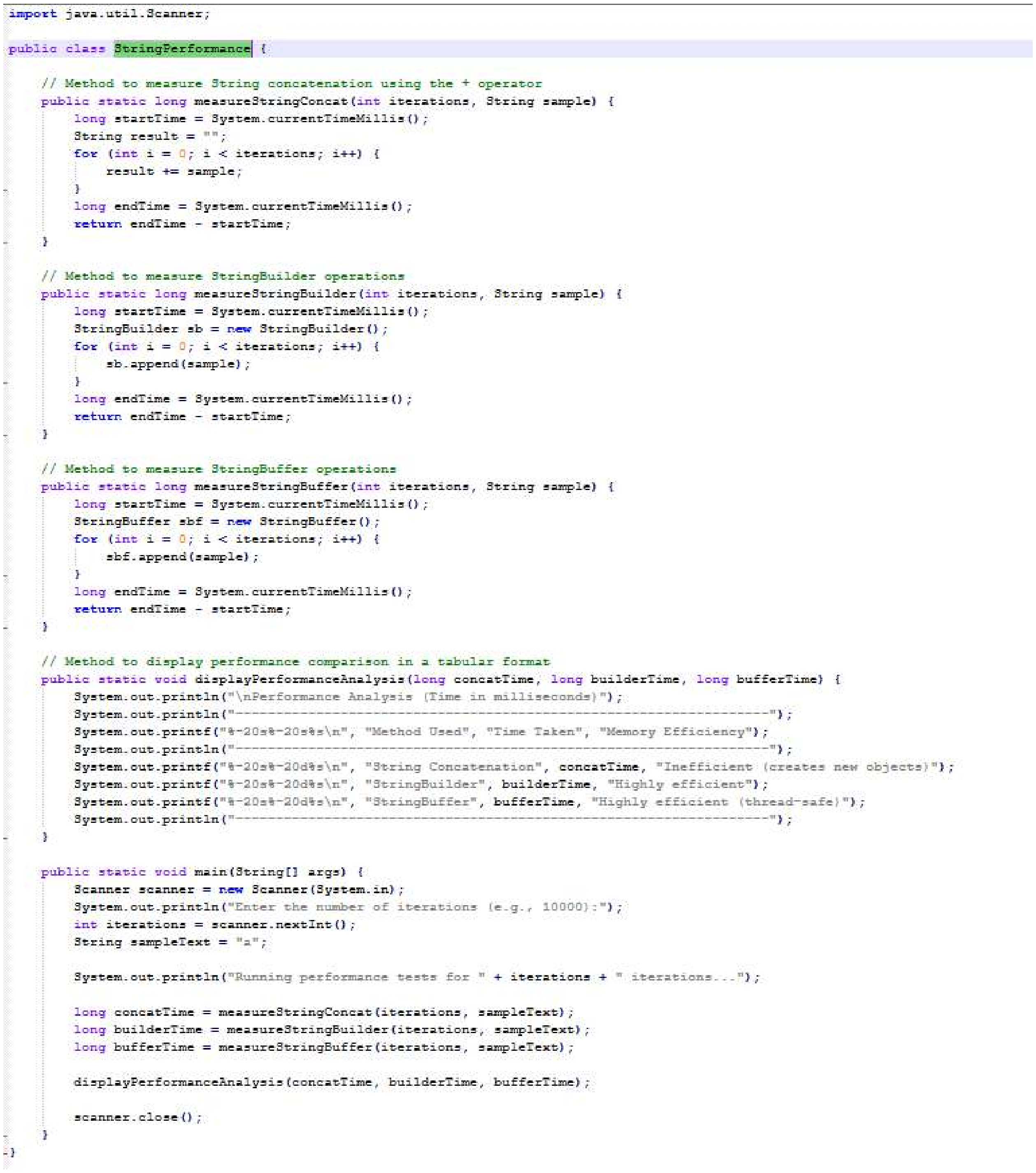


```java
import java.util.ArrayList;
import java.util.Scanner;

public class SubstringReplacer {

    // Method to find all starting positions of a substring
    public static int[] findAllOccurrences(String mainText, String subText) {
        ArrayList<Integer> indices = new ArrayList<>();
        int fromIndex = 0;
        int foundIndex;
        while ((foundIndex = mainText.indexOf(subText, fromIndex)) != -1) {
            indices.add(foundIndex);
            fromIndex = foundIndex + 1;
        }
        return indices.stream().mapToInt(i -> i).toArray();
    }

    // Method to manually replace a substring
    public static String manualReplace(String mainText, String subText, String replacementText) {
        StringBuilder newText = new StringBuilder();
        int lastIndex = 0;
        int foundIndex;
        while ((foundIndex = mainText.indexOf(subText, lastIndex)) != -1) {
            // Append characters from the original string up to the found index
            newText.append(mainText.substring(lastIndex, foundIndex));
            // Append the replacement string
            newText.append(replacementText);
            // Move the index past the replaced substring
            lastIndex = foundIndex + subText.length();
        }
        // Append any remaining characters from the original string
        newText.append(mainText.substring(lastIndex));
        return newText.toString();
    }

    public static boolean compareResults(String result1, String result2) {
        return result1.equals(result2);
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the main text:");
        String mainText = scanner.nextLine();
        System.out.println("Enter the substring to find:");
        String subText = scanner.nextLine();
        System.out.println("Enter the replacement text:");
        String replacementText = scanner.nextLine();

        // Perform manual replacement
        String manualResult = manualReplace(mainText, subText, replacementText);

        // Perform built-in replacement
        String builtInResult = mainText.replace(subText, replacementText);

        System.out.println("\nManual replacement result: " + manualResult);
        System.out.println("Built-in replacement result: " + builtInResult);
        System.out.println("Are the results the same? " + compareResults(manualResult, builtInResult));

        scanner.close();
    }
}
```

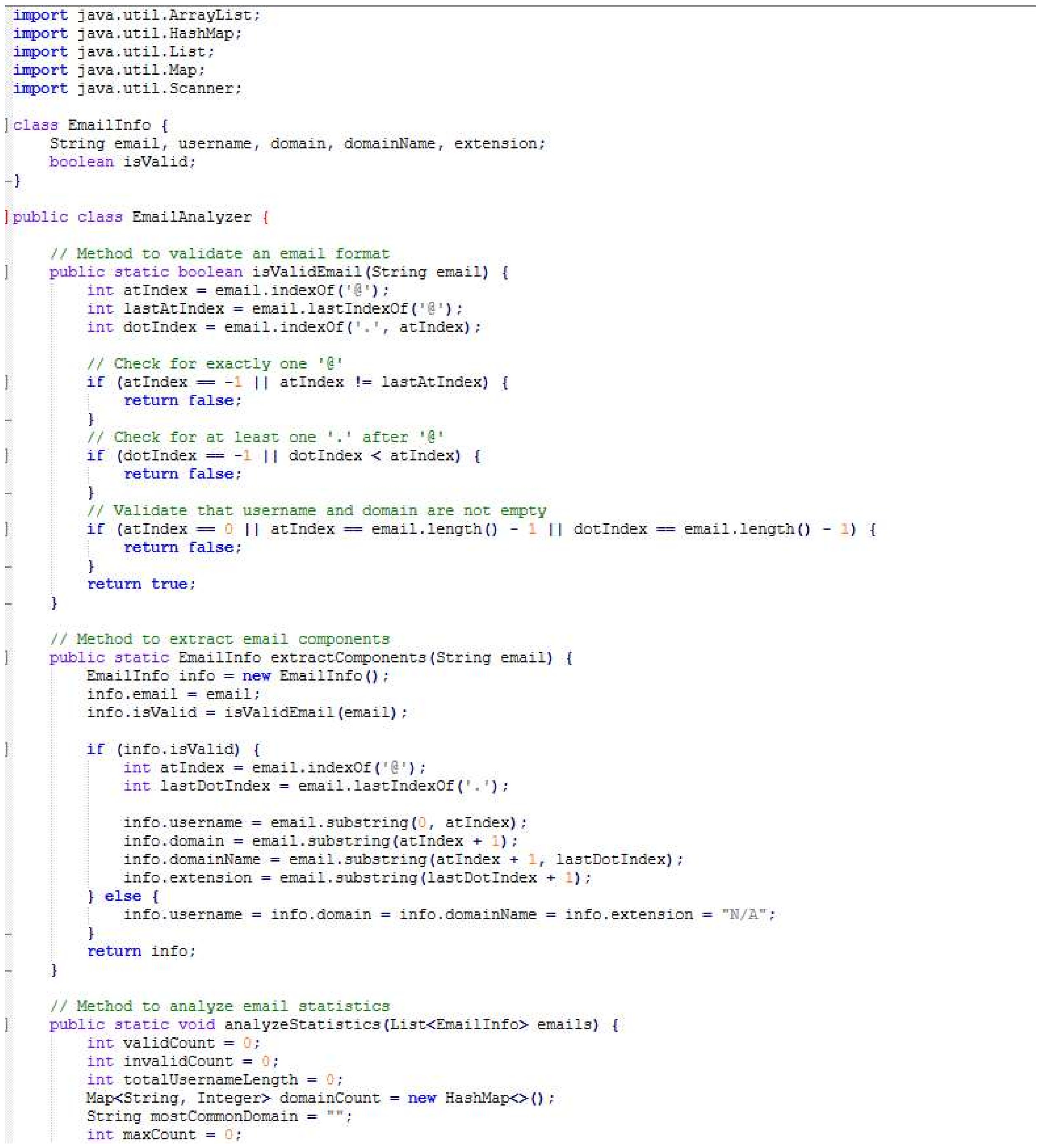


```java
import java.util.Scanner;

public class TextConverter {

    public static String toUpperCase(String text) {
        StringBuilder result = new StringBuilder();
        for (int i = 0; i < text.length(); i++) {
            char ch = text.charAt(i);
            if (ch >= 'a' && ch <= 'z') {
                result.append((char) (ch - 32)); // Convert to uppercase
            } else {
                result.append(ch);
            }
        }
        return result.toString();
    }

    public static String toLowerCase(String text) {
        StringBuilder result = new StringBuilder();
        for (int i = 0; i < text.length(); i++) {
            char ch = text.charAt(i);
            if (ch >= 'A' && ch <= 'Z') {
                result.append((char) (ch + 32)); // Convert to lowercase
            } else {
                result.append(ch);
            }
        }
        return result.toString();
    }

    public static String toTitleCase(String text) {
        StringBuilder result = new StringBuilder();
        boolean capitalizeNext = true;
        for (int i = 0; i < text.length(); i++) {
            char ch = text.charAt(i);
            if (ch == ' ') {
                capitalizeNext = true;
                result.append(ch);
            } else if (capitalizeNext) {
                result.append(toUpperCase(String.valueOf(ch)));
                capitalizeNext = false;
            } else {
                result.append(toLowerCase(String.valueOf(ch)));
            }
        }
        return result.toString();
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter a sentence:");
        String userInput = scanner.nextLine();

        String userUpper = toUpperCase(userInput);
        String userLower = toLowerCase(userInput);
        String userTitle = toTitleCase(userInput);

        String builtInUpper = userInput.toUpperCase();
        String builtInLower = userInput.toLowerCase();

        System.out.println("\nCase Type\t\tUser-Defined\t\tBuilt-in");
        System.out.println("--------------------------------------------------------------");
        System.out.printf("Original\t\t%s\n", userInput);
        System.out.printf("UPPERCASE\t\t%s\t\t%s\n", userUpper, builtInUpper);
        System.out.printf("lowercase\t\t%s\t\t%s\n", userLower, builtInLower);
        System.out.printf("Title Case\t\t%s\n", userTitle);
        System.out.println("--------------------------------------------------------------");
```

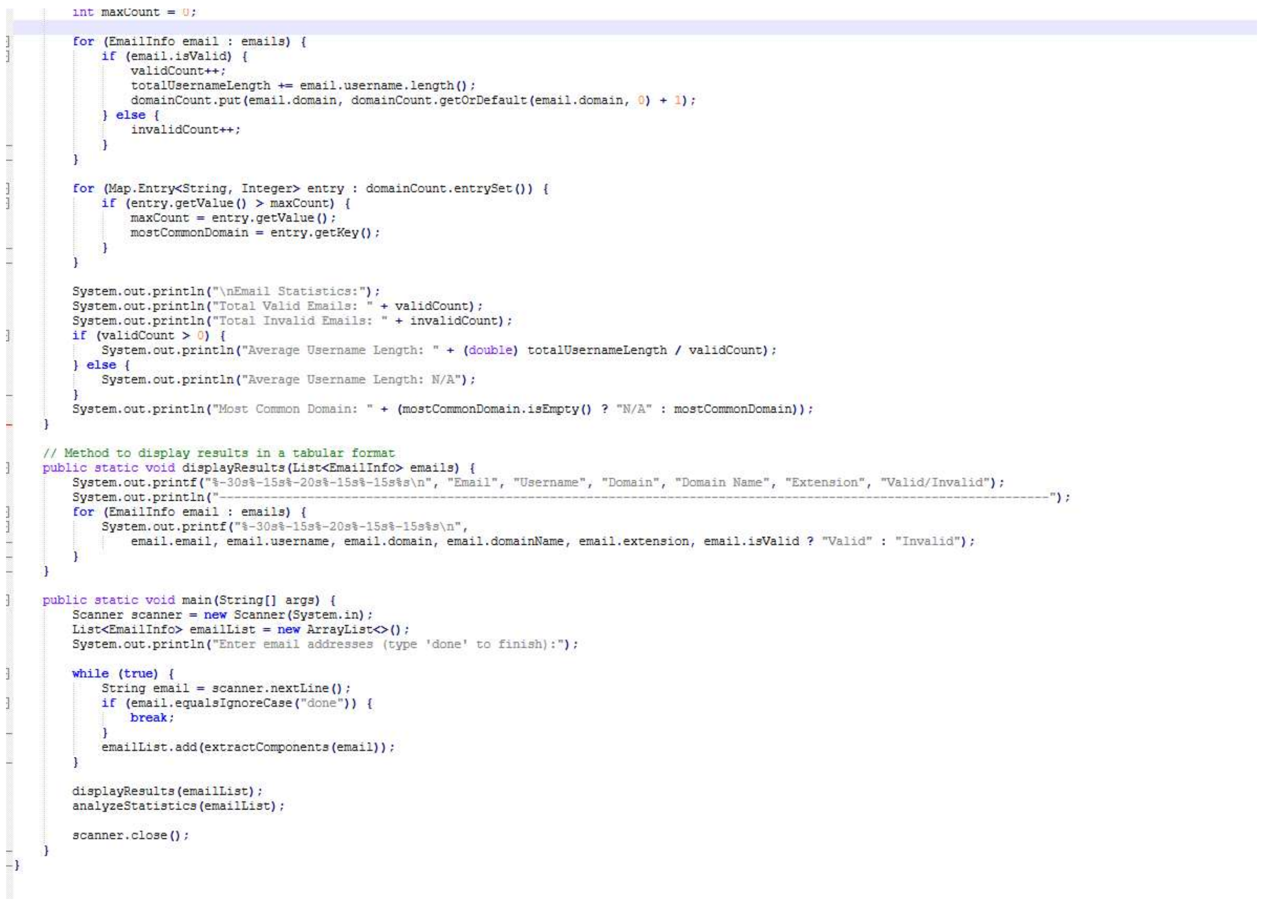


```java
import java.util.Scanner;

public class StringPerformance {

    // Method to measure String concatenation using the + operator
    public static long measureStringConcat(int iterations, String sample) {
        long startTime = System.currentTimeMillis();
        String result = "";
        for (int i = 0; i < iterations; i++) {
            result += sample;
        }
        long endTime = System.currentTimeMillis();
        return endTime - startTime;
    }

    // Method to measure StringBuilder operations
    public static long measureStringBuilder(int iterations, String sample) {
        long startTime = System.currentTimeMillis();
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < iterations; i++) {
            sb.append(sample);
        }
        long endTime = System.currentTimeMillis();
        return endTime - startTime;
    }

    // Method to measure StringBuffer operations
    public static long measureStringBuffer(int iterations, String sample) {
        long startTime = System.currentTimeMillis();
        StringBuffer sbf = new StringBuffer();
        for (int i = 0; i < iterations; i++) {
            sbf.append(sample);
        }
        long endTime = System.currentTimeMillis();
        return endTime - startTime;
    }

    // Method to display performance comparison in a tabular format
    public static void displayPerformanceAnalysis(long concatTime, long builderTime, long bufferTime) {
        System.out.println("\nPerformance Analysis (Time in milliseconds)");
        System.out.println("-------------------------------------------------------------------");
        System.out.printf("%-20s%-20s%s\n", "Method Used", "Time Taken", "Memory Efficiency");
        System.out.println("-------------------------------------------------------------------");
        System.out.printf("%-20s%-20d%s\n", "String Concatenation", concatTime, "Inefficient (creates new objects)");
        System.out.printf("%-20s%-20d%s\n", "StringBuilder", builderTime, "Highly efficient");
        System.out.printf("%-20s%-20d%s\n", "StringBuffer", bufferTime, "Highly efficient (thread-safe)");
        System.out.println("-------------------------------------------------------------------");
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the number of iterations (e.g., 10000):");
        int iterations = scanner.nextInt();
        String sampleText = "a";

        System.out.println("Running performance tests for " + iterations + " iterations...");

        long concatTime = measureStringConcat(iterations, sampleText);
        long builderTime = measureStringBuilder(iterations, sampleText);
        long bufferTime = measureStringBuffer(iterations, sampleText);

        displayPerformanceAnalysis(concatTime, builderTime, bufferTime);

        scanner.close();
    }
}
```

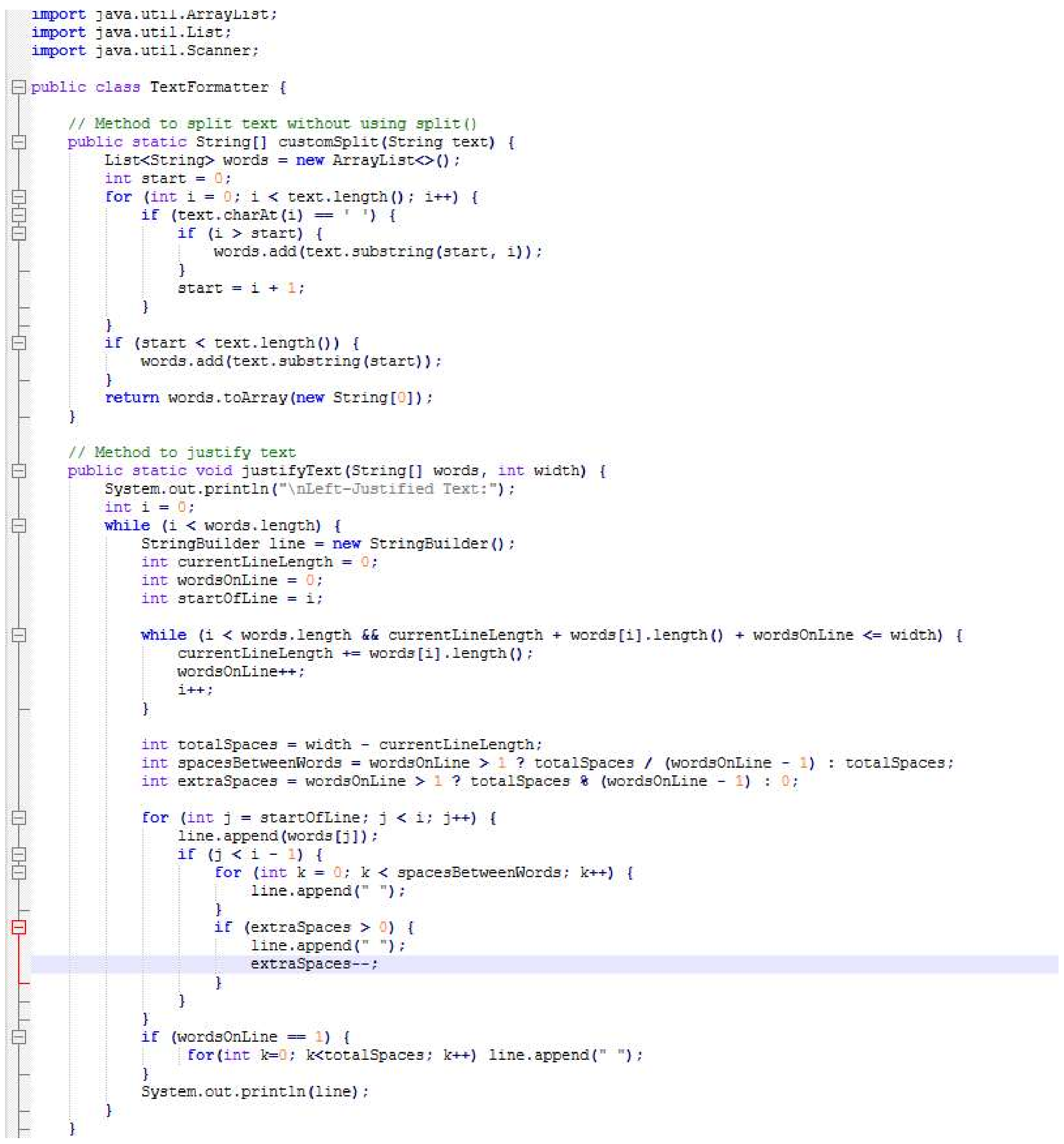
```java
import java.util.Scanner;

public class CaesarCipher {

    // Method to encrypt text
    public static String encrypt(String text, int shift) {
        StringBuilder result = new StringBuilder();
        shift = shift % 26; // Handle shifts larger than 26

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

            if (ch >= 'a' && ch <= 'z') {
                ch = (char) ('a' + (ch - 'a' + shift + 26) % 26);
            } else if (ch >= 'A' && ch <= 'Z') {
                ch = (char) ('A' + (ch - 'A' + shift + 26) % 26);
            }
            result.append(ch);
        }
        return result.toString();
    }

    // Method to decrypt text
    public static String decrypt(String text, int shift) {
        return encrypt(text, -shift); // Decryption is just encryption with a negative shift
    }

    // Method to validate that decryption returns the original text
    public static boolean validate(String original, String decrypted) {
        return original.equals(decrypted);
    }

    // Method to display a character's ASCII value
    public static void displayAscii(char ch) {
        System.out.println("Character: '" + ch + "', ASCII: " + (int) ch);
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter text to encrypt:");
        String originalText = scanner.nextLine();
        System.out.println("Enter the shift value:");
        int shift = scanner.nextInt();

        // Display ASCII values of a sample character before and after encryption
        char sampleChar = originalText.charAt(0);
        char encryptedSampleChar = encrypt(String.valueOf(sampleChar), shift).charAt(0);
        System.out.println("\nASCII values before and after encryption for '" + sampleChar + "':");
        displayAscii(sampleChar);
        displayAscii(encryptedSampleChar);

        // Encrypt and decrypt the text
        String encryptedText = encrypt(originalText, shift);
        String decryptedText = decrypt(encryptedText, shift);

        System.out.println("\nOriginal Text: " + originalText);
        System.out.println("Encrypted Text: " + encryptedText);
        System.out.println("Decrypted Text: " + decryptedText);

        // Validate that decryption returns the original text
        boolean isValid = validate(originalText, decryptedText);
        System.out.println("\nDecryption validated: " + isValid);

        scanner.close();
    }
}
```



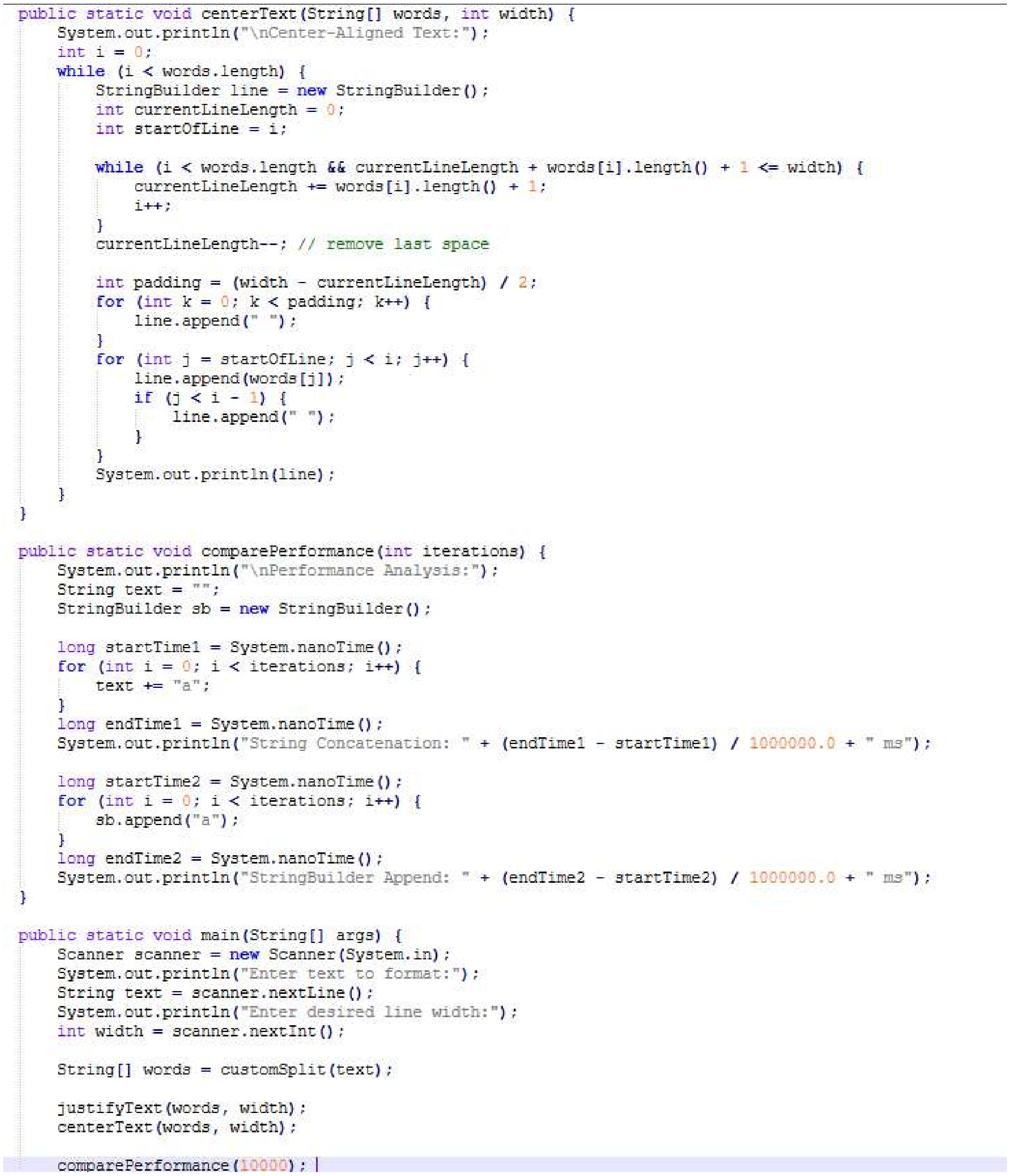
```java
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;

class EmailInfo {
    String email, username, domain, domainName, extension;
    boolean isValid;
}

public class EmailAnalyzer {

    // Method to validate an email format
    public static boolean isValidEmail(String email) {
        int atIndex = email.indexOf('@');
        int lastAtIndex = email.lastIndexOf('@');
        int dotIndex = email.indexOf('.', atIndex);

        // Check for exactly one '@'
        if (atIndex == -1 || atIndex != lastAtIndex) {
            return false;
        }
        // Check for at least one '.' after '@'
        if (dotIndex == -1 || dotIndex < atIndex) {
            return false;
        }
        // Validate that username and domain are not empty
        if (atIndex == 0 || atIndex == email.length() - 1 || dotIndex == email.length() - 1) {
            return false;
        }
        return true;
    }

    // Method to extract email components
    public static EmailInfo extractComponents(String email) {
        EmailInfo info = new EmailInfo();
        info.email = email;
        info.isValid = isValidEmail(email);

        if (info.isValid) {
            int atIndex = email.indexOf('@');
            int lastDotIndex = email.lastIndexOf('.');

            info.username = email.substring(0, atIndex);
            info.domain = email.substring(atIndex + 1);
            info.domainName = email.substring(atIndex + 1, lastDotIndex);
            info.extension = email.substring(lastDotIndex + 1);
        } else {
            info.username = info.domain = info.domainName = info.extension = "N/A";
        }
        return info;
    }

    // Method to analyze email statistics
    public static void analyzeStatistics(List<EmailInfo> emails) {
        int validCount = 0;
        int invalidCount = 0;
        int totalUsernameLength = 0;
        Map<String, Integer> domainCount = new HashMap<>();
        String mostCommonDomain = "";
        int maxCount = 0;
```



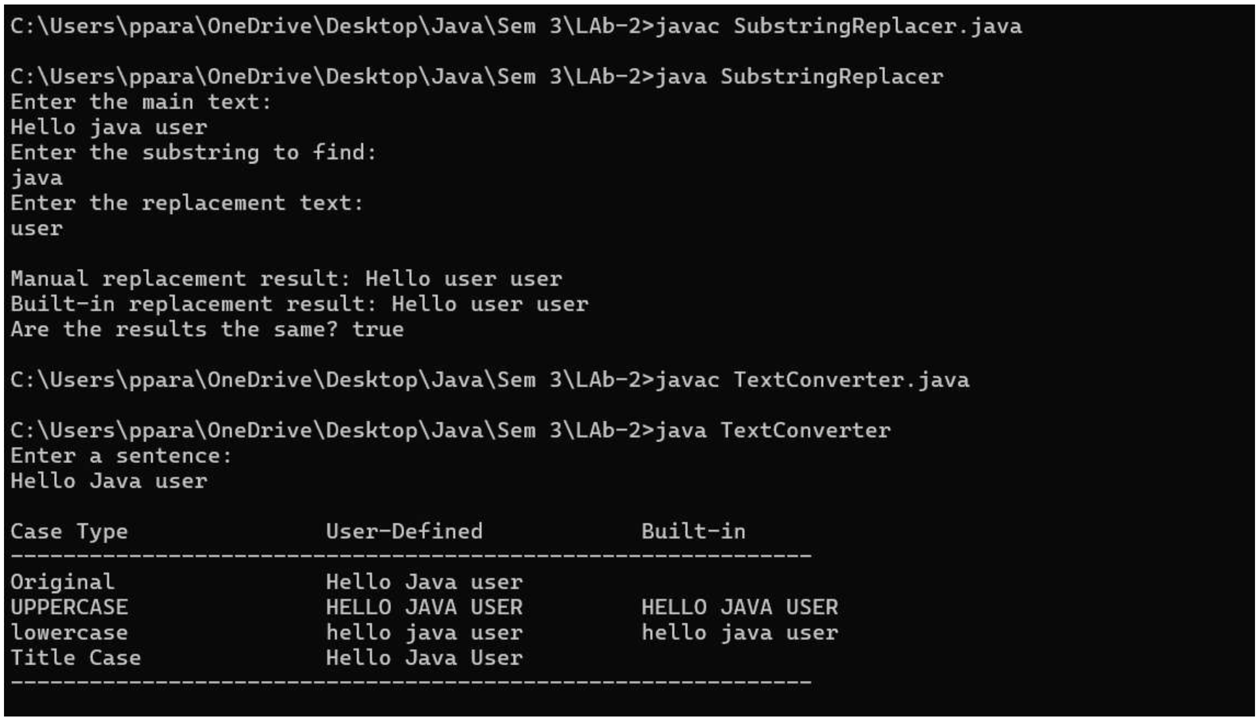
```java
        int maxCount = 0;

        for (EmailInfo email : emails) {
            if (email.isValid) {
                validCount++;
                totalUsernameLength += email.username.length();
                domainCount.put(email.domain, domainCount.getOrDefault(email.domain, 0) + 1);
            } else {
                invalidCount++;
            }
        }

        for (Map.Entry<String, Integer> entry : domainCount.entrySet()) {
            if (entry.getValue() > maxCount) {
                maxCount = entry.getValue();
                mostCommonDomain = entry.getKey();
            }
        }

        System.out.println("\nEmail Statistics:");
        System.out.println("Total Valid Emails: " + validCount);
        System.out.println("Total Invalid Emails: " + invalidCount);
        if (validCount > 0) {
            System.out.println("Average Username Length: " + (double) totalUsernameLength / validCount);
        } else {
            System.out.println("Average Username Length: N/A");
        }
        System.out.println("Most Common Domain: " + (mostCommonDomain.isEmpty() ? "N/A" : mostCommonDomain));
    }

    // Method to display results in a tabular format
    public static void displayResults(List<EmailInfo> emails) {
        System.out.printf("%-30s%-15s%-20s%-15s%-15s%s\n", "Email", "Username", "Domain", "Domain Name", "Extension", "Valid/Invalid");
        System.out.println("-------------------------------------------------------------------------------------------------------------------");
        for (EmailInfo email : emails) {
            System.out.printf("%-30s%-15s%-20s%-15s%-15s%s\n",
                email.email, email.username, email.domain, email.domainName, email.extension, email.isValid ? "Valid" : "Invalid");
        }
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        List<EmailInfo> emailList = new ArrayList<>();
        System.out.println("Enter email addresses (type 'done' to finish):");

        while (true) {
            String email = scanner.nextLine();
            if (email.equalsIgnoreCase("done")) {
                break;
            }
            emailList.add(extractComponents(email));
        }

        displayResults(emailList);
        analyzeStatistics(emailList);

        scanner.close();
    }
}
```



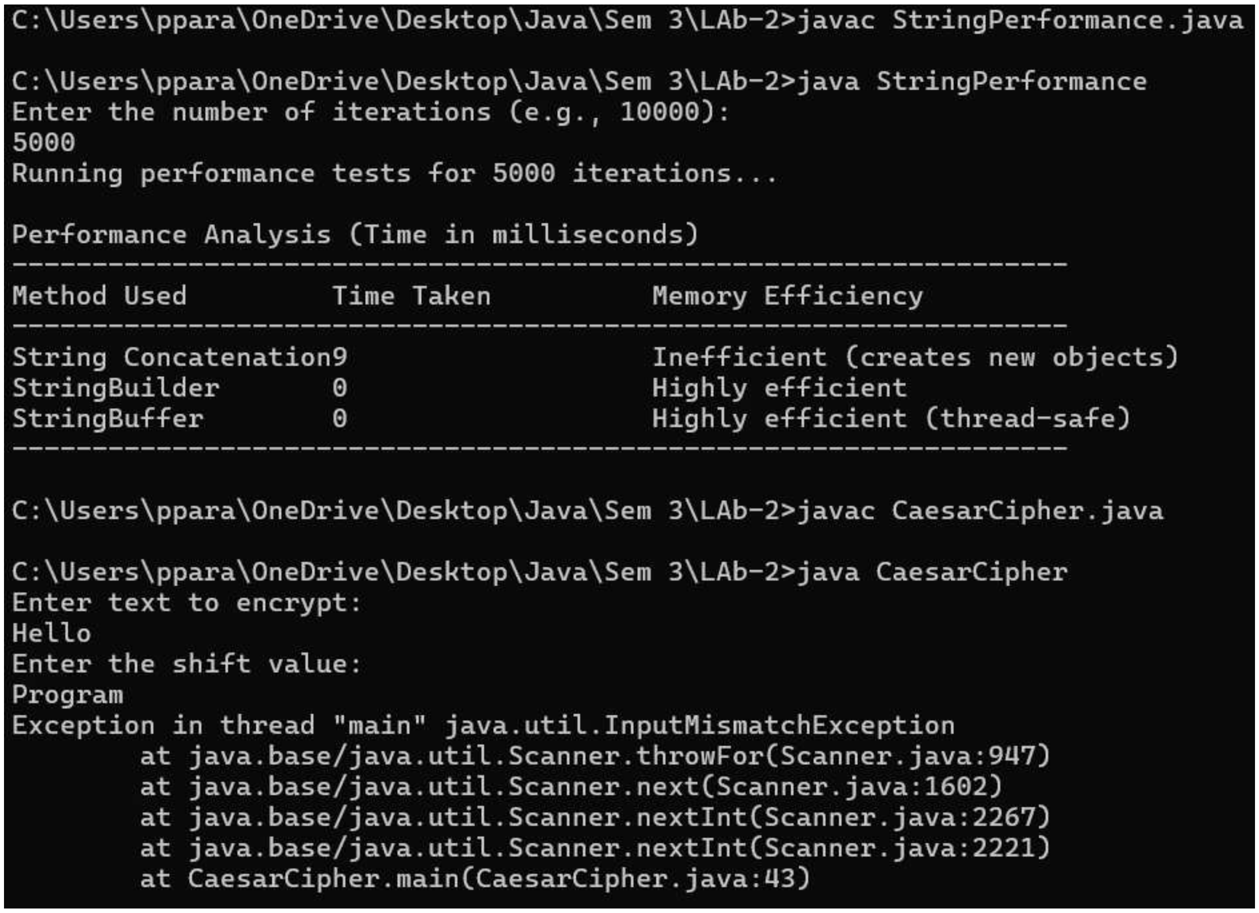
```java
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class TextFormatter {

    // Method to split text without using split()
    public static String[] customSplit(String text) {
        List<String> words = new ArrayList<>();
        int start = 0;
        for (int i = 0; i < text.length(); i++) {
            if (text.charAt(i) == ' ') {
                if (i > start) {
                    words.add(text.substring(start, i));
                }
                start = i + 1;
            }
        }
        if (start < text.length()) {
            words.add(text.substring(start));
        }
        return words.toArray(new String[0]);
    }

    // Method to justify text
    public static void justifyText(String[] words, int width) {
        System.out.println("\nLeft-Justified Text:");
        int i = 0;
        while (i < words.length) {
            StringBuilder line = new StringBuilder();
            int currentLineLength = 0;
            int wordsOnLine = 0;
            int startOfLine = i;

            while (i < words.length && currentLineLength + words[i].length() + wordsOnLine <= width) {
                currentLineLength += words[i].length();
                wordsOnLine++;
                i++;
            }

            int totalSpaces = width - currentLineLength;
            int spacesBetweenWords = wordsOnLine > 1 ? totalSpaces / (wordsOnLine - 1) : totalSpaces;
            int extraSpaces = wordsOnLine > 1 ? totalSpaces % (wordsOnLine - 1) : 0;

            for (int j = startOfLine; j < i; j++) {
                line.append(words[j]);
                if (j < i - 1) {
                    for (int k = 0; k < spacesBetweenWords; k++) {
                        line.append(" ");
                    }
                    if (extraSpaces > 0) {
                        line.append(" ");
                        extraSpaces--;
                    }
                }
            }
            if (wordsOnLine == 1) {
               for(int k=0; k<totalSpaces; k++) line.append(" ");
            }
            System.out.println(line);
        }
    }
```



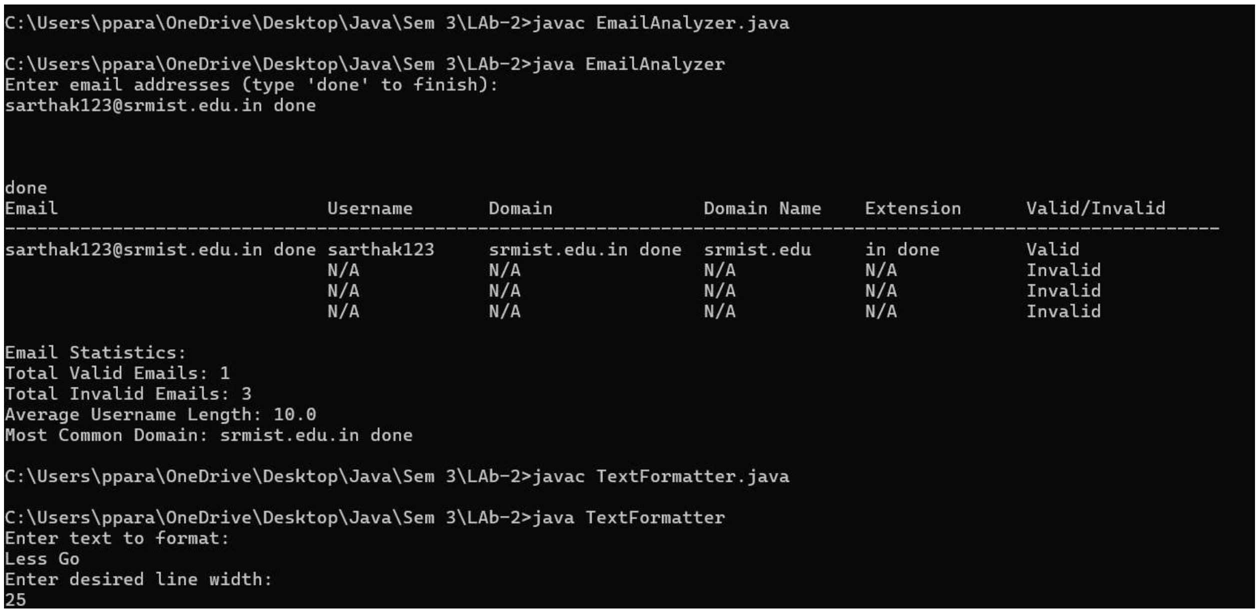
```java
public static void centerText(String[] words, int width) {
    System.out.println("\nCenter-Aligned Text:");
    int i = 0;
    while (i < words.length) {
        StringBuilder line = new StringBuilder();
        int currentLineLength = 0;
        int startOfLine = i;

        while (i < words.length && currentLineLength + words[i].length() + 1 <= width) {
            currentLineLength += words[i].length() + 1;
            i++;
        }
        currentLineLength--; // remove last space

        int padding = (width - currentLineLength) / 2;
        for (int k = 0; k < padding; k++) {
            line.append(" ");
        }
        for (int j = startOfLine; j < i; j++) {
            line.append(words[j]);
            if (j < i - 1) {
                line.append(" ");
            }
        }
        System.out.println(line);
    }
}

public static void comparePerformance(int iterations) {
    System.out.println("\nPerformance Analysis:");
    String text = "";
    StringBuilder sb = new StringBuilder();

    long startTime1 = System.nanoTime();
    for (int i = 0; i < iterations; i++) {
        text += "a";
    }
    long endTime1 = System.nanoTime();
    System.out.println("String Concatenation: " + (endTime1 - startTime1) / 1000000.0 + " ms");

    long startTime2 = System.nanoTime();
    for (int i = 0; i < iterations; i++) {
        sb.append("a");
    }
    long endTime2 = System.nanoTime();
    System.out.println("StringBuilder Append: " + (endTime2 - startTime2) / 1000000.0 + " ms");
}

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Enter text to format:");
    String text = scanner.nextLine();
    System.out.println("Enter desired line width:");
    int width = scanner.nextInt();

    String[] words = customSplit(text);

    justifyText(words, width);
    centerText(words, width);

    comparePerformance(10000);
```



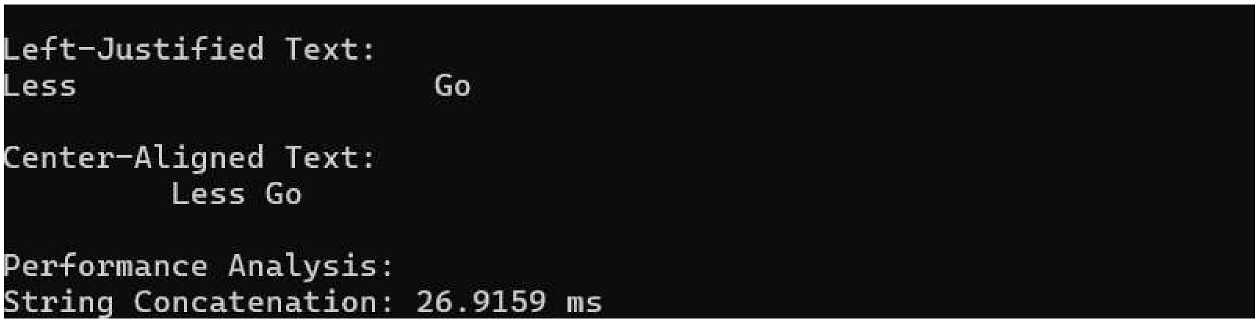
```
C:\Users\ppara\OneDrive\Desktop\Java\Sem 3\LAb-2>javac SubstringReplacer.java

C:\Users\ppara\OneDrive\Desktop\Java\Sem 3\LAb-2>java SubstringReplacer
Enter the main text:
Hello java user
Enter the substring to find:
java
Enter the replacement text:
user

Manual replacement result: Hello user user
Built-in replacement result: Hello user user
Are the results the same? true

C:\Users\ppara\OneDrive\Desktop\Java\Sem 3\LAb-2>javac TextConverter.java

C:\Users\ppara\OneDrive\Desktop\Java\Sem 3\LAb-2>java TextConverter
Enter a sentence:
Hello Java user

Case Type               User-Defined            Built-in
------------------------------------------------------------
Original                Hello Java user
UPPERCASE               HELLO JAVA USER         HELLO JAVA USER
lowercase               hello java user         hello java user
Title Case              Hello Java User
------------------------------------------------------------
```



```
C:\Users\ppara\OneDrive\Desktop\Java\Sem 3\LAb-2>javac StringPerformance.java

C:\Users\ppara\OneDrive\Desktop\Java\Sem 3\LAb-2>java StringPerformance
Enter the number of iterations (e.g., 10000):
5000
Running performance tests for 5000 iterations...

Performance Analysis (Time in milliseconds)
-----------------------------------------------------------------
Method Used         Time Taken          Memory Efficiency
-----------------------------------------------------------------
String Concatenation9                   Inefficient (creates new objects)
StringBuilder       0                   Highly efficient
StringBuffer        0                   Highly efficient (thread-safe)
-----------------------------------------------------------------


C:\Users\ppara\OneDrive\Desktop\Java\Sem 3\LAb-2>javac CaesarCipher.java

C:\Users\ppara\OneDrive\Desktop\Java\Sem 3\LAb-2>java CaesarCipher
Enter text to encrypt:
Hello
Enter the shift value:
Program
Exception in thread "main" java.util.InputMismatchException
        at java.base/java.util.Scanner.throwFor(Scanner.java:947)
        at java.base/java.util.Scanner.next(Scanner.java:1602)
        at java.base/java.util.Scanner.nextInt(Scanner.java:2267)
        at java.base/java.util.Scanner.nextInt(Scanner.java:2221)
        at CaesarCipher.main(CaesarCipher.java:43)
```



```
C:\Users\ppara\OneDrive\Desktop\Java\Sem 3\LAb-2>javac EmailAnalyzer.java

C:\Users\ppara\OneDrive\Desktop\Java\Sem 3\LAb-2>java EmailAnalyzer
Enter email addresses (type 'done' to finish):
sarthak123@srmist.edu.in done



done
Email                        Username       Domain              Domain Name    Extension      Valid/Invalid
------------------------------------------------------------------------------------------------------------------
sarthak123@srmist.edu.in done sarthak123     srmist.edu.in done  srmist.edu     in done        Valid
                             N/A            N/A                 N/A            N/A            Invalid
                             N/A            N/A                 N/A            N/A            Invalid
                             N/A            N/A                 N/A            N/A            Invalid

Email Statistics:
Total Valid Emails: 1
Total Invalid Emails: 3
Average Username Length: 10.0
Most Common Domain: srmist.edu.in done

C:\Users\ppara\OneDrive\Desktop\Java\Sem 3\LAb-2>javac TextFormatter.java

C:\Users\ppara\OneDrive\Desktop\Java\Sem 3\LAb-2>java TextFormatter
Enter text to format:
Less Go
Enter desired line width:
25
```



```
Left-Justified Text:
Less                    Go

Center-Aligned Text:
         Less Go

Performance Analysis:
String Concatenation: 26.9159 ms
```