Name : Tejaswini Anil Kamble

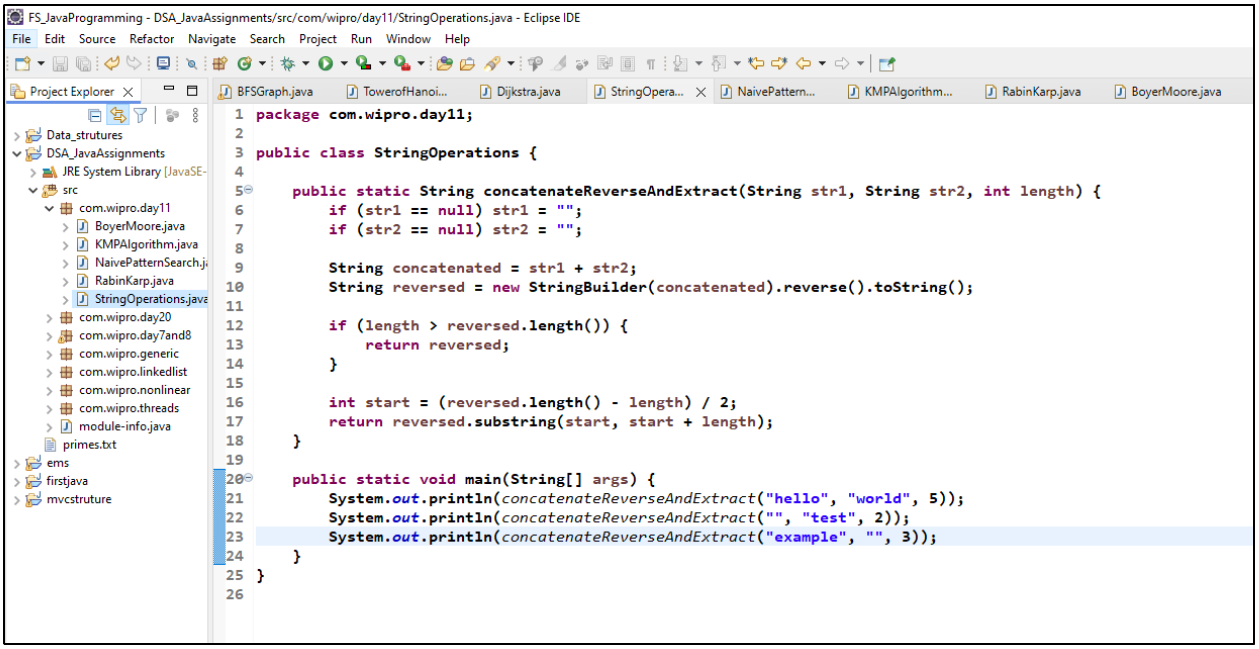
Email : teju000kamble@gmail.com

Assignments : Day 11

Task 1: String Operations

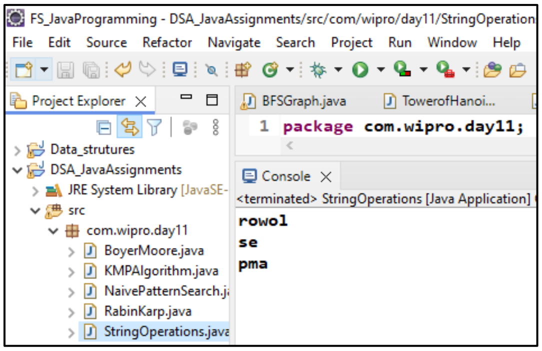
Write a method that takes two strings, concatenates them, reverses the result, and then extracts the middle substring of the given length. Ensure your method handles edge cases, such as an empty string or a substring length larger than the concatenated string.

Ans: Source Code



```java
package com.wipro.day11;

public class StringOperations {

    public static String concatenateReverseAndExtract(String str1, String str2, int length) {
        if (str1 == null) str1 = "";
        if (str2 == null) str2 = "";

        String concatenated = str1 + str2;
        String reversed = new StringBuilder(concatenated).reverse().toString();

        if (length > reversed.length()) {
            return reversed;
        }

        int start = (reversed.length() - length) / 2;
        return reversed.substring(start, start + length);
    }

    public static void main(String[] args) {
        System.out.println(concatenateReverseAndExtract("hello", "world", 5));
        System.out.println(concatenateReverseAndExtract("", "test", 2));
        System.out.println(concatenateReverseAndExtract("example", "", 3));
    }
}

```

Output:


FS_JavaProgramming - DSA_JavaAssignments/src/com/wipro/day11/StringOperation
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
BFSGraph.java
TowerofHanoi...
Data_strutures
DSA_JavaAssignments
JRE System Library [JavaSE-
src
com.wipro.day11
BoyerMoore.java
KMPAlgorithm.java
NaivePatternSearch.j
RabinKarp.java
StringOperations.java
Console
<terminated> StringOperations [Java Application]

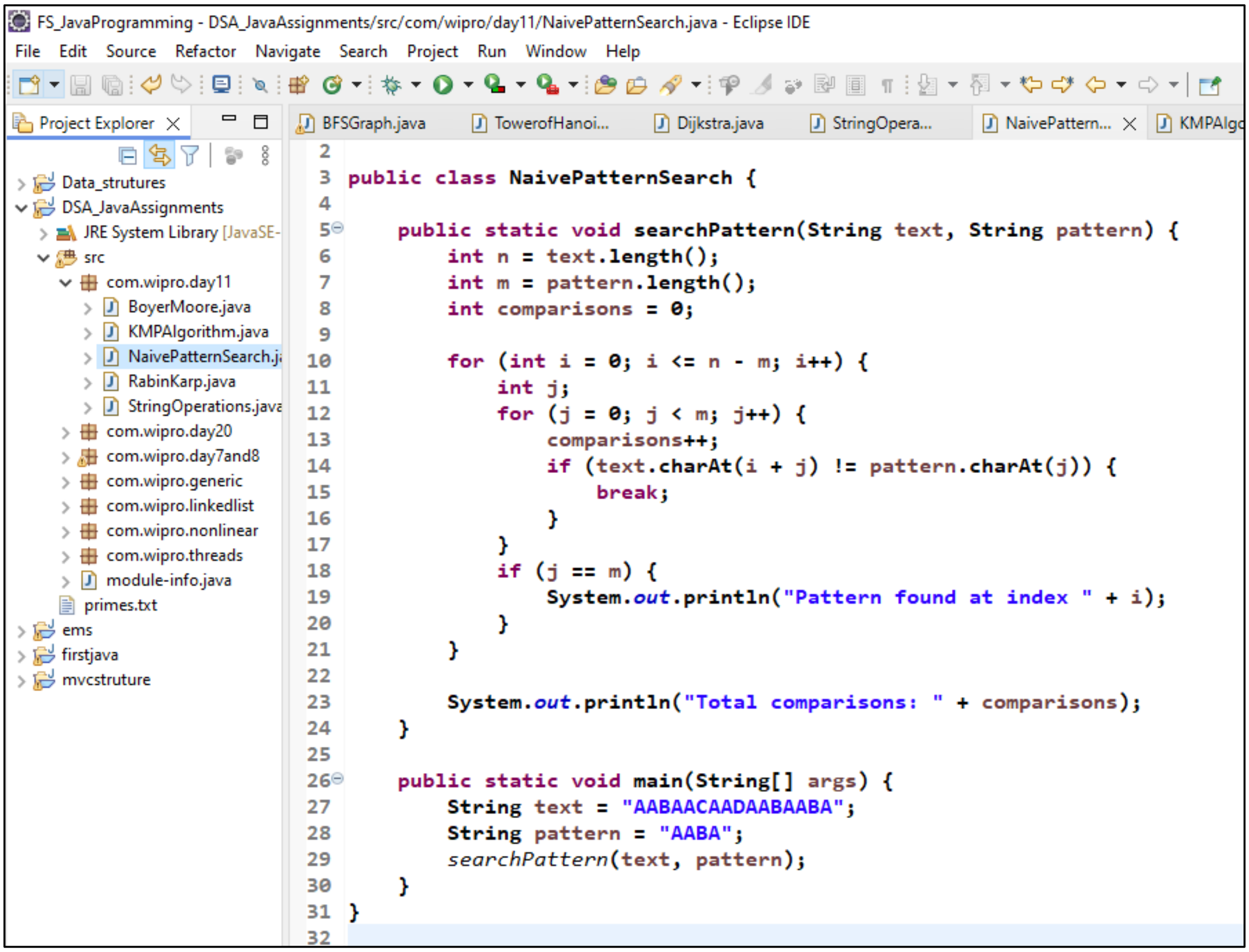
```java
package com.wipro.day11;
```

```
rowol
se
pma
```

Task 2: Naive Pattern Search

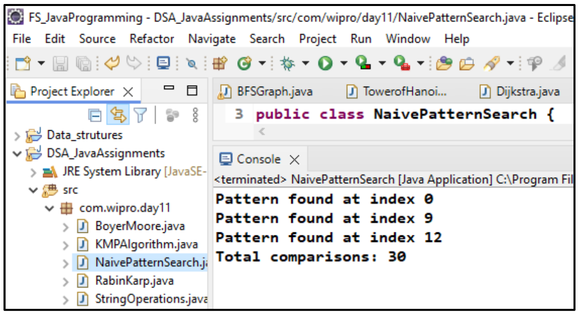
Implement the naive pattern searching algorithm to find all occurrences of a pattern within a given text string. Count the number of comparisons made during the search to evaluate the efficiency of the algorithm.

Ans: Source Code



```java

public class NaivePatternSearch {

    public static void searchPattern(String text, String pattern) {
        int n = text.length();
        int m = pattern.length();
        int comparisons = 0;

        for (int i = 0; i <= n - m; i++) {
            int j;
            for (j = 0; j < m; j++) {
                comparisons++;
                if (text.charAt(i + j) != pattern.charAt(j)) {
                    break;
                }
            }
            if (j == m) {
                System.out.println("Pattern found at index " + i);
            }
        }

        System.out.println("Total comparisons: " + comparisons);
    }

    public static void main(String[] args) {
        String text = "AABAACAADAABAABA";
        String pattern = "AABA";
        searchPattern(text, pattern);
    }
}

```

Output:

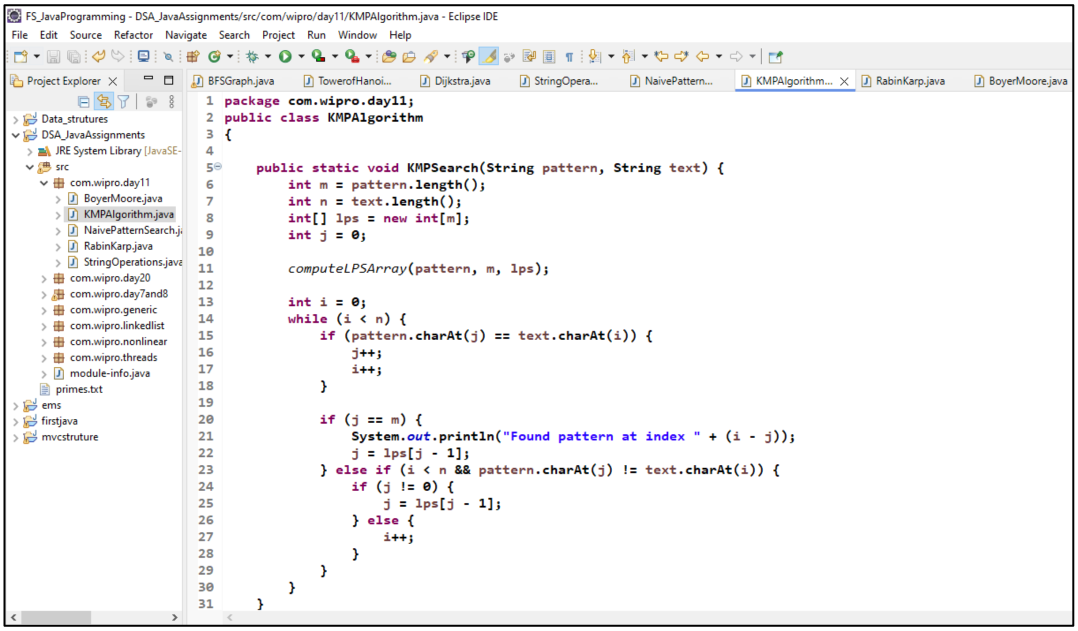

FS_JavaProgramming - DSA_JavaAssignments/src/com/wipro/day11/NaivePatternSearch.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
BFSGraph.java
TowerofHanoi...
Dijkstra.java
Data_strutures
DSA_JavaAssignments
JRE System Library [JavaSE-
src
com.wipro.day11
BoyerMoore.java
KMPAlgorithm.java
NaivePatternSearch.j
RabinKarp.java
StringOperations.java
3 public class NaivePatternSearch {
Console

```
<terminated> NaivePatternSearch [Java Application] C:\Program Fil
Pattern found at index 0
Pattern found at index 9
Pattern found at index 12
Total comparisons: 30
```

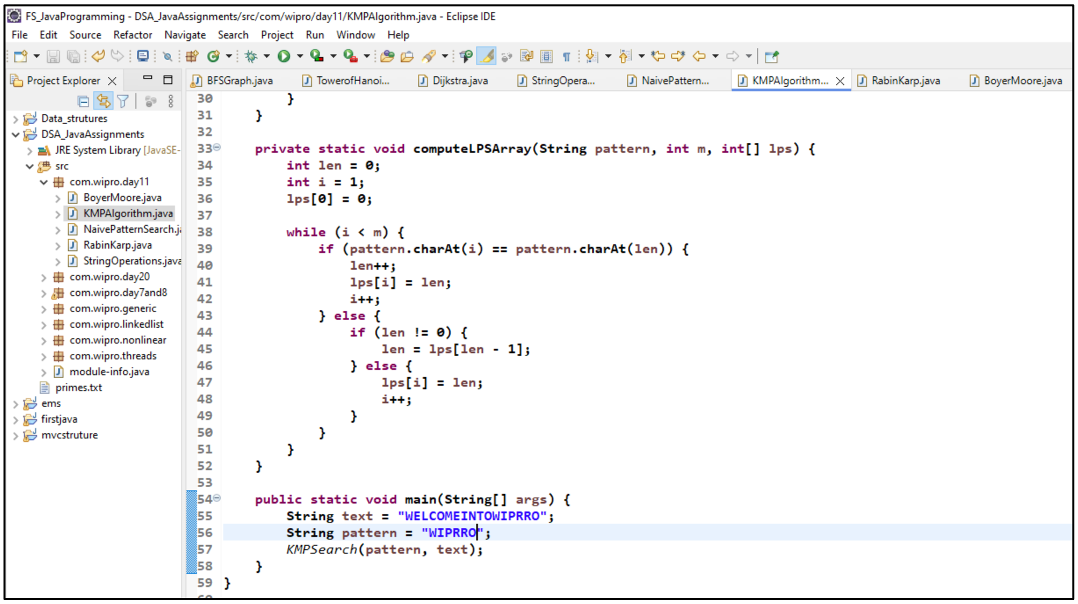
Task 3: Implementing the KMP Algorithm

Code the Knuth-Morris-Pratt (KMP) algorithm in C# for pattern searching which pre-processes the pattern to reduce the number of comparisons. Explain how this pre-processing improves the search time compared to the naive approach.

Ans: Source Code

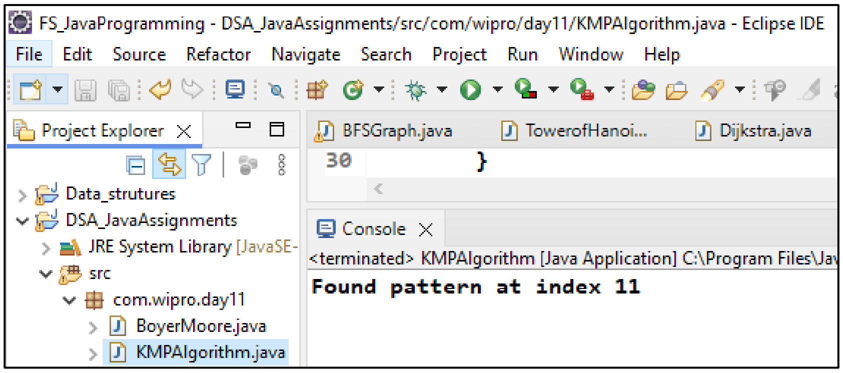


```java
package com.wipro.day11;
public class KMPAlgorithm
{

    public static void KMPSearch(String pattern, String text) {
        int m = pattern.length();
        int n = text.length();
        int[] lps = new int[m];
        int j = 0;

        computeLPSArray(pattern, m, lps);

        int i = 0;
        while (i < n) {
            if (pattern.charAt(j) == text.charAt(i)) {
                j++;
                i++;
            }

            if (j == m) {
                System.out.println("Found pattern at index " + (i - j));
                j = lps[j - 1];
            } else if (i < n && pattern.charAt(j) != text.charAt(i)) {
                if (j != 0) {
                    j = lps[j - 1];
                } else {
                    i++;
                }
            }
        }
    }
```



```java
        }
    }

    private static void computeLPSArray(String pattern, int m, int[] lps) {
        int len = 0;
        int i = 1;
        lps[0] = 0;

        while (i < m) {
            if (pattern.charAt(i) == pattern.charAt(len)) {
                len++;
                lps[i] = len;
                i++;
            } else {
                if (len != 0) {
                    len = lps[len - 1];
                } else {
                    lps[i] = len;
                    i++;
                }
            }
        }
    }

    public static void main(String[] args) {
        String text = "WELCOMEINTOWIPRRO";
        String pattern = "WIPRRO";
        KMPSearch(pattern, text);
    }
}
```

Output:


FS_JavaProgramming - DSA_JavaAssignments/src/com/wipro/day11/KMPAlgorithm.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Data_strutures
DSA_JavaAssignments
JRE System Library [JavaSE-
src
com.wipro.day11
BoyerMoore.java
KMPAlgorithm.java
BFSGraph.java
TowerofHanoi...
Dijkstra.java
Console

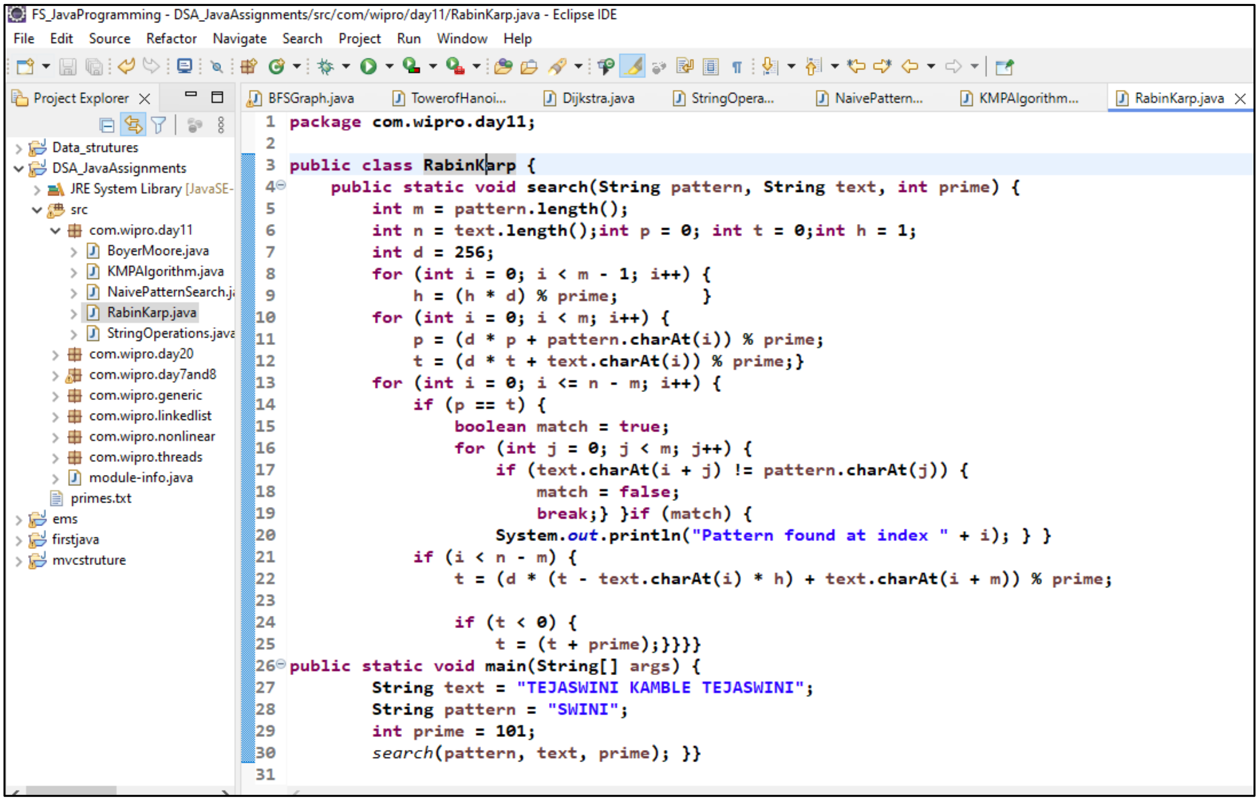
```
        }
```

```
<terminated> KMPAlgorithm [Java Application] C:\Program Files\Ja
Found pattern at index 11
```

Task 4: Rabin-Karp Substring Search

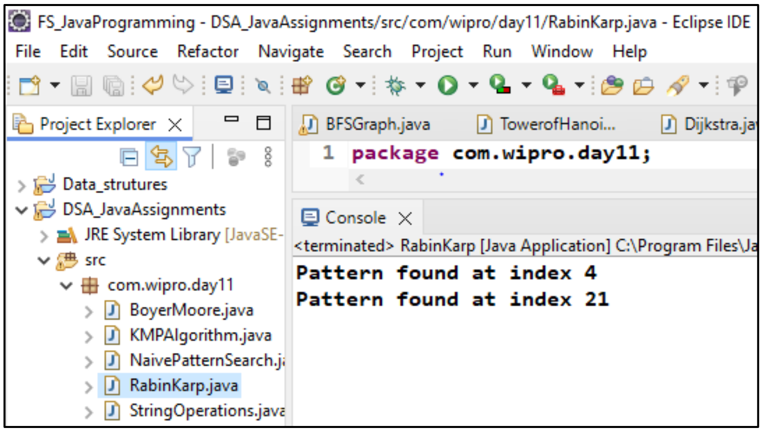
Implement the Rabin-Karp algorithm for substring search using a rolling hash. Discuss the impact of hash collisions on the algorithm's performance and how to handle them.

Ans: Source Code



```java
package com.wipro.day11;

public class RabinKarp {
    public static void search(String pattern, String text, int prime) {
        int m = pattern.length();
        int n = text.length();int p = 0; int t = 0;int h = 1;
        int d = 256;
        for (int i = 0; i < m - 1; i++) {
            h = (h * d) % prime;          }
        for (int i = 0; i < m; i++) {
            p = (d * p + pattern.charAt(i)) % prime;
            t = (d * t + text.charAt(i)) % prime;}
        for (int i = 0; i <= n - m; i++) {
            if (p == t) {
                boolean match = true;
                for (int j = 0; j < m; j++) {
                    if (text.charAt(i + j) != pattern.charAt(j)) {
                        match = false;
                        break;} }if (match) {
                    System.out.println("Pattern found at index " + i); } }
            if (i < n - m) {
                t = (d * (t - text.charAt(i) * h) + text.charAt(i + m)) % prime;

                if (t < 0) {
                    t = (t + prime);}}}}
public static void main(String[] args) {
        String text = "TEJASWINI KAMBLE TEJASWINI";
        String pattern = "SWINI";
        int prime = 101;
        search(pattern, text, prime); }}

```

Output:


FS_JavaProgramming - DSA_JavaAssignments/src/com/wipro/day11/RabinKarp.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Data_strutures
DSA_JavaAssignments
JRE System Library [JavaSE-
src
com.wipro.day11
BoyerMoore.java
KMPAlgorithm.java
NaivePatternSearch.j
RabinKarp.java
StringOperations.java
BFSGraph.java
TowerofHanoi...
Dijkstra.ja
Console
<terminated> RabinKarp [Java Application] C:\Program Files\Ja

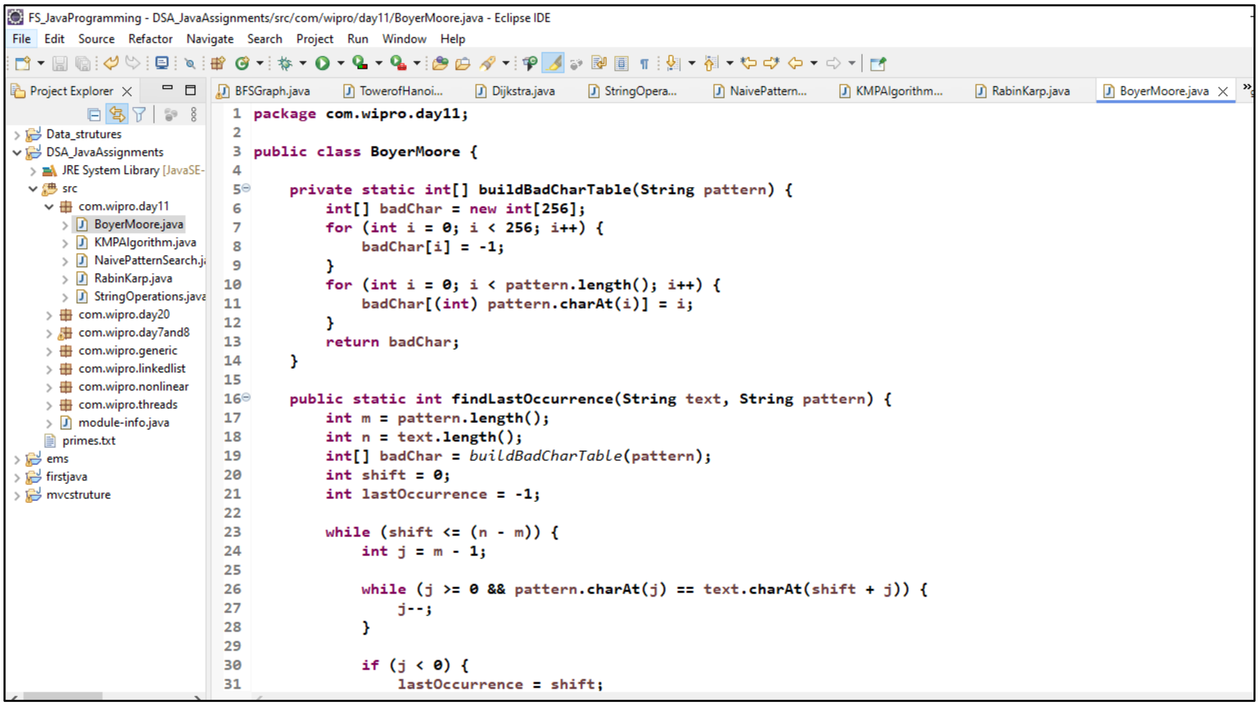
```
package com.wipro.day11;
```

```
Pattern found at index 4
Pattern found at index 21
```

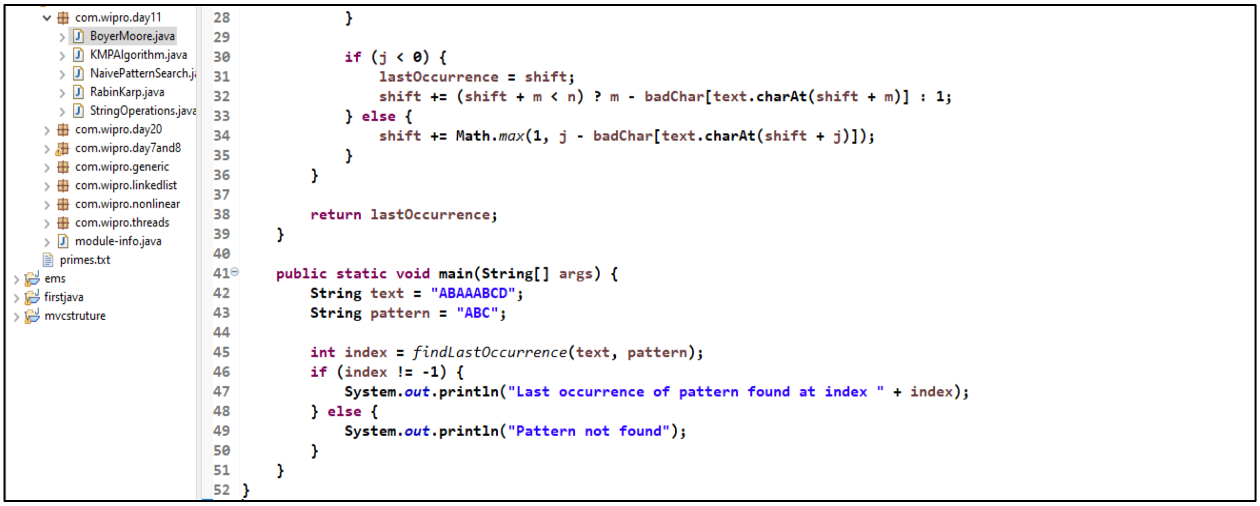
Task 5: Boyer-Moore Algorithm Application

Use the Boyer-Moore algorithm to write a function that finds the last occurrence of a substring in a given string and returns its index. Explain why this algorithm can outperform others in certain scenarios.

Ans : Source Code

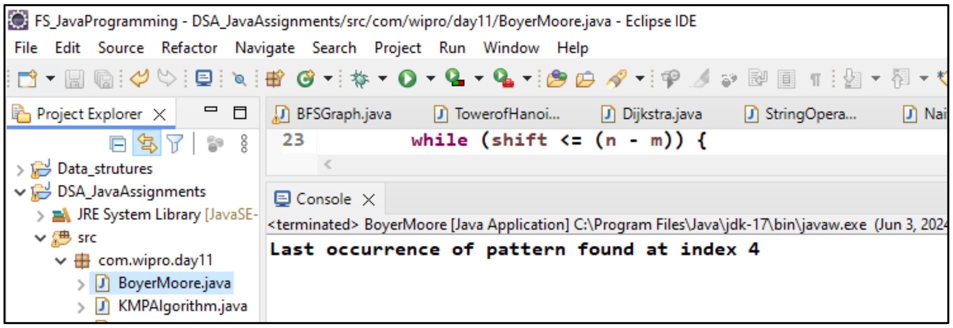


```
package com.wipro.day11;

public class BoyerMoore {

    private static int[] buildBadCharTable(String pattern) {
        int[] badChar = new int[256];
        for (int i = 0; i < 256; i++) {
            badChar[i] = -1;
        }
        for (int i = 0; i < pattern.length(); i++) {
            badChar[(int) pattern.charAt(i)] = i;
        }
        return badChar;
    }

    public static int findLastOccurrence(String text, String pattern) {
        int m = pattern.length();
        int n = text.length();
        int[] badChar = buildBadCharTable(pattern);
        int shift = 0;
        int lastOccurrence = -1;

        while (shift <= (n - m)) {
            int j = m - 1;

            while (j >= 0 && pattern.charAt(j) == text.charAt(shift + j)) {
                j--;
            }

            if (j < 0) {
                lastOccurrence = shift;
```



```
            }

            if (j < 0) {
                lastOccurrence = shift;
                shift += (shift + m < n) ? m - badChar[text.charAt(shift + m)] : 1;
            } else {
                shift += Math.max(1, j - badChar[text.charAt(shift + j)]);
            }
        }

        return lastOccurrence;
    }

    public static void main(String[] args) {
        String text = "ABAAABCD";
        String pattern = "ABC";

        int index = findLastOccurrence(text, pattern);
        if (index != -1) {
            System.out.println("Last occurrence of pattern found at index " + index);
        } else {
            System.out.println("Pattern not found");
        }
    }
}
```

Output:


FS_JavaProgramming - DSA_JavaAssignments/src/com/wipro/day11/BoyerMoore.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
BFSGraph.java
TowerofHanoi...
Dijkstra.java
StringOpera...
Nai
Data_strutures
DSA_JavaAssignments
JRE System Library [JavaSE-
src
com.wipro.day11
BoyerMoore.java
KMPAlgorithm.java
Console

```
        while (shift <= (n - m)) {
```

```
<terminated> BoyerMoore [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (Jun 3, 2024
Last occurrence of pattern found at index 4
```