RPS DAY 11 Assignments

Assignment 4

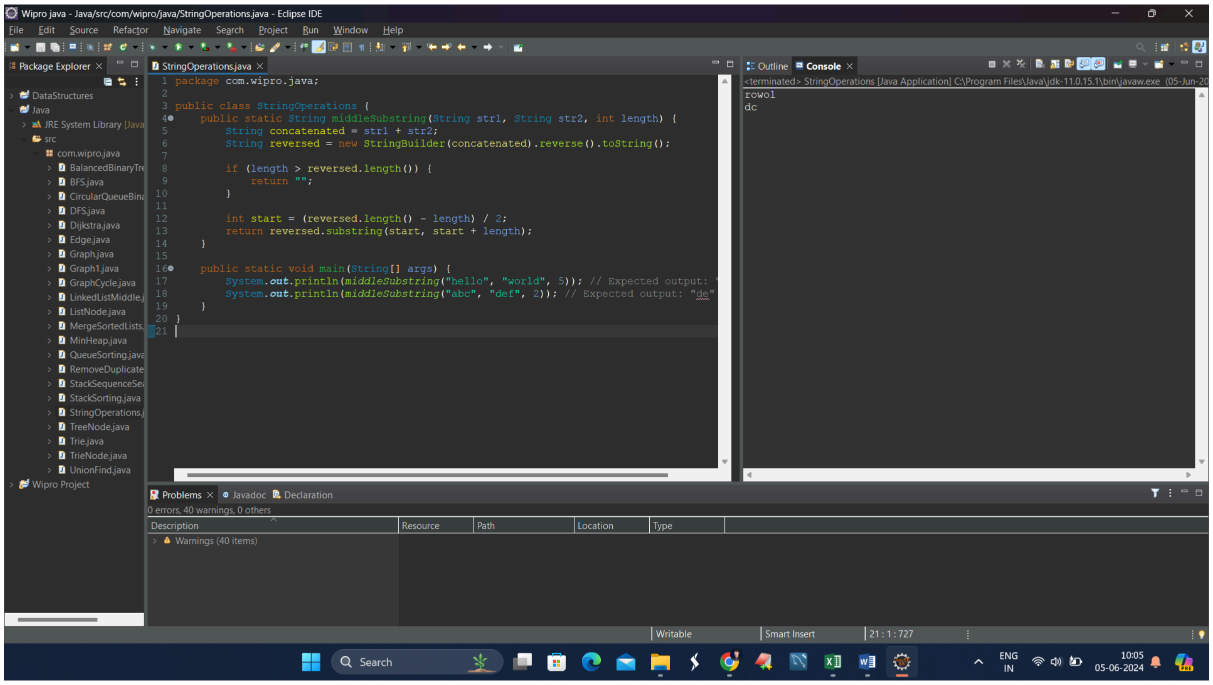
Name: Akshada Baad
Batch - CPPE

Day 11:

Task 1: String Operations

Wite 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.

```
public class StringOperations {
    public static String middleSubstring(String str1, String str2, int length) {
        String concatenated = str1 + str2;
        String reversed = new StringBuilder(concatenated).reverse().toString();

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

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

    public static void main(String[] args) {
        System.out.println(middleSubstring("hello", "world", 5)); // Expected output: "olleh"
        System.out.println(middleSubstring("abc", "def", 2)); // Expected output: "de"
    }
}
```



```
package com.wipro.java;

public class StringOperations {
    public static String middleSubstring(String str1, String str2, int length) {
        String concatenated = str1 + str2;
        String reversed = new StringBuilder(concatenated).reverse().toString();

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

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

    public static void main(String[] args) {
        System.out.println(middleSubstring("hello", "world", 5)); // Expected output: '
        System.out.println(middleSubstring("abc", "def", 2)); // Expected output: "de"
    }
}
```

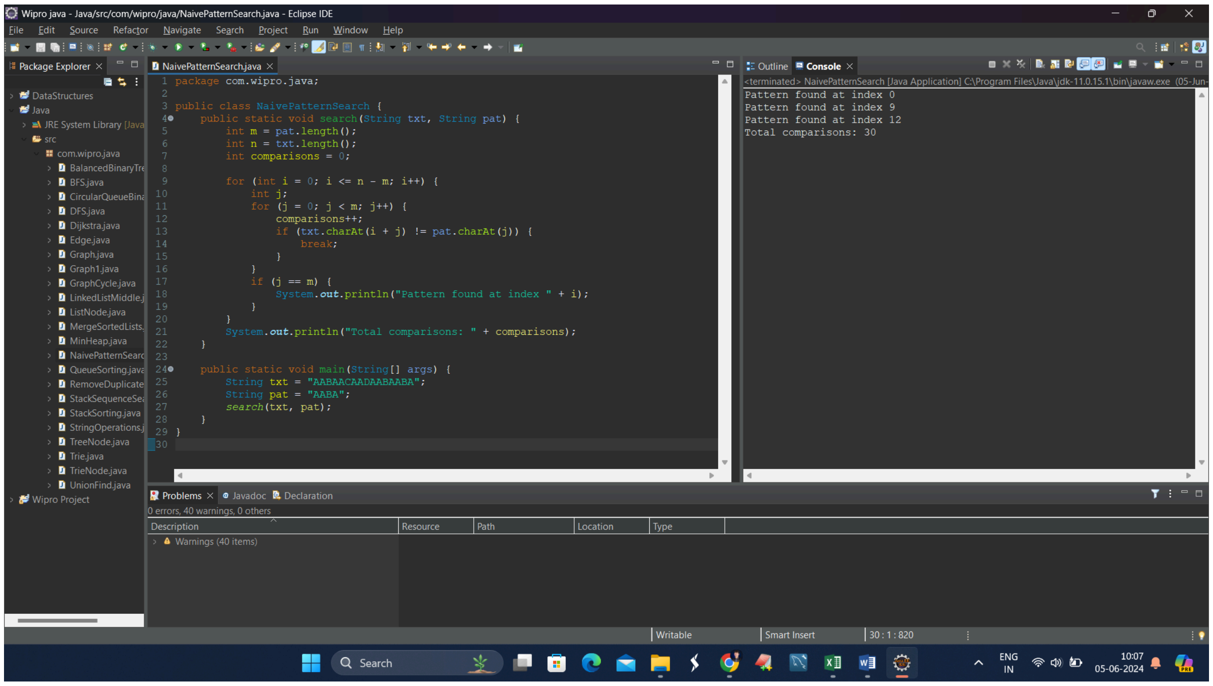
```
<terminated> StringOperations [Java Application] C:\Program Files\Java\jdk-11.0.15.1\bin\javaw.exe (05-Jun-20
rowol
dc
```

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.

```
public class NaivePatternSearch {
  public static void search(String txt, String pat) {
    int m = pat.length();
    int n = txt.length();
    int comparisons = 0;

    for (int i = 0; i <= n - m; i++) {
      int j;
      for (j = 0; j < m; j++) {
        comparisons++;
        if (txt.charAt(i + j) != pat.charAt(j)) {
          break;
```

```java
                }
            }
            if (j == m) {
                System.out.println("Pattern found at index " + i);
            }
        }
        System.out.println("Total comparisons: " + comparisons);
    }

    public static void main(String[] args) {
        String txt = "AABAACAADAABAABA";
        String pat = "AABA";
        search(txt, pat);
    }
}
```


Wipro java - Java/src/com/wipro/java/NaivePatternSearch.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
NaivePatternSearch.java
Outline
Console
Problems
Javadoc
Declaration
0 errors, 40 warnings, 0 others
Warnings (40 items)
Writable
Smart Insert
30 : 1 : 820

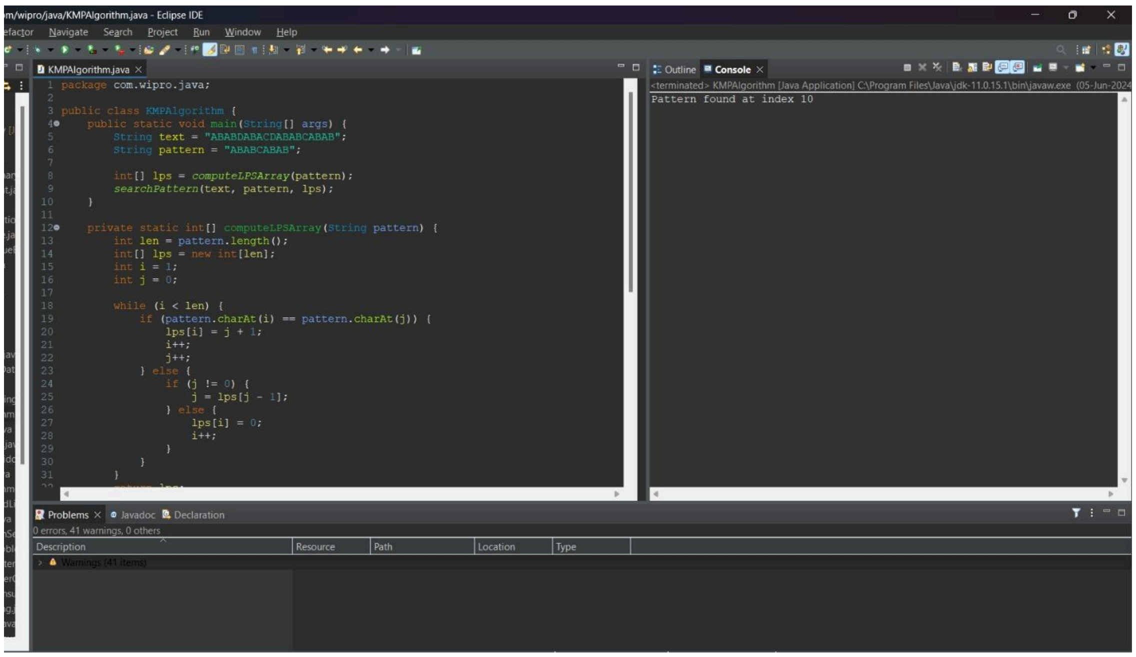
```java
package com.wipro.java;

public class NaivePatternSearch {
    public static void search(String txt, String pat) {
        int m = pat.length();
        int n = txt.length();
        int comparisons = 0;

        for (int i = 0; i <= n - m; i++) {
            int j;
            for (j = 0; j < m; j++) {
                comparisons++;
                if (txt.charAt(i + j) != pat.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 txt = "AABAACAADAABAABA";
        String pat = "AABA";
        search(txt, pat);
    }
}
```

```
<terminated> NaivePatternSearch [Java Application] C:\Program Files\Java\jdk-11.0.15.1\bin\javaw.exe  (05-Jun-
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 java 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.

```
public class KMPAlgorithm {
    public static void main(String[] args) {
        String text = "ABABDABACDABABCABAB";
        String pattern = "ABABCABAB";

        int[] lps = computeLPSArray(pattern);
        searchPattern(text, pattern, lps);
    }

    private static int[] computeLPSArray(String pattern) {
        int len = pattern.length();
        int[] lps = new int[len];
        int i = 1;
        int j = 0;

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

private static void searchPattern(String text, String pattern, int[] lps) {
    int m = text.length();
    int n = pattern.length();
    int i = 0;
    int j = 0;

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

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



om/wipro/java/KMPAlgorithm.java - Eclipse IDE

```
package com.wipro.java;

public class KMPAlgorithm {
    public static void main(String[] args) {
        String text = "ABABDABACDABABCABAB";
        String pattern = "ABABCABAB";

        int[] lps = computeLPSArray(pattern);
        searchPattern(text, pattern, lps);
    }

    private static int[] computeLPSArray(String pattern) {
        int len = pattern.length();
        int[] lps = new int[len];
        int i = 1;
        int j = 0;

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

```
<terminated> KMPAlgorithm [Java Application] C:\Program Files\Java\jdk-11.0.15.1\bin\javaw.exe (05-Jun-2024
Pattern found at index 10
```

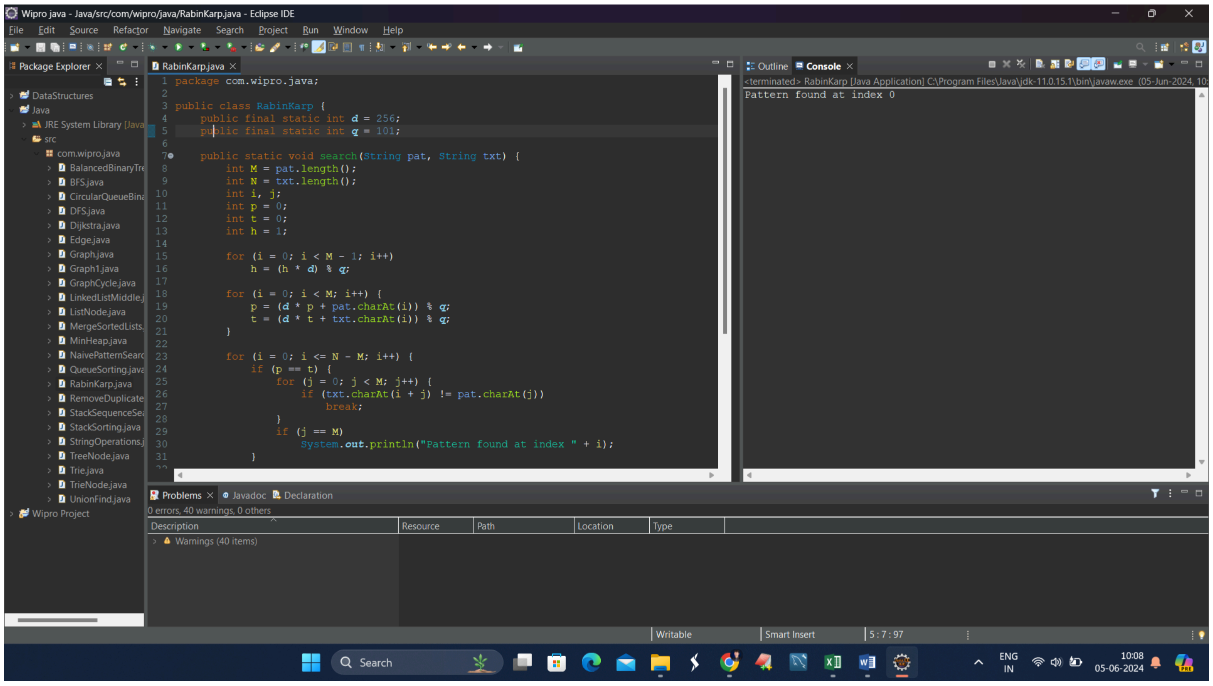
0 errors, 41 warnings, 0 others

Description	Resource	Path	Location	Type
Warnings (41 items)				

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.

```
 public class RabinKarp {
    public final static int d = 256;
    public final static int q = 101;

    public static void search(String pat, String txt) {
        int M = pat.length();
        int N = txt.length();
        int i, j;
        int p = 0;
        int t = 0;
        int h = 1;

        for (i = 0; i < M - 1; i++)
            h = (h * d) % q;

        for (i = 0; i < M; i++) {
            p = (d * p + pat.charAt(i)) % q;
            t = (d * t + txt.charAt(i)) % q;
        }

        for (i = 0; i <= N - M; i++) {
            if (p == t) {
                for (j = 0; j < M; j++) {
                    if (txt.charAt(i + j) != pat.charAt(j))
                        break;
                }
                if (j == M)
                    System.out.println("Pattern found at index " + i);
            }

            if (i < N - M) {
                t = (d * (t - txt.charAt(i) * h) + txt.charAt(i + 1)) % q;
                if (t < 0)
                    t = (t + q);
            }
        }
    }

    public static void main(String[] args) {
        String txt = "GEEKS FOR GEEKS";
        String pat = "GEEK";
        search(pat, txt);
    }
}
```



```
package com.wipro.java;

public class RabinKarp {
    public final static int d = 256;
    public final static int q = 101;

    public static void search(String pat, String txt) {
        int M = pat.length();
        int N = txt.length();
        int i, j;
        int p = 0;
        int t = 0;
        int h = 1;

        for (i = 0; i < M - 1; i++)
            h = (h * d) % q;

        for (i = 0; i < M; i++) {
            p = (d * p + pat.charAt(i)) % q;
            t = (d * t + txt.charAt(i)) % q;
        }

        for (i = 0; i <= N - M; i++) {
            if (p == t) {
                for (j = 0; j < M; j++) {
                    if (txt.charAt(i + j) != pat.charAt(j))
                        break;
                }
                if (j == M)
                    System.out.println("Pattern found at index " + i);
            }
```

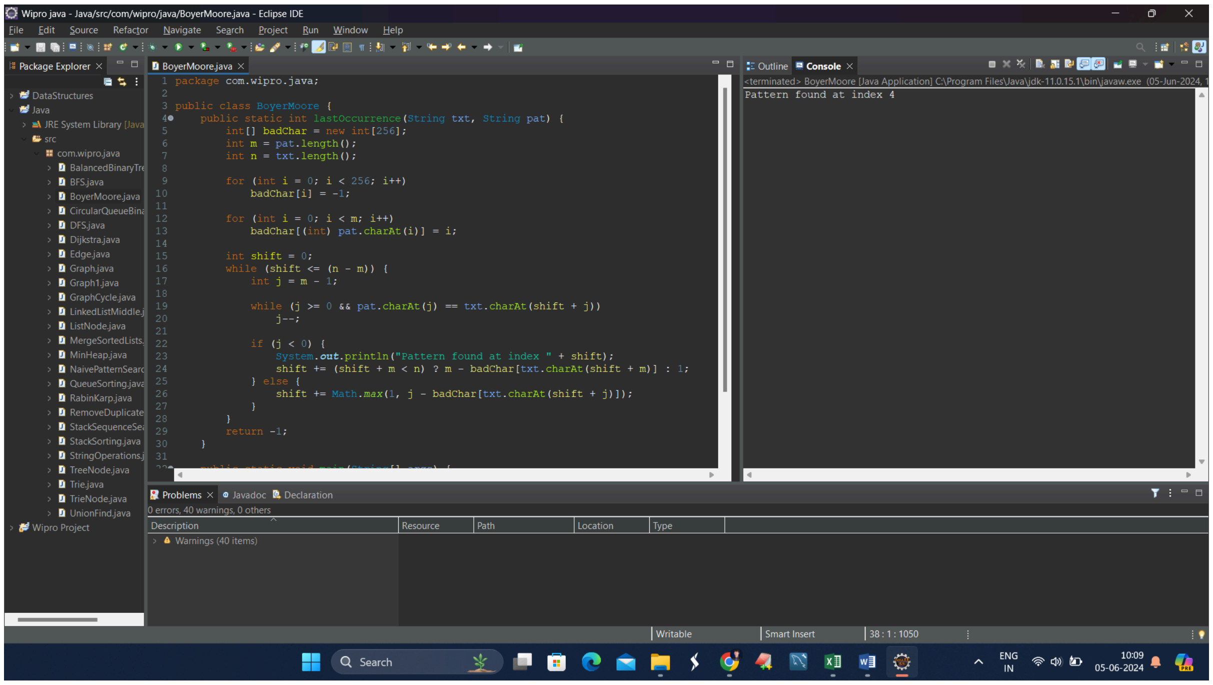
```
<terminated> RabinKarp [Java Application] C:\Program Files\Java\jdk-11.0.15.1\bin\javaw.exe (05-Jun-2024, 10:
Pattern found at index 0
```

```
0 errors, 40 warnings, 0 others
Description                Resource    Path    Location    Type
> Warnings (40 items)
```

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.

```
public class BoyerMoore {
    public static int lastOccurrence(String txt, String pat) {
        int[] badChar = new int[256];
        int m = pat.length();
        int n = txt.length();
    }

        for (int i = 0; i < 256; i++)
            badChar[i] = -1;

        for (int i = 0; i < m; i++)
            badChar[(int) pat.charAt(i)] = i;

        int shift = 0;
        while (shift <= (n - m)) {
            int j = m - 1;

            while (j >= 0 && pat.charAt(j) == txt.charAt(shift + j))
                j--;

            if (j < 0) {
                System.out.println("Pattern found at index " + shift);
                shift += (shift + m < n) ? m - badChar[txt.charAt(shift + m)] : 1;
            } else {
                shift += Math.max(1, j - badChar[txt.charAt(shift + j)]);
            }
        }
        return -1;
    }

    public static void main(String[] args) {
        String txt = "ABAAABCD";
        String pat = "ABC";
        lastOccurrence(txt, pat);
    }
}
```



```java
package com.wipro.java;

public class BoyerMoore {
    public static int lastOccurrence(String txt, String pat) {
        int[] badChar = new int[256];
        int m = pat.length();
        int n = txt.length();

        for (int i = 0; i < 256; i++)
            badChar[i] = -1;

        for (int i = 0; i < m; i++)
            badChar[(int) pat.charAt(i)] = i;

        int shift = 0;
        while (shift <= (n - m)) {
            int j = m - 1;

            while (j >= 0 && pat.charAt(j) == txt.charAt(shift + j))
                j--;

            if (j < 0) {
                System.out.println("Pattern found at index " + shift);
                shift += (shift + m < n) ? m - badChar[txt.charAt(shift + m)] : 1;
            } else {
                shift += Math.max(1, j - badChar[txt.charAt(shift + j)]);
            }
        }
        return -1;
    }

```

```
<terminated> BoyerMoore [Java Application] C:\Program Files\Java\jdk-11.0.15.1\bin\javaw.exe (05-Jun-2024,
Pattern found at index 4
```

0 errors, 40 warnings, 0 others

Warnings (40 items)