## Day 16

### Task 2: 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.

**A)**

### Java code:

```
package Day16;
public class StringOperations {
            public static String processStrings(String str1, String str2, int subLength) {
      // Concatenate the strings
      String concatenated = str1 + str2;

      // Check if the concatenated string is empty
      if (concatenated.isEmpty()) {
         return "";
      }

      // Reverse the concatenated string
      String reversed = new StringBuilder(concatenated).reverse().toString();

      // Handle case where subLength is larger than or equal to the reversed string
      if (subLength >= reversed.length()) {
         return reversed;
      }

      // Calculate start index for middle substring extraction
      int startIndex = Math.max(0, (reversed.length() - subLength) / 2);

      // Extract the middle substring
      return reversed.substring(startIndex, startIndex + subLength);
   }

   public static void main(String[] args) {
      // Test cases
      System.out.println(processStrings("hello", "world", 5));  // Expected output: "oworl"
      System.out.println(processStrings("abc", "def", 4));      // Expected output: "cdeb"
      System.out.println(processStrings("", "", 3));            // Expected output: ""
      System.out.println(processStrings("hi", "there", 10));    // Expected output: "erehtih"
      System.out.println(processStrings("apple", "pie", 3));    // Expected output: "eli"
      System.out.println(processStrings("single", "", 3));      // Expected output: "gni"
      System.out.println(processStrings("", "alone", 4));       // Expected output: "nola"
   }
}
```

## *Explanation:*

1. Concatenation**:** The method concatenates str1 and str2.

2. Empty String Check**:** If the concatenated string is empty, it returns an empty string.

3. Reversing: The concatenated string is reversed using StringBuilder.

4. Handling Large subLength: If subLength is larger than or equal to the length of the reversed string, the entire reversed string is returned.

5. Middle Substring Extraction**:**

- The start index for extracting the middle substring is calculated using the formula (reversed.length() - subLength) / 2.

- The Math.max ensures the start index is not negative, which can happen in edge cases.

- The substring is extracted from the reversed string using the calculated start index and subLength.

### *Additional Edge Cases Handled:*

- One or both input strings are empty.

- The specified substring length is larger than or equal to the length of the concatenated string.

- Typical cases where the input strings and substring length are within normal ranges.

This implementation is robust, handling various edge cases effectively while ensuring the correct substring is extracted from the reversed concatenated string.