HƯỚNG DẪN GIẢI BÀI TẬP
LẬP TRÌNH ỨNG DỤNG MẠNG
TUẦN 02

BÀI 1:

Một số phương pháp để tách chuỗi phép toán:

➔ Sử dụng hàm split.

➔ Sử dụng regrex

➔ Sử dụng lớp StringTokenizer

Dưới đây là đoạn code minh họa sử dụng regrex

W2_Ex1.java

```java
package Week2;

import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class W2_Ex1 {
    public static void main(String[] args) {
        String input = "1234567*890";
        String regrex = "(\\d+)([\\-+*/])(\\d+)";
        Pattern pattern = Pattern.compile(regrex);
        Matcher matcher = pattern.matcher(input);
        if(matcher.find()){
            String num1 = matcher.group(1);
            String op = matcher.group(2);
            String num2 = matcher.group(3);
            System.out.println(num1 + " " + op + " " + num2);
        }
    }
}
```

Sinh viên có thể tham khảo cách sử dụng regrex trong Java qua một số link:

➔ https://codelearn.io/sharing/regular-expression-trong-java

➔ https://viettuts.vn/java/su-dung-regex-trong-java

Dưới đây là đoạn code minh họa sử dụng StringTokenizer

W2_Ex1.java

```java
package Week2;

import java.util.StringTokenizer;

public class W2_Ex1 {
    public static void main(String[] args) {
        String input = "1234567*890";
        StringTokenizer st = new StringTokenizer(input, "+-*/", true);
        String num1 = st.nextToken();
        String op = st.nextToken();
        String num2 = st.nextToken();
        System.out.println(num1 + " " + op + " " + num2);
    }
}
```

Trong hai phương pháp trên, sử dụng StringTokenizer có lợi thế hơn Regrex, bởi nó có thể tách chuỗi phép toán có độ dài bất kỳ, không giới hạn bởi format như regrex, ví dụ đoạn code dưới đây sẽ tách và xuất ra màn hình tất cả các số và phép toán trong chuỗi:

W2_Ex1.java

```java
package Week2;

import java.util.StringTokenizer;

public class W2_Ex1 {
    public static void main(String[] args) {
        String input = "12+34-56*78/90*12/45-67";
        StringTokenizer st = new StringTokenizer(input, "+-*/", true);
        while(st.hasMoreTokens())
            System.out.println(st.nextToken());
    }
}
```

BÀI 2:

Một trong những phương pháp loại bỏ từ trùng nhau đơn giản và hiệu quả là sử dụng cấu trúc dữ liệu HashMap. Tuy nhiên, để giữ lại từ gốc, đồng thời so sánh được từ trùng nhau, ta cần sử dụng HashMap được thiết kế như sau:

<key, value> = <từ đã được chuyển thành chữ thường nhằm so sánh, từ gốc trong văn bản>

HashMap giải quyết được từ trùng nhau, nhưng sẽ gặp vấn đề về thứ tự các từ do cấu trúc dữ liệu này không đảm bảo thứ tự tương ứng lúc thêm. Để giải quyết vấn đề này, ta sử dụng cấu trúc dữ liệu tương tự HashMap là LinkedHashMap:

W2_Ex2.java

```java
package Week2;

import java.util.LinkedHashMap;
import java.util.StringTokenizer;

public class W2_Ex2 {

  public static void main(String[] args) {
    String input =
      " Dai   hoc  Sai Gon la mot trong nhung truong dai hoc lau doi nhat Sai Gon " ;
    StringTokenizer st = new StringTokenizer(input, " ");
    LinkedHashMap<String, String> map = new LinkedHashMap<>();
    while (st.hasMoreTokens()) {
      String value = st.nextToken();
      String key = value.toLowerCase();
      map.putIfAbsent(key, value);
    }
    StringBuilder output = new StringBuilder();
    map.forEach((key, value) -> {
      output.append(value).append(" ");
    });
    System.out.println(output);
  }
}
```