|  |  |  |  |
| --- | --- | --- | --- |
| **Ex. No. 01** | **TOKEN SEPERATION** | | |
| Date of Exercise | \_\_\_\_\_\_\_\_\_\_\_\_\_ | Date of Output Verification | \_\_\_\_\_\_\_\_\_\_\_\_\_ |

**Question**

Write a program to perform token separation for the subset of Java programming language. Accept the input as a “.java” file and perform token separation on the program.

**Program**

import java.io.\*;

import java.util.\*;

public class CompilerLabExp1TokenSeperation {

public static Scanner in = new Scanner(System.in);

public static BufferedReader filedata;

public static String line = "", temp;

public static StringTokenizer st;

public static ArrayList pkeywords = new ArrayList();

public static ArrayList psymbols = new ArrayList();

public static ArrayList poperators = new ArrayList();

public static ArrayList pliterals = new ArrayList();

public static ArrayList pvariables = new ArrayList();

public static ArrayList keywords = new ArrayList();

public static ArrayList symbols = new ArrayList();

public static ArrayList operators = new ArrayList();

public static ArrayList variables = new ArrayList();

public static ArrayList constants = new ArrayList();

public static ArrayList declarations = new ArrayList();

public static void main(String[] args) throws FileNotFoundException, IOException {

addpredata();

filedata = new BufferedReader(new FileReader("..\\samplejavacode.txt"));

while (!(temp = filedata.readLine()).contains("class")) {

if (temp.contains("import")) {

declarations.add(temp);

}

}

while ((line = filedata.readLine()) != null) {

st = new StringTokenizer(line, " +-/\*%^,=;{}( )", true);

while (st.hasMoreTokens()) {

temp = st.nextToken();

System.out.println(temp);

if (poperators.contains(temp)) {

if (operators.contains(temp)) {

continue;

}

operators.add(temp);

}

if (pkeywords.contains(temp)) {

if (keywords.contains(temp)) {

continue;

}

keywords.add(temp);

}

if (psymbols.contains(temp)) {

if (symbols.contains(temp)) {

continue;

}

symbols.add(temp);

}

if (pvariables.contains(temp)) {

while (!(temp = st.nextToken()).contains(";")) {

if (!temp.contains(" ") && !temp.contains(",") && !temp.contains("=")) {

if ((int) temp.charAt(0) >= 48 && (int) temp.charAt(0) <= 57) {

constants.add(temp);

} else {

variables.add(temp);

}

}

}

}

}

}

display();

}

public static void display() {

System.out.println("\n\n--------All Tokens------");

System.out.println("\nDeclarations: ");

for (int i = 0; i < declarations.size(); i++) {

System.out.println(declarations.get(i));

}

System.out.println("\nKeywords: ");

for (int i = 0; i < keywords.size(); i++) {

System.out.println(keywords.get(i));

}

System.out.println("\nSymbols: ");

for (int i = 0; i < symbols.size(); i++) {

System.out.println(symbols.get(i));

}

System.out.println("\nOperators: ");

for (int i = 0; i < operators.size(); i++) {

System.out.println(operators.get(i));

}

System.out.println("\nVariables: ");

for (int i = 0; i < variables.size(); i++) {

System.out.println(variables.get(i));

}

System.out.println("\nConstants: ");

for (int i = 0; i < constants.size(); i++) {

System.out.println(constants.get(i));

}

}

public static void addpredata() {

pkeywords.add("abstract");

pkeywords.add("break");

pkeywords.add("case");

pkeywords.add("catch");

pkeywords.add("class");

pkeywords.add("const");

pkeywords.add("continue");

pkeywords.add("default");

pkeywords.add("do");

pkeywords.add("else");

pkeywords.add("enum");

pkeywords.add("extends");

pkeywords.add("finally");

pkeywords.add("for");

pkeywords.add("goto");

pkeywords.add("if");

pkeywords.add("implements");

pkeywords.add("import");

pkeywords.add("instanceof");

pkeywords.add("interface");

pkeywords.add("native");

pkeywords.add("new");

pkeywords.add("package");

pkeywords.add("private");

pkeywords.add("protected");

pkeywords.add("public");

pkeywords.add("return");

pkeywords.add("static");

pkeywords.add("strictfp");

pkeywords.add("super");

pkeywords.add("switch");

pkeywords.add("synchronized");

pkeywords.add("this");

pkeywords.add("throw");

pkeywords.add("throws");

pkeywords.add("transient");

pkeywords.add("try");

pkeywords.add("void");

pkeywords.add("volatile");

pkeywords.add("while");

pkeywords.add("false");

pkeywords.add("null");

pkeywords.add("true");

pkeywords.add("System.out.println");

pkeywords.add("int");

pkeywords.add("char");

pkeywords.add("float");

pkeywords.add("String");

pkeywords.add("double");

pkeywords.add("short");

pkeywords.add("long");

pvariables.add("int");

pvariables.add("char");

pvariables.add("float");

pvariables.add("String");

pvariables.add("double");

pvariables.add("short");

pvariables.add("long");

psymbols.add("#");

psymbols.add("/");

psymbols.add("{");

psymbols.add("}");

psymbols.add("|");

psymbols.add("!");

psymbols.add("`");

psymbols.add("~");

psymbols.add("$");

psymbols.add("(");

psymbols.add(")");

poperators.add("+");

poperators.add("-");

poperators.add("\*");

poperators.add("+");

poperators.add("/");

poperators.add("%");

poperators.add("=");

}

}

**Input**

SampleJavaCode.java

import java.io.BufferedReader;

import java.util.StringTokenizer;

public class CompilerLabExp1Sample {

public static void main(String[] args) {

int a = 10, b=5, c;

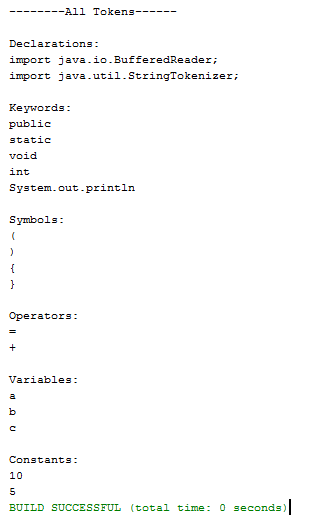
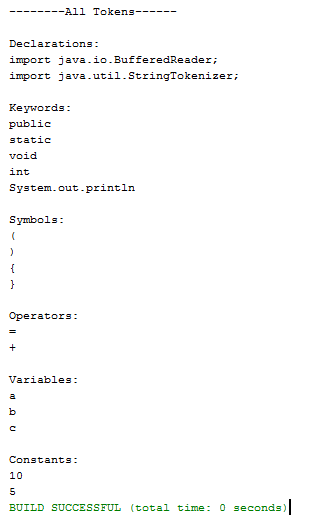
c=a+b;

System.out.println("c: " + c);

}

}

**Output**



**Result**

The Implementation of Token Separation is successfully done in java.

[Signature of the Staff In-charge]

Name of the Staff In – charge: Mr. Jeban Chandir Moses

Date: \_\_\_\_\_\_\_\_\_\_\_\_\_