**##Step -> 1 ::** Update code style for better consistency

import java.io.\*;

import java.text.\*;

import java.util.\*;

public class StudentList {

public static void main(String[] args) {

// Check arguments

if (args[0].equals("a")) {

System.out.println("Loading data ...");

try {

BufferedReader s = new BufferedReader(new InputStreamReader(new FileInputStream("students.txt")));

String r = s.readLine();

String i[] = r.split(",");

for (String j : i) {

System.out.println(j);

}

} catch (Exception e) {

}

System.out.println("Data Loaded.");

} else if (args[0].equals("r")) {

System.out.println("Loading data ...");

try {

BufferedReader s = new BufferedReader(new InputStreamReader(new FileInputStream("students.txt")));

String r = s.readLine();

System.out.println(r);

String i[] = r.split(",");

Random x = new Random();

int y = x.nextInt();

System.out.println(i[y]);

} catch (Exception e) {

}

System.out.println("Data Loaded.");

} else if (args[0].contains("+")) {

System.out.println("Loading data ...");

try {

BufferedWriter s = new BufferedWriter(new FileWriter("students.txt", true));

String t = args[0].substring(1);

Date d = new Date();

String df = "dd/mm/yyyy-hh:mm:ss a";

DateFormat dateFormat = new SimpleDateFormat(df);

String fd = dateFormat.format(d);

s.write(", " + t + "\nList last updated on " + fd);

s.close();

} catch (Exception e) {

}

System.out.println("Data Loaded.");

} else if (args[0].contains("?")) {

System.out.println("Loading data ...");

try {

BufferedReader s = new BufferedReader(new InputStreamReader(new FileInputStream("students.txt")));

String r = s.readLine();

String i[] = r.split(",");

boolean done = false;

String t = args[0].substring(1);

for (int idx = 0; idx < i.length && !done; idx++) {

if (i[idx].equals(t)) {

System.out.println("We found it!");

done = true;

}

}

} catch (Exception e) {

}

System.out.println("Data Loaded.");

} else if (args[0].contains("c")) {

System.out.println("Loading data ...");

try {

BufferedReader s = new BufferedReader(new InputStreamReader(new FileInputStream("students.txt")));

String D = s.readLine();

char a[] = D.toCharArray();

boolean in\_word = false;

int count = 0;

for (char c : a) {

if (c == ' ') {

if (!in\_word) {

count++;

in\_word = true;

} else {

in\_word = false;

}

}

}

System.out.println(count + " word(s) found " + a.length);

} catch (Exception e) {

}

System.out.println("Data Loaded.");

}

}

}

**## Step -> 2::** Application now terminates early if the number of arguments passed into it is wrong, fix it.

import java.io.\*;

import java.text.\*;

import java.util.\*;

public class StudentList {

public static void main(String[] args) {

// Check arguments

if(args == null || args.length != 1){

System.out.println("Usage: (a | r | c | +WORD | ?WORD)");

return; //Exit early.

}

if (args[0].equals("a")) {

System.out.println("Loading data ...");

try {

BufferedReader s = new BufferedReader(new InputStreamReader(new FileInputStream("students.txt")));

String r = s.readLine();

String i[] = r.split(",");

for (String j : i) {

System.out.println(j);

}

} catch (Exception e) {

}

System.out.println("Data Loaded.");

} else if (args[0].equals("r")) {

System.out.println("Loading data ...");

try {

BufferedReader s = new BufferedReader(new InputStreamReader(new FileInputStream("students.txt")));

String r = s.readLine();

System.out.println(r);

String i[] = r.split(",");

Random x = new Random();

int y = x.nextInt();

System.out.println(i[y]);

} catch (Exception e) {

}

System.out.println("Data Loaded.");

} else if (args[0].contains("+")) {

System.out.println("Loading data ...");

try {

BufferedWriter s = new BufferedWriter(new FileWriter("students.txt", true));

String t = args[0].substring(1);

Date d = new Date();

String df = "dd/mm/yyyy-hh:mm:ss a";

DateFormat dateFormat = new SimpleDateFormat(df);

String fd = dateFormat.format(d);

s.write(", " + t + "\nList last updated on " + fd);

s.close();

} catch (Exception e) {

}

System.out.println("Data Loaded.");

} else if (args[0].contains("?")) {

System.out.println("Loading data ...");

try {

BufferedReader s = new BufferedReader(new InputStreamReader(new FileInputStream("students.txt")));

String r = s.readLine();

String i[] = r.split(",");

boolean done = false;

String t = args[0].substring(1);

for (int idx = 0; idx < i.length && !done; idx++) {

if (i[idx].equals(t)) {

System.out.println("We found it!");

done = true;

}

}

} catch (Exception e) {

}

System.out.println("Data Loaded.");

} else if (args[0].contains("c")) {

System.out.println("Loading data ...");

try {

BufferedReader s = new BufferedReader(new InputStreamReader(new FileInputStream("students.txt")));

String D = s.readLine();

char a[] = D.toCharArray();

boolean in\_word = false;

int count = 0;

for (char c : a) {

if (c == ' ') {

if (!in\_word) {

count++;

in\_word = true;

} else {

in\_word = false;

}

}

}

System.out.println(count + " word(s) found " + a.length);

} catch (Exception e) {

}

System.out.println("Data Loaded.");

}

}

}

**##Step -> 3 ::** Makes improvements to variable names

import java.io.\*;

import java.text.\*;

import java.util.\*;

public class StudentList {

public static void main(String[] args) {

// Check arguments

if(args == null || args.length != 1){

System.out.println("Usage: (a | r | c | +WORD | ?WORD)");

return; //Exit early.

}

if (args[0].equals("a")) {

System.out.println("Loading data ...");

try {

BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(new FileInputStream("students.txt")));

String fileContents = bufferedReader.readLine();

String words[] = fileContents.split(",");

for (String word : words) {

System.out.println(word);

}

} catch (Exception e) {

}

System.out.println("Data Loaded.");

} else if (args[0].equals("r")) {

System.out.println("Loading data ...");

try {

BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(new FileInputStream("students.txt")));

String fileContents = bufferedReader.readLine();

String words[] = fileContents.split(",");

Random randomGenerator = new Random();

int randomNumber = randomGenerator.nextInt(words.length);

System.out.println(words[randomNumber]);

} catch (Exception e) {

}

System.out.println("Data Loaded.");

} else if (args[0].contains("+")) {

System.out.println("Loading data ...");

try {

BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter("students.txt", true));

String word = args[0].substring(1);

Date date = new Date();

String dateFormatSample = "dd/mm/yyyy-hh:mm:ss a";

DateFormat dateFormat = new SimpleDateFormat(dateFormatSample);

String finalDate = dateFormat.format(date);

bufferedWriter.write(", " + word + "\nList last updated on " + finalDate);

bufferedWriter.close();

} catch (Exception e) {

}

System.out.println("Data Loaded.");

} else if (args[0].contains("?")) {

System.out.println("Loading data ...");

try {

BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(new FileInputStream("students.txt")));

String fileContents = bufferedReader.readLine();

String words[] = fileContents.split(",");

boolean done = false;

String word = args[0].substring(1);

for (int idx = 0; idx < words.length && !done; idx++) {

if (words[idx].equals(word)) {

System.out.println("We found it!");

done = true;

}

}

} catch (Exception e) {

}

System.out.println("Data Loaded.");

} else if (args[0].contains("c")) {

System.out.println("Loading data ...");

try {

BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(new FileInputStream("students.txt")));

String fileContents = bufferedReader.readLine();

char charArray[] = fileContents.toCharArray();

boolean in\_word = false;

int count = 0;

for (char character : charArray) {

if (character == ' ') {

if (!in\_word) {

count++;

in\_word = true;

} else {

in\_word = false;

}

}

}

System.out.println(count + " word(s) found " + charArray.length);

} catch (Exception e) {

}

System.out.println("Data Loaded.");

}

}

}

**Step -> 4::** Refactors duplicate file read and write logic into methods

import java.io.\*;

import java.text.\*;

import java.util.\*;

public class StudentList {

public static void main(String[] args) {

// Check arguments

if(args == null || args.length != 1){

System.out.println("Usage: (a | r | c | +WORD | ?WORD)");

return; //Exit early.

}

String fileContents = readFileContents("students.txt");

if (args[0].equals("a")) {

System.out.println("Loading data ...");

String words[] = fileContents.split(",");

for (String word : words) {

System.out.println(word);

}

System.out.println("Data Loaded.");

} else if (args[0].equals("r")) {

System.out.println("Loading data ...");

String words[] = fileContents.split(",");

Random randomGenerator = new Random();

int randomNumber = randomGenerator.nextInt(words.length);

System.out.println(words[randomNumber]);

System.out.println("Data Loaded.");

} else if (args[0].contains("+")) {

System.out.println("Loading data ...");

String word = args[0].substring(1);

Date date = new Date();

String dateFormatSample = "dd/mm/yyyy-hh:mm:ss a";

DateFormat dateFormat = new SimpleDateFormat(dateFormatSample);

String finalDate = dateFormat.format(date);

writeToFile("students.txt",", " + word + "\nList last updated on " + finalDate);

System.out.println("Data Loaded.");

} else if (args[0].contains("?")) {

System.out.println("Loading data ...");

String words[] = fileContents.split(",");

boolean done = false;

String word = args[0].substring(1);

for (int idx = 0; idx < words.length && !done; idx++) {

if (words[idx].equals(word)) {

System.out.println("We found it!");

done = true;

}

}

System.out.println("Data Loaded.");

} else if (args[0].contains("c")) {

System.out.println("Loading data ...");

char charArray[] = fileContents.toCharArray();

boolean in\_word = false;

int count = 0;

for (char character : charArray) {

if (character == ' ') {

if (!in\_word) {

count++;

in\_word = true;

} else {

in\_word = false;

}

}

}

System.out.println(count + " word(s) found ");

System.out.println("Data Loaded.");

}

}

public static String readFileContents(String filename) {

try {

BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(new FileInputStream(filename)));

String fileContents = bufferedReader.readLine();

bufferedReader.close();

return fileContents;

} catch (Exception e) {

return null;

}

}

public static void writeToFile(String filename, String data) {

try {

BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(filename, true));

bufferedWriter.write(data);

bufferedWriter.close();

} catch (Exception e) {

e.printStackTrace();

}

}

}

**Step -> 5::** Replaces string literals with constants, storing those constants in a new class called Constants.java

import java.io.\*;

import java.text.\*;

import java.util.\*;

public class StudentList {

public static void main(String[] args) {

// Check arguments

if(args == null || args.length != 1){

System.out.println("Usage: (a | r | c | +WORD | ?WORD)");

return; //Exit early.

}

String fileContents = readFileContents(Constants.FILE\_NAME);

if (args[0].equals(Constants.ShowAll)) {

String words[] = fileContents.split(Constants.StudentEntryDelimiter);

for (String word : words) {

System.out.println(word);

}

} else if (args[0].equals(Constants.ShowRandom)) {

String words[] = fileContents.split(Constants.StudentEntryDelimiter);

Random randomGenerator = new Random();

int randomNumber = randomGenerator.nextInt(words.length);

System.out.println(words[randomNumber]);

} else if (args[0].contains(Constants.AddEntry)) {

String word = args[0].substring(1);

Date date = new Date();

DateFormat dateFormat = new SimpleDateFormat(Constants.DATE\_FORMAT);

String finalDate = dateFormat.format(date);

writeToFile(Constants.FILE\_NAME,finalDate,word);

} else if (args[0].contains(Constants.FindEntry)) {

String words[] = fileContents.split(Constants.StudentEntryDelimiter);

boolean done = false;

String word = args[0].substring(1);

for (int idx = 0; idx < words.length && !done; idx++) {

if (words[idx].equals(word)) {

System.out.println(Constants.FoundText);

done = true;

}

}

} else if (args[0].contains(Constants.ShowCount)) {

char charArray[] = fileContents.toCharArray();

boolean in\_word = false;

int count = 0;

for (char character : charArray) {

if (character == Constants.Space) {

if (!in\_word) {

count++;

in\_word = true;

} else {

in\_word = false;

}

}

}

System.out.println(count + Constants.WordsFound);

}

System.out.println(Constants.DataLoadedText);

}

public static String readFileContents(String filename) {

System.out.println(Constants.LoadingDataText);

try {

BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(new FileInputStream(filename)));

String fileContents = bufferedReader.readLine();

bufferedReader.close();

return fileContents;

} catch (Exception e) {

return null;

}

}

public static void writeToFile(String filename, String finalDate, String word) {

String data = ", " + word + "\nList last updated on " + finalDate;

try {

BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(filename, true));

bufferedWriter.write(data);

bufferedWriter.close();

} catch (Exception e) {

e.printStackTrace();

}

}

}

//Constants.java

public class Constants {

public static final String ShowAll = "a";

public static final String ShowRandom = "r";

public static final String AddEntry = "+";

public static final String FindEntry = "?";

public static final String ShowCount = "c";

public static final String FILE\_NAME = "students.txt";

public static final String DATE\_FORMAT = "dd/MM/yyyy-hh:mm:ss a";

public static final String StudentEntryDelimiter = ", ";

public static final String LoadingDataText = "Loading data ...";

public static final String DataLoadedText = "Data Loaded.";

public static final String FoundText = "We found it!";

public static final String WordsFound = " word(s) found ";

public static final char Space = ' ';

}

**Step -> 6::** Remove Temporary variables

import java.io.\*;

import java.text.\*;

import java.util.\*;

public class StudentList {

public static void main(String[] args) {

// Check arguments

if(args == null || args.length != 1){

System.out.println("Usage: (a | r | c | +WORD | ?WORD)");

return; //Exit early.

}

String fileContents = readFileContents(Constants.FILE\_NAME);

if (args[0].equals(Constants.ShowAll)) {

String words[] = fileContents.split(Constants.StudentEntryDelimiter);

for (String word : words) {

System.out.println(word);

}

} else if (args[0].equals(Constants.ShowRandom)) {

String words[] = fileContents.split(Constants.StudentEntryDelimiter);

int randomNumber = new Random().nextInt(words.length);

System.out.println(words[randomNumber]);

} else if (args[0].contains(Constants.AddEntry)) {

writeToFile(Constants.FILE\_NAME,new SimpleDateFormat(Constants.DATE\_FORMAT).format(new Date()),args[0].substring(1));

} else if (args[0].contains(Constants.FindEntry)) {

String words[] = fileContents.split(Constants.StudentEntryDelimiter);

boolean done = false;

String word = args[0].substring(1);

for (int idx = 0; idx < words.length && !done; idx++) {

if (words[idx].equals(word)) {

System.out.println(Constants.FoundText);

done = true;

}

}

} else if (args[0].contains(Constants.ShowCount)) {

char charArray[] = fileContents.toCharArray();

boolean in\_word = false;

int count = 0;

for (char character : charArray) {

if (character == Constants.Space) {

if (!in\_word) {

count++;

in\_word = true;

} else {

in\_word = false;

}

}

}

System.out.println(count + Constants.WordsFound);

}

System.out.println(Constants.DataLoadedText);

}

public static String readFileContents(String filename) {

System.out.println(Constants.LoadingDataText);

try {

BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(new FileInputStream(filename)));

String fileContents = bufferedReader.readLine();

bufferedReader.close();

return fileContents;

} catch (Exception e) {

return null;

}

}

public static void writeToFile(String filename, String finalDate, String word) {

try {

BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(filename, true));

bufferedWriter.write(", " + word + "\nList last updated on " + finalDate);

bufferedWriter.close();

} catch (Exception e) {

e.printStackTrace();

}

}

}

**Step ->7 ::** Eliminates the ‘done’ control-flow variable. Adds better response for search operation.

import java.io.\*;

import java.text.\*;

import java.util.\*;

public class StudentList {

public static void main(String[] args) {

// Check arguments

if(args == null || args.length != 1){

System.out.println("Usage: (a | r | c | +WORD | ?WORD)");

return; //Exit early.

}

String fileContents = readFileContents(Constants.FILE\_NAME);

if (args[0].equals(Constants.ShowAll)) {

showAll(fileContents);

} else if (args[0].equals(Constants.ShowRandom)) {

showRandom(fileContents);

} else if (args[0].contains(Constants.AddEntry)) {

addEntry(Constants.FILE\_NAME,args[0].substring(1));

} else if (args[0].contains(Constants.FindEntry)) {

findEntry(fileContents,args[0].substring(1));

} else if (args[0].contains(Constants.ShowCount)) {

showWordCount(fileContents);

}

System.out.println(Constants.DataLoadedText);

}

public static void showWordCount(String fileContents) {

char charArray[] = fileContents.toCharArray();

boolean in\_word = false;

int count = 0;

for (char character : charArray) {

if (character == Constants.Space) {

if (!in\_word) {

count++;

in\_word = true;

} else {

in\_word = false;

}

}

}

System.out.println(count + Constants.WordsFound);

}

public static void findEntry(String fileContents, String word) {

String words[] = fileContents.split(Constants.StudentEntryDelimiter);

boolean found = false;

for (String w : words) {

if (w.equals(word)) {

System.out.println(Constants.FoundText);

found = true;

break;

}

}

if (!found) {

System.out.println("Word not found.");

}

}

public static void addEntry(String filename, String word) {

writeToFile(filename, new SimpleDateFormat(Constants.DATE\_FORMAT).format(new Date()), word);

}

public static void showRandom(String fileContents) {

String words[] = fileContents.split(Constants.StudentEntryDelimiter);

if (words.length > 0) {

int randomNumber = new Random().nextInt(words.length);

System.out.println(words[randomNumber]);

} else {

System.out.println("No words found.");

}

}

public static void showAll(String fileContents) {

String words[] = fileContents.split(Constants.StudentEntryDelimiter);

for (String word : words) {

System.out.println(word);

}

}

public static String readFileContents(String filename) {

System.out.println(Constants.LoadingDataText);

try {

BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(new FileInputStream(filename)));

String fileContents = bufferedReader.readLine();

bufferedReader.close();

return fileContents;

} catch (Exception e) {

return null;

}

}

public static void writeToFile(String filename, String finalDate, String word) {

try {

BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(filename, true));

bufferedWriter.write(", " + word + "\nList last updated on " + finalDate);

bufferedWriter.close();

} catch (Exception e) {

e.printStackTrace();

}

}

}

**Step -> 8::** Simplifies the logic behind the count operation

import java.io.\*;

import java.text.\*;

import java.util.\*;

public class StudentList {

public static void main(String[] args) {

// Check arguments

if(args == null || args.length != 1){

System.out.println("Usage: (a | r | c | +WORD | ?WORD)");

return; //Exit early.

}

String fileContents = readFileContents(Constants.FILE\_NAME);

if (args[0].equals(Constants.ShowAll)) {

showAll(fileContents);

} else if (args[0].equals(Constants.ShowRandom)) {

showRandom(fileContents);

} else if (args[0].contains(Constants.AddEntry)) {

addEntry(Constants.FILE\_NAME,args[0].substring(1));

} else if (args[0].contains(Constants.FindEntry)) {

findEntry(fileContents,args[0].substring(1));

} else if (args[0].contains(Constants.ShowCount)) {

showWordCount(fileContents);

}

System.out.println(Constants.DataLoadedText);

}

public static void showWordCount(String fileContents) {

String[] words = fileContents.split(Constants.StudentEntryDelimiter);

System.out.println(words.length + Constants.WordsFound);

}

public static void findEntry(String fileContents, String word) {

String words[] = fileContents.split(Constants.StudentEntryDelimiter);

boolean found = false;

for (String w : words) {

if (w.equals(word)) {

System.out.println(Constants.FoundText);

found = true;

break;

}

}

if (!found) {

System.out.println("Word not found.");

}

}

public static void addEntry(String filename, String word) {

writeToFile(filename, new SimpleDateFormat(Constants.DATE\_FORMAT).format(new Date()), word);

}

public static void showRandom(String fileContents) {

String words[] = fileContents.split(Constants.StudentEntryDelimiter);

if (words.length > 0) {

int randomNumber = new Random().nextInt(words.length);

System.out.println(words[randomNumber]);

} else {

System.out.println("No words found.");

}

}

public static void showAll(String fileContents) {

String words[] = fileContents.split(Constants.StudentEntryDelimiter);

for (String word : words) {

System.out.println(word);

}

}

public static String readFileContents(String filename) {

System.out.println(Constants.LoadingDataText);

try {

BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(new FileInputStream(filename)));

String fileContents = bufferedReader.readLine();

bufferedReader.close();

return fileContents;

} catch (Exception e) {

return null;

}

}

public static void writeToFile(String filename, String finalDate, String word) {

try {

BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(filename, true));

bufferedWriter.write(", " + word + "\nList last updated on " + finalDate);

bufferedWriter.close();

} catch (Exception e) {

e.printStackTrace();

}

}

}

**Step -> 9::** Adds handling for case when user enters invalid arguments

import java.io.\*;

import java.text.\*;

import java.util.\*;

public class StudentList {

public static void main(String[] args) {

// Check arguments

if(args == null || args.length != 1){

System.out.println("Usage: (a | r | c | +WORD | ?WORD)");

return; //Exit early.

}

String fileContents = readFileContents(Constants.FILE\_NAME);

if (args[0].equals(Constants.ShowAll)) {

showAll(fileContents);

} else if (args[0].equals(Constants.ShowRandom)) {

showRandom(fileContents);

} else if (args[0].contains(Constants.AddEntry)) {

addEntry(Constants.FILE\_NAME,args[0].substring(1));

} else if (args[0].contains(Constants.FindEntry)) {

findEntry(fileContents,args[0].substring(1));

} else if (args[0].contains(Constants.ShowCount)) {

showWordCount(fileContents);

}else{

System.out.println(Constants.InvalidArgument + args[0]);

}

System.out.println(Constants.DataLoadedText);

}

public static void showWordCount(String fileContents) {

String[] words = fileContents.split(Constants.StudentEntryDelimiter);

System.out.println(words.length + Constants.WordsFound);

}

public static void findEntry(String fileContents, String word) {

String words[] = fileContents.split(Constants.StudentEntryDelimiter);

boolean found = false;

for (String w : words) {

if (w.equals(word)) {

System.out.println(Constants.FoundText);

found = true;

break;

}

}

if (!found) {

System.out.println("Word not found.");

}

}

public static void addEntry(String filename, String word) {

writeToFile(filename, new SimpleDateFormat(Constants.DATE\_FORMAT).format(new Date()), word);

}

public static void showRandom(String fileContents) {

String words[] = fileContents.split(Constants.StudentEntryDelimiter);

if (words.length > 0) {

int randomNumber = new Random().nextInt(words.length);

System.out.println(words[randomNumber]);

} else {

System.out.println("No words found.");

}

}

public static void showAll(String fileContents) {

String words[] = fileContents.split(Constants.StudentEntryDelimiter);

for (String word : words) {

System.out.println(word);

}

}

public static String readFileContents(String filename) {

System.out.println(Constants.LoadingDataText);

try {

BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(new FileInputStream(filename)));

String fileContents = bufferedReader.readLine();

bufferedReader.close();

return fileContents;

} catch (Exception e) {

return null;

}

}

public static void writeToFile(String filename, String finalDate, String word) {

try {

BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(filename, true));

bufferedWriter.write(", " + word + "\nList last updated on " + finalDate);

bufferedWriter.close();

} catch (Exception e) {

e.printStackTrace();

}

}

}

//Constants.java

public class Constants {

public static final String ShowAll = "a";

public static final String ShowRandom = "r";

public static final String AddEntry = "+";

public static final String FindEntry = "?";

public static final String ShowCount = "c";

public static final String FILE\_NAME = "students.txt";

public static final String DATE\_FORMAT = "dd/MM/yyyy-hh:mm:ss a";

public static final String StudentEntryDelimiter = ", ";

public static final String LoadingDataText = "Loading data ...";

public static final String DataLoadedText = "Data Loaded.";

public static final String FoundText = "We found it!";

public static final String WordsFound = " word(s) found ";

public static final String InvalidArgument = "Invalid argument: ";

public static final char Space = ' ';

}

**Step -> 10::** Add more comments and makes more naming improvements

import java.io.\*;

import java.text.\*;

import java.util.\*;

public class StudentList {

public static void main(String[] args) {

// Check arguments

if(args == null || args.length != 1){

System.out.println("Usage: (a | r | c | +WORD | ?WORD)");

return; //Exit early.

}

//Process file contents

String fileContents = readFileContents(Constants.FILE\_NAME);

// Process user input

if (args[0].equals(Constants.ShowAll)) {

showAllStudents(fileContents);

} else if (args[0].equals(Constants.ShowRandom)) {

showRandomStudent(fileContents);

} else if (args[0].contains(Constants.AddEntry)) {

addEntryToFile(Constants.FILE\_NAME,args[0].substring(1));

} else if (args[0].contains(Constants.FindEntry)) {

findEntryInFile(fileContents,args[0].substring(1));

} else if (args[0].contains(Constants.ShowCount)) {

showWordCount(fileContents);

}else{

System.out.println(Constants.InvalidArgument + args[0]);

}

System.out.println(Constants.DataLoadedText);

}

// Method to display the word count

public static void showWordCount(String fileContents) {

String[] words = fileContents.split(Constants.StudentEntryDelimiter);

System.out.println(words.length + Constants.WordsFound);

}

// Method to find and display an entry in the file

public static void findEntryInFile(String fileContents, String word) {

String words[] = fileContents.split(Constants.StudentEntryDelimiter);

boolean found = false;

for (String w : words) {

if (w.equals(word)) {

System.out.println(Constants.FoundText);

found = true;

break;

}

}

if (!found) {

System.out.println("Word not found.");

}

}

// Method to add an entry to the file

public static void addEntryToFile(String filename, String word) {

writeToFile(filename, new SimpleDateFormat(Constants.DATE\_FORMAT).format(new Date()), word);

}

// Method to display a random entry from the file

public static void showRandomStudent(String fileContents) {

String words[] = fileContents.split(Constants.StudentEntryDelimiter);

if (words.length > 0) {

int randomNumber = new Random().nextInt(words.length);

System.out.println(words[randomNumber]);

} else {

System.out.println("No words found.");

}

}

// Method to display all entries in the file

public static void showAllStudents(String fileContents) {

String words[] = fileContents.split(Constants.StudentEntryDelimiter);

for (String word : words) {

System.out.println(word);

}

}

// Method to read file contents

public static String readFileContents(String filename) {

System.out.println(Constants.LoadingDataText);

try {

BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(new FileInputStream(filename)));

String fileContents = bufferedReader.readLine();

bufferedReader.close();

return fileContents;

} catch (Exception e) {

return null;

}

}

// Method to write to the file

public static void writeToFile(String filename, String finalDate, String word) {

try {

BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(filename, true));

bufferedWriter.write(", " + word + "\nList last updated on " + finalDate);

bufferedWriter.close();

} catch (Exception e) {

e.printStackTrace();

}

}

}