Lab 1

Lab 1

As you read a text file, you go line by line until you reach the end of the file. What happens if you want to go back to a specific line of text? A better way of doing this is to read a text file into an ArrayList. This way you can easily reference any line from the file.

Before reading the file, create the `path` variable with the file path, and instantiate the ArrayList `text`.

```
//add code below this line
String path = "studentFolder/labs/filesLab1.txt";
ArrayList<String> text = new ArrayList<String>();
//add code above this line
```

Use a `try... catch` block to handle input/output exceptions. In the `try` portion, create a `BufferedReader` object and read through the file. Add each line to the ArrayList `text`. Print any erros in the `catch` portion, and print a message that the file has been read in the `finally` portion.

```
//add code below this line
String path = "studentFolder/labs/filesLab1.txt";
ArrayList<String> text = new ArrayList<String>();
try {
  BufferedReader reader = new BufferedReader(new
    FileReader(path));
  while (reader.ready()) {
    text.add(reader.readLine());
  }
  reader.close();
} catch (IOException e) {
  System.out.println(e);
} finally {
  System.out.println("Finished reading a file.\n");
}
//add code above this line
```

The contents of the text file now reside in the variable `text`. Since the text file is one paragraph (a single newline character at the end of the file), there is only one element in `text`. Go outside of the `try... catch` block and print the first element of the ArrayList `text`.

```java
//add code below this line
String path = "studentFolder/labs/filesLab1.txt";
ArrayList<String> text = new ArrayList<String>();
try {
  BufferedReader reader = new BufferedReader(new
    FileReader(path));
  while (reader.ready()) {
    text.add(reader.readLine());
  }
  reader.close();
} catch (IOException e) {
  System.out.println(e);
} finally {
  System.out.println("Finished reading a file.\n");
}

System.out.println(text.get(0) + "\n");
//add code above this line
```

You should see a passage from Bram Stoker's *Dracula*. You can access the paragraph again by referencing the variable `text`. We can print each word of the text file by splitting the string on the spaces and printing each element of the resulting array.

```java
//add code below this line
String path = "studentFolder/labs/filesLab1.txt";
ArrayList<String> text = new ArrayList<String>();
try {
  BufferedReader reader = new BufferedReader(new
    FileReader(path));
  while (reader.ready()) {
    text.add(reader.readLine());
  }
  reader.close();
} catch (IOException e) {
  System.out.println(e);
} finally {
  System.out.println("Finished reading a file.\n");
}

System.out.println(text.get(0) + "\n");
String[] words = text.get(0).split(" ");
for(String word : words) {
  System.out.println(word);
}
//add code above this line
```

Finally, we can print each sentence from the text file by splitting on the periods. **Note**, that the `split` method uses regular expressions. The `.` has special meaning for regular expressions. Instead use `\\.` as the delimiter, that tells Java to use a period and not a regular expression.

```java
//add code below this line
String path = "studentFolder/labs/filesLab1.txt";
ArrayList<String> text = new ArrayList<String>();
try {
  BufferedReader reader = new BufferedReader(new
    FileReader(path));
  while (reader.ready()) {
    text.add(reader.readLine());
  }
  reader.close();
} catch (IOException e) {
  System.out.println(e);
} finally {
  System.out.println("Finished reading a file.\n");
}

System.out.println(text.get(0) + "\n");
String[] sentences = text.get(0).split("\\.");
for(String sentence : sentences) {
  System.out.println(sentence + "\n");
}
//add code above this line
```

Lab 2

Lab 2

This lab uses a comma delimited CSV file `filesLab2.csv`, which contains integers. There are three columns and four rows. The program below will print the sum for each row in the CSV.

Start by having the variable `path` contain the file path for the CSV file. For now, use a `try` block to create a `CSVReader` object. Print any errors with a `catch` block, and leave a message that the CSV file was read with a `finally` block. Running the code should print the message from the `finally` block.

```java
//add code below this line
String path = "studentFolder/labs/filesLab2.csv";
try {
  CSVReader reader = new CSVReader(new FileReader(path));

  reader.close();
} catch (Exception e) {
  System.out.println(e);
} finally {
  System.out.println("Finished reading a CSV");
}

//add code above this line
```

Iterate over the file with an enhanced loop. For each iteration create the variable `total` and set its value to 0. Use a nested enhanced loop to iterate over each row from the CSV file. Increment `total` with each number from the file. **Remember**, data from the CSV file is read as a string, so type cast the value to an integer. Running the code should print the message from the `finally` block.

```java
//add code below this line
String path = "studentFolder/labs/filesLab2.csv";
try {
  CSVReader reader = new CSVReader(new FileReader(path));
  for (String[] row : reader) {
    int total = 0;
    for (String number : row) {
      total += Integer.parseInt(number);
    }
  }

  reader.close();
} catch (Exception e) {
  System.out.println(e);
} finally {
  System.out.println("Finished reading a CSV");
}

//add code above this line
```

Once the inner loop stops running the variable `total` should have to the sum of the row of integers. Print the value of `total` with a short explanation.

```java
//add code below this line
String path = "studentFolder/labs/filesLab2.csv";
try {
  CSVReader reader = new CSVReader(new FileReader(path));
  for (String[] row : reader) {
    int total = 0;
    for (String number : row) {
      total += Integer.parseInt(number);
    }
    System.out.println("The total value is: " +
    String.valueOf(total));
  }

  reader.close();
} catch (Exception e) {
  System.out.println(e);
} finally {
  System.out.println("Finished reading a CSV");
}

//add code above this line
```

Your program should print the following output:

```
The total value is: 10
The total value is: 151
The total value is: 63
The total value is: 127
```

Lab 3

Lab 3

The following program will ask the user to enter the name of a superhero and then to enter their superpower. These two pieces of information will be written to the CSV file called `superheroes.csv`. When the user enters `stop` for the superhero name, the program will stop running.

Start by creating the variable `path` with the file path for the CSV file. In addition, create a variable to represent a `Scanner` object as you will be entering information through the terminal.

```
//add code below this line
String path = "studentFolder/labs/superheroes.csv";
Scanner sc = new Scanner(System.in);
//add code above this line
```

Next, create a `try... catch... finally` block that creates a `CSVWriter` object that appends to the file, prints any exceptions, and ends with a message that the program wrote to the CSV file. Running the program should print the message from the `finally` block (even though nothing has been written to file).

```
//add code below this line
String path = "studentFolder/labs/superheroes.csv";
Scanner sc = new Scanner(System.in);
try {
  CSVWriter writer = new CSVWriter(new FileWriter(path,
    true));

  writer.close();
} catch (Exception e) {
  System.out.println(e);
} finally {
  System.out.println("Finished writing to a CSV file");
}

//add code above this line
```

The program should run until the user enters `stop`. Use a while loop and prompt the user to enter the superhero's name. Be sure to check if the name is `stop`. If so, break out of the loop. When you run the program, you should see the prompt to enter the name. Add a name and make sure you are prompted again. Enter `stop` and verify that you see the message from the `finally` block.

```java
//add code below this line
String path = "studentFolder/labs/superheroes.csv";
Scanner sc = new Scanner(System.in);
try {
  CSVWriter writer = new CSVWriter(new FileWriter(path,
    true));
  while (true) {
    System.out.println("Enter the superhero's name (type
    'stop' to quit):");
    String name = sc.nextLine();
    if (name.equals("stop")) {
      break;
    }
  }

  writer.close();
} catch (Exception e) {
  System.out.println(e);
} finally {
  System.out.println("Finished writing to a CSV file");
}

//add code above this line
```

Finally, prompt the user for the superhero's power. Combine the name and power into an array of strings and write the array to the CSV file. Run the program and add several superheroes. After entering `stop`, click on the link below to open the CSV file and make sure that the information is written to file. Run it again to verify that additional information is being appending to the file.

```java
//add code below this line
String path = "studentFolder/labs/superheroes.csv";
Scanner sc = new Scanner(System.in);
try {
  CSVWriter writer = new CSVWriter(new FileWriter(path,
    true));
  while (true) {
    System.out.println("Enter the superhero's name (type
    'stop' to quit):");
    String name = sc.nextLine();
    if (name.equals("stop")) {
      break;
    }
    System.out.println("Enter the superhero's power:");
    String power = sc.nextLine();
    String[] row = {name, power};
    writer.writeNext(row);
  }

  writer.close();
} catch (Exception e) {
  System.out.println(e);
} finally {
  System.out.println("Finished writing to a CSV file");
}

//add code above this line
```

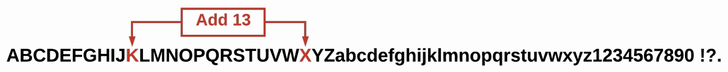
Open CSV File

Lab 4

Lab 4

The lab will cover how to encrypt the contents of a file with a Caesar cipher. This cipher will work with any characters that are letters (uppercase and lowercase), the digits 0 to 9, and the symbols space, exclamation, question mark, and the period. Other symbols will not be encrypted. The Caesar cipher requires a key to work as well. The key is any number from 0 to 25.

The Caesar cipher works by having a list of all of the symbols that can be encrypted. Take the letter you want to encrypt and find its place in the list. Add the value of the key to the position to get the encrypted letter. If the new position is greater than the end of the list, keep counting from the beginning of the list. The example below assumes a key of 13, and shows that a `"K"` becomes `"X"` with the Caesar cipher.


Add 13
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890 !?.

Caesar Cipher

Reading the Source File

This program will read from a file called `source.txt` with the path of `student_folder/labs`. The encrypted text will be written into a file called `encrypted.txt` with the path of `student_folder/text`. Start by making the customary file paths for the two text files.

```
//add code below this line
String readPath = "studentFolder/labs/source.txt";
String writePath = "studentFolder/labs/encrypted.txt";
//add code above this line
```

Now add the `try`, `catch`, and `finally` blocks. Create `BufferedReader` and `BufferedWriter` objects in the `try` block, while printing the appropriate messages in the other blocks. Running the program should print the message about reading and writing to a file.

```java
//add code below this line
String readPath = "studentFolder/labs/source.txt";
String writePath = "studentFolder/labs/encrypted.txt";
try {
  BufferedReader source = new BufferedReader(new
    FileReader(readPath));
  BufferedWriter destination = new BufferedWriter(new
    FileWriter(writePath));


  source.close();
  destination.close();
} catch (Exception e) {
  System.out.println(e);
} finally {
  System.out.println("Finished reading and writing to a
    file");
}
//add code above this line
```

Next, you need set the key (a number from 0 to 25), the cipher mode (encrypt or decrypt), and the list of symbols.

```java
//add code below this line
String readPath = "studentFolder/labs/source.txt";
String writePath = "studentFolder/labs/encrypted.txt";
try {
  BufferedReader source = new BufferedReader(new
    FileReader(readPath));
  BufferedWriter destination = new BufferedWriter(new
    FileWriter(writePath));
  int key = 13;
  String mode = "encrypt";
  final String SYMBOLS =
    "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890
    !?.,-";


  source.close();
  destination.close();
} catch (Exception e) {
  System.out.println(e);
} finally {
  System.out.println("Finished reading and writing to a
    file");
}
//add code above this line
```

▼ **What does `final` mean?**

There is a type of variable called a constant. This variable should never change its value. The `final` keyword turns a variable into a constant. The Java community also uses the convention of writing constants in all caps as well.

Use a while loop to iterate through `source` as long as the end of the file has not yet been reached. Read the next line of the file. The string `newIndex` will represented the index of the encrypted (or decrypted) character, and `newText` will be the encrypted (or decrypted) text.

```java
//add code below this line
String readPath = "studentFolder/labs/source.txt";
String writePath = "studentFolder/labs/encrypted.txt";
try {
  BufferedReader source = new BufferedReader(new
    FileReader(readPath));
  BufferedWriter destination = new BufferedWriter(new
    FileWriter(writePath));
  int key = 13;
  String mode = "encrypt";
  final String SYMBOLS =
    "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890
    !?.,-";

  String newText = "";

  while (source.ready()) {
    String line = source.readLine();
    int newIndex = -1;
    String newText = "";

  }

  source.close();
  destination.close();
} catch (Exception e) {
  System.out.println(e);
} finally {
  System.out.println("Finished reading and writing to a
    file");
}
//add code above this line
```

The Caesar Cipher

The Caesar cipher can only encrypt those characters that are in the `SYMBOLS` variable. Iterate over the string `line`, where `ch` represents each character of the string. Create the variable `charIndex` and set its value to `-1`. This variable will represent if ch is in SYMBOLS. Next, iterate over SYMBOLS. Ask if ch is equal to each character of SYMBOLS. If yes, set the value of charIndex to i which is the index of the character in the string SYMBOLS. To summarize, if charIndex is -1, then the character in line is not found in SYMBOLS. If charIndex is not equal to -1, then the character in line is found in SYMBOLS and charIndex represents the character's index in SYMBOLS. Add the code below after the declaration of newText.

```
    for (char ch : line.toCharArray()) {
      int charIndex = -1;
      for (int i = 0; i < SYMBOLS.length(); i++) {
        if (ch == SYMBOLS.charAt(i)) {
          charIndex = i;
        }
      }

    }
```

You only want to encrypt characters that appear in SYMBOLS, so determine if charIndex is not equal to -1 (the character is in SYMBOLS). Then determine if you are encrypting or decrypting the text. If encrypting, add key to charIndex; if you are decrypting, subtract key from charIndex.

```
    for (char ch : line.toCharArray()) {
      int charIndex = -1;
      for (int i = 0; i < SYMBOLS.length(); i++) {
        if (ch == SYMBOLS.charAt(i)) {
          charIndex = i;
        }
      }
      if (charIndex != -1) {
        if (mode.equals("encrypt")) {
          newIndex = key + charIndex;
        } else if (mode.equals("decrypt")) {
          newIndex = charIndex - key;
        }
      }

    }
```

It is possible that newIndex is less than 0 or greater than the length of SYMBOLS. These indexes will cause a problem. If newIndex is negative, add it to the length of SYMBOLS. If newIndex is greater than the length of SYMBOLS, subtract the length of SYMBOLS.

```
    for (char ch : line.toCharArray()) {
      int charIndex = -1;
      for (int i = 0; i < SYMBOLS.length(); i++) {
        if (ch == SYMBOLS.charAt(i)) {
          charIndex = i;
        }
      }
      if (charIndex != -1) {
        if (mode.equals("encrypt")) {
          newIndex = key + charIndex;
        } else if (mode.equals("decrypt")) {
          newIndex = charIndex - key;
        }
      }
      if (newIndex < 0) {
        newIndex = newIndex + SYMBOLS.length();
      } else if (newIndex > SYMBOLS.length()) {
        newIndex = newIndex - SYMBOLS.length();
      }

    }
```

Now that you have a new index, find the character in `SYMBOLS` with this new index and append it to the variable `newText`. After iterating through the entire line, write `newText` to file `destination`. This program reads up until a newline character, but it does not write a newline character to the destination file. Use the `newLine` method to add a newline character to the destination file.

```
        for (char ch : line.toCharArray()) {
          int charIndex = -1;
          for (int i = 0; i < SYMBOLS.length(); i++) {
            if (ch == SYMBOLS.charAt(i)) {
              charIndex = i;
            }
          }
          if (charIndex != -1) {
            if (mode.equals("encrypt")) {
              newIndex = key + charIndex;
            } else if (mode.equals("decrypt")) {
              newIndex = charIndex - key;
            }
          }
          if (newIndex < 0) {
            newIndex = newIndex + SYMBOLS.length();
          } else if (newIndex > SYMBOLS.length()) {
            newIndex = newIndex - SYMBOLS.length();
          }
          newText += SYMBOLS.charAt(newIndex);
        }
        destination.write(newText);
        destination.newLine();
```

Open Encrypted File

▼ **Code**

```
import java.io.*;

public class Lab4 {
  public static void main(String args[]) {

    //add code below this line
    String readPath = "studentFolder/labs/source.txt";
    String writePath = "studentFolder/labs/encrypted.txt";
    try {
      BufferedReader source = new BufferedReader(new
      FileReader(readPath));
      BufferedWriter destination = new BufferedWriter(new
      FileWriter(writePath));
      int key = 13;
      String mode = "encrypt";
      final String SYMBOLS =
      "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890
      !?.,-";

      while (source.ready()) {
        String line = source.readLine();
        int newIndex = -1;
        String newText = "";
        for (char ch : line.toCharArray()) {
          int charIndex = -1;
          for (int i = 0; i < SYMBOLS.length(); i++) {
            if (ch == SYMBOLS.charAt(i)) {
              charIndex = i;
            }
          }
          if (charIndex != -1) {
            if (mode.equals("encrypt")) {
              newIndex = key + charIndex;
            } else if (mode.equals("decrypt")) {
              newIndex = charIndex - key;
            }
          }
          if (newIndex < 0) {
            newIndex = newIndex + SYMBOLS.length();
          } else if (newIndex > SYMBOLS.length()) {
            newIndex = newIndex - SYMBOLS.length();
          }
          newText += SYMBOLS.charAt(newIndex);
        }
        destination.write(newText);
        destination.newLine();
      }

      source.close();
      destination.close();
    } catch (Exception e) {
      System.out.println(e);
    } finally {
      System.out.println("Finished reading and writing to a
      file");
    }
    //add code above this line
  }
}
```

Decrypting the File

To decrypt the file, a few changes need to be made to your code. The source file should be the encrypted file, and the destination file will be the file `decrypted.txt`.

```
    //add code below this line
    String readPath = "studentFolder/labs/encrypted.txt";
    String writePath = "studentFolder/labs/decrypted.txt";
```

Finally, change the mode of the cipher to `"decrypt"`.

```
        String mode = "decrypt";
```

Open Decrypted File

▼ **Source Text**

The original text for this lab is the opening paragraph from L. Frank Baum’s *The Wizard of Oz*.

Lab Challenge

Lab Challenge

Write a program that reads the text file. This file is stored in the variable `path`. **Do not alter this line of code.** The file contains several instances of the word `Burma`. Replace each instance of `Burma` with `Myanmar`, and print the results of this transformation. The final output of your program should be:

```
Myanmar is a country in Southeast Asia.
The capital of Myanmar is Naypyidaw.
Its population is about 54 million people.
Myanmar is a former British colony.
```

▼ **Where is the code visualizer?**

Unfortunately, the code visualizer does not work with external files, so it cannot be used for this problem.