Creating a Java program in Notepad step-by-step guide:

- Open Notepad: You can open Notepad by searching for it in the Start menu (on Windows) or by typing notepad in the command line interface.
- Write Your Java Code: In Notepad, write your Java code. For example, a simple "Hello, World!" program looks like this:

```
public class HelloWorld {

   public static void main(String[] args) {

      System.out.println("Hello, World!");

   }

}
```

- Save the File: After writing your Java code, save the file with a .java extension. For example, you can save it as HelloWorld.java.
- Compile the Java Program: Open a command prompt (cmd) and navigate to the directory where you saved your Java file. Then, compile your Java program using the javac command followed by the name of your Java file. For example:

  javac HelloWorld.java

  If there are no syntax errors in your code, this will create a HelloWorld.class file in the same directory.

- Run the Java Program: After successfully compiling your Java program, you can run it using the java command followed by the name of the class containing the main method (without the .class extension). For example:
  java HelloWorld

  This will execute your Java program, and you should see the output printed to the console.

Command Line Arguments

simple Java program that takes command line arguments and prints them:

```
public class CommandLineArgs {
    public static void main(String[] args) {
        // Check if any arguments are passed
        if (args.length == 0) {
            System.out.println("No arguments provided.");
        } else {
            System.out.println("Arguments passed:");
            // Loop through each argument and print it
            for (int i = 0; i < args.length; i++) {
                System.out.println("Argument " + (i+1) + ": " + args[i]);
            }
        }
    }
}
```

Save this code in a file named CommandLineArgs.java.

To compile this program, open a command prompt, navigate to the directory where CommandLineArgs.java is saved, and run:

javac CommandLineArgs.java

Then, to run the compiled program with command line arguments, use:

java CommandLineArgs arg1 arg2 arg3

Replace arg1, arg2, arg3, etc., with the arguments you want to pass. The program will print each argument you provided. If no arguments are provided, it will print "No arguments provided."

Only when File-handling topic is done

A simple file copying program using Command line Args. This program will take two command line arguments: the path of the source file to be copied and the path of the destination where the file will be copied to.

```
import java.io.*;

public class FileCopy {
   public static void main(String[] args) {
      // Check if the correct number of arguments is provided
      if (args.length != 2) {
         System.out.println("Usage: java FileCopy <source-file> <destination-file>");
         return;
      }

      String sourceFile = args[0];
      String destinationFile = args[1];

      // Try to copy the file
      try {
         FileInputStream fis = new FileInputStream(sourceFile);
         FileOutputStream fos = new FileOutputStream(destinationFile);
         byte[] buffer = new byte[1024];
         int bytesRead;

         // Read from the source file and write to the destination file
         while ((bytesRead = fis.read(buffer)) != -1) {
            fos.write(buffer, 0, bytesRead);
         }

         // Close streams
         fis.close();
         fos.close();

         System.out.println("File copied successfully.");
      } catch (IOException e) {
         System.out.println("An error occurred while copying the file: " + e.getMessage());
      }
   }
}
```

To use this program:

Save the code above into a file named FileCopy.java.
Compile the program by running javac FileCopy.java.
Run the program with the source and destination file paths as command line arguments:
java FileCopy source.txt destination.txt