Java Input/Output (I/O) is the process of reading data from or writing data to external sources, such as files, console, or network sockets. Java provides a rich set of classes and methods for performing I/O operations. Here, I'll explain the basics of Java I/O with two code examples: one for reading input from the console and another for reading and writing data to a file.

Example 1: Reading Input from Console

In this example, we'll use Java's `Scanner` class to read input from the console.

```java
import java.util.Scanner;

public class ConsoleInputExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter your name: ");
        String name = scanner.nextLine();

        System.out.print("Enter your age: ");
        int age = scanner.nextInt();

        System.out.println("Hello, " + name + "! You are " + age + " years old.");

        // Don't forget to close the scanner to release resources.
        scanner.close();
    }
}
```

In this code:

- We import the `java.util.Scanner` class to handle console input.
- We create a `Scanner` object to read from the `System.in` input stream, which represents the console.
- We use `nextLine()` and `nextInt()` methods to read a line of text and an integer from the console, respectively.
- Finally, we print out the collected input.

Example 2: Reading and Writing to a File

In this example, we'll demonstrate how to read data from one file and write it to another file.

```java
```

```
import java.io.*;

public class FileIOExample {
    public static void main(String[] args) {
        try {
            // Reading from a file
            BufferedReader reader = new BufferedReader(new FileReader("input.txt"));
            String line;

            // Writing to a file
            BufferedWriter writer = new BufferedWriter(new FileWriter("output.txt"));

            while ((line = reader.readLine()) != null) {
                // Process the data (in this case, just writing to another file)
                writer.write(line);
                writer.newLine(); // Add a newline character
            }

            // Close the reader and writer to release resources
            reader.close();
            writer.close();

            System.out.println("File copied successfully!");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

In this code:

- We use `BufferedReader` and `FileReader` to read data from the "input.txt" file and `BufferedWriter` and `FileWriter` to write data to the "output.txt" file.
- We read data line by line from the input file and write it to the output file.
- The `newLine()` method is used to add a newline character after each line in the output file.
- Finally, we close the reader and writer to release resources.

Make sure to replace "input.txt" and "output.txt" with the actual file paths you want to use.