

# Practical 2 Instructions

Jean-Pierre du Plessis

August 14, 2018

**Due Date: August 27, 2018 at  
midnight.**

## 1 Introduction

In this practical we will make use of the strategy and decorator pattern to create re-usable components that can be customised. Before proceeding with the practical, be sure to correct all errors from the previous practical.

## 2 Program specifications

Modify your application from the previous practical and make use of the provided library to implement the following:

1. Class implementations that adapt the provided components to re-usable **ITerminalWindow** or **IInputWindow** components.
2. An **ITextWriter** implementation that can write text in colour.
3. Various borders for the components.

## **3 Terminal Window Sections**

Our final application will consist of three types of sections. The purpose of each section is described in the sections below. Keep these sections in mind when building your reusable components. You should ensure that each section be be resized and repositioned if necessary. Each section is described below.

### **3.1 Title Section**

This part of the terminal should be located at the top of the terminal. For the purpose of this practical, it should display your name and student number.

### **3.2 Editor Section**

This part of the terminal will be used to input text and modify the contents of a file. You only need to worry about displaying text at this point. We will implement the input functionality at a later stage.

### **3.3 Command Section**

This part of the terminal will allow the user to run commands or to view information about the document, such as the current line/column of the cursor and so on. For this practical, you need only place some placeholder text.

The command sections's text should be in colour.

### **3.4 Implementation**

Make use of the provided component library along with the adapter, strategy and decorator pattern to build re-usable components that can then be used to construct the user interface. For example, it should be possible in theory to add multiple editor sections to the terminal window (figure below).

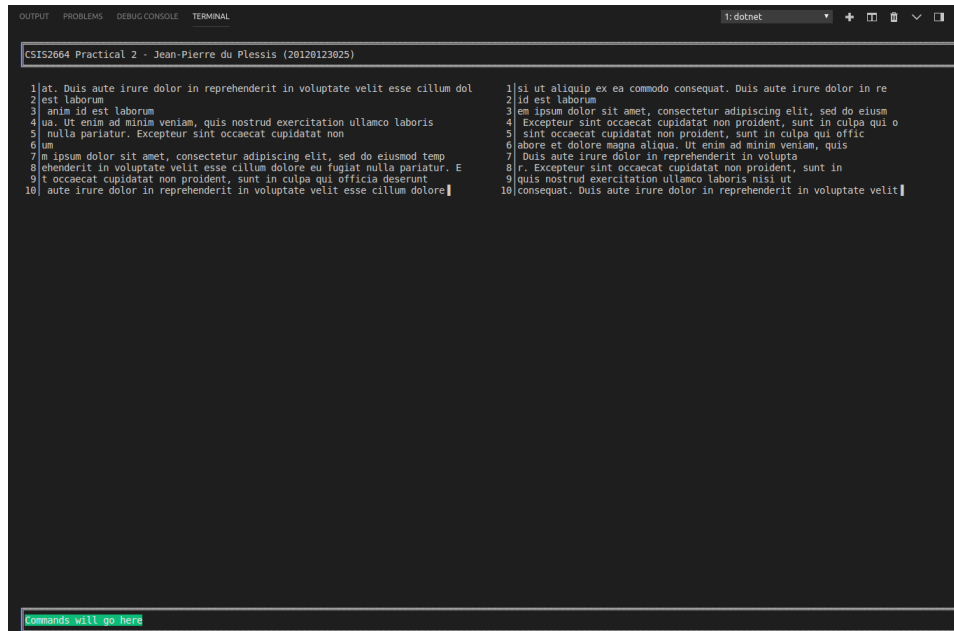


Figure 1: Application with two editing sections.

## 4 Borders for components

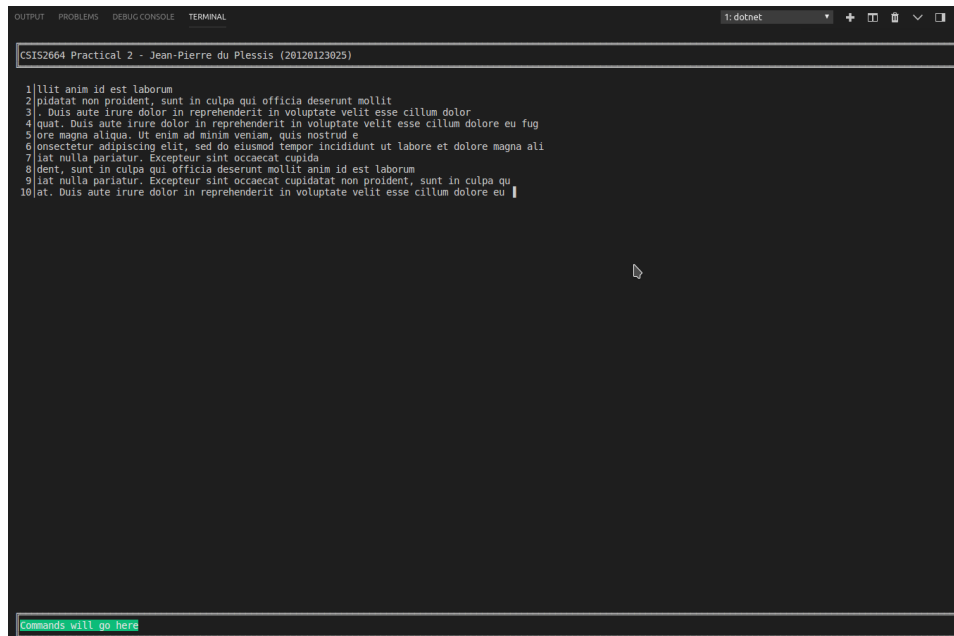
It should also be possible to add a border to **ITerminalWindow** objects. You should implement your borders in such a way that similar borders can be reused and borders can be joined if necessary. Note that a border will not always surround the entire component. For example, I may choose to add a border to only the top and bottom of the title section. Your design should make it easy to customize how a component's borders are displayed without resorting to long if-else statements in each component. Utilise the decorator and strategy pattern in your implementation for this.

**Note:** While your solution should provide a mechanism through which it is possible to customise the border, you may construct all borders using four sides for the moment.

## 5 Coloured text writer

Create a coloured text writer using the decorator pattern. This writer will write out text using a user-specified background and/or foreground colour. Make use of this text writer in your command section.

Demonstrate that your program is working by building a user interface using the components you created as part of this practical to resemble to image below. You should also add text to the editor section to test that the line numbers are working.



```
CSIS2664 Practical 2 - Jean-Pierre du Plessis (20120123025)

1|llit anim id est laborum
2|ddidat non proident, sunt in culpa qui officia deserunt mollit
3| Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolor
4|quat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fug
5|ore magna aliqua. Ut enim ad minim veniam, quis nostrud e
6|onsectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna ali
7|lat nulla pariatur. Excepteur sint occaecat cupida
8|dent, sunt in culpa qui officia deserunt mollit anim id est laborum
9|lat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui
10|at. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu |

Commands will go here
```

Figure 2: Sample output of this practical.

**NB: Code that does not compile or run will not be marked.**

## 6 UML Diagram

Create a complete UML class diagram of your solution using the appropriate notation. This diagram should be exported to a PDF document and handed in along with your coded solution.

<b>NB: Diagrams not in the PDF format will not be marked.</b>
---

## 7 Bonus

Make use of the Template pattern along with generics to minimise the amount of code you need to write to adapt the provided components to the **IInputWindow** interface in the case of the editor and command sections.

## 8 Submission

Create a zip (or 7z) file of your application and UML diagram. Name the zip file Practical2\_YourStudentNumber.zip. Upload the zip file to the Practical 2 submission link on Blackboard. If you are working with someone (in a pair) be sure to include both your student details in the UML diagram.