|  |  |
| --- | --- |
| **Laboratory Activity No. 8** | |
| **Converting TUI to GUI Programs** | |
| **Course Code:** CPE103 | **Program:** BSCPE |
| **Course Title:** Object-Oriented Programming | **Date Performed:** March 15, 2025 |
| **Section:** 1 - A | **Date Submitted:** March 15, 2025 |
| **Name:** Hermosura, Leigh B. | **Instructor:** Maria Rizette M. Sayo |
| **1. Objective(s):** | |
| This activity aims to convert a TUI program to GUI program with the Pycharm framework | |
| **2. Intended Learning Outcomes (ILOs):** | |
| The students should be able to:   * 1. Identify the main components in a GUI Application   2. Create a simple GUI Application that converts TUI program to GUI program | |
| **3. Discussion:** | |
| In general, programs consist of three components—input, processing, and output. In TUI programs, input is usually obtained from an input statement or by importing data from a file. Output is usually given by a print statement or stored in a file. When we convert a TUI program to a GUI program, we replace input and print statements with Label/Entry pairs. Processing data and inputting and outputting data to files works much the same in both types of programs. The primary difference is that the processing in GUI programs is usually  triggered by an event | |
| **4. Materials and Equipment:** | |
| Desktop Computer with Anaconda Python or Pycharm Windows Operating System | |
| **5. Procedure:** | |

1. Type these codes in Pycharm: #TUI Form

def main():

# Find the largest number among three numbers L = []

num1 = eval(input("Enter the first number:")) L.append(num1)

num2 = eval(input("Enter the second number:")) L.append(num2)

num3 = eval(input("Enter the third number:")) L.append(num3)

print("The largest number among the three is:",str(max(L))) main()

|  |
| --- |
| 2. Run the program and observe the output.    Figure 1. TUI form    Figure 1(a) TUI form with three input numbers    Figure 1(b) TUI form with output “The largest number among the three”  Method 1 above shows a TUI program and a possible output in Figures 1(a) and (b) while Figure 2 shows the output of the GUI program in Method 2. |

|  |
| --- |
| **5. Procedure:** |
| Method 2  from tkinter import \*  window = Tk()  window.title("Find the largest number") window.geometry("400x300+20+10")  def findLargest(): L = []  L.append(eval(conOfent2.get())) L.append(eval(conOfent3.get())) L.append(eval(conOfent4.get())) conOfLargest.set(max(L))  lbl1 = Label(window, text = "The Program that Finds the Largest Number") lbl1.grid(row=0, column=1, columnspan=2,sticky=EW)  lbl2 = Label(window,text = "Enter the first number:") lbl2.grid(row=1, column = 0,sticky=W)  conOfent2 = StringVar()  ent2 = Entry(window,bd=3,textvariable=conOfent2) ent2.grid(row=1, column = 1)  lbl3 = Label(window,text = "Enter the second number:") lbl3.grid(row=2, column=0)  conOfent3=StringVar()  ent3 = Entry(window,bd=3,textvariable=conOfent3) ent3.grid(row=2,column=1)  lbl4 = Label(window,text="Enter the third number:") lbl4.grid(row=3,column =0, sticky=W)  conOfent4 = StringVar()  ent4 = Entry(window,bd=3,textvariable=conOfent4) ent4.grid(row=3, column=1) |

|  |
| --- |
| btn1 = Button(window,text = "Find the largest no.",command=findLargest) btn1.grid(row=4, column = 1)  lbl5 = Label(window,text="The largest number:") lbl5.grid(row=5,column=0,sticky=W) conOfLargest = StringVar()  ent5 = Entry(window,bd=3,state="readonly",textvariable=conOfLargest) ent5.grid(row=5,column=1)  mainloop()  Results 2    Figure 2. GUI program to find the largest number |

|  |
| --- |
| **Questions**   1. What is TUI in Python?   TUI stands for Text User Interface. It is a form of user interface that is text based which uses the keyboard for navigating and typing commands rather than using the mouse.   1. How to make a TUI in Python?   The main function that allows user interface is “input()” in which the user must input a value to be stored by the program. This information can then be manipulated by the programmer to perform specific tasks.   1. What is the difference between TUI and GUI?   The main difference between the two is that GUI or Graphic User Interface enables the user to utilize their mouse or touch screen controls to navigate and generate input. |
| **6. Supplementary Activity:** |
| TUI Implementation  # Simple TUI Calculator  def add(a, b): return a + b  def subtract(a, b): return a - b  def multiply(a, b): return a \* b  def divide(a, b): if b != 0:  return a / b else:  return "Error! Division by zero."  def main():  print("Simple Calculator") print("Options:")  print("1. Add") print("2. Subtract") print("3. Multiply") print("4. Divide")  choice = input("Select operation (1/2/3/4): ")  num1 = float(input("Enter first number: ")) num2 = float(input("Enter second number: "))  if choice == '1':  print(f"{num1} + {num2} = {add(num1, num2)}") elif choice == '2':  print(f"{num1} - {num2} = {subtract(num1, num2)}") elif choice == '3':  print(f"{num1} \* {num2} = {multiply(num1, num2)}") elif choice == '4': |

|  |
| --- |
| print(f"{num1} / {num2} = {divide(num1, num2)}") else:  print("Invalid input.")  if name == " main ": main()  GUI Conversion of the Calculator:  import tkinter as tk  # Functions for calculation def add():  result.set(float(entry1.get()) + float(entry2.get()))  def subtract():  result.set(float(entry1.get()) - float(entry2.get()))  def multiply():  result.set(float(entry1.get()) \* float(entry2.get()))  def divide(): try:  result.set(float(entry1.get()) / float(entry2.get())) except ZeroDivisionError:  result.set("Error! Division by zero.")  # Create the main window root = tk.Tk() root.title("Simple Calculator")  # Create StringVar to hold the result result = tk.StringVar()  # Create the layout  tk.Label(root, text="Enter first number:").grid(row=0, column=0) entry1 = tk.Entry(root)  entry1.grid(row=0, column=1)  tk.Label(root, text="Enter second number:").grid(row=1, column=0) entry2 = tk.Entry(root)  entry2.grid(row=1, column=1)  # Buttons for operations  tk.Button(root, text="Add", command=add).grid(row=2, column=0) tk.Button(root, text="Subtract", command=subtract).grid(row=2, column=1) tk.Button(root, text="Multiply", command=multiply).grid(row=3, column=0) tk.Button(root, text="Divide", command=divide).grid(row=3, column=1)  # Label to show result  tk.Label(root, text="Result:").grid(row=4, column=0) result\_label = tk.Label(root, textvariable=result) result\_label.grid(row=4, column=1)  # Start the main loop root.mainloop() |

|  |
| --- |
| Once you've successfully created the GUI version of the calculator, try adding the following features to enhance the program:   1. **Clear Button**: Add a button to clear the input fields and reset the result. 2. **History Feature**: Add a list or label to show the history of operations performed. 3. **Advanced Operations**: Implement additional operations such as square roots, powers, or trigonometric functions. 4. **Input Validation**: Add validation to ensure that the user only enters numeric values in the input fields. 5. **Styling**: Experiment with different styles (font sizes, button colors) to improve the appearance of the GUI.   For the supplementary activity, please refer to this link <https://github.com/Leigh-Hermosura/CPE-103-OOP-1-A/tree/main/Laboratory-No.8/Supplementary%20Activity> |
| **6. Conclusion** |
| TUI (Text User Interface) relies on text-based commands and keyboard inputs for navigation and interaction, whereas GUI (Graphical User Interface) allows for more intuitive interaction through graphical elements like icons, buttons, and touch screen controls. The input() function plays a crucial role in TUIs by allowing users to provide text-based input that the program can process. While TUIs are more focused on text and command-driven interactions, GUIs provide a more visually interactive experience, utilizing pointing devices like mice or touch screens. Both have their advantages, with TUIs being efficient for more technical tasks and GUIs offering a more user-friendly and accessible interface. |