Laboratory Activity No. 7 - Converting TUI to GUI Programs	
Course Code: OOP 009B	**Program:** Computer Engineering
Course Title: Object Oriented Programming 2	**Date Performed:** November 11, 2024
Section: CPE21S4	**Date Submitted:** November 11, 2024
Group Members: Virtucio, Dominic Joseph P. Bonifacio, Nyko Adrein L. Magistrado, Aira Pauleen M. Planta, Calvin Earl L. Solis, Paul Vincent M.	**Instructor:** Ma'am Maria Rizette Sayo

Procedure

1. **TUI Form - Method 1**

```python
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()
```

Figure 1. TUI form

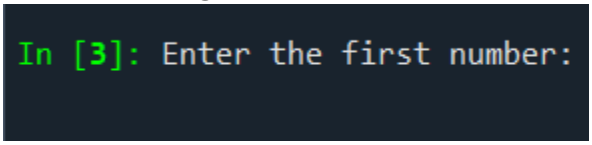


Figure 1(a) TUI form with three input numbers

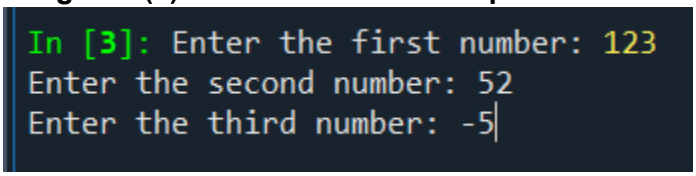


Figure 1(b) TUI form with output “The largest number among the three

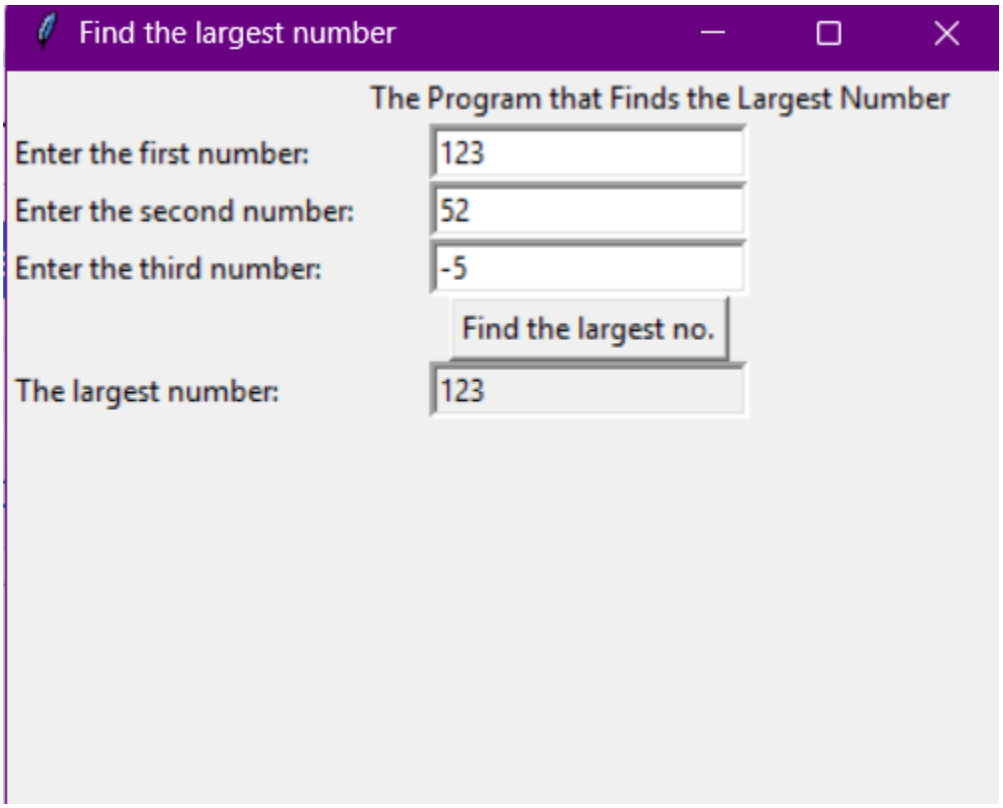
```
Enter the first number: 123
Enter the second number: 52
Enter the third number: -5
The largest number among the three is: 123

Process finished with exit code 0
```

2. Method 2 (tkinter)

Python

```
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)

window.mainloop()
```



Questions

1. **What is TUI in Python?**

- A text User Interface, or TUI, is related to a basic text-based text-based app in which you enter commands to interact. Compared to GUI, which has buttons and images, it is more simple. Imagine it as a vintage computer game that you play by simply typing.

2. **How to make a TUI in Python?**

- You can use python libraries like urwid or curses to create a TUI. With the help of these libraries, you can manipulate the terminal window and show text in various colors and locations. It's similar to building blocks, only that text is used in place of bricks.

3. **What is the difference between TUI and GUI?**

- The way you interact with a TUI and GUI is the biggest difference between both. In contrast to a GUI, which functions more like a touch screen phone, a TUI is similar to speaking to a robot using

only words. For certain jobs, TUIs can be quicker and more effective than GUIs, which are typically easier to use.
Supplementary Activity

TUI Implementation

Python

```python
# 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()
```

Output:

```
Simple Calculator
Options:
1. Add
2. Subtract
3. Multiply
4. Divide
Select operation (1/2/3/4): 1
Enter first number: 23
Enter second number: 34
23.0 + 34.0 = 57.0
```

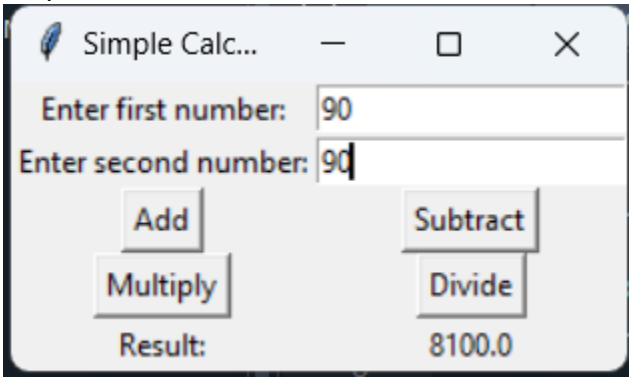
GUI Conversion of the Calculator:

C/C++

```
import tkinter as tk

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.")

root = tk.Tk()
root.title("Simple Calculator")

result = tk.StringVar()

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)

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)
```

```
tk.Label(root, text="Result:").grid(row=4, column=0)
result_label = tk.Label(root, textvariable=result)
result_label.grid(row=4, column=1)

root.mainloop()
```

Output:



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.

Python

```
from tkinter import *
from tkinter import messagebox
import math

class MyWindow:
    def __init__(self, win):
        self.Label1 = Label(win, bg="Orange", fg="Black", text="Calculator",
font=("Impact", 35))
        self.Label1.place(x=100, y=10)

        self.Label2 = Label(win, bg="Orange", fg="Black", text="Number 1:",
font=("Impact", 18))
        self.Label2.place(x=75, y=90)
        self.Entry1 = Entry(win, bd=8)
```

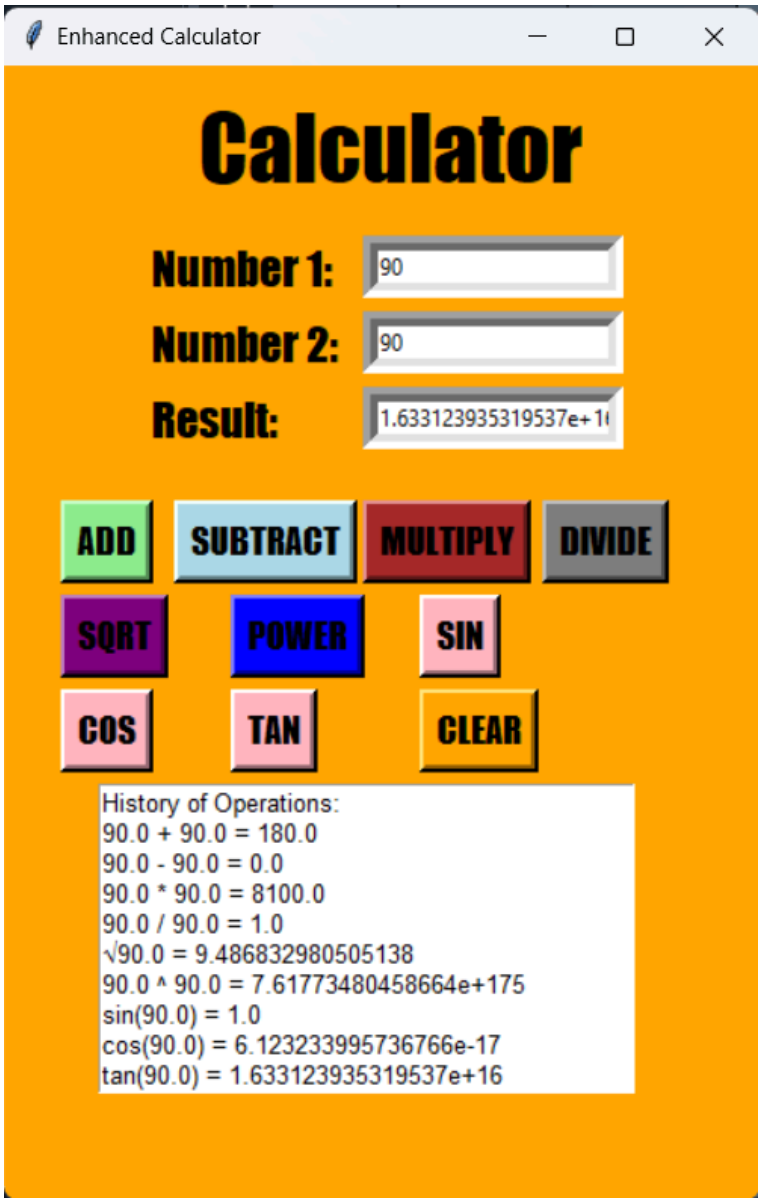
```
        self.Entry1.place(x=190, y=90)

        self.Label3 = Label(win, bg="Orange", fg="Black", text="Number 2:",
font=("Impact", 18))
        self.Label3.place(x=75, y=130)
        self.Entry2 = Entry(win, bd=8)
        self.Entry2.place(x=190, y=130)

        self.Label4 = Label(win, bg="Orange", fg="Black", text="Result:",
font=("Impact", 18))
        self.Label4.place(x=75, y=170)
        self.Entry3 = Entry(win, bd=8)
        self.Entry3.place(x=190, y=170)

        self.Button1 = Button(win, bd=4, bg="LightGreen", fg="Black",
text="ADD", font=("Impact", 14), command=self.add)
        self.Button1.place(x=30, y=230)
        self.Button2 = Button(win, bd=4, bg="LightBlue", fg="Black",
text="SUBTRACT", font=("Impact", 14), command=self.sub)
        self.Button2.place(x=90, y=230)
        self.Button3 = Button(win, bd=4, bg="Brown", fg="Black",
text="MULTIPLY", font=("Impact", 14), command=self.multiply)
        self.Button3.place(x=190, y=230)
        self.Button4 = Button(win, bd=4, bg="Gray", fg="Black", text="DIVIDE",
font=("Impact", 14), command=self.divide)
        self.Button4.place(x=285, y=230)

        self.Button5 = Button(win, bd=4, bg="Purple", fg="Black", text="SQRT",
font=("Impact", 14), command=self.sqrt)
        self.Button5.place(x=30, y=280)
        self.Button6 = Button(win, bd=4, bg="Blue", fg="Black", text="POWER",
font=("Impact", 14), command=self.power)
        self.Button6.place(x=120, y=280)

        # Buttons for trigonometric functions
        self.ButtonSin = Button(win, bd=4, bg="LightPink", fg="Black",
text="SIN", font=("Impact", 14), command=self.sine)
        self.ButtonSin.place(x=220, y=280)
        self.ButtonCos = Button(win, bd=4, bg="LightPink", fg="Black",
text="COS", font=("Impact", 14), command=self.cosine)
        self.ButtonCos.place(x=30, y=330)
        self.ButtonTan = Button(win, bd=4, bg="LightPink", fg="Black",
text="TAN", font=("Impact", 14), command=self.tangent)
        self.ButtonTan.place(x=120, y=330)

        self.ClearButton = Button(win, bd=4, bg="Orange", fg="Black",
text="CLEAR", font=("Impact", 14), command=self.clear)
        self.ClearButton.place(x=220, y=330)

        self.history_text = Text(win, height=10, width=40, font=("Arial", 10))
```

```
        self.history_text.place(x=50, y=380)
        self.history_text.insert(END, "History of Operations:\n")

    def validate_input(self, entry):
        try:
            return float(entry.get())
        except ValueError:
            messagebox.showerror("Invalid input", "Please enter valid numbers.")
            return None

    def add(self):
        num1 = self.validate_input(self.Entry1)
        num2 = self.validate_input(self.Entry2)
        if num1 is not None and num2 is not None:
            result = num1 + num2
            self.Entry3.delete(0, 'end')
            self.Entry3.insert(END, str(result))
            self.add_to_history(f"{num1} + {num2} = {result}")

    def sub(self):
        num1 = self.validate_input(self.Entry1)
        num2 = self.validate_input(self.Entry2)
        if num1 is not None and num2 is not None:
            result = num1 - num2
            self.Entry3.delete(0, 'end')
            self.Entry3.insert(END, str(result))
            self.add_to_history(f"{num1} - {num2} = {result}")

    def multiply(self):
        num1 = self.validate_input(self.Entry1)
        num2 = self.validate_input(self.Entry2)
        if num1 is not None and num2 is not None:
            result = num1 * num2
            self.Entry3.delete(0, 'end')
            self.Entry3.insert(END, str(result))
            self.add_to_history(f"{num1} * {num2} = {result}")

    def divide(self):
        num1 = self.validate_input(self.Entry1)
        num2 = self.validate_input(self.Entry2)
        if num1 is not None and num2 is not None:
            if num2 == 0:
                self.Entry3.insert(END, "Error! Division by zero.")
                self.add_to_history("Error! Division by zero.")
            else:
                result = num1 / num2
                self.Entry3.delete(0, 'end')
                self.Entry3.insert(END, str(result))
                self.add_to_history(f"{num1} / {num2} = {result}")
```

```
    def sqrt(self):
        num1 = self.validate_input(self.Entry1)
        if num1 is not None:
            result = math.sqrt(num1)
            self.Entry3.delete(0, 'end')
            self.Entry3.insert(END, str(result))
            self.add_to_history(f"√{num1} = {result}")

    def power(self):
        num1 = self.validate_input(self.Entry1)
        num2 = self.validate_input(self.Entry2)
        if num1 is not None and num2 is not None:
            result = num1 ** num2
            self.Entry3.delete(0, 'end')
            self.Entry3.insert(END, str(result))
            self.add_to_history(f"{num1} ^ {num2} = {result}")

    def sine(self):
        num1 = self.validate_input(self.Entry1)
        if num1 is not None:
            result = math.sin(math.radians(num1))
            self.Entry3.delete(0, 'end')
            self.Entry3.insert(END, str(result))
            self.add_to_history(f"sin({num1}) = {result}")

    def cosine(self):
        num1 = self.validate_input(self.Entry1)
        if num1 is not None:
            result = math.cos(math.radians(num1))
            self.Entry3.delete(0, 'end')
            self.Entry3.insert(END, str(result))
            self.add_to_history(f"cos({num1}) = {result}")

    def tangent(self):
        num1 = self.validate_input(self.Entry1)
        if num1 is not None:
            result = math.tan(math.radians(num1))
            self.Entry3.delete(0, 'end')
            self.Entry3.insert(END, str(result))
            self.add_to_history(f"tan({num1}) = {result}")

    def clear(self):
        self.Entry1.delete(0, 'end')
        self.Entry2.delete(0, 'end')
        self.Entry3.delete(0, 'end')
        self.history_text.delete("2.0", END)

    def add_to_history(self, operation):
        self.history_text.insert(END, operation + "\n")
```

```
window = Tk()
MyWindow = MyWindow(window)

window.geometry("400x600")
window.title("Enhanced Calculator")
window.configure(bg='Orange')

window.mainloop()
```

OUTPUT:



Conclusion

In this lab activity, we successfully converted a TUI program into a GUI program. We discovered that the primary distinction between the two is that, in terms of visual representations, TUI programs can be displayed in a text-based format, whereas GUI programs have visuals that include labels, entries, and buttons. Method 1 taught us about the main differences between them, specifically how they handle input, processing, and output.

Method 2 entails creating a simple GUI for Method 1. Both methods serve the same purpose of determining the largest number based on user input, but the main difference is that Method 2 incorporates visual elements, which improve user interaction and make the program more interactive. This section made us realize that GUI is very helpful in the overall user experience because it allows for more organized presentation of the information.

In the supplementary activity, we optimized a basic calculator app with Python and Tkinter. To that we included a Clear button, a history log where people may track their calculations and functions such as square root and power. We also made the program to conduct a validity check on the numbers entered before performing and operations to avoid this. Moreover, with colors and better layout, they replaced the design to look more friendly and easier to navigate. All in all the enhancements done on the calculator made the calculator more useful and easy to use.