EGR680 High Level Implementation on FPGA

Laboratory 08

Graphical User Interface (GUI) in Python

Professor: Dr. C. Parikh

Student: Dimitri Häring

November 1, 2018

Contents

List of Tables

List of Figures

Listings

1 Introduction

The goal of laboratory eight is to familiarize the student with tkinter, and turtle to build a graphical user interface (GUI) in python.

2 Design

In this section the design and decisions that where made to achieve the laboratory are discussed.

2.1 Tkinter

According to wiki.python.org Tkinter is Python's de-facto standard GUI (Graphical User Interface) package. It is a thin object-oriented layer on top of Tcl/Tk.

Tkinter is not the only GuiProgramming toolkit for Python. It is however the most commonly used one. CameronLaird calls the yearly decision to keep TkInter "one of the minor traditions of the Python world."

Tkinter provides a canvas to draw in it and place widgets that allows simple interaction with the user. Widgets are Buttons, Labels, Entrys, and Frams to name some of the most commonly used ones. Important of tinker is to understand that it provides three layout managing systems as the are pack, grid and place. In the lab pack was used because it is the most discussed and if it is properly understood most likely the most power full among the three. Pack does group objects in containers as example the main window, a canvas or a frame. now the trick is that frames can be cascade and and placed in order which allows the organization of objects most commonly widgets. Figure 1 shows how the main window which is the window him self with title encloses the canvas. Then packed on top is a turtle animation followed by the control frame. In the control frame is on top a welcome frame that packes a welcome text or the pin entry wiggeds. Followed by three frames left center and right that show labels, input output, and commands. And the last one is a frame packed to the bottom of the control frame.

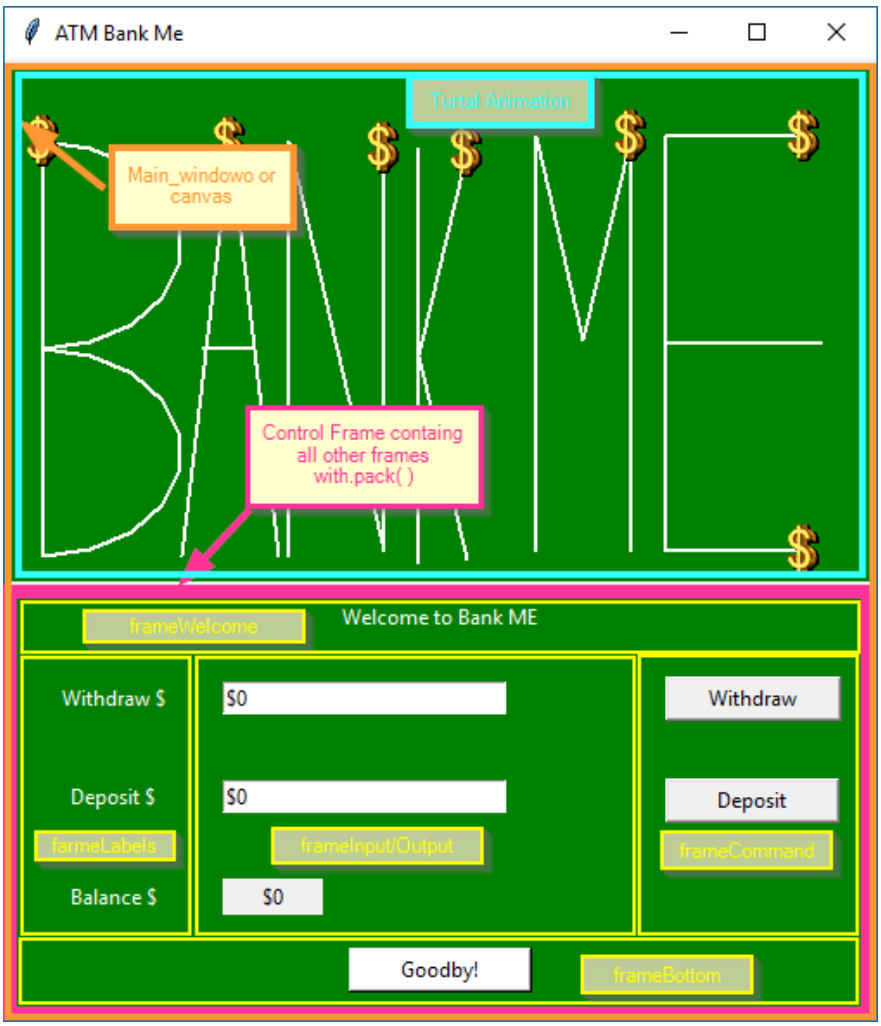

Figure 1: Frames arranged with pack.

An example code example for a empty tkinter window is shown in Listing 1.

```python
# import module
import tkinter as tk

# define class with an empty main window
my_class(tk.Frame)
    def __init__(self, master=None):
    self.main_window = master
    self.main_window.title("ATM Bank Me")

# instanciate class
root = tk.Tk()
class_instance = my_class(root)
class_instance.main_window.mainloop()
```

Listing 1: Python tkinter example code to create an empty window.

An example code example for a canvas is shown in Listing 2.

```python
  #Make canvaas child of root
  canvas = tk.Canvas(master = main_window, width = 500, height = 300, bg = 'green')
  canvas.pack(side = 'top', fill = 'both', expand = 'yes')
```

Listing 2: Python tkinter example code to create a canvas.

An example code example for a frame is shown in Listing 3.

```python
  # A frame is an invisible widget that holds other widgets. This frame goes
  # on the right hand side of the window and holds the buttons and Entry widgets.
  self.frameControl = tk.Frame(master = main_window, width=500, height=100, background="
    green")
  self.frameControl.pack(side = tk.TOP, fill=tk.BOTH, ipady = 50)
```

Listing 3: Python tkinter example code for a frame.

An example code example for a button and callback is shown in Listing 4. The call back opens a pop up message box with a title and a text message that can be used to inform a user about an event as example. Furthermore it contains code for an Entry and a Label widget.

```python
  # callback for button
    def messagebox(self):
    tk.messagebox.showinfo('Title','Text Message')
  # Button
  # With command = lambda: self.callback(arg1, arg2) a command with arguments is possible
  # without being executed at once on instantiation of the class.
    btTEST = tk.Button(master = self.frameControl, text = "Test", command = self.messagebox)
    btTEST.pack(side = tk.LEFT, anchor='w', ipadx=20)
    # Entry
    # with .bind('<return>', self.handler ) an callback can be restiered
    # which would be executed on a return key press as further example
    enDeposit = tk.Entry(master = self.frameControl)
    enDeposit.pack(side = tk.TOP, anchor='w', ipadx=20, expand=True)
    enDeposit.insert(0, '$0') # set default value
    # Label
    # A label can also be used a s placeholder to maintain order with using of .pack()
    lbBalanceAmount = tk.Label(master = self.frameControl, text='balance')
    lbBalanceAmount.pack(side = tk.TOP, anchor='w', ipadx=20, expand=True)
```

Listing 4: Python tkinter example code for a Button with callabck that opens a message box.

2.2 Turtle

According to docs.python.org/3.3/library/turtle, Turtle graphics is a popular way for introducing programming to kids. It was part of the original Logo programming language developed by Wally Feurzig and Seymour Papert in 1966. Imagine a robotic turtle starting at (0, 0) in the x-y plane. After an import turtle, give it the command turtle.forward(15), and it moves (on-screen!) 15 pixels in the direction it is facing, drawing a line as it moves. Give it the command turtle.right(25), and it rotates in-place 25 degrees clockwise.

The Bank Me Logo of the ATM is animated with turtle that allows a simple and fast way for a simple animation. Listening 5 shows an example code that uses turtle to draw the letter 'B'.

```python
import turtle

self.t = turtle.RawTurtle(canvas) # creatse a canvas to draw on
screen = self.t.getscreen() # returns the screen object
# print(t.Screen().screensize())

# This sets the lower left corner to 0,0 and the upper right corner to 600,600.
screen.setworldcoordinates(0,0,250,200)
screen.bgcolor("green")

# With the lines below, the "turtle" will look like a dollar sign
# and can be placed wit t.stamp().
screen.register_shape("dollar_resize.gif")
self.t.shape("dollar_resize.gif")

# Animation
t.pencolor("#FFFFFF") # WHITE
t.pensize(2)

t.penup()   # Regarding one of the comments
t.forward(10)
t.left(90)
t.forward(10)
t.pendown() # Regarding one of the comments
# writes the letter 'B'
t.right(90)
t.circle(40, 180)
t.right(180)
t.circle(40, 180)
t.stamp()
```

Listing 5: Python turtle example code for a letter 'B'.

2.3 ATM machine

The ATM gui was build with tkinter and turtle. The error log and receipt print out is made with file I/O. The logic because it of simple logic flow could be drastically reduced compared to the previous lab. Therefore, no additional class was written. A complete code listening can be found in the appendix Section 4.1.

Figure 2 shows the program after the animation is done and the user ask to enter the his personal identification number (PIN). After the pin has been entered in the Entry field the user can press the Enter button which will call the PIN verification method. Notice, that with the Goodbye button the program can be closed at any time.

As the user entered a valid PIN the pin entry objects are forgotten and destroyed because there is no need anymore for them. Instead the welcome message is shown and the objects to make a withdraw deposit and balance appear. By making an entry in one of the Entry fields for deposit or withdraw and pressing the button the user can either withdraw an amount of money from his account, not more then it contains or deposit amount of money.

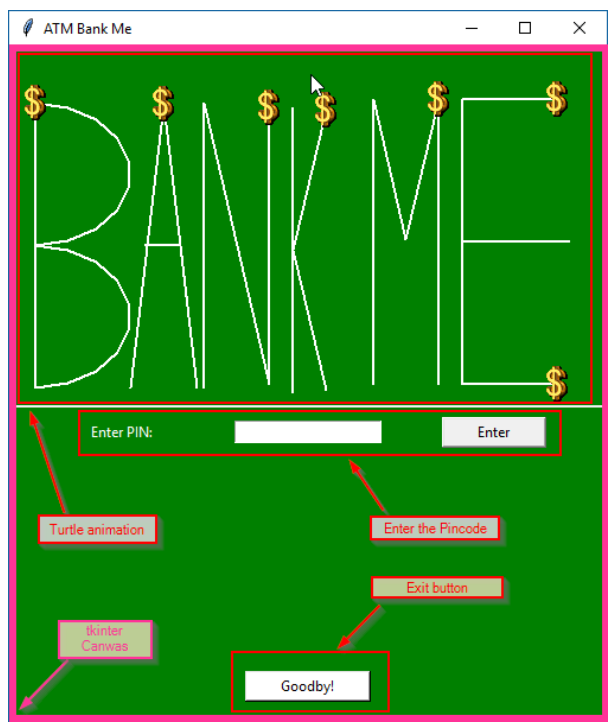

Figure 2: GUI at start time after animation is done.

Figure 3: GUI after successful PIN entry.

The following listening shows the error log output file content. This can be easily used to check which errors has been tested or for debugging. The error log is continuously appended into a text file and has to be managed manually. This could be automated with a script.

Listing 6: Error log saved as log_error.txt file.

```
Thu Nov  1 01:32:15 2018 ATM program starts
Thu Nov  1 01:32:33 2018 Error! Make sure you only use numbers from 0-9 in PIN
Thu Nov  1 01:32:35 2018 Error! Make sure you only use numbers from 0-9 in PIN
Thu Nov  1 01:32:38 2018 Error! Make sure you only use numbers from 0-9 in PIN
Thu Nov  1 01:32:47 2018 Program Closed
Thu Nov  1 01:33:15 2018 ATM program starts
Thu Nov  1 01:33:34 2018 Program Closed
Thu Nov  1 01:33:37 2018 ATM program starts
Thu Nov  1 01:33:54 2018 Invalid PIN!
Thu Nov  1 01:33:59 2018 Program Closed
Thu Nov  1 01:39:03 2018 ATM program starts
Thu Nov  1 01:39:26 2018 Withdraw amount too big or not anouth balance
Thu Nov  1 01:39:31 2018 Withdraw amount too big or not anouth balance
Thu Nov  1 01:40:22 2018 Program Closed
Thu Nov  1 01:40:29 2018 ATM program starts
Thu Nov  1 01:40:50 2018 Program Closed
Thu Nov  1 11:06:25 2018 ATM program starts
Thu Nov  1 11:06:58 2018 Program Closed
Thu Nov  1 11:13:43 2018 ATM program starts
Thu Nov  1 11:21:39 2018 Withdraw amount too big or not anouth balance
Thu Nov  1 11:21:44 2018 Program Closed
```

Figure 4 shows the receipt output generated by the program and opened in notepad by exiting the program by clicking the "Goodby!" button.

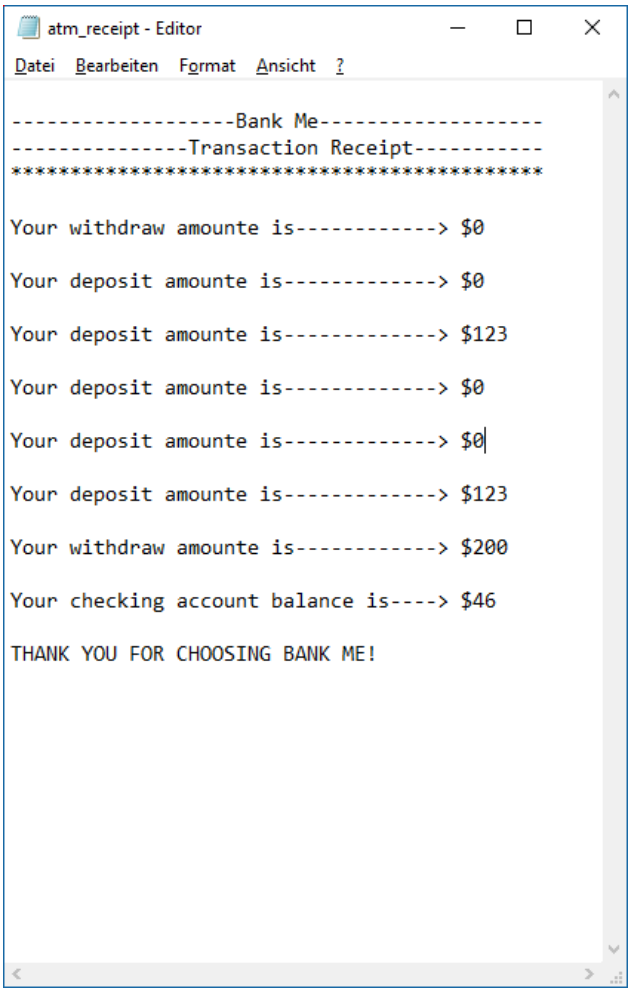

Figure 4: Printed receipt.

Further work would be to write a better GUI class and separate logic and GUI a more distinguishably.

3 Conclusion

The lab demonstrates the use of the python as simple and fast scripting language that allows access to vast number of packages that allows an decreased development time. The syntax is easy to learn but it is possible to lose the overview by having too many continuations of statements. Furthermore, it allows file I/O to save data on a disk and objective oriented programming with classes.

4 Appendix

The appendix contains code listening and other large information parts that contain partial or complete relevance to the reports topic.

4.1 Python code Listings

```python
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""
import tkinter as tk
from ATM_GUI_class import ATM_GUI

root = tk.Tk()
ATM = ATM_GUI(root)
ATM.main_window.mainloop()
```

Listing 7: Python ATM invoke of ATM_GUI_class shown in Listening 8.

```python
# -*- coding: utf-8 -*-
"""
Created on Thu Oct 25 19:22:21 2018

@author: schwa
"""

#import tkinter as tk
import tkinter as tk
import turtle
import re
from time import asctime
import webbrowser
#from tkinter import Tk, Label, Button, messagebox\
#,Canvas

class ATM_GUI(tk.Frame):
    balance ='$0'
    withdraw ='$0'
    deposit = '$0'
    pin = '1234'
    file_error = 'error_log.txt'
    file_receipt = 'atm_receipt.txt'
    fo = None
    fr = None

    def __init__(self, master=None):
        self.fo = self.init_error_file(self.fo, self.file_error)
        self.fr = self.init_receipt_file(self.fr, self.file_receipt)
        self.main_window = master

#         self.main_window = tk.Tk()
        self.main_window.title("ATM Bank Me")
        # .geometry() fixes windows size instead of downsizing with pack()
        # but does not work with turtle
#         self.main_window.geometry('800x600+10+50')
        self.display_CANVAS(self.main_window)
        self.welcomeAnim(self.t)
        self.frameQuit(self.frameBottom)
        self.framePin(self.frameWelcome)
        # Enter the tkinter main loop
#         self.main_window.mainloop()

    def init_error_file(self, file_descriptor, file_name):
```

```
        try:
            file_descriptor = open(file_name, "a")
            file_descriptor.write(asctime()+ ' ATM program starts '+ '\n')
            return file_descriptor
        except IOError as e:
            print('File '+e.filename+' could not be opened or writen to!')

    def init_receipt_file(self, file_descriptor, file_name):
        try:
            file_descriptor = open(file_name, "w")
            file_descriptor.write('\n--------------------Bank Me--------------------\n')
            file_descriptor.write('----------------Transaction Receipt------------\n')
            file_descriptor.write('*******************************************\n\n')
            return file_descriptor
        except IOError as e:
            print('File '+e.filename+' could not be opened or writen to!')

    def log_error(self, string_error):
        if self.fo != None:
            self.fo.write(asctime()+' '+string_error+'\n')
        else:
            print('No error log file defined or initialiesd, use funtion \
                  init_error_file(self, file_descriptor, file_name)')
    def log_receipt(self, string_add_to_receipt):
        if self.fr != None:
            self.fr.write(string_add_to_receipt)
        else:
            print('No receipt file defined or initialiesd, use funtion \
                  init_receipt_file(self, file_descriptor, file_name)')
            self.log_error('No receipt file defined or initialiesd, use funtion \
                  init_receipt_file(self, file_descriptor, file_name)')

    def clsoeFile(self, file_descriptor):
        file_descriptor.close()

    def withdraw(self):
#        print("Withdraw!") # debugging only
#        print(int(re.sub('[^0-9]+', '', self.enWithdraw.get())) )  # debugging only
        if int(re.sub('[^0-9]+', '', self.balance)) >= int(re.sub('[^0-9]+', '', self.enWithdraw.get())):
            self.log_receipt('Your withdraw amounte is------------> $'\
                             +re.sub('[^0-9]+', '', self.enWithdraw.get())+'\n\n')
            self.setBalance( str(int(re.sub('[^0-9]+', '', self.balance)) - \
                                 int(re.sub('[^0-9]+', '', self.enWithdraw.get())) ) )
        else:
            self.log_error('Withdraw amount too big or not anouth balance')
#        print("Withdraw: " + self.balance) # debugging only
        self.setWithdraw('0')
        self.enWithdraw.delete(0, tk.END)
        self.enWithdraw.insert(0, self.withdraw)

    def messagebox(self): # only for debugin
        tk.messagebox.showinfo('Test1','Think you did part I!')
#        print("messagebox!") # debug only

    def deposit(self):
        #tk.messagebox.showinfo('Test1','Think you did part I!')
#        print(int(re.sub('[^0-9]+', '', self.enDeposit.get())) ) # debugging only
        self.log_receipt('Your deposit amounte is------------> $'\
                         +re.sub('[^0-9]+', '', self.enDeposit.get())+'\n\n')
        self.setBalance( str(int(re.sub('[^0-9]+', '', self.balance)) + \
                             int(re.sub('[^0-9]+', '', self.enDeposit.get())) ) )
#        print("Deposit: " + self.balance) # debugging only
        self.setDeposit('0')
#        print(self.deposit) # debug only
        self.enDeposit.delete(0, tk.END)
        self.enDeposit.insert(0, self.deposit)
#        self.setDeposit(self.enDeposit.get())
```

```python

    def getBalance(self):
        return int(re.sub('[^0-9]+', '', self.balance))

    def setBalance(self, balance):
        string = re.sub('[^0-9]+', '', balance)
        self.balance = '$'+str(string)
        self.lbBalanceAmount['text'] = self.balance

    def getWithdraw(self):
        return int(re.sub('[^0-9]+', '', self.withdraw))

    def setWithdraw(self, withdraw):
        string = re.sub('[^0-9]+', '', withdraw)
        self.withdraw = '$'+str(string)

    def getDeposit(self):
        return int(re.sub('[^0-9]+', '', self.withdraw))

    def setDeposit(self, deposit):
        string = re.sub('[^0-9]+', '', deposit)
        self.deposit = '$'+str(string)

    def pinValidation(self, inVal, pin):
#        print('Enter\n') # debug only
#        print(inVal) # debug only
        ret = 0
#        if self.pin == pin:
        if not re.match("^[0-9]{4}$", inVal):
#            print("Error! Make sure you only use numbers from 0-9 in PIN\n")
            self.log_error("Error! Make sure you only use numbers from 0-9 in PIN")
            inVal = 'Z'
            ret = 0
        else:
            if inVal == pin:
#                print("\nCorect PIN") # debug only
                self.enPIN.pack_forget()
                self.lbEnterPin.pack_forget()
                self.btPIN.pack_forget()
                self.enPIN.destroy()
                self.lbEnterPin.destroy()
                self.btPIN.destroy()
#                self.framePIN.pack_forget()
                self.welcome(self.frameWelcome)
                self.frameLabel(self.frameLeft)
                self.frameInput(self.frameCenter)
                self.frameCommand(self.frameRight)
                inVal = 'Z'
                ret = 1
    #                break
            else:
                self.log_error("Invalid PIN!")
#                print("Invalid PIN!\n")
                inVal = 'Z'
                ret = 0
        return ret

    def closeProgram(self):
        self.log_error('Program Closed')
        self.log_receipt('Your checking account balance is----> $'\
                    +re.sub('[^0-9]+', '', self.balance)+'\n\n')
        self.log_receipt('THANK YOU FOR CHOOSING BANK ME!')
        self.clsoeFile(self.fo) # to ensure file is closed
        self.clsoeFile(self.fr) # to ensure file is closed
        webbrowser.open(self.file_receipt)
        self.main_window.destroy()
#        print('Goodby!')  # only for debuging

```

```
    def display_CANVAS(self, main_window):
        global canvas, logo
        #-----------------------------------------
        #Make canvaas child of root
        canvas = tk.Canvas(master = main_window, width = 500, height = 300, bg = 'green')
#        canvas.pack(side = 'top', fill = 'both', expand = 'yes')
#        explain = """Welcome to Bank Me's ATM """
#        canvas.create_text(170, 50, text=explain, font=('times',18), fill = 'blue')
#        close_button = tk.Button(canvas, text="Goodby!",
#                                 command=self.main_window.destroy)
#        close_button.configure(width = 10, activebackground = "#33B5E5", relief = 'flat')
#        canvas.create_window(10, 10, anchor='nw', window=close_button)
        canvas.pack(side = tk.TOP)

        self.t = turtle.RawTurtle(canvas)
        screen = self.t.getscreen()# print(t.Screen().screensize())
        # This sets the lower left corner to 0,0 and the upper right corner to 600,600.
        screen.setworldcoordinates(0,0,250,200)
        screen.bgcolor("green")

        # With the lines below, the "turtle" will look like a pencil.
        screen.register_shape("dollar_resize.gif")
        self.t.shape("dollar_resize.gif")

        # A frame is an invisible widget that holds other widgets. This frame goes
        # on the right hand side of the window and holds the buttons and Entry widgets.
        self.frameControl = tk.Frame(master = main_window, width=500, height=100, background="green")
        self.frameControl.pack(side = tk.TOP,fill=tk.BOTH, ipady = 50)
        self.frameWelcome = tk.Frame(master = self.frameControl, width=300, height=20, background="green")
        self.frameWelcome.pack(side = tk.TOP,fill=tk.BOTH, ipady = 8)
#        self.framePIN = tk.Frame(master = self.frameControl, width=300, height=20, background="green")
#        self.framePIN.pack(side = tk.TOP,fill=tk.BOTH, ipady = 8)
        self.frameLeft = tk.Frame(master = self.frameControl, width=50, height=80, background = 'green')
        self.frameLeft.pack(side = tk.LEFT,fill=tk.BOTH, ipadx = 10)
        self.frameCenter = tk.Frame(master = self.frameControl, width=50, height=80, background = 'green')
        self.frameCenter.pack(side = tk.LEFT,fill=tk.BOTH, ipadx = 10)
        self.frameRight = tk.Frame(master = self.frameControl, width=50, height=80, background="green")
        self.frameRight.pack(side = tk.RIGHT,fill=tk.BOTH, ipadx = 10)
        self.frameBottom = tk.Frame(master = main_window, width=150, height=80, background="green")
        self.frameBottom.pack(side = tk.BOTTOM,fill=tk.BOTH, ipadx =8, ipady = 8)

    def framePin(self, frame ):
        self.lbEnterPin = tk.Label(master = frame, text = 'Enter PIN:', background='green', foreground = 'white')
        self.lbEnterPin.pack(side = tk.LEFT, anchor='center', ipadx=41, expand=True)
        self.enPIN = tk.Entry(master = frame)
        self.enPIN.pack(side = tk.LEFT, anchor='w', ipadx=0, expand=True)
        self.enPIN["textvariable"] = 'test'
#        self.enPIN.bind("<Return>", lambda: self.pinValidation(self.enPIN.get(), self.pin)) # problem with lambda
#        print('enPIN: ' + self.enPIN.get() +'\n')  # only for debuging
        self.btPIN = tk.Button(master = frame, text = "Enter", command = lambda: self.pinValidation(self.enPIN.get(), self.pin))
        self.btPIN.pack(side = tk.LEFT, anchor='w', ipadx=25, expand = True)
#        btTEST = tk.Button(master = frameWelcome, text = "Test", command = self.messagebox) # only for debuging
#        btTEST.pack(side = tk.LEFT, anchor='w', ipadx=20)

    def welcome(self, frame ):
        tk.Label(master = frame,
                 text = 'Welcome to Bank ME',
```

```
                 background='green',
                 foreground = 'white').pack(side = tk.TOP, anchor='center', expand=True)

    def frameLabel(self, frame):
        # frameLeft
        lbWithdraw = tk.Label(master = frame, text = "Withdraw $", background = "green", foreground='white')
        lbWithdraw.pack(side = tk.TOP, anchor='center', ipadx=20, expand=True)
        lbDeposit = tk.Label(master = frame, text = "Deposit $", background = "green", foreground='white')
        lbDeposit.pack(side = tk.TOP, anchor='center', ipadx=20, expand=True)
        lbBalance = tk.Label(master = frame, text = "Balance $", background = "green", foreground='white')
        lbBalance.pack(side = tk.TOP, anchor='center', ipadx=20, expand=True)

    def frameInput(self, frameCenter):
        # frameCenter
        self.enWithdraw = tk.Entry(master = frameCenter)
        self.enWithdraw.pack(side = tk.TOP, anchor='w', ipadx=20, expand=True)
        self.enWithdraw.insert(0,'$0')
        self.enDeposit = tk.Entry(master = frameCenter)
        self.enDeposit.pack(side = tk.TOP, anchor='w', ipadx=20, expand=True)
        self.enDeposit.insert(0, '$0')
        self.lbBalanceAmount = tk.Label(master = frameCenter, text=self.balance)
        self.lbBalanceAmount.pack(side = tk.TOP, anchor='w', ipadx=20, expand=True)

    def frameCommand(self, frameRight):
        # frameRight
        btWithdraw = tk.Button(master = frameRight, text = "Withdraw", command = self.withdraw)
        btWithdraw.pack(side = tk.TOP, anchor='w', ipadx=20, expand=True)
        btDeposit = tk.Button(master = frameRight, text = "Deposit", command = self.deposit)
        btDeposit.pack(side = tk.TOP, anchor='w', ipadx=25, expand=True)
#        tk.Button(master = frameRight, text = "space",background = "green",
#                  activebackground = "green", relief=tk.FLAT).pack(side = tk.TOP, anchor='w', ipadx=20, expand=True)
        tk.Label(master = frameRight, text='',background = "green").pack(side = tk.TOP, anchor='w', ipadx=20, expand=True)

    def frameQuit(self, frameBottom):
        # frameRight
        btQuit = tk.Button(master = frameBottom, text="Goodby!", background = "#FFFFFF",
                  activebackground = "#33B5E5",
                  command=self.closeProgram)
        btQuit.pack(side = tk.TOP, anchor='center', ipadx=25)

    def welcomeAnim(self, t):
        # Animation
        t.pencolor("#FFFFFF") # WHITE
        t.pensize(2)

        t.penup()   # Regarding one of the comments
        t.forward(10)
        t.left(90)
        t.forward(10)
        t.pendown() # Regarding one of the comments
        # B
        t.right(90)
        t.circle(40, 180)

        t.right(180)
        t.circle(40, 180)
        t.stamp()
        t.left(90)
        t.forward(160)
        t.penup()   # Regarding one of the comments
        t.left(90)
        t.forward(40)
```

```
        t.pendown()   # Regarding one of the comments
        # A
        t.left(85)
        t.forward(160)
        t.stamp()
        t.right(170)
        t.forward(160)
        t.backward(80)
        t.left(265)
        t.forward(15)
        t.penup()   # Regarding one of the comments
        t.backward(25)
        t.left(90)
        t.pendown()   # Regarding one of the comments
        # N
        t.forward(80)
        t.backward(160)
        t.right(360-10)
        t.forward(160)
        t.left(180-10)
        t.forward(155)
        t.stamp()
        t.penup()   # Regarding one of the comments
        t.right(90)
        t.forward(10)
        t.pendown()   # Regarding one of the comments
        # K
        t.right(90)
        t.forward(160)
        t.backward(80)
        t.left(180-10)
        t.forward(80)
        t.stamp()
        t.backward(80)
        t.right(160)
        t.forward(80)
        t.penup()   # Regarding one of the comments
        t.left(90)
        t.forward(20)
        t.pendown()   # Regarding one of the comments
        # M
        t.left(80)
        t.forward(160)
        t.right(180-10)
        t.forward(80)
        t.left(180-20)
        t.forward(80)
        t.right(180-10)
        t.stamp()
        t.forward(160)
        t.penup()   # Regarding one of the comments
        t.left(90)
        t.forward(10)
        t.pendown()   # Regarding one of the comments
        # E
        t.left(90)
        t.forward(160)
        t.right(90)
        t.forward(40)
        t.stamp()
        t.backward(40)
        t.right(90)
        t.forward(80)
        t.left(90)
        t.forward(45)
        t.backward(45)
        t.right(90)
        t.forward(80)
```

```
        t.left(90)
        t.forward(40)
        t.stamp()


#GUI =Tk()
#GUI.mainloop()
```

Listing 8: Python GUI for an ATM.