```
1  import tkinter as tk
2  import tkinter.messagebox
3  import time
4
5
6  class Application(tk.Frame):
7      def __init__(self, master, *args, **
   kwargs):
8          tk.Frame.__init__(self, master, *args, *
   *kwargs)
9          self.master = master
10         self.running = False
11         self.time = 0
12         self.hours = 0
13         self.mins = 0
14         self.secs = 0
15         self.build_interface()
16
17     def build_interface(self):
18         self.time_entry = tk.Entry(self)
19         self.time_entry.grid(row=0,
   column=1)
20
21         self.clock = tk.Label(self,
   text="00:00:00", font=("Courier", 20),
   width=10)
22         self.clock.grid(row=1, column=1,
   stick="S")
23
24         self.time_label = tk.Label(self,
   text="hour   min   sec", font=("Courier", 10),
   width=15)
25         self.time_label.grid(row=2,      =1,
```

```
21      text= 00:00:00 , font=( Courier , 20),
        width=10)
22          self.clock.grid(row=1, column=1,
        stick="S")
23
24          self.time_label = tk.Label(self,
        text="hour   min   sec", font=("Courier", 10),
        width=15)
25          self.time_label.grid(row=2, column=1,
        sticky="N")
26
27          self.power_button = tk.Button(self,
        text="Start", command=lambda: self.
        start())
28          self.power_button.grid(row=3,
        column=0, sticky="NE")
29
30          self.reset_button = tk.Button(self,
        text="Reset", command=lambda: self.
        reset())
31          self.reset_button.grid(row=3,
        column=1, sticky="NW")
32
33          self.quit_button = tk.Button(self,
        text="Quit", command=lambda: self.quit())
34          self.quit_button.grid(row=3,
        column=3, sticky="NE")
35
36          self.pause_button = tk.Button(self,
        text="Pause", command=lambda: self.
        pause())
37          self.pause_button.grid(row =
        column=2, sticky = "NW")
38
```

```
34        self.quit_button.grid(row=3,
column=3, sticky="NE")
35
36        self.pause_button = tk.Button(self,
text="Pause", command=lambda: self.
pause())
37        self.pause_button.grid(row = 3,
column=2, sticky = "NW")
38
39        self.master.bind("<Return>", lambda
x: self.start())
40        self.time_entry.bind("<Key>", lambda
v: self.update())
41
42    def calculate(self):
43        """time calculation"""
44        self.hours = self.time // 3600
45        self.mins = (self.time // 60) % 60
46        self.secs = self.time % 60
47        return "{:02d}:{:02d}:{:02d}".
format(self.hours, self.mins, self.secs)
48
49    def update(self):
50        """validation"""
51        self.time = int(self.time_entry.get())
52        try:
53            self.clock.configure(text=self.
calculate())
54        except:
55            self.clock.
configure(text="00:00:00")
56
57    def timer(self):
```

```
        validation
        self.time = int(self.time_entry.get())
        try:
            self.clock.configure(text=self.
calculate())
        except:
            self.clock.
configure(text="00:00:00")

    def timer(self):
        """display time"""
        if self.running:
            if self.time <= 0:
                self.clock.configure(text="Time's
up!")
            else:
                self.clock.configure(text=self.
calculate())
                self.time -= 1
                self.after(1000, self.timer)

    def start(self):
        """start timer"""
        try:
            self.time = int(self.time_entry.get())
            self.time_entry.delete(0, 'end')
        except:
            self.time = self.time
        self.power_button.
configure(text="Stop", command=lambda:
self.stop())
        self.master.bind("<Return>", lambda
x: self.stop())
        self.running = True
```

```
70              self.time = int(self.time_entry.get())
71              self.time_entry.delete(0, 'end')
72          except:
73              self.time = self.time
74          self.power_button.
    configure(text="Stop", command=lambda:
    self.stop())
75          self.master.bind("<Return>", lambda
    x: self.stop())
76          self.running = True
77          self.timer()
78
79      def stop(self):
80          """Stop timer"""
81          self.power_button.
    configure(text="Start", command=lambda:
    self.start())
82          self.master.bind("<Return>", lambda
    x: self.start())
83          self.running = False
84
85      def reset(self):
86          """Resets the timer to 0."""
87          self.power_button.
    configure(text="Start", command=lambda:
    self.start())
88          self.master.bind("<Return>", lambda
    x: self.start())
89          self.running = False
90          self.time = 0
91          self.clock["text"] = "00:00:00"
92
93      def quit(self):
```

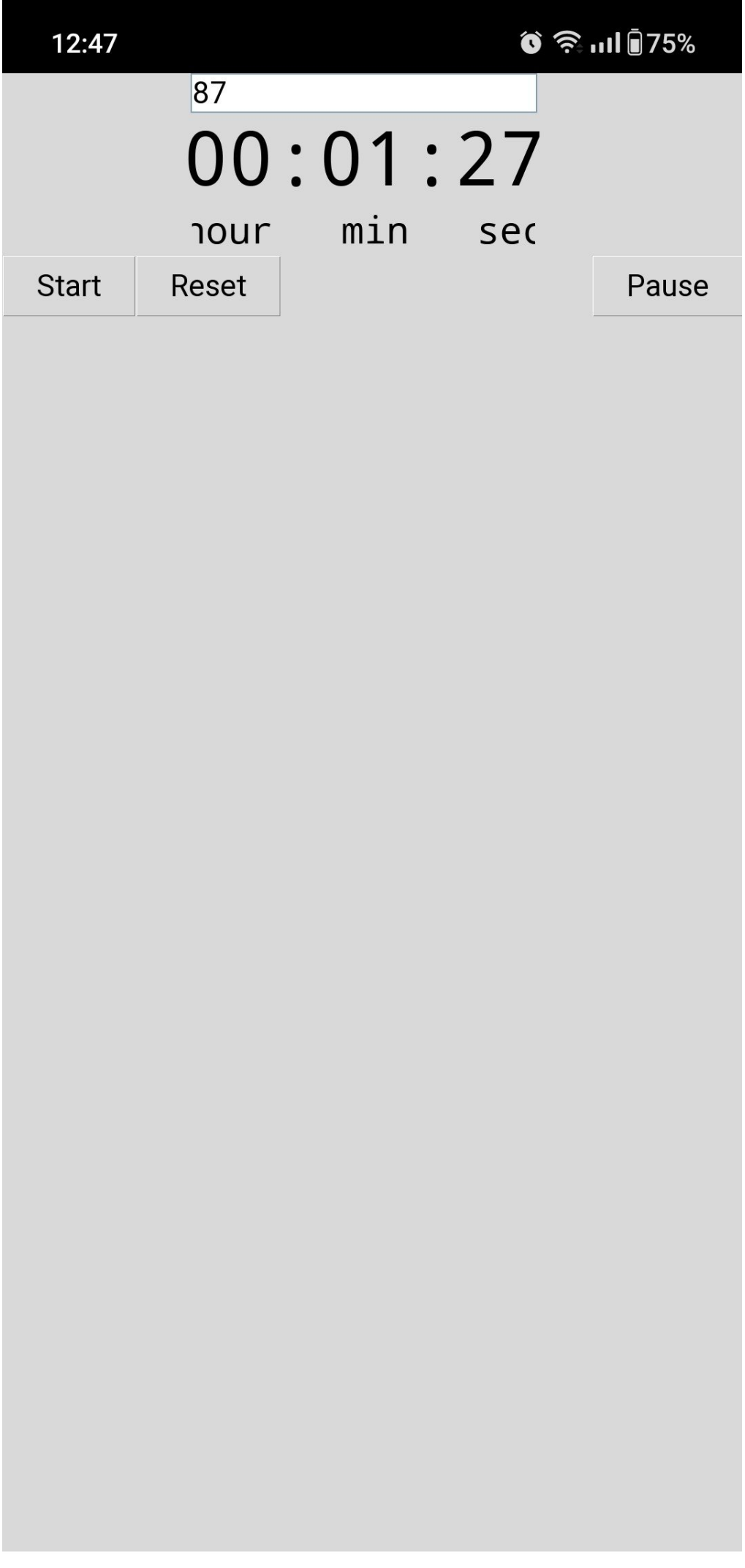
```
    def reset(self):
        """Resets the timer to 0."""
        self.power_button.
configure(text="Start", command=lambda:
self.start())
        self.master.bind("<Return>", lambda
x: self.start())
        self.running = False
        self.time = 0
        self.clock["text"] = "00:00:00"

    def quit(self):
        """quit the window"""
        if tk.messagebox.askokcancel("Quit",
"Do you want to quit?"):
            root.destroy()

    def pause(self):
        """Pause timer"""
        self.pause_button.
configure(text="Resume",
command=lambda: self.resume())
        self.master.bind("<Return>", lambda
x: self.resume())
        if self.running == True:
            self.running = False
        self.timer()


    def resume(self):
        """Resume timer"""
        self.pause_button.
configure(text="Pause",
command=lambda: self.pause())
```

```
105
106
107     def resume(self):
108         """Resume timer"""
109         self.pause_button.
    configure(text="Pause",
    command=lambda: self.pause())
110         self.master.bind("<Return>", lambda
    x: self.pause())
111         if self.running == False:
112             self.running = True
113         self.timer()
114
115
116
117
118
119 if __name__ == "__main__":
120     """Main loop of timer"""
121     root = tk.Tk()
122     root.title("TIMER")
123     Application(root).pack(side="top",
    fill="both", expand=True)
124     root.mainloop()
```


12:47
75%
87
00:01:27
hour
min
sec
Start
Reset
Pause