Code Examples for Lecture 6

Long-Running Tasks: Application Stub

```
import time
import random
import tkinter as tk
from tkinter import ttk

class App(ttk.Frame):

    def __init__(self, parent):
        super().__init__(parent, padding="3 3 12 12")
        self.style = ttk.Style()
        self.style.theme_use("alt")
        parent.rowconfigure(0, weight=1)
        parent.columnconfigure(0, weight=1)
        self.grid(row=0, column=0, sticky="NEWS")
        self.create_widgets()

    def create_widgets(self):
        self.rowconfigure(0, weight=1)
        self.rowconfigure(1, weight=1)
        self.columnconfigure(0, weight=1)

        # control variables
        self.progress1 = tk.IntVar()
        self.status1 = tk.StringVar()
        self.status2 = tk.StringVar()

        # subframe for Task 1
        self.frame1 = ttk.LabelFrame(self, text="Task 1")
        self.frame1.grid(row=0, column=0, sticky="news", padx=5, pady=5)
        self.frame1.rowconfigure(0, weight=1)
        self.frame1.rowconfigure(1, weight=1)
        self.frame1.columnconfigure(0, weight=1)
        self.bar1 = ttk.Progressbar(self.frame1, length=500,
            variable=self.progress1, mode="determinate")
        self.label1 = ttk.Label(self.frame1, text="Stopped",
            textvariable=self.status1)
        self.start1 = ttk.Button(self.frame1, text="Start",
            command=self.run_task1)
        self.bar1.grid(row=0, column=0, sticky="sew", padx=10)
        self.label1.grid(row=1, column=0, sticky="wn", padx=10)
        self.start1.grid(row=0, column=1, rowspan=2, padx=10, pady=10)

        # subframe for Task 2
        self.frame2 = ttk.LabelFrame(self, text="Task 2")
        self.frame2.grid(row=1, column=0, sticky="news", padx=5, pady=5)
        self.frame2.rowconfigure(0, weight=1)
        self.frame2.rowconfigure(1, weight=1)
        self.frame2.columnconfigure(0, weight=1)
        self.bar2 = ttk.Progressbar(self.frame2, length=500,
            mode="indeterminate")
        self.label2 = ttk.Label(self.frame2, text="Stopped",
            textvariable=self.status2)
        self.start2 = ttk.Button(self.frame2, text="Start", c
            ommand=self.run_task2)
        self.bar2.grid(row=0, column=0, sticky="sew", padx=10)
        self.label2.grid(row=1, column=0, sticky="wn", padx=10)
        self.start2.grid(row=0, column=1, rowspan=2, padx=10, pady=10)

        self.quit = ttk.Button(self, text="Quit", command=root.destroy)
        self.quit.grid(row=2, column=0, padx=5, pady=5, sticky="s")

        self.status1.set("Stopped")
        self.status2.set("Stopped")

    def run_task1(self):
        print("Running task 1...")
        self.task1()
```

```
    def run_task2(self):
        print("Running task 2...")
        self.task2()

    def task1(self):   # simulate determinate long-running task
        time.sleep(10)

    def task2(self):   # simulate indeterminate long-running task
        time.sleep(random.randrange(5,10)


if __name__ == "__main__":
    root = tk.Tk()
    root.title("Long-Running Tasks")
    app = App(root)
    root.mainloop()
```

Basic Animation: Application Stub

```
import time
import tkinter as tk
import tkinter.ttk as ttk

class App(ttk.Frame):

    def __init__(self, parent):
        super().__init__(parent)
        parent.rowconfigure(0, weight=1)
        parent.columnconfigure(0, weight=1)
        self.grid(row=0, column=0, sticky="news")
        self.rowconfigure(0, weight=1)
        self.columnconfigure(0, weight=1)
        self.columnconfigure(1, weight=1)
        self.create_widgets()
        self.is_animating = False

    def create_widgets(self):
        self.canvas = tk.Canvas(self, borderwidth=0,
            highlightthickness=0, bg="yellow")
        self.canvas.grid(row=0, column=0, columnspan=2,
            sticky="news", padx=10, pady=10)
        self.btn_animate = ttk.Button(self, text="Animate",
            command=self.toggle_animation)
        self.btn_animate.grid(row=1, column=0, pady=10)
        ttk.Button(self, text="Quit", command=root.destroy).grid(
            row=1, column=1, pady=10)

    def toggle_animation(self):
        self.is_animating = not self.is_animating
        if self.is_animating:
            self.btn_animate.config(text="Stop")
            self.animate()
        else:
            self.btn_animate.config(text="Animate")

    def animate(self):
        # put animation update code here
        #   :

        # schedule the next update
        if self.is_animating:
            self.after(33, self.animate)


if __name__ == "__main__":
    root = tk.Tk()
    root.title("Basic Animation")
    root.geometry("300x300")
    app = App(root)
    root.mainloop()
```