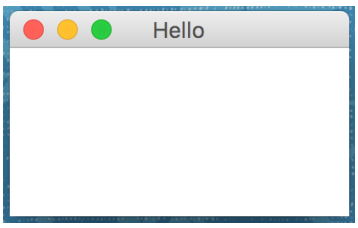
객체지향 프로그래밍
Graphical User Interface (GUI) 활용

소프트웨어와 사용자가 의사소통하는 창구를 **사용자 인터페이스**user interface라고 한다. 지금까지 작성한 프로그램에서 사용자 인터페이스는 표준입출력창이 창구가 되는 **명령어 인터페이스**command line interface이다. 표준입력창에서 키보드를 통하여 문자열로 프로그램에 입력 데이터를 전달하고, 프로그램은 계산 결과를 문자열로 표준출력창에 보여 준다. 그런데 이 사용자 인터페이스는 명령창에서 문자열로만 의사소통을 하므로 사용자 편의성 면에서 불편하다.

마우스가 발명되면서 키보드 이외의 다양한 의사소통 경로가 생겼는데, 이를 활용한 사용자 인터페이스를 **그래픽 사용자 인터페이스**graphical user interface라고 하고 하며, 줄여서 GUI('구이'라고 읽음)라고 하기도 한다. 이 장에서는 Python에서 GUI를 활용하는 프로그램을 만들 수 있는 도구를 모아놓은 표준 라이브러리 모듈인 **Tkinter**(Tk interface의 약자로 '티케이인터'라고 읽음)를 공부한다[1].

Hello 1

먼저 아래 왼쪽과 같은 빈 창은 아래 오른쪽 코드로 만든다.

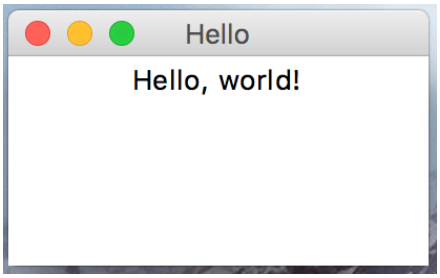



```
1 from tkinter import *
2
3 window = Tk()
4 window.title("Hello")
5 window.geometry("200x100")
6 window.mainloop()
```

tkinter 모듈을 쓰기 위해서는 일단 가져와야 하므로 줄 1과 같이 하여 Tkinter 모듈을 통채로 가져온다. Tkinter 모듈에는 Tk GUI 프로그램을 만드는데 필요한 도구가 모두 마련되어 있다. Tkinter를 시작하려면 GUI가 활동할 영역인 창window 객체 Tk를 먼저 생성해야 한다. 줄 3에서 Tk를 생성하여 이름을 window라 하였다. 창의 간판명은 줄 4와 같이 title 메소드를 호출하여 지정할 수 있고, 창의 크기는 줄 5와 같이 geometry 메소드를 호출하여 지정할 수 있다. 그래픽 화면으로 보는 이미지는 아주 작은 정사각형 점들이 모여서 이루어지는데 이 점 하나를 픽셀pixel(화소)이라고 하며, 그래픽 화면에서 크기를 나타내는 단위로 사용한다. 창의 크기는 "200x100"과 같이 하나의 문자열 형식으로 geometry 메소드의 인수로 지정하는데, 창의 크기가 가로로 200 픽셀, 세로 100픽셀 이라는 뜻이다. geometry 메소드의 크기 인수는 이와 같은 형식으로 문자열로 만들어야 함을 명심하자.

이제 빈 창이 생겼으니 이 창에 다음과 같이 Hello, world!를 프린트해보자. 이 창을 만드는 프로그램은 다음과 같이 작성한다.

[1] https://wiki.python.org/moin/TkInter

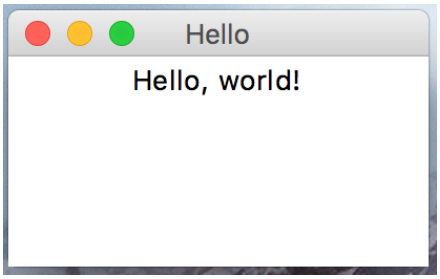



```
1  from tkinter import *
2
3  window = Tk()
4  window.title("Hello")
5  window.geometry("200x100")
6  Label(window, text="Hello, world!").pack()
7  window.mainloop()
```

GUI에는 버튼button, 스크롤바scroll bar, 메뉴menu, 입력 상자entry, 텍스트 상자text box와 같은 다양한 종류의 인터페이스 요소가 있는데, 이를 통틀어 **위젯**widget이라고 한다. 줄 6에서는 Label 위젯 객체를 하나 만든다. Label 위젯은 텍스트나 이미지를 갖고 있는 위젯이다. 여기서 만든 Label 객체는 "Hello, world!" 문자열을 텍스트로 갖고 있는 window 소속의 위젯 객체이다. pack() 메소드를 호출하면 Label 위젯 객체의 창에서 텍스트의 위치를 적절하게 지정한다. 그리고 줄 7과 같이 window.mainloop()를 호출하면, 창이 생기면서 이벤트event 루프에 돌입하여 Label 객체의 내용을 창의 지정한 위치에서 보여준다.

GUI 프로그램은 **이벤트 구동**event-driven 방식으로 작동한다. 생성된 위젯은 모두 이벤트가 발생하기를 상시 기다리고 있으므로 특정 이벤트가 발생하면 이에 맞는 반응을 하도록 프로그램을 작성해야 한다. Label 위젯은 스스로 이벤트를 발생시켜 내용을 보여주는 기능만 가진 일방형 위젯이다.

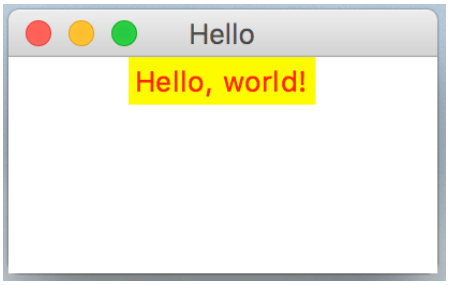
여러 위젯은 모아놓을 수 있는 컨테이너를 Frame이라 하는데, 위에서 본 window 자체도 프레임이지만 따로 프레임을 만들어 쓰는게 좋다. 위의 App을 Frame 클래스를 상속받아 다음과 같이 만들 수 있다.




```
1  from tkinter import *
2
3  class App(Frame):
4      def __init__(self, window):
5          super().__init__(window)
6          self.pack()
7          label = Label(self)
8          label['text'] = "Hello, world!"
9          label.pack()
10
11 window = Tk()
12 window.title("Hello")
13 window.geometry("200x100")
14 App(window)
15 window.mainloop()
```

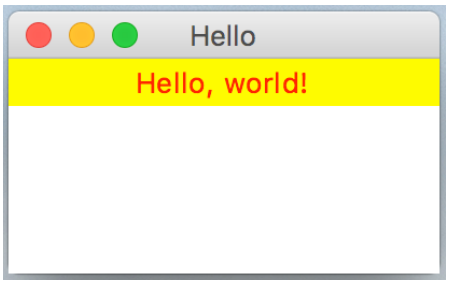
이제 보이지는 않지만 창 안에 Frame 객체가 있고, 그 안에 Label 객체가 있다. App 클래스에서 self 는 window 안에 속한 Frame 자신을 가리킨다. 그 프레임(self) 안에 Label 객체가 있다.

위젯은 다양한 옵션을 갖고 있다. 예를 들어 배경과 글자의 색깔을 다음과 같이 지정할 수 있다.



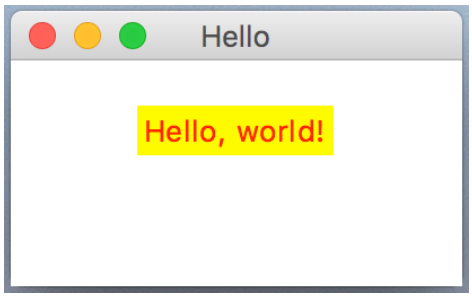

```
1  from tkinter import *
2
3  class App(Frame):
4      def __init__(self, window):
5          super().__init__(window)
6          self.pack()
7          label = Label(self)
8          label['text'] = "Hello, world!"
9          label['bg'] = "yellow"
10         label['fg'] = "red"
11         label.pack()
12
13 window = Tk()
14 window.title("Hello")
15 window.geometry("200x100")
16 App(window)
17 window.mainloop()
```

pack() 메소드의 옵션도 다양하게 있다. fill 옵션은 여백을 모두 배경 색깔로 다음과 같이 채운다. X는 수평으로 채우라는 표시이고, Y는 수직으로 채우라는 표시이다.




```
1  from tkinter import *
2
3  class App(Frame):
4      def __init__(self, window):
5          super().__init__(window)
6          self.pack()
7          label = Label(self)
8          label['text'] = "Hello, world!"
9          label['bg'] = "yellow"
10         label['fg'] = "red"
11         label.pack(fill=X)
12
13 window = Tk()
14 window.title("Hello")
15 window.geometry("200x100")
16 App(window)
17 window.mainloop()
```

Frame의 주위에 다음과 같이 빈 여백을 줄 수도 있다.




```
1  from tkinter import *
2
3  class App(Frame):
4      def __init__(self, window):
5          super().__init__(window)
6          self.pack(padx=20, pady=20)
7          label = Label(self)
8          label['text'] = "Hello, world!"
9          label['bg'] = "yellow"
10         label['fg'] = "red"
11         label.pack()
12
13 window = Tk()
14 window.title("Hello")
15 window.geometry("200x100")
16 App(window)
17 window.mainloop()
```

똑 같은 프로그램이지만 다음과 같이 표현할 수도 있다.

```
1  from tkinter import *
2
3  class App(Frame):
4      def __init__(self, window):
5          super().__init__(window)
6          self.pack(padx=20, pady=20)
7          Label(self, text="Hello, world!", bg="yellow", fg="red").pack()
8
9  window = Tk()
10 window.title("Hello")
11 window.geometry("200x100")
12 App(window)
13 window.mainloop()
```

지금까지는 창에 직접 그렸지만 이번엔 사용자가 이벤트를 발생시키면 이에 반응을 하는 위젯을 만들어보자. 오른쪽과 같이 Quit 단추를 만들어 이를 클릭하면 창을 닫는다.

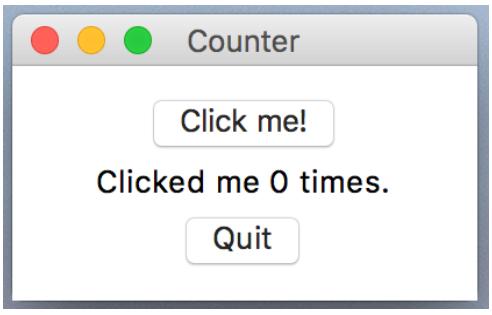
```
1  from tkinter import *
2
3  class App(Frame):
4      def __init__(self, window):
5          super().__init__(window)
6          self.pack(padx=20, pady=20)
7          Label(self, text="Hello, world!", bg="yellow", fg="red").pack()
8          Button(self, text="Quit", command=self.quit).pack()
9
10 window = Tk()
11 window.title("Hello")
12 window.geometry("200x100")
13 App(window)
14 window.mainloop()
```

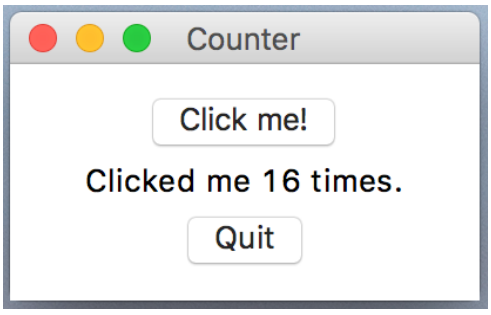
줄 8에서는 Button 위젯을 만든다. command 옵션에는 이 버튼을 클릭했을 때 실행할 메소드의 이름을 적는다. quit 메소드는 Frame에 내장되어 있는 메소드로 호출하면 창을 닫아준다. 여기서 self.quit은 메소드 호출이 아님을 주의하자. ()가 없지 않은가! 단순히 메소드 이름만 전달해줄 뿐이다. 언젠가 이 Button을 클릭하는 이벤트가 발생하면 이 메소드가 저절로 호출된다.

이와 같이 Tkinter 모듈은 GUI 프로그램을 작성하는데 필요한 다양한 도구를 갖추고 있다. 여기서 도구를 모두 다 살펴보기는 위젯의 종류도 너무 많고 옵션도 다양하므로, 몇 가지만 살펴본다. 자세한 문서는 위키사이트인 https://wiki.python.org/moin/TkInter 에서 찾아볼 수 있다. GUI의 프로그램을 잘 하기 위해서는 다양한 위젯과 관련 옵션들을 잘 꿰고 있어야 한다. 많이 접해보는 수 밖에 지금길은 없다.

Counter

이번에는 카운터를 만들어보자. 다음과 같은 창에서 단추를 클릭할 때마다 1씩 증가하여 보여주도록 한다. 즉, 아래의 왼쪽 창에서 시작하여 16번 Click me! 단추를 클릭하면 오른쪽 창과 같이 되어야 한다.

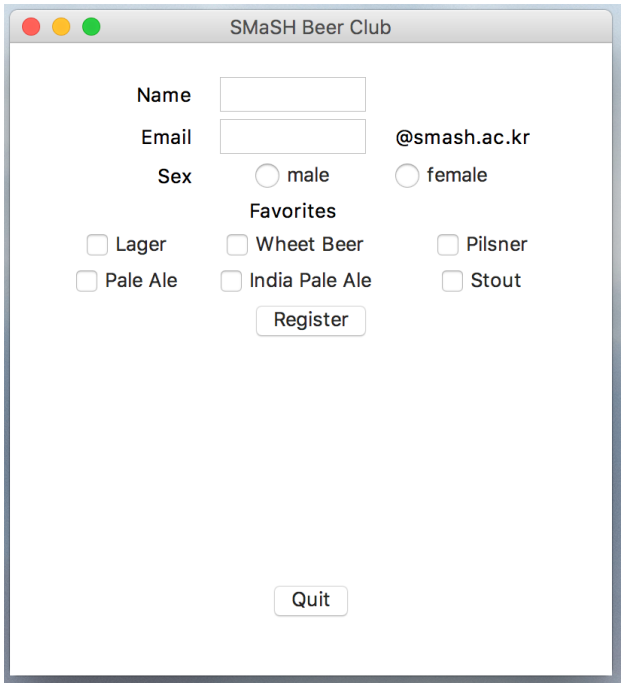


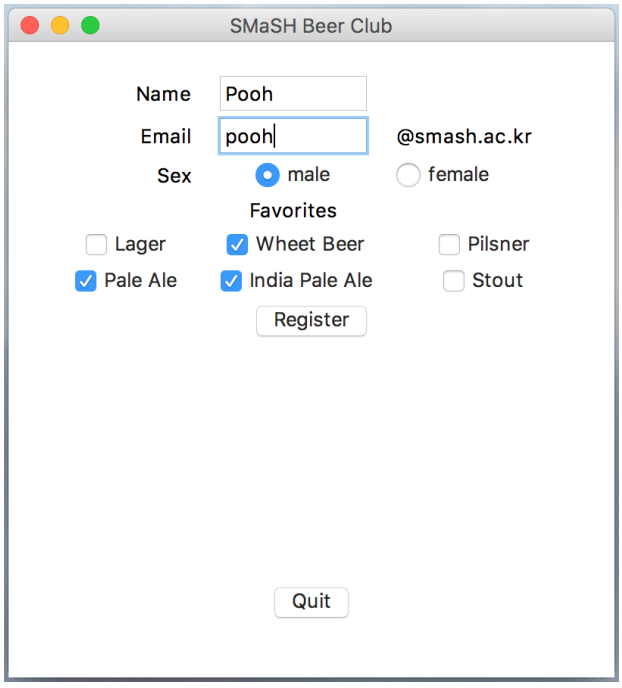


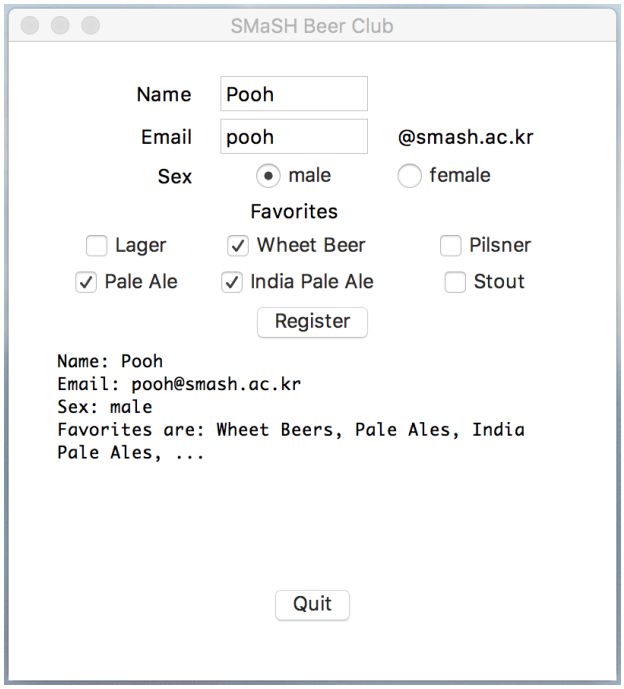
```
from tkinter import *

class App(Frame):
    def __init__(self, master):
        super().__init__(master)
        self.pack(padx = 10, pady = 10)
        self.button_clicks = 0
        self.create_widgets()

    def create_widgets(self):
        self.button = Button(self)
        self.button["text"] = "Click me!"
        self.button["command"] = self.update_count
        self.button.pack()
        self.label = Label(self)
        self.label["text"] = "Clicked me " + str(self.button_clicks) + " times."
        self.label.pack()
        Button(self, text="Quit", command=self.quit).pack()

    def update_count(self):
        self.button_clicks += 1
        self.label["text"] = "Clicked me " + str(self.button_clicks) + " times."

root = Tk()
root.title("Counter")
root.geometry("200x100")
app = App(root)
root.mainloop()
```

Beer Club

이 그림과 같이 Label, Entry, Radiobutton, Checkbutton, Button 위젯을 만들어 Frame에 배치한 다음, 아래 왼쪽 그림과 같이 입력하고 선택하고 Register 단추를 클릭했을 때, 아래 오른쪽 그림과 같이 입력 및 선택 사항을 정리하여 보여주는 프로그램을 만들어보자.



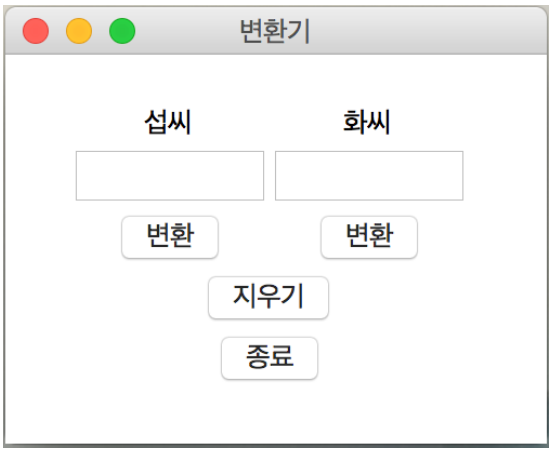




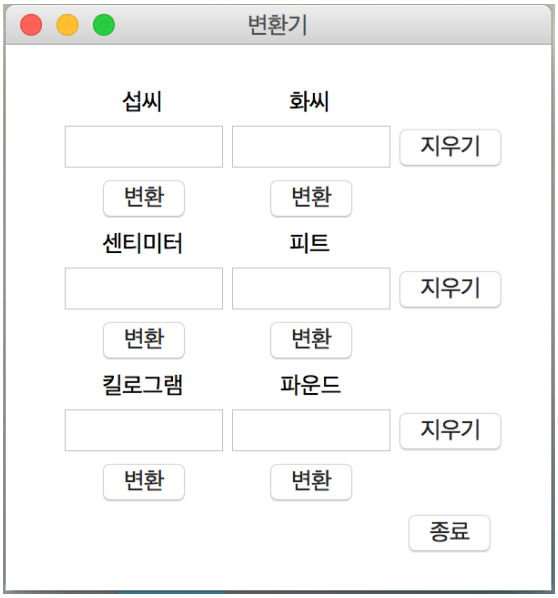
```
from tkinter import *

class App(Frame):
    def __init__(self, master):
        super().__init__(master)
        self.pack(padx=20, pady=20)
        self.create_widgets()

    def create_widgets(self):
        Label(self, text="Name").grid(row=0, column=0, sticky=E)
        self.name = Entry(self, width=10)
        self.name.grid(row=0, column=1)
        Label(self, text="Email").grid(row=1, column=0, sticky=E)
        self.email = Entry(self, width=10)
        self.email.grid(row=1, column=1)
        Label(self, text="@smash.ac.kr").grid(row=1, column=2, sticky=W)
        self.sex = StringVar()
        self.sex.set(None)
        Label(self, text="Sex").grid(row=2, column=0, sticky=E)
        Radiobutton(self, text='male',
                    variable=self.sex, value='male'
                    ).grid(row=2, column=1)
        Radiobutton(self, text='female',
                    variable=self.sex, value='female'
                    ).grid(row=2, column=2, sticky=W)
        Label(self, text="Favorites").grid(row=3, column=1)
        self.lagers = BooleanVar()
        Checkbutton(self, text="Lager", variable=self.lagers
                    ).grid(row=4, column=0)
        self.wheetbeer = BooleanVar()
        Checkbutton(self, text="Wheet Beer", variable=self.wheetbeer
                    ).grid(row=4, column=1)
        self.pilsners = BooleanVar()
        Checkbutton(self, text="Pilsner", variable=self.pilsners
                    ).grid(row=4, column=2)
        self.paleales = BooleanVar()
        Checkbutton(self, text="Pale Ale", variable=self.paleales
                    ).grid(row=5, column=0)
        self.indiapaleales = BooleanVar()
        Checkbutton(self, text="India Pale Ale", variable=self.indiapaleales
                    ).grid(row=5, column=1)
        self.stouts = BooleanVar()
        Checkbutton(self, text="Stout", variable=self.stouts
                    ).grid(row=5, column=2)
```

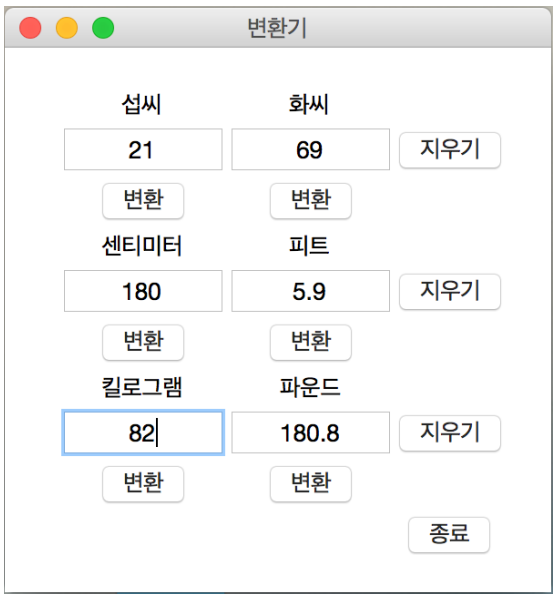
```
45          Button(self, text="Register",
46                 command=self.write_summary
47                 ).grid(row=6, column=0, columnspan=3, sticky=S)
48          self.summary = Text(self, width=48, height=10, wrap=WORD)
49          self.summary.grid(row=7, column=0, columnspan=3, sticky=S)
50          Button(self, text="Quit", command=self.quit
51                 ).grid(row=8, column=0, columnspan=3)
52
53      def write_summary(self):
54          summary = "Name: " + self.name.get() + "\n"
55          summary += "Email: " + self.email.get() + "@smash.ac.kr\n"
56          summary += "Sex: " + self.sex.get() + "\n"
57          summary += "Favorites are: "
58          if self.lagers.get():
59              summary += "Lagers, "
60          if self.wheetbeer.get():
61              summary += "Wheet Beers, "
62          if self.pilsners.get():
63              summary += "Pilsners, "
64          if self.paleales.get():
65              summary += "Pale Ales, "
66          if self.indiapaleales.get():
67              summary += "India Pale Ales, "
68          if self.stouts.get():
69              summary += "Stouts, "
70          summary += "..."
71          self.summary.delete(0.0, END)
72          self.summary.insert(0.0, summary)
73
74  # main
75  root = Tk()
76  root.title("SMaSH Beer Club")
77  root.geometry("400x420")
78  App(root)
79  root.mainloop()
```

실습 1 : 계량 변환 애플리케이션



conversion.py 파일에 위의 창과 같이 섭씨 온도를 화씨로 화씨 온도를 섭씨로 변환하는 애플리케이션이 들어있다. 변환하고 싶은 온도를 정수로 입력하고 바로 밑의 변환버튼을 누르면 변환한 값이 정수로 옆 창에 나타난다. 지우기버튼을 누르면 온도데이터가 모두 지워지고, 종료버튼을 누르면 창이 닫힌다. 이 파일의 코드를 모두 이해하고 이를 확장하여 다음 창과 같은 기능을 추가로 갖추도록 애플리케이션을 확장하자.





추가로 구현해야 하는 기능은 다음과 같다.

- 센티미터를 피트로 변환하거나, 피트를 센티미터로 변환 (입력은 정수만 허용, 출력은 소수점 첫째자리 미만에서 반올림)
- 킬로그램을 파운드로 변환하거나, 파운드를 킬로그램으로 변환 (입력은 정수만 허용, 출력은 소수점 첫째자리 미만에서 반올림)
- 해당 입력창에 정수를 입력하고 바로 아래 버튼을 누르면 변환한 값이 옆 창에 나타난다.
- 지우기버튼을 누르면 해당 행의 데이터가 모두 지워진다.

- 종료버튼을 누르면 창이 닫힌다.

변환 공식은 다음과 같다.

centimeters	feets
1	0.03281
30.48	1

kilograms	pounds
1	2.2046
0.4536	1

 ©도경구(2016)