Занятие 26. Основы TKinter. Обмен сообщениями

ПЛАН ЗАНЯТИЯ

1. ОБМЕН СООБЩЕНИЯМИ

Откройте свой проект и включите в него файл window_08.py:

window_08.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

# импортирование модулей python
from Tkinter import *

# класс главного окна
class main:
  def __init__(self, master):
    self.master = master
    self.master.title('parent')
    self.master.geometry('200x150+300+225')
    self.button = Button(self.master,
                         text = 'dialog',
                         command = self.openDialog)
    self.button.pack(side = BOTTOM)
    self.text = Text(self.master,
                     background = 'white')
    self.text.pack(side = TOP,
                   fill = BOTH,
                   expand = YES)
    self.master.mainloop()

  def openDialog(self):
    self.dialog = child(self.master)
    self.sendValue = self.text.get('0.0', END)
    self.returnValue = self.dialog.go(self.sendValue)
    if self.returnValue:
      self.text.delete('0.0', END)
      self.text.insert('0.0', self.returnValue)

# класс дочернего окна
class child:
  def __init__(self, master):
    self.slave = Toplevel(master)
    self.slave.title('child')
    self.slave.geometry('200x150+500+375')
    self.button = Button(self.slave,
```

```
                    text = 'accept',
                    command = self.accept)
    self.button.pack(side = BOTTOM)
    self.text = Text(self.slave,
                     background = 'white')
    self.text.pack(side = TOP,
                   fill = BOTH,
                   expand = YES)

  def go(self, myText = ''):
    self.text.insert('0.0', myText)
    self.newValue = None
    self.slave.grab_set()
    self.slave.focus_set()
    self.slave.wait_window()
    return self.newValue

  def accept(self):
    self.newValue = self.text.get('0.0', END)
    self.slave.destroy()

# создание окна
root = Tk()

# запуск окна
main(root)
```

Мы хотим, чтобы информация шла от родительского окна к дочернему И от дочернего к родительскому. Последнего можно добиться, добавив к дочернему окну кнопку accept и метод для регистрации всех изменений, произведенных в передаваемом тексте; также введем метод go как для создания экземпляра дочернего окна, так и для управления процессом обмена информацией. Такое применение методов [go] для открытия дочерних окон - полезный инструмент, который приобретет особую важность, когда мы будем иметь дело с более сложными операциями в последующих руководствах. Сейчас же он создает пустую переменную newValue, в которую будет записан измененный текст [если он был изменен].

```
# класс дочернего окна
```

```
class child:
  def __init__(self, master):
    ...
    self.button = Button(self.slave,
                         text = 'accept',
                         command = self.accept)
    self.button.pack(side = BOTTOM)
    ...

  def go(self, myText = ''):
    self.text.insert('0.0', myText)
    self.newValue = None
    self.slave.grab_set()
    self.slave.focus_set()
    self.slave.wait_window()
    return self.newValue

  def accept(self):
    self.newValue = self.text.get('0.0', END)
    self.slave.destroy()
```

При вызове дочернего метода `go` введенный текст вставляется в дочернее текстовое окно. Если текст редактировался и пользователь нажимает кнопку **accept**, исправленный текст возвращается как **newValue**. Если дочернее окно просто закрывается, `newValue` возвращается с пустым значением.

```
  def openDialog(self):
    self.dialog = child(self.master)
    self.sendValue = self.text.get('0.0', END)
    self.returnValue = self.dialog.go(self.sendValue)
    if self.returnValue:
      self.text.delete('0.0', END)
      self.text.insert('0.0', self.returnValue)
```

При возврате метод **openDialog** класса main осуществляет проверку. Если возвращаемая строка не пустая, возвращаемый текст будет вставлен в текстовое окно **main.**

Запускаем **window_08.py.** Убедитесь, что вы "уловили", как работает **go.** Нам придется еще довольно много иметь дело с этим методом...

2. УСЛОЖНЕНИЕ ДИАЛОГОВ

Откройте свой проект и включитие в него файл window_09.py:

window_09.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

# импортирование модулей python
from Tkinter import *

# класс главного окна
class main:
  def __init__(self, master):
    self.master = master
    self.master.title('parent')
    self.master.geometry('400x300+200+150')
    self.button = Button(self.master,
                         text = 'dialog',
                         command = self.openDialog)
    self.button.pack(side = BOTTOM)
    self.text = Text(self.master,
                     background = 'white')
    self.text.pack(side = TOP,
                   fill = BOTH,
                   expand = YES)
    self.master.protocol('WM_DELETE_WINDOW',
                         self.exitMethod)
    self.master.mainloop()

  def openDialog(self):
    self.dialog = child(self.master)
    self.sendValue = self.text.get('0.0', END)
    self.returnValue = self.dialog.go(self.sendValue)
    if self.returnValue:
      self.text.delete('0.0', END)
      self.text.insert('0.0', self.returnValue)

  def exitMethod(self):
    self.dialog = yesno(self.master)
    self.returnValue = self.dialog.go('question',
```

```
                                'Do you want to exit?')
    if self.returnValue:
      self.master.destroy()

# класс дочернего окна
class child:
  def __init__(self, master):
    self.slave = Toplevel(master)
    self.slave.title('child')
    self.slave.geometry('200x150+500+375')
    self.frame = Frame(self.slave)
    self.frame.pack(side = BOTTOM)
    self.accept_button = Button(self.frame,
                                text = 'accept',
                                command = self.accept)
    self.accept_button.pack(side = LEFT)
    self.cancel_button = Button(self.frame,
                                text = 'cancel',
                                command = self.cancel)
    self.cancel_button.pack(side = RIGHT)
    self.text = Text(self.slave, background = 'white')
    self.text.pack(side = TOP, fill = BOTH, expand = YES)
    self.slave.protocol('WM_DELETE_WINDOW', self.cancel)

  def go(self, myText = ''):
    self.text.insert('0.0', myText)
    self.newValue = None
    self.slave.grab_set()
    self.slave.focus_set()
    self.slave.wait_window()
    return self.newValue

  def accept(self):
    self.newValue = self.text.get('0.0', END)
    self.slave.destroy()

  def cancel(self):
    self.slave.destroy()

# класс диалогового окна выхода
class yesno:
  def __init__(self, master):
    self.slave = Toplevel(master)
    self.slave.title('exit dialog')
    self.slave.geometry('200x100+300+250')
```

```
    self.frame = Frame(self.slave)
    self.frame.pack(side = BOTTOM)
    self.yes_button = Button(self.frame,
                             text = 'yes',
                             command = self.yes)
    self.yes_button.pack(side = LEFT)
    self.no_button = Button(self.frame,
                            text = 'no',
                            command = self.no)
    self.no_button.pack(side = RIGHT)
    self.label = Label(self.slave)
    self.label.pack(side = TOP, fill = BOTH, expand = YES)
    self.slave.protocol('WM_DELETE_WINDOW', self.no)

  def go(self, title = '', message = ''):
    self.slave.title(title)
    self.label.configure(text = message)
    self.booleanValue = TRUE
    self.slave.grab_set()
    self.slave.focus_set()
    self.slave.wait_window()
    return self.booleanValue

  def yes(self):
    self.booleanValue = TRUE
    self.slave.destroy()

  def no(self):
    self.booleanValue = FALSE
    self.slave.destroy()

# создание окна
root = Tk()

# запуск окна
main(root)
```

Мы приближаемся к концу начала. Рассмотрим еще несколько приемов, которые будут использоваться позднее в "реальном" приложении: кнопки принятия (accept) и отмены (cancel), а также "перехват" закрытия основного окна с соответствующим диалогом.

```
class child:
```

```
  def __init__(self, master):
    ...
    self.frame = Frame(self.slave)
    self.frame.pack(side = BOTTOM)
    self.accept_button = Button(self.frame,
                                text = 'accept',
                                command = self.accept)
    self.accept_button.pack(side = LEFT)
    self.cancel_button = Button(self.frame,
                                text = 'cancel',
                                command = self.cancel)
    self.cancel_button.pack(side = RIGHT)
    ...

  def accept(self):
    self.newValue = self.text.get('0.0', END)
    self.slave.destroy()

  def cancel(self):
    self.slave.destroy()
```

Пока все просто. Размещение в дочерних диалоговых окнах кнопки отмены (**cancel**) - довольно стандартный прием. При этом происходит уничтожение дочернего окна без изменения **newValue.**

```
# класс главного окна
class main:
  def __init__(self, master):
    ...
    self.master.protocol('WM_DELETE_WINDOW', self.exitMethod)
    ...
```

```
  def exitMethod(self):
    self.dialog = yesno(self.master)
    self.returnValue = self.dialog.go('question', 'Do you want to exit?')
    if self.returnValue:
      self.master.destroy()
```

```
    ...
```

```
# класс диалогового окна выхода
class yesno:
  def __init__(self, master):
    self.slave = Toplevel(master)
```

```
    self.slave.title('exit dialog')
    self.slave.geometry('200x100+300+250')
    self.frame = Frame(self.slave)
    self.frame.pack(side = BOTTOM)
    self.yes_button = Button(self.frame,
                             text = 'yes',
                             command = self.yes)
    self.yes_button.pack(side = LEFT)
    self.no_button = Button(self.frame,
                            text = 'no',
                            command = self.no)
    self.no_button.pack(side = RIGHT)
    self.label = Label(self.slave)
    self.label.pack(side = TOP, fill = BOTH, expand = YES)
    self.slave.protocol('WM_DELETE_WINDOW', self.no)

  def go(self, title = '', message = ''):
    self.slave.title(title)
    self.label.configure(text = message)
    self.booleanValue = TRUE
    self.slave.grab_set()
    self.slave.focus_set()
    self.slave.wait_window()
    return self.booleanValue

  def yes(self):
    self.booleanValue = TRUE
    self.slave.destroy()

  def no(self):
    self.booleanValue = FALSE
    self.slave.destroy()
```

"Перехват" закрытия пользователем главного окна немного сложнее. Для чего это может потребоваться? Допустим, в случае текстового редактора нам бы хотелось фиксировать ситуации, когда не был сохранен измененный файл, чтобы предохранить пользователя от нечаянного уничтожения плодов всех его усилий из-за преждевременного закрытия окна.

Секрет спрятался в строке **self.master.protocol('WM_DELETE_WINDOW', self.exitMethod). WM_DELETE_WINDOW -** это часть оконного протокола, которая обычно бывает связана с **self.destroy().** Но в данной строке вместо этого она

связывается с одним из разработанных нами методов. Итак, взгляните на **def exitMethod(self)**: здесь вызывается **message** и выход происходит, только если message (сообщение) равно **TRUE (ИСТИНА). message** управляется целым новым классом **yesno**, обладающим своим собственным методом **go.**

Запустите **window_09.py**, чтобы увидеть программу в действии.

Djghjcs