

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

ІКТА  
Кафедра ЗІ



**Звіт**  
**до лабораторної роботи №10**

з курсу: «Програмування скриптовими мовами» на тему: «Розробка графічного інтерфейсу користувача засобами пакету tkinter»

**Виконав:**

студент групи КБ-305

Семенчук А.А.

**Прийняв:**

к.т.н., доцент

Совин Я. Р.

Львів 2024

Мета роботи – ознайомитись з особливостями створення графічного інтерфейсу користувача засобами пакету tkinter мови Python, зокрема навчитися працювати з типовими елементами інтерфейсу (віджетами), вивчити їх основні властивості і методи, вміти розміщувати віджети у вікні та обробляти події.

## Завдання

1. Написати програму, яка використовуючи класи з лабор. роботи №9 створює додаток на базі бібліотеки tkinter, що дозволяє виконувати такі операції:

- a. Вивести весь список.
- b. Додавати елементи до списку.
- c. Відсортувати список за заданим атрибутом.
- d. Видаляти елементи за заданим атрибутом.
- e. Видаляти елемент за заданим індексом.

f. Виводити всі елементи за заданим атрибутом.

7	Пиво	Назва, виробник, міцність, ціна, термін зберігання в днях
---	------	---

## Виконання лабораторної роботи

```
import tkinter as tk
from tkinter import messagebox, simpledialog
from typing import List, Any

class Beer:
    def __init__(self, name: str, manufacturer: str, strength: float, price: int,
storage_days: int):
        self.__name = name
        self.__manufacturer = manufacturer
        self.__strength = strength
        self.__price = price
        self.__storage_days = storage_days

    @property
    def name(self) -> str:
        return self.__name

    @property
    def manufacturer(self) -> str:
        return self.__manufacturer

    @property
    def strength(self) -> float:
        return self.__strength
```

```

@property
def price(self) -> int:
    return self.__price

@property
def storage_days(self) -> int:
    return self.__storage_days

def __lt__(self, other: 'Beer') -> bool:
    return self.__storage_days < other.storage_days

def __repr__(self) -> str:
    return f"{self.name} ({self.manufacturer}): {self.strength}% ABV,
${self.price}, {self.storage_days} days"

class BeerCatalog:
    def __init__(self):
        self.beers = [
            Beer("Obolon", "Ukraine", 5.0, 30, 180),
            Beer("Lvivske", "Ukraine", 4.5, 25, 150),
            Beer("Heineken", "Netherlands", 5.0, 50, 365),
            Beer("Corona", "Mexico", 4.6, 45, 300),
            Beer("Budweiser", "USA", 5.0, 40, 180)
        ]

    def print_catalog(self) -> List[str]:
        return [str(beer) for beer in self.beers]

    def add_beer(self, beer: Beer) -> None:
        self.beers.append(beer)

    def sort_by_attribute(self, attribute: str) -> None:
        self.beers.sort(key=lambda beer: getattr(beer, attribute))

    def delete_by_attribute(self, attribute: str, value: Any) -> None:
        self.beers = [beer for beer in self.beers if getattr(beer, attribute) !=
value]

    def delete_by_index(self, index: int) -> None:
        if 0 <= index < len(self.beers):
            self.beers.pop(index)

    def filter_by_attribute(self, attribute: str, value: Any) -> List[str]:
        return [str(beer) for beer in self.beers if getattr(beer, attribute) ==
value]

class BeerApp:
    def __init__(self, root):
        self.catalog = BeerCatalog()
        self.root = root
        self.root.title("Beer Catalog Management")

```

```

# Listbox to display the catalog
self.listbox = tk.Listbox(root, width=50)
self.listbox.pack()

# Display Buttons
tk.Button(root, text="Show All", command=self.show_all).pack()
tk.Button(root, text="Add Beer", command=self.add_beer).pack()
tk.Button(root, text="Sort By Attribute",
command=self.sort_by_attribute).pack()
tk.Button(root, text="Delete By Attribute",
command=self.delete_by_attribute).pack()
tk.Button(root, text="Delete By Index",
command=self.delete_by_index).pack()
tk.Button(root, text="Filter By Attribute",
command=self.filter_by_attribute).pack()

# Initial display of catalog
self.show_all()

def show_all(self):
    self.listbox.delete(0, tk.END)
    for beer in self.catalog.print_catalog():
        self.listbox.insert(tk.END, beer)

def add_beer(self):
    AddBeerWindow(self.catalog, self)

def sort_by_attribute(self):
    attribute = self.ask_attribute("Enter attribute to sort by (name,
manufacturer, strength, price, storage_days):")
    if attribute in ["name", "manufacturer", "strength", "price",
"storage_days"]:
        self.catalog.sort_by_attribute(attribute)
        self.show_all()
    else:
        messagebox.showerror("Error", "Invalid attribute name for sorting.")

def delete_by_attribute(self):
    attribute = self.ask_attribute("Enter attribute to delete by (name,
manufacturer, strength, price, storage_days):")
    if attribute in ["name", "manufacturer", "strength", "price",
"storage_days"]:
        value = self.ask_value("Enter the value of the attribute:")
        if attribute in ["strength", "price", "storage_days"]:
            try:
                value = float(value) if attribute == "strength" else int(value)
            except ValueError:
                messagebox.showerror("Error", f"Invalid value type for
attribute '{attribute}'.")
            return
        self.catalog.delete_by_attribute(attribute, value)
        self.show_all()

```

```

        else:
            messagebox.showerror("Error", "Invalid attribute name for deletion.")

    def delete_by_index(self):
        index = self.ask_index("Enter index to delete:")
        if index is not None:
            self.catalog.delete_by_index(index)
            self.show_all()

    def filter_by_attribute(self):
        attribute = self.ask_attribute("Enter attribute to filter by (name, manufacturer, strength, price, storage_days):")
        if attribute in ["name", "manufacturer", "strength", "price", "storage_days"]:
            value = self.ask_value("Enter the value of the attribute:")
            if attribute in ["strength", "price", "storage_days"]:
                try:
                    value = float(value) if attribute == "strength" else int(value)
                except ValueError:
                    messagebox.showerror("Error", f"Invalid value type for attribute '{attribute}'.")
                    return
            self.listbox.delete(0, tk.END)
            for beer in self.catalog.filter_by_attribute(attribute, value):
                self.listbox.insert(tk.END, beer)
        else:
            messagebox.showerror("Error", "Invalid attribute name for filtering.")

    def ask_attribute(self, message):
        return simpdialog.askstring("Input", message, parent=self.root)

    def ask_value(self, message):
        return simpdialog.askstring("Input", message, parent=self.root)

    def ask_index(self, message):
        response = simpdialog.askstring("Input", message, parent=self.root)
        if response is None:
            return None
        try:
            return int(response)
        except ValueError:
            messagebox.showerror("Error", "Invalid index")
            return None

class AddBeerWindow:
    def __init__(self, catalog, app):
        self.catalog = catalog
        self.app = app
        self.window = tk.Toplevel()
        self.window.title("Add Beer")

        # Fields for Beer properties

```

```

tk.Label(self.window, text="Name:").pack()
self.name_entry = tk.Entry(self.window)
self.name_entry.pack()

tk.Label(self.window, text="Manufacturer:").pack()
self.manufacturer_entry = tk.Entry(self.window)
self.manufacturer_entry.pack()

tk.Label(self.window, text="Strength:").pack()
self.strength_entry = tk.Entry(self.window)
self.strength_entry.pack()

tk.Label(self.window, text="Price:").pack()
self.price_entry = tk.Entry(self.window)
self.price_entry.pack()

tk.Label(self.window, text="Storage Days:").pack()
self.storage_entry = tk.Entry(self.window)
self.storage_entry.pack()

tk.Button(self.window, text="Add", command=self.add).pack()

def add(self):
    name = self.name_entry.get()
    manufacturer = self.manufacturer_entry.get()
    try:
        strength = float(self.strength_entry.get())
        price = int(self.price_entry.get())
        storage_days = int(self.storage_entry.get())
        new_beer = Beer(name, manufacturer, strength, price, storage_days)
        self.catalog.add_beer(new_beer)
        self.app.show_all()
        self.window.destroy()
    except ValueError:
        messagebox.showerror("Error", "Please enter valid values for strength,
price, and storage days.")

if __name__ == "__main__":
    root = tk.Tk()
    app = BeerApp(root)
    root.mainloop()

```

Obolon (Ukraine): 5.0% ABV, \$30, 180 days  
Lvivske (Ukraine): 4.5% ABV, \$25, 150 days  
Heineken (Netherlands): 5.0% ABV, \$50, 365 days  
Budweiser (USA): 5.0% ABV, \$40, 180 days

Buttons for beer management:

- Show All
- Add Beer
- Sort By Attribute
- Delete By Attribute
- Delete By Index
- Filter By Attribute

Obolon (Ukraine): 5.0% ABV, \$30, 180 days  
Lvivske (Ukraine): 4.5% ABV, \$25, 150 days  
Heineken (Netherlands): 5.0% ABV, \$50, 365 days  
Budweiser (USA): 5.0% ABV, \$40, 180 days

Enter attribute to filter by (name, manufacturer, strength, name)

OK Cancel

Buttons for beer management:

- Show All
- Add Beer
- Sort By Attribute
- Delete By Attribute
- Delete By Index
- Filter By Attribute

Obolon (Ukraine): 5.0% ABV, \$30, 180 days  
Lvivske (Ukraine): 4.5% ABV, \$25, 150 days  
Heineken (Netherlands): 5.0% ABV, \$50, 365 days  
Budweiser (USA): 5.0% ABV, \$40, 180 days



**Invalid attribute name for filtering.**

OK

Sort By Attribute

Delete By Attribute

Delete By Index

Filter By Attribute

Name: 5.0% ABV, \$30, 180 days

5.0% ABV, \$25, 150 days

Manufacturer: 5.0% ABV, \$50, 365 days

5.0% ABV, \$40, 180 days

Strength:

Price:

Storage Days:

Show All

Add

Add Beer

Sort By Attribute

Delete By Attribute

Delete By Index

Filter By Attribute



## **Висновок**

Виконуючи дану роботу, я ознайомився з основами створення графічного інтерфейсу користувача в Python, використовуючи бібліотеку Tkinter. Я навчився працювати з типовими елементами інтерфейсу (віджетами) та їх властивостями і методами. У процесі роботи я зрозумів, як розміщувати віджети у вікні за допомогою методів `pack`, `grid`, `place`, а також вивчив підходи до обробки подій, пов'язаних із взаємодією користувача з інтерфейсом. Це дозволило мені створювати прості графічні програми, які забезпечують зручний та зрозумілий інтерфейс для користувачів.