Unit:3; Class:4

Polymorphism, Abstraction & Tkinter

Instructor: Musfique Ahmed

What is Polymorphism?

📌 **Definition:**

- *Polymorphism* means **"many forms."**
- The same function or method can work differently depending on the object.

🎯 **Example (Real Life):**

- A **teacher** can teach **Math, Science, or English**—same person, different subjects!
- A **smartphone** can be used to **call, play games, or browse the internet**—same device, different functionalities!

🛠️ **Python Example:**

```
print(len("Hello"))  # Output: 5 (counts characters)
print(len([1, 2, 3]))  # Output: 3 (counts list items)
```

📝 **Practice Problem:**

- Create a function that calculates the area of a **circle**, **square**, and **rectangle** using the same method name.

Method Overriding vs. Overloading

🟢 **Method Overriding:**

- A **child class changes the behavior** of a method inherited from the parent class.

🔴 **Method Overloading:**

- The **same method name** with different parameters (not fully supported in Python).
- *Python achieves overloading using default arguments.*

Method Overriding vs. Overloading

🛠️ **Example:**

```
class Animal:
    def speak(self):
        return "Animal makes a sound"

class Dog(Animal):
    def speak(self):
        return "Woof! Woof!"

dog = Dog()
print(dog.speak())  # Output: Woof! Woof!
```

📝 **Practice Problem:**

- Create a `Bird` class with a `fly()` method.
- Override it in `Eagle` (returns `"Can fly"`) and `Penguin` (returns `"Cannot fly"`).

Polymorphism with Classes

🎯 Using Polymorphism in Inheritance

- Different classes can **use the same method** name but behave differently.

🛠️ Example: Vehicle Class

```
class Vehicle:
    def move(self):
        return "This vehicle is moving"

class Car(Vehicle):
    def move(self):
        return "The car is driving!"

class Bike(Vehicle):
    def move(self):
        return "The bike is riding!"

def vehicle_movement(vehicle):
    print(vehicle.move())

car = Car()
bike = Bike()

vehicle_movement(car)  # Output: The car is driving!
vehicle_movement(bike) # Output: The bike is riding!
```

📝 Practice Problem:

- Create a `Shape` class with a `draw()` method.
- Override it in `Circle` and `Square` classes to return `"Drawing a circle"` and `"Drawing a square"`.

Introduction to Abstraction (abc Module)

What is Abstraction?

- Hides unnecessary details and shows only relevant parts.
- Uses **abstract base classes (ABC module)**.

🛠️ **Example:**

```python
from abc import ABC, abstractmethod

class Animal(ABC):
    @abstractmethod
    def make_sound(self):
        pass

class Dog(Animal):
    def make_sound(self):
        return "Bark!"

dog = Dog()
print(dog.make_sound())  # Output: Bark!
```

📝 **Practice Problem:**

- Create an abstract class `Appliance` with a method `turn_on()`.
- Implement it in `Fan` and `TV` classes.

Python Developers & Jobs

Rahiim Hussein · 24m ·

I'm prepared this Project with Python GUI Programming language. This Project Retail Billing System 💯✅

Introduction to Tkinter

📌 What is Tkinter?

- Tkinter is Python's built-in **GUI (Graphical User Interface) library**.
- Used to create **windows, buttons, labels, and input fields**.

🎯 Why Learn Tkinter?

✅ Easy to use.
✅ Great for simple apps, dashboards, and tools.

Documentation , Geeks_for_Geeks

🛠️ Basic Tkinter Window Example:

```
import tkinter as tk

root = tk.Tk()
root.title("My First GUI")
root.geometry("300x200")

label = tk.Label(root, text="Hello, Tkinter!")
label.pack()

root.mainloop()
```

Tkinter Widgets

📌 Common Tkinter Widgets

Widget	Description	Widget	Description
Label	Displays text or images.	**Button**	A clickable button to trigger events.
Entry	A single-line text input field.	**Frame**	A container for organizing widgets.
Text	A multi-line text input field.	**Checkbutton**	A checkbox for selecting/deselecting.
Radiobutton	Allows selecting one option from a group.	**Listbox**	Displays a list of selectable items.
Combobox	A drop-down list for selection.	**Spinbox**	A numeric input with increment/decrement arrows.
Scale	A slider for selecting a numerical value.	**Scrollbar**	Adds scrolling to widgets like Text & Listbox.
Canvas	Used for drawing shapes, images, and graphics.	**Menu**	A menu bar with drop-down options.
PanedWindow	A window split into resizable sections.	**Toplevel**	A separate pop-up window.
Message	Like Label, but for displaying longer text.	**LabelFrame**	A container with a visible title label.
Progressbar	Shows task progress with a graphical bar.	**Notebook**	Implements a tabbed interface.
Treeview	Displays hierarchical data in a tree-like format.	**Separator**	Creates a horizontal or vertical line.
Sizegrip	A draggable corner for resizing windows.	**Tooltip**	Shows a message when hovering over a widget.

📌 More Tkinter Widgets

Widget	Description	Widget	Description
Scrollbar	Provides scrolling functionality for widgets like Text and Listbox.	**Buttonbox**	A group of buttons for user selections.
OptionMenu	A drop-down menu to select from predefined options.	**Listbox**	A scrollable list for multiple item selection.
Checkbutton	A checkbox used for binary options (checked/unchecked).	**Radiobutton**	A group of mutually exclusive radio buttons.
LabelFrame	A container widget that has a label on top.	**PanedWindow**	A window that can be split into resizable sections.
Message	Displays text in a block with automatic wrapping.	**Scrollbar**	Adds vertical or horizontal scrolling capability.
Text	A widget to display and edit multi-line text.	**Scale**	A slider widget for adjusting a value.
Spinbox	A widget to input a number using up/down arrows.	**Toplevel**	A separate pop-up window.
Treeview	A widget to display hierarchical data in a tree format.	**Progressbar**	A graphical widget to display progress.
Canvas	A widget for drawing graphics like shapes, lines, etc.	**Entry**	A simple one-line text input widget.
Text	A multi-line text widget for longer user input.	**ComboBox**	A drop-down menu to choose from several options.
Sizegrip	A resizable corner of a window to adjust its size.	**Separator**	A widget that creates a line to separate sections.

Adding a Button


```python
import tkinter as tk

root = tk.Tk()
root.title("Simple App")

def say_hello():
    print("Hello, Tkinter!")

button = tk.Button(root, text="Click Me", command=say_hello)
button.pack()

root.mainloop()
```

📝 **Practice Problem:**

- Create a Tkinter window with a **label** and **two buttons** (one prints `"Hello"`, another prints `"Goodbye"`).

Real-World Project Idea 🚀

💡 Project: GUI-Based Calculator

- Create a **simple calculator** using Tkinter.
- Use buttons for numbers and basic operations (+, -, *, /).

Calculator

```
import tkinter as tk

def calculate():
    result.set(eval(entry.get()))

root = tk.Tk()
root.title("Calculator")

entry = tk.Entry(root)
entry.pack()

result = tk.StringVar()
label = tk.Label(root, textvariable=result)
label.pack()

button = tk.Button(root, text="Calculate", command=calculate)
button.pack()

root.mainloop()
```

🛠️ Other Code Idea:

- Catch_the_ball
- Snake_game
- Tic_tac_toe

Summary

✅ **Polymorphism** allows the same method to have **different behaviors**.
✅ **Method Overriding** lets a child class change a parent class method.
✅ **Abstract Classes** enforce structure using `abc` module.
✅ **Tkinter** helps create **interactive Python applications**.
✅ **Real-world applications:** AI assistants, GUI apps, game development.

Thank You

Do the Quiz Please, you have 10 minutes to do that!