**Laporan Program Python**

**Praktikum Lab – 13**

**Mata Kuliah:**

Pemrograman Berorientasi Objek

## Oleh:

Gerry Moeis Mahardika Dwi Putra

23091397164

2023E

[https://github.com/gerrymoeis/pbo-](https://github.com/gerrymoeis/pbo-1)4



# PROGRAM STUDI MANAJEMEN INFORMATIKA FAKULTAS VOKASI

**UNIVERSITAS NEGERI SURABAYA 2024**

## Laporan Tugas 1: Polymorphism – Suara Binatang!

|  |  |
| --- | --- |
| **Langkah** | **Praktikum** |
| **1.** | import tkinter as tk  Import modul yang dibutuhkan dalam hal ini tkinter as tk.  class Animal:      def make\_sound(self):          return "Miaw Miaw Ninja!"  Selanjutnya buat class Animal sebagai Parent Class Utama.  Class Animal ini memiliki method berupa make\_sound(), yang mengembalikan nilai berupa Teks Suara yang akan ditampilkan. |
| **2.** | # Kelas turunan Bird  class Bird(Animal):      def make\_sound(self):          return "Twiwit Twiwit"  # Kelas turunan Dog  class Dog(Animal):      def make\_sound(self):          return "Anjengg!"  Selanjutnya, buat Class anak dari Animal.  Disini instruksi modul membuat class Bird dan Dog. |
| **3.** | root = tk.Tk()  root.title("Suara Binatang!")  # Label untuk menampilkan hasil suara  label\_result = tk.Label(root, text="Klik salah satu tombol untuk mendengar suara hewan.", font=("Arial", 20))  label\_result.pack(pady=20)  label\_sound = tk.Label(root, text="...", font=("Arial", 16))  label\_sound.pack(pady=20)  # Fungsi untuk menampilkan suara berdasarkan jenis hewan yang dipilih  def show\_sound(animal):      label\_sound.config(text=animal.make\_sound())  Setelah itu kita siapkan untuk variable-variable yang diperlukan.  root dari modul tk sebagai window program.  label\_result sebagai label text.  label\_sound sebagai tempat output suara nantinya.  Lalu kita buat fungsi show\_sound sebagai polymorphism dari method make\_sound milik masing-masing animal yang diperoleh dari parameter animal. |
| **4.** | # Tombol untuk memilih Burung  button\_bird = tk.Button(root, text="Burung", font=("Arial", 16), command=lambda: show\_sound(Bird()))  button\_bird.pack(pady=10)  # Tombol untuk memilih Anjing  button\_dog = tk.Button(root, text="Anjing", font=("Arial", 16), command=lambda: show\_sound(Dog()))  button\_dog.pack(pady=10)  # Tombol untuk memilih Hewan Umum  button\_animal = tk.Button(root, text="Hewan Umum", font=("Arial", 16), command=lambda: show\_sound(Animal()))  button\_animal.pack(pady=10)  # Menjalankan aplikasi Tkinter  root.mainloop()  Nah langkah terakhir, disini kita buat tombol button untuk masing-masing animal. Umumnya saya memakai looping untuk membuat button-button ini agar tidak mengulang, tetapi sekarang lagi banyak tugas, jadi seyogyanya dulu.  button disini dibuat dari tk.Button, dengan input argument, root, text, font, dan command untuk fungsi yang akan dilaksanakan bila button di klik.  Terakhir, agar program berjalan terus selama belum di close, maka pastikan untuk menambahkan root.mainloop(). |
| **Tugas** | # Kelas turunan Bird  class Babi(Animal):      def make\_sound(self):          return "Ang Eng Ang"  # Kelas turunan Monyet  class Monyet(Animal):      def make\_sound(self):          return "U u A A"  # Tombol untuk memilih Babi  button\_pig = tk.Button(root, text="Babi", font=("Arial", 16), command=lambda: show\_sound(Babi()))  button\_pig.pack(pady=10)  # Tombol untuk memilih Monyet  button\_monkey = tk.Button(root, text="Monyet", font=("Arial", 16), command=lambda: show\_sound(Monyet()))  button\_monkey.pack(pady=10)  Disini saya menambahkan 2 Class untuk hewan-hewan baru dengan output suaranya masing-masing. |
| **Output** |  |

**Laporan Tugas 2: Data Hiding – Bank Toyyib**

|  |  |
| --- | --- |
| **Langkah** | **Praktikum** |
| **1.** | import tkinter as tk  from tkinter import messagebox  Untuk persiapan awal disini kita import tkinter dan messagebox.  class BankAccount:      def \_\_init\_\_(self):          self.\_\_\_balance = 0 # Private attribute      def get\_balance(self):          return self.\_\_\_balance      def deposit(self, amount):          if amount > 0:              self.\_\_\_balance += amount          else:              raise ValueError("Invalid deposit amount!")        def reset\_balance(self):          self.\_\_\_balance = 0  Disini kita membuat satu class Utama yaitu class BankAccount.  Atributnya balance dan dibuat Private dengan menambahkan underscore sebanyak 3 untuk menyembunyikan atribut ini dari akses public.  Class BankAccount ini memiliki 3 method yaitu get\_balance(), deposit(), dan reset\_balance(). Pembuatan method ini mengimplementasikan konsep Data Hiding. |
| **2.** | def add\_balance():      try:          amount = int(entry\_amount.get())          account.deposit (amount)          label\_balance.config(text=f"Balance: {account.get\_balance()}", font=("Arial", 20))      except ValueError as e:          messagebox.showerror("Error", str(e))  Selanjutnya kita membuat method add\_balance() di luar Class. Pembuatan method ini juga memperhatikan konsep Data Hiding. |
| **3.** | # GUI Tkinter  root = tk.Tk()  root.title("Bank Toyyib")  account = BankAccount()  label\_balance = tk.Label(root, text=f"Balance: {account.get\_balance()}", font=("Arial", 20))  label\_balance.pack()  entry\_amount = tk.Entry(root, font=("Arial", 20))  entry\_amount.pack()  button\_deposit = tk.Button(root, text="Deposit", command=add\_balance, font=("Arial", 20))  button\_deposit.pack()  root.mainloop()  Lalu untuk langkah terakhir. Kita membuat variable-variable yang dibutuhkan.  Root, pembuatan Objek account, entry\_amount, dan button\_deposit.  Sisanya memakai root.mainloop() untuk memproses looping Program agar terus berjalan. |
| **Tugas** | def reset\_balance(self):          self.\_\_\_balance = 0  def reset():      try:          account.reset\_balance()          label\_balance.config(text=f"Balance: {account.get\_balance()}", font=("Arial", 20))          entry\_amount.delete(0, tk.END)          entry\_amount.insert(0, 0)      except ValueError as e:          messagebox.showerror("Error", str(e))  reset\_button = tk.Button(root, text="Reset", command=reset, font=("Arial", 20))  reset\_button.pack()  Disini saya menambahkan tombol reset dengan integrasi reset\_balance di dalam Class, lalu fungsi reset diluar Class, setelah itu membuat reset\_button dengan tkinter. |
| **Output** | **Reset** |

**Laporan Tugas 3: Overriding – Kalkulasi Bentuk2**

|  |  |
| --- | --- |
| **Langkah** | **Praktikum** |
| **1.** | import tkinter as tk  from tkinter import messagebox  from math import pi  Oke langsung saja, import tkinter, messagebox, dan pi dari math.  class Shape:      def area(self):          return "Not implemented"      def perimeter(self):          return "Not implemented"  Buat class Shape, sebagai Parent Class Utama. |
| **2.** | class Rectangle(Shape):      def \_\_init\_\_(self, length, width):          self.length = length          self.width = width      def area(self):          return self.length \* self.width        def perimeter(self):          return 2 \* (self.length + self.width)  class Circle(Shape):      def \_\_init\_\_(self, radius):          self.radius = radius      def area(self):          return pi \* self.radius \*\* 2        def perimeter(self):          return 2 \* pi \* self.radius  Selanjutnya buat 2 Class tambahan yang inherit dari Shape. Yaitu Rectangle dan Circle, dengan override method dan penambahan atribut untuk masing-masing class. |
| **3.** | root = tk.Tk()  root.title("Kalkulasi Bentuk2")  shape\_var = tk.StringVar(value="Rectangle")  tk.Radiobutton (root, text="Rectangle", variable=shape\_var, value="Rectangle", font=("Arial", 20)).pack()  tk.Radiobutton (root, text="Circle", variable=shape\_var, value="Circle", font=("Arial", 20)).pack()  tk.Label(root, text="Parameter 1:", font=("Arial", 20)).pack()  entry\_param1 = tk.Entry(root, font=("Arial", 20))  entry\_param1.pack()  tk.Label(root, text="Parameter 2 (if Rectangle):", font=("Arial", 20)).pack()  entry\_param2 = tk.Entry(root, font=("Arial", 20))  entry\_param2.pack()  Buat dan siapkan variable-variable yang diperlukan.  Root, shape\_var.  Lalu buat UI yang diperlukan seperti pilihan Radiobutton dan Label. |
| **4.** | def calculate():      try:          shape\_type = shape\_var.get()          if shape\_type == "Rectangle":              shape = Rectangle(int(entry\_param1.get()), int(entry\_param2.get()))          elif shape\_type == "Circle":              shape = Circle(int (entry\_param1.get()))          else:              raise ValueError("Invalid shape")            label\_result.config(text=f"Area: {shape.area()}, Perimeter: {shape.perimeter()}", font=("Arial", 20))      except ValueError as e:          messagebox.showerror("Error", str(e))  Setelah itu buat fungsi Calculate diluar class-class sebelumnya.  Fungsi ini memiliki try except sebagai error handlingnya. Proses yang terjadi sederhananya memilih opsi kalkulasi, lalu menghitung dengan Class masing-masing. |
| **5.** | button\_calculate = tk.Button(root, text="Calculate", command=calculate)  button\_calculate.pack()  label\_result = tk.Label(root, text="Area:, Perimeter: ", font=("Arial", 20))  label\_result.pack()  root.mainloop()  Oke terakhir disini kita buat button dan label setelah itu kita run root.mainloop(). |
| **Tugas** | def reset():      try:          label\_result.config(text="Area:, Perimeter: ", font=("Arial", 20))            entry\_param1.delete(0, tk.END)          entry\_param1.insert(0, 0)          entry\_param2.delete(0, tk.END)          entry\_param2.insert(0, 0)      except ValueError as e:          messagebox.showerror("Error", str(e))  reset\_button = tk.Button(root, text="Reset", command=reset, font=("Arial", 20))  reset\_button.pack()  Disini kita buat fungsi reset untuk mengembalikan nilai dan input kembali ke 0.  Tidak lupa membuat reset\_button untuk UI nya. |
| **Output** |  |

**Laporan Tugas 4: Exception Handling – Bagi Membagi**

|  |  |
| --- | --- |
| **Langkah** | **Praktikum** |
| **1.** | import tkinter as tk  from tkinter import messagebox  Import tkinter dan messagebox.  def divide\_numbers():      try:          num1 = int (entry\_num1.get())          num2 = int (entry\_num2.get())          result = num1 / num2          label\_result.config(text=f"Result: {result}", font=("Arial", 20))      except ZeroDivisionError:          messagebox.showerror("Error", "Division by zero is not allowed.")      except ValueError:          messagebox.showerror("Error", "Please enter valid numbers.")  Buat fungsi divide\_numbers dengan try dan except untuk ExceptionHandling, spesifiknya disini kita memakai ZeroDivisionError, artinya kita tidak boleh membagi suatu angka dengan angka 0. |
| **2.** | root = tk.Tk()  root.title("Bagi Membagi")  tk.Label(root, text="Number 1:", font=("Arial", 20)).pack()  entry\_num1 = tk.Entry(root, font=("Arial", 20))  entry\_num1.pack()  tk.Label(root, text="Number 2:", font=("Arial", 20)).pack()  entry\_num2 = tk.Entry(root, font=("Arial", 20))  entry\_num2.pack()  button\_divide = tk.Button(root, text="Divide", command=divide\_numbers, font=("Arial", 20))  button\_divide.pack()  label\_result = tk.Label(root, text="Result: ", font=("Arial", 20))  label\_result.pack()  root.mainloop()  Kode utama membuat root.  Label masing-masing dan juga button.  Ditutup dengan root.mainloop(). |
| **Tugas** | def reset():      try:          label\_result.config(text="Result: ", font=("Arial", 20))            entry\_num1.delete(0, tk.END)          entry\_num1.insert(0, 0)          entry\_num2.delete(0, tk.END)          entry\_num2.insert(0, 0)      except ValueError as e:          messagebox.showerror("Error", str(e))  reset\_button = tk.Button(root, text="Reset", command=reset, font=("Arial", 20))  reset\_button.pack()  Buat reset\_button yang memanggil fungsi reset  Fungsi reset ini menggunakan Error Handling juga. Dengan proses utamanya untuk mengembalikan nilai entry dan label ke 0 atau kosong. |
| **Output** |  |

***--#BeraksiBerprestasiBersinergi--***