LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK (PBO) – 4

Disusun Oleh

ABEL FORTINO 123140111

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

INSTITUT TEKNOLOGI SUMATERA

2025

Soal nomor 1 Menghitung akar kuadrat

Program ini meminta pengguna untuk memasukkan angka.

Jika inputnya bukan angka, program akan menampilkan pesan *"Input tidak valid. Harap masukkan angka yang valid."*

Jika angka yang dimasukkan negatif atau nol, program akan menampilkan pesan error yang relevan, seperti *"Akar kuadrat dari nol tidak diperbolehkan."*

Jika inputnya valid (angka positif), program akan menghitung dan menampilkan akar kuadratnya.

Terapkan exception dan error handling dalam menangani eror yang terjadi di dalam program.

contoh output:

```
Masukkan angka: Hello
Input tidak valid. Harap masukkan angka yang valid.
Masukkan angka: -5
Input tidak valid. Harap masukkan angka positif.
Masukkan angka: 0
Error: Akar kuadrat dari nol tidak diperbolehkan.
Masukkan angka: 25
Akar kuadrat dari 25 adalah 5.0
```

Penjelasan

1. Import math: digunakan untuk menghitung akar kuadrat fungsi math.sqrt.
2. Def hitung_akar_kuadrat:
    - Jika input bukan angka, ValueError akan mengeluarkan output "Input tidak valid. Harap masukkan angka yang valid."
    - Jika angka yang dimasukkan negatif, program mengeluarkan output "Input tidak valid. Harap masukkan angka positif."
    - Jika angka yang dimasukkan adalah nol, program mengeluarkan output "Error: Akar kuadrat dari nol tidak diperbolehkan."
    - Jika input valid (angka positif), program akan menghitung dan menampilkan akar kuadratnya.

Source Code

```python
import math

def hitung_akar_kuadrat():
    while True:
        try:
            angka = input("Masukkan angka: ")
            angka = float(angka)
            if angka < 0:
                print("Input tidak valid. Harap masukkan angka positif.")
            elif angka == 0:
                print("Error: Akar kuadrat dari nol tidak diperbolehkan.")
            else:
                hasil = math.sqrt(angka)
                print(f"Akar kuadrat dari {angka} adalah {hasil}")
                break
        except ValueError:
            print("Input tidak valid. Harap masukkan angka yang valid.")

if __name__ == "__main__":
    hitung_akar_kuadrat()
```

Output

```
PS C:\Users\ASUS\Documents\Kuliah\Semester 4\PBO> &
Masukkan angka: 144
Akar kuadrat dari 144.0 adalah 12.0
PS C:\Users\ASUS\Documents\Kuliah\Semester 4\PBO> &
Masukkan angka: 90
Akar kuadrat dari 90.0 adalah 9.486832980505138
PS C:\Users\ASUS\Documents\Kuliah\Semester 4\PBO> &
Masukkan angka: Hai
Input tidak valid. Harap masukkan angka yang valid.
```

Soal Nomor 2 Manajemen Daftar Tugas (To-Do List)

Program ini meminta pengguna untuk menambahkan, menghapus, dan menampilkan daftar tugas.

Program ini menangani beberapa **exception** yang mungkin terjadi, seperti input yang tidak valid, mencoba menghapus tugas yang tidak ada, dan input kosong.

Terapkan exception dan error handling, serta penerapan raising exception dalam menangani error yang terjadi di dalam program.

contoh output:

```
Pilih aksi:
1. Tambah tugas
2. Hapus tugas
3. Tampilkan daftar tugas
4. Keluar
Masukkan pilihan (1/2/3/4): 1
Masukkan tugas yang ingin ditambahkan: Belajar Python
Tugas berhasil ditambahkan!

Pilih aksi:
1. Tambah tugas
2. Hapus tugas
3. Tampilkan daftar tugas
4. Keluar
Masukkan pilihan (1/2/3/4): 3
Daftar Tugas:
- Belajar Python

Pilih aksi:
1. Tambah tugas
2. Hapus tugas
3. Tampilkan daftar tugas
4. Keluar
Masukkan pilihan (1/2/3/4): 2
Masukkan nomor tugas yang ingin dihapus: 2
Error: Tugas dengan nomor 2 tidak ditemukan.

Pilih aksi:
1. Tambah tugas
2. Hapus tugas
3. Tampilkan daftar tugas
4. Keluar
Masukkan pilihan (1/2/3/4): 4
Keluar dari program.
```

Penjelasan

1. Def tambah_tugas: digunakan untuk menambahkan tugas ke dalam daftar tugas.
2. Def hapus_tugas: digunakan untuk menghapus tugas dari daftar tugas berdasarkan nomor tugas.
3. Def tampilkan_daftar_tugas: digunakan untuk menampilkan semua tugas yang ada dalam daftar tugas.
4. Enumerate: menghasilkan pasangan indeks dan elemen dari daftar_tugas.
5. Def main: merupakan menu untuk user dan memanggil fungsi yang sesuai dengan pilihan user.

Source code

```python
def tambah_tugas(daftar_tugas):
    tugas = input("Masukkan tugas yang ingin ditambahkan: ").strip()
    if not tugas:
        raise ValueError("Tugas tidak boleh kosong.")
    daftar_tugas.append(tugas)
    print("Tugas berhasil ditambahkan!")

def hapus_tugas(daftar_tugas):
    try:
        nomor_tugas = int(input("Masukkan nomor tugas yang ingin dihapus: "))
        if nomor_tugas < 1 or nomor_tugas > len(daftar_tugas):
            raise IndexError("Tugas dengan nomor tersebut tidak ditemukan.")
        tugas = daftar_tugas.pop(nomor_tugas - 1)
        print(f"Tugas '{tugas}' berhasil dihapus!")
    except ValueError:
        print("Input tidak valid. Harap masukkan nomor tugas yang valid.")
    except IndexError as e:
        print(f"Error: {e}")

def tampilkan_daftar_tugas(daftar_tugas):
    if not daftar_tugas:
        print("Daftar tugas kosong.")
    else:
        print("Daftar Tugas:")
        for index, tugas in enumerate(daftar_tugas, start=1):
            print(f"{index}. {tugas}")

def main():
    daftar_tugas = []
    while True:
        print("\nPilih aksi:")
        print("1. Tambah tugas")
        print("2. Hapus tugas")
        print("3. Tampilkan daftar tugas")
        print("4. Keluar")
        pilihan = input("Masukkan pilihan (1/2/3/4): ").strip()

        if pilihan == "1":
            try:
                tambah_tugas(daftar_tugas)
            except ValueError as e:
                print(f"Error: {e}")
        elif pilihan == "2":
            hapus_tugas(daftar_tugas)
        elif pilihan == "3":
            tampilkan_daftar_tugas(daftar_tugas)
        elif pilihan == "4":
            print("Keluar dari program.")
            break
        else:
            print("Pilihan tidak valid. Silakan coba lagi.")

if __name__ == "__main__":
    main()
```

Output

```
Pilih aksi:
1. Tambah tugas
2. Hapus tugas
3. Tampilkan daftar tugas
4. Keluar
Masukkan pilihan (1/2/3/4): 1
Masukkan tugas yang ingin ditambahkan: PBO
Tugas berhasil ditambahkan!

Pilih aksi:
1. Tambah tugas
2. Hapus tugas
3. Tampilkan daftar tugas
4. Keluar
Masukkan pilihan (1/2/3/4): 1
Masukkan tugas yang ingin ditambahkan: Basdat
Tugas berhasil ditambahkan!

Pilih aksi:
1. Tambah tugas
2. Hapus tugas
3. Tampilkan daftar tugas
4. Keluar
Masukkan pilihan (1/2/3/4): 3
Daftar Tugas:
1. PBO
2. Basdat

Pilih aksi:
1. Tambah tugas
2. Hapus tugas
3. Tampilkan daftar tugas
4. Keluar
Masukkan pilihan (1/2/3/4): 2
Masukkan nomor tugas yang ingin dihapus: 2
Tugas 'Basdat' berhasil dihapus!
```

Soal nomor 3 Sistem Manajemen Hewan (Zoo Management System)

Pada kasus ini, kalian akan membuat sebuah sistem untuk mengelola berbagai jenis hewan di kebun binatang.

Setiap hewan memiliki karakteristik dan perilaku yang berbeda, dan sistem ini akan mengimplementasikan konsep-konsep OOP yang telah dipelajari sebelumnya beserta eror handling yang kita pelajari hari ini.

Deskripsi Program:

1. **Polimorfisme:** Setiap hewan dapat memiliki metode `make_sound()` yang berbeda-beda sesuai dengan jenis hewan tersebut. Metode ini akan dipanggil melalui objek hewan yang berbeda jenis (misalnya, anjing, kucing, dll), kalo mau tambah metode lain boleh.
2. **Abstraksi:** Kalian akan membuat kelas abstrak `Animal` yang mendefinisikan metode `make_sound()` yang harus diimplementasikan oleh semua kelas turunan.
3. **Enkapsulasi:** Data terkait hewan akan disembunyikan di dalam kelas dan hanya bisa diakses atau dimodifikasi melalui metode tertentu.
4. **Inheritance:** Semua hewan akan mewarisi kelas dasar `Animal` dan dapat menambahkan perilaku atau atribut mereka sendiri.
5. **Exception Handling:** Program akan menangani error jika ada input yang tidak valid, misalnya, saat menambahkan hewan tanpa nama atau usia yang benar.

refreshing otak 4 pilar OOP:

1. Encapsulation: semua properti harus bersifat private dan hanya bisa diakses lewat method, getter untuk mengambil dan setter untuk merubah nilainya.
2. Inheritance: suatu class menjadi subclass dari superclass yang mana subclass mempunyai properti dan method yang sama dengan superclass.
3. Abstraction: Membuat class yang mana class tersebut merupakan blue print dari kelas lain, kelas yang untuk blueprint class lain tidak bisa dijadikan blueprint objek, dan juga akan memaksa class blueprint untuk memakai abstractmethod yang ada di dalam kelas abstrak(wajib dipake semuanya), abstractmethod di kelas abstract tidak perlu diimplementasikan di kelas abstract(isi 'pass' aja)
4. Polimorphism: Mengoverride suatu fungsi di dalam kelas induk di kelas anaknya

Penjelasan

1. Class Animal: Kelas yang mendefinisikan atribut name dan age dengan getter dan setter untuk enkapsulasi, make_sound memunculkan NotImplementedError untuk memastikan bahwa kelas turunan harus mengimplementasikan metode ini.
2. Class Dog, cat, bird: Kelas turunan dari Animal yang mengimplementasikan make_sound dengan suara yang sesuai.
3. Def main: merupakan menu untuk user dan memanggil fungsi yang sesuai dengan pilihan user.

Source code

```python
class Animal:
    def __init__(self, name, age):
        self.__name = name
        self.__age = age

    def make_sound(self):
        raise NotImplementedError("Subclasses must implement this method")

    def get_name(self):
        return self.__name

    def set_name(self, name):
        if not name:
            raise ValueError("Nama tidak boleh kosong.")
        self.__name = name

    def get_age(self):
        return self.__age

    def set_age(self, age):
        if age < 0:
            raise ValueError("Usia tidak boleh negatif.")
        self.__age = age

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

class Cat(Animal):
    def make_sound(self):
        return "Meow!"

class Bird(Animal):
    def make_sound(self):
        return "Tweet!"

def main():
    animals = []

    while True:
        print("\nPilih aksi:")
        print("1. Tambah hewan")
        print("2. Tampilkan suara hewan")
        print("3. Keluar")
        pilihan = input("Masukkan pilihan (1/2/3): ").strip()

        if pilihan == "1":
            try:
                jenis = input("Masukkan jenis hewan (Dog/Cat/Bird):
").strip().capitalize()
                name = input("Masukkan nama hewan: ").strip()
                age = int(input("Masukkan usia hewan: "))

                if jenis == "Dog":
                    animal = Dog(name, age)
                elif jenis == "Cat":
                    animal = Cat(name, age)
                elif jenis == "Bird":
                    animal = Bird(name, age)
                else:
                    raise ValueError("Jenis hewan tidak valid.")

                animals.append(animal)
                print(f"{jenis} bernama {name} berhasil ditambahkan!")
            except ValueError as e:
                print(f"Error: {e}")
        elif pilihan == "2":
            if not animals:
                print("Tidak ada hewan di kebun binatang.")
            else:
                for animal in animals:
                    print(f"{animal.get_name()} ({animal.__class__.__name__}):
{animal.make_sound()}")
        elif pilihan == "3":
            print("Keluar dari program.")
            break
        else:
            print("Pilihan tidak valid. Silakan coba lagi.")

if __name__ == "__main__":
    main()
```

Output

```
Pilih aksi:
1. Tambah hewan
2. Tampilkan suara hewan
3. Keluar
Masukkan pilihan (1/2/3): 1
Masukkan jenis hewan (Dog/Cat/Bird): Dog
Masukkan nama hewan: Abel
Masukkan usia hewan: 20
Dog bernama Abel berhasil ditambahkan!

Pilih aksi:
1. Tambah hewan
2. Tampilkan suara hewan
3. Keluar
Masukkan pilihan (1/2/3): 2
Abel (Dog): Woof!

Pilih aksi:
1. Tambah hewan
2. Tampilkan suara hewan
3. Keluar
Masukkan pilihan (1/2/3): 1
Masukkan jenis hewan (Dog/Cat/Bird): Abel
Masukkan nama hewan: Abel
Masukkan usia hewan: -20
Error: Jenis hewan tidak valid.
```