Laporan Praktikum Ke-7

Pemograman Berorientasi Objek

Nama : Stefen Tjung

NIM : 121140057

Kelas : RB

Abstract Class

Sebuah kelas abstrak dapat dianggap sebagai cetak biru (blueprint) untuk kelas-kelas lainnya. Hal ini memungkinkan Anda untuk membuat sekumpulan metode yang harus dibuat dalam setiap kelas turunan yang dibangun dari kelas abstrak tersebut. Kelas yang mengandung satu atau lebih metode abstrak disebut kelas abstrak. Metode abstrak adalah metode yang memiliki deklarasi namun tidak memiliki implementasi. Ketika kita merancang unit-unit fungsional yang besar, kita menggunakan kelas abstrak. Ketika kita ingin memberikan antarmuka umum untuk implementasi-implementasi yang berbeda dari sebuah komponen, kita menggunakan kelas abstrak.

Secara default, Python tidak menyediakan kelas-kelas abstrak. Cara kerja Abstract Class ini sendiri adalah python dilengkapi dengan modul yang menyediakan dasar untuk mendefinisikan kelas-kelas Abstract Base (ABC), dan nama modul tersebut adalah ABC. ABC bekerja dengan mendekorasi metode-metode kelas dasar sebagai abstrak, dan kemudian mendaftarkan kelas-kelas konkret sebagai implementasi dari basis abstrak tersebut. Sebuah metode menjadi abstrak ketika didekorasi dengan kata kunci @abstractmethod.

Contoh Implementasi Abstract Class adalah sebagai berikut :

```python
from abc import ABC, abstractmethod

class Shape(ABC):
    @abstractmethod
    def area(self):
        pass

    @abstractmethod
    def perimeter(self):
        pass

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 3.14 * self.radius ** 2

    def perimeter(self):
        return 2 * 3.14 * self.radius

# Membuat objek-objek dari kelas turunan Shape
rectangle = Rectangle(5, 3)
circle = Circle(7)

# Memanggil metode-metode dari objek-objek tersebut
print("Luas persegi panjang:", rectangle.area())
print("Keliling persegi panjang:", rectangle.perimeter())

print("Luas lingkaran:", circle.area())
print("Keliling lingkaran:", circle.perimeter())
```

Interface

Python menyediakan beberapa opsi untuk mengembangkan Antarmuka Pengguna Grafis (Graphical User Interface/GUI). Dari semua metode GUI tersebut, tkinter adalah metode yang paling umum digunakan. Ini adalah antarmuka Python standar untuk toolkit GUI Tk yang dikirimkan dengan Python. Python dengan tkinter adalah cara yang paling cepat dan mudah untuk membuat aplikasi GUI. Membuat GUI menggunakan tkinter adalah tugas yang mudah.

Untuk membuat aplikasi tkinter:

1. Impor modul - tkinter.

2. Buat jendela utama (container).

3. Tambahkan sejumlah widget ke jendela utama.

4. Terapkan pemicu acara pada widget-widget tersebut.

Impor tkinter dilakukan dengan cara yang sama seperti mengimpor modul lain dalam kode Python. Perhatikan bahwa nama modulnya adalah 'Tkinter' untuk Python 2.x dan 'tkinter' untuk Python 3.x.

Ada dua metode utama yang perlu diingat oleh pengguna saat membuat aplikasi Python dengan GUI:

1. `Tk(screenName=None, baseName=None, className='Tk', useTk=1)`: Untuk membuat jendela utama, tkinter menyediakan metode `Tk(screenName=None, baseName=None, className='Tk', useTk=1)`. Untuk mengubah nama jendela, Anda dapat mengganti className sesuai yang diinginkan.

2. `mainloop()`: Terdapat metode yang dikenal dengan nama mainloop() yang digunakan ketika aplikasi Anda siap dijalankan. mainloop() adalah sebuah perulangan tak terbatas yang digunakan untuk menjalankan aplikasi, menunggu kejadian terjadi, dan memproses kejadian tersebut selama jendela tidak ditutup.

Contoh Implementasi Interface/GUI pada python adalah

```python
import tkinter as tk

def button_click():
    label.config(text="Button clicked!")

# Membuat jendela utama
window = tk.Tk()

# Membuat label
label = tk.Label(window, text="Hello, GUI World!")
label.pack()

# Membuat tombol
button = tk.Button(window, text="Click Me!", command=button_click)
button.pack()

# Menjalankan jendela utama
window.mainloop()
```

MetaClass

Metakelas adalah kelas yang secara langsung mewarisi dari tipe (type). Metode yang harus diimplementasikan oleh metakelas kustom adalah metode __new__. Argumen yang disebutkan dalam metode __new__ metakelas mencerminkan dalam metode __new__ kelas tipe (type). Terdapat empat argumen posisi. Berikut adalah penjelasan mengenai keempat argumen tersebut:

1. Argumen pertama adalah metakelas itu sendiri.
2. Argumen kedua adalah nama kelas.
3. Argumen ketiga adalah superclass (dalam bentuk tuple).
4. Argumen keempat adalah atribut-atribut kelas (dalam bentuk kamus/dictionary).

Contoh Implementasi MetaClass Pada program python adalah

```python
class MyMeta(type):
    def __new__(cls, name, bases, attrs):
        # Melakukan modifikasi pada atribut 'attrs'
        attrs['new_attribute'] = 100

        # Membuat instance baru dari kelas dengan metakelas yang dimodifikasi
        new_class = super().__new__(cls, name, bases, attrs)

        return new_class

# Menggunakan metakelas dalam mendefinisikan kelas
class MyClass(metaclass=MyMeta):
    my_attribute = 42

# Membuat objek dari kelas dengan metakelas
obj = MyClass()

# Mengakses atribut yang ditambahkan oleh metakelas
print(obj.new_attribute)  # Output: 100
```

Kesimpulan

Graphical User Interface (GUI) adalah suatu antarmuka pengguna (user interface) di mana pengguna berinteraksi melalui objek-objek grafis (yang disebut widgets), seperti tombol (button), checkbox, menu, dsb. Dalam bahasa Python, kita dapat memanfaatkan module tkinter untuk membuat GUI. GUI digunakan pada saat membuat program yang rapi dan membutuhkan tampil objek yang menyampaikan informasi dan mewakili tindakan yang dapat diambil oleh pengguna.

Kelas abstrak adalah kelas yang masih dalam bentuk abstrak. Karena bentuknya masih abstrak, dia tidak bisa dibuat langsung menjadi objek. Kelas Abstrak digunakan pada saat membuat program yang dapat menentukan antarmuka Program Aplikasi (API) umum untuk sekelompok kelas anak. Kemampuan ini sangat berguna dalam situasi di mana pihak ketiga akan menyediakan implementasi, seperti dengan plugin, namun juga dapat membantu saat bekerja dalam tim besar atau dengan basis kode yang besar di mana sulit atau tidak mungkin untuk mengingat semua kelas dalam pikiran Anda.

Kelas Konkret adalah kelas dipython yang tidak memiliki kelas yang tidak ada implementasinya didalam kelas tersebut, atau bisa dibilang kelas konkret adalah kelas yang tidak memiliki kelas abstract didalamnya. Kelas konkret digunakan pada saat program program dibutuhkan detail yang berbeda beda didalamnya dan setiap tim harus mengerti detail sehingga dapat digunakan apabila sewaktu waktu kode tersebut ingin dimodifikasi.

Metakelas adalah kelas yang secara langsung mewarisi dari tipe (type). Sederhananya Metakelas digunakan untuk menyesuaikan perilaku sebuah kelas, sementara pewarisan (inheritance) digunakan untuk membuat kelas baru yang mewarisi atribut dan metode dari kelas yang sudah ada.

DAFTAR PUSTAKA

https://www.geeksforgeeks.org/abstract-classes-in-python/

https://www.geeksforgeeks.org/python-gui-tkinter/

https://www.geeksforgeeks.org/python-metaclasses/

https://www.tutorialspoint.com/differences-between-abstract-class-and-concrete-class-in-java

https://www.petanikode.com/java-oop-abstract/