UJIAN TENGAH SEMESTER – Praktikum Pemrograman Berorientasi Objek

Nama: Abdul Hakeem Putra Porfiriko

NIM: 23/517061/SV/22738

1. Susunlah program menggunakan Python untuk mengkonversi input angka dalam satuan mil menjadi angka dalam satuan kilometer! Gunakan fungsi print untuk menampilkan output dengan format "… mil = … kilometer".

```python
mile = float(input())

mtk = '%.3f'%(mile*1.60934)

print(f"{mile} Mil = {mtk} Kilometer")
```

```
hakeem@tpkd-ThinkPad  UTS  master  python .\UTS-CPMK1.py
62
62.0 Mil = 99.779 Kilometer
hakeem@tpkd-ThinkPad  UTS  master
```

2. Jelaskan pengertian dari istilah berikut:
    a. Class

Class merupakan sebuah cetak biru yang dapat dibuat oleh pengguna, untuk membuat sebuah object. Class berfungsi untuk membungkus data dan fungsionalitas menjadi satu paket yang dapat digunakan berkali-kali.

    b. Object

Object merupakan sebuah instance dari sebuah class, dengan nilai yang diberikan yang kemudian dapat dioperasikan dengan method yang telah ada dalam class.

    c. Inheritance

Inheritance adalah sebuah konsep yang mengizinkan sebuah class untuk mewariskan method dan property dari class lain. Class yang mewarisi (inherits) dari class lain disebut parent class, sementara class yang mendapatkan method dan property yang diwariskan adalah child class.

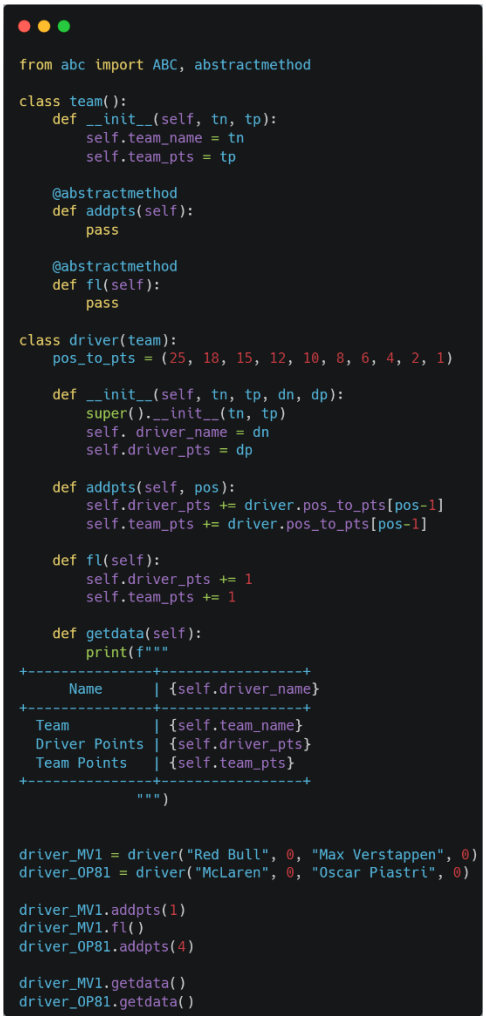
    d. Polymorphism

Polymorphism adalah kemampuan suatu objek untuk memiliki banyak bentuk. Bentuk-bentuk ini dapat dibuat dengan mewariskan satu parent class pada dua atau lebih child class. Semua class dapat digunakan bersamaan, namun output yang dapat muncul dapat berbeda, tergantung pada class yang digunakan.

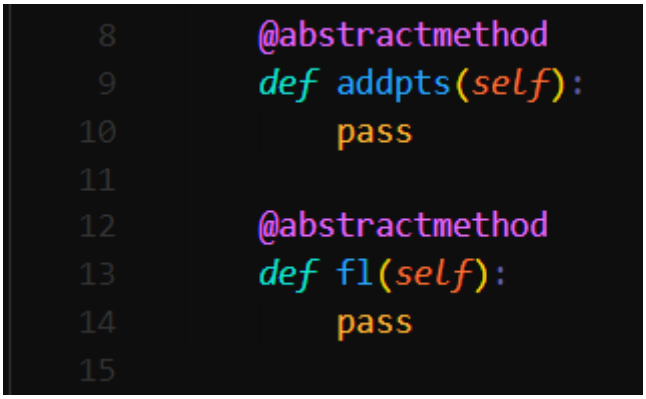
e. Abstract Base Class

Abstract base class meurpakan sebuah class yang memiliki fungsi virtual dan dapat digunakan sebagai cetak biru untuk class lain yang mewarisi dari sebuah abstract base class. Child class tersebut kemudian dapat mengimplementasikan fungsi/method tersebut secara independen. (read: polymorphism but no fallback method)

3. Susunlah sebuah program Python yang di dalamnya menggunakan konsep inheritance, polymorphism, dan abstract base class. Berikan penjelasan singkat di bagian mana konsep tersebut digunakan pada program.

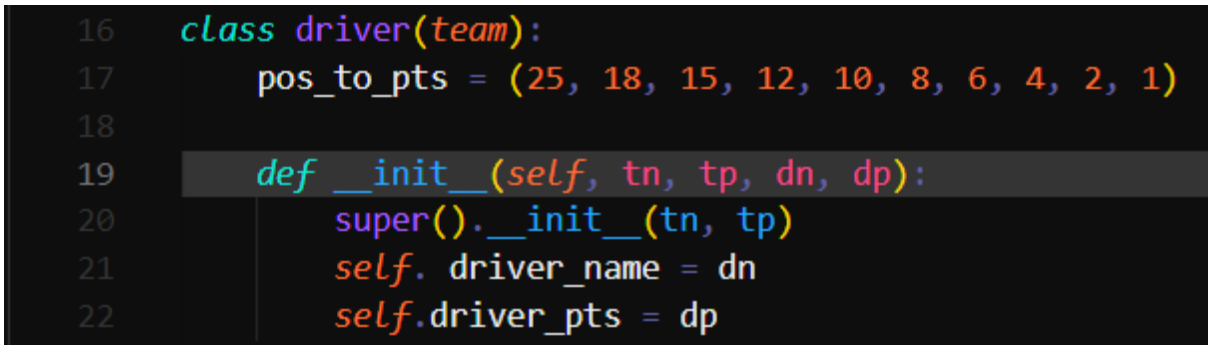


```python
from abc import ABC, abstractmethod

class team():
    def __init__(self, tn, tp):
        self.team_name = tn
        self.team_pts = tp

    @abstractmethod
    def addpts(self):
        pass

    @abstractmethod
    def fl(self):
        pass

class driver(team):
    pos_to_pts = (25, 18, 15, 12, 10, 8, 6, 4, 2, 1)

    def __init__(self, tn, tp, dn, dp):
        super().__init__(tn, tp)
        self. driver_name = dn
        self.driver_pts = dp

    def addpts(self, pos):
        self.driver_pts += driver.pos_to_pts[pos-1]
        self.team_pts += driver.pos_to_pts[pos-1]

    def fl(self):
        self.driver_pts += 1
        self.team_pts += 1

    def getdata(self):
        print(f"""
+---------------+----------------+
      Name      | {self.driver_name}
+---------------+----------------+
  Team          | {self.team_name}
  Driver Points | {self.driver_pts}
  Team Points   | {self.team_pts}
+---------------+----------------+
              """)


driver_MV1 = driver("Red Bull", 0, "Max Verstappen", 0)
driver_OP81 = driver("McLaren", 0, "Oscar Piastri", 0)

driver_MV1.addpts(1)
driver_MV1.fl()
driver_OP81.addpts(4)

driver_MV1.getdata()
driver_OP81.getdata()
```



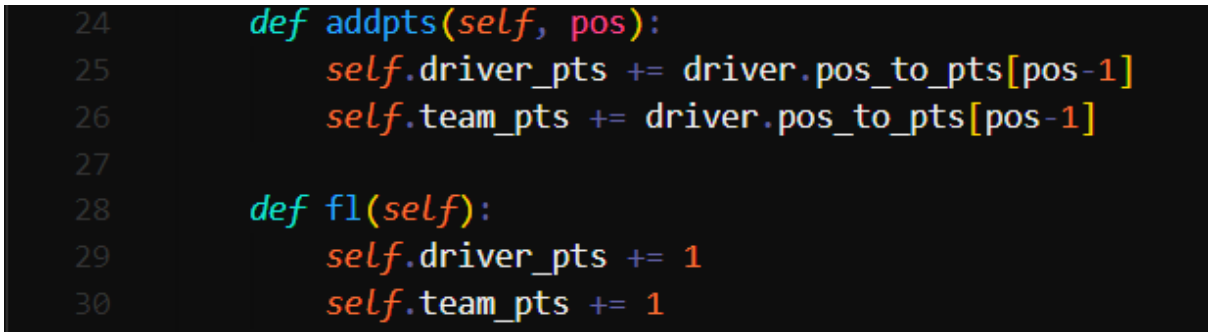
```
8        @abstractmethod
9        def addpts(self):
10           pass
11
12       @abstractmethod
13       def fl(self):
14           pass
15
```

Gambar 1: Implementasi Abstract Base Classs



```
16   class driver(team):
17       pos_to_pts = (25, 18, 15, 12, 10, 8, 6, 4, 2, 1)
18
19       def __init__(self, tn, tp, dn, dp):
20           super().__init__(tn, tp)
21           self. driver_name = dn
22           self.driver_pts = dp
```

Gambar 2: Implementasi Inheritance



```
24       def addpts(self, pos):
25           self.driver_pts += driver.pos_to_pts[pos-1]
26           self.team_pts += driver.pos_to_pts[pos-1]
27
28       def fl(self):
29           self.driver_pts += 1
30           self.team_pts += 1
```

Gambar 3: Implementasi Polymorphism (function inherited dari class team, lalu dapat diimplementasikan)

4. Buatlah sebuah program dengan ketentuan sebagai berikut. Pada program tersebut buatlah parent class dengan nama Rekening dan child class dengan nama Nasabah. Parent class memiliki 3 atribut, yaitu Nama, No. Rekening, dan Saldo. Child class memiliki method untuk menampilkan data nasabah (3 atribut). Setelah membuat kelas, lanjutkan dengan membuat dua object dari class Nasabah dengan data sebagai berikut.

   Selanjutnya panggil method untuk menampilkan data nasabah dan screenshot output program.

```python
class rekening():
    def __init__(self, name:str, accnum:int,
balance:float):
        self.name = name
        self.accnum = accnum
        self.balance = balance

class customer(rekening):
    def __init__(self, name, accnum, balance):
        super().__init__(name, accnum, balance)

    def showdata(self):
        print(f"Nama: {self.name}")
        print(f"No. Rekening: {self.accnum}")
        print(f"Saldo: {self.balance}")



nasabah1 = customer("Budi", 5555, 500000)
nasabah2 = customer("Wati", 6666, 2000000)

nasabah1.showdata()
nasabah2.showdata()
```

```
hakeem@tpkd-ThinkPad  UTS  master  python .\UTS-CPMK3-SOAL4.PY
Nama: Budi
No. Rekening: 5555
Saldo: 500000
Nama: Wati
No. Rekening: 6666
Saldo: 2000000
hakeem@tpkd-ThinkPad  UTS  master
```

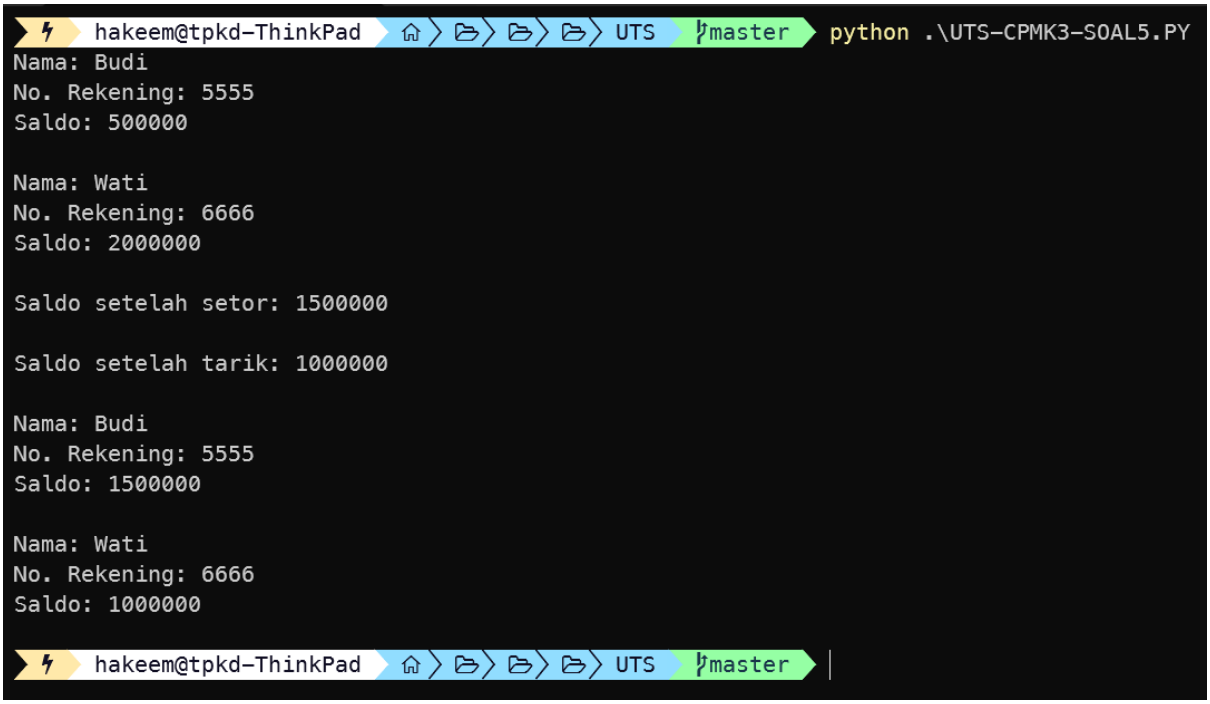
5. Kembangkan program pada soal no. 4 sebagai berikut. Pada class Rekening, tambahkan method Setor_tunai dan Tarik_tunai. Kedua method tersebut tentunya akan berpengaruh pada nominal Saldo dari Nasabah. Selanjutnya, simulasikan transaksi ATM secara sederhana sebagai berikut.

   • Nasabah 1 setor tunai 1.000.000

   • Nasabah 2 tarik tunai 1.000.000

   Tampilkan juga data nasabah setelah transaksi tersebut.

```python
class rekening():
    def __init__(self, name:str, accnum:int,
balance:float):
        self.name = name
        self.accnum = accnum
        self.balance = balance

class customer(rekening):
    def __init__(self, name, accnum, balance):
        super().__init__(name, accnum, balance)

    def showdata(self):
        print(f"Nama: {self.name}")
        print(f"No. Rekening: {self.accnum}")
        print(f"Saldo: {self.balance}\n")

    def deposit(self, amount):
        self.balance += amount
        print(f"Saldo setelah setor: {self.balance}\n")

    def withdraw(self, amount):
        if amount > self.balance:
            print("Saldo tidak mencukupi")
        else:
            self.balance -= amount
            print(f"Saldo setelah tarik: {self.balance}\n")



nasabah1 = customer("Budi", 5555, 500000)
nasabah2 = customer("Wati", 6666, 2000000)

nasabah1.showdata()
nasabah2.showdata()

nasabah1.deposit(1000000)
nasabah2.withdraw(1000000)

nasabah1.showdata()
nasabah2.showdata()
```



6. Kembangkan lagi program dari soal no. 5 dengan menambahkan method Transfer pada class Rekening. Method transfer akan mengurangi saldo pengirim dan pada saat yang sama akan menambah saldo penerima. Selanjutnya simulasikan transaksi transfer sebagai berikut:

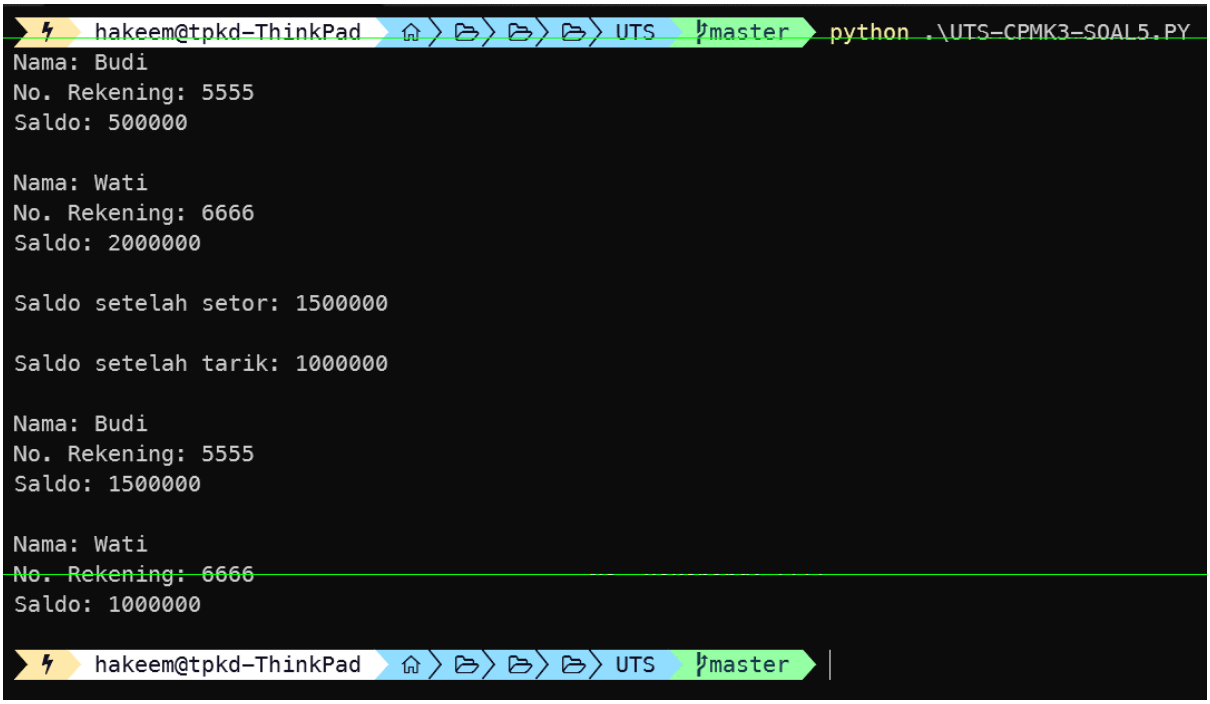
   • Pengirim : Nasabah 1

   • Penerima : Nasabah 2

   • Nominal transfer : 500.000

   Pada method Transfer, tambahkan juga laporan transaksi transfer misalnya “Transfer sebesar xxxx dari rekening xxxx ke rekening xxxx berhasil”. Tampilkan juga data nasabah setelah transaksi tersebut.

```python
# IMPORTANT: ZIP THIS FILE. NOT THE OTHER ONE #
class rekening():
    def __init__(self, name:str, accnum:int, balance:float):
        self.name = name
        self.accnum = accnum
        self.balance = balance

class customer(rekening):
    def __init__(self, name, accnum, balance):
        super().__init__(name, accnum, balance)

    def showdata(self):
        print(f"Nama: {self.name}")
        print(f"No. Rekening: {self.accnum}")
        print(f"Saldo: {self.balance}\n")

    def deposit(self, amount):
        self.balance += amount
        print(f"Saldo {self.name} setelah setor: {self.balance}\n")

    def withdraw(self, amount):
        if amount > self.balance:
            print("Saldo tidak mencukupi")
        else:
            self.balance -= amount
            print(f"Saldo {self.name} setelah tarik: {self.balance}\n")

    def transfer(self, recepient, amount):
        if amount > self.balance:
            print("Saldo tidak mencukupi")
        else:
            self.balance -= amount
            recepient.balance += amount
            print(f"Saldo {self.name} setelah transfer: {self.balance}\n")
            print(f"Saldo {recepient.name} setelah transfer: {recepient.balance}\n")



nasabah1 = customer("Budi", 5555, 500000)
nasabah2 = customer("Wati", 6666, 2000000)

nasabah1.showdata()
nasabah2.showdata()

methodchoice = int(input("Pilih metode untuk didemonstrasikan:\n1. Setor dan Tarik\n2. Transfer\n"))

if methodchoice == 1:
    nasabah1.deposit(1000000)
    nasabah2.withdraw(1000000)
    nasabah1.showdata()
    nasabah2.showdata()
elif methodchoice == 2:
    nasabah1.transfer(nasabah2, 500000)
    nasabah1.showdata()
    nasabah2.showdata()
else:
    print("Pilihan tidak valid")
```


hakeem@tpkd-ThinkPad
UTS
master
python .\UTS-CPMK3-SOAL5.PY
Nama: Budi
No. Rekening: 5555
Saldo: 500000
Nama: Wati
No. Rekening: 6666
Saldo: 2000000
Saldo setelah setor: 1500000
Saldo setelah tarik: 1000000
Nama: Budi
No. Rekening: 5555
Saldo: 1500000
Nama: Wati
No. Rekening: 6666
Saldo: 1000000
hakeem@tpkd-ThinkPad
UTS
master