Homework 3

By 張皓鈞

OOP的外送員派單系統，含銀行系統

Account Class

帳戶類

In [ ]:
```
class Account:

    def __init__(self, money: float):
        self.money = money

    @property
    def money(self) -> float:
        return self._money

    @money.setter
    def money(self, money: float):
        self._money = money
```

Deliveryman Class

外送員類

In [ ]:
```
class Deliveryman(Account):

    def __init__(self, name: str, pos: tuple, money = 10000):
        self._name = name
        self._pos = pos
        super().__init__(money)

    def __str__(self):
        return "{}({}, {})".format(self._name, self._pos[0], self._pos[1])

    @property
    def name(self) -> str:
        return self._name
```

```
    @property
    def pos(self) -> tuple:
        return self._pos

    @pos.setter
    def pos(self, pos: tuple):
        self._pos = pos
```

Buyer Class

顧客類

In [ ]:

```
class Buyer:

    def __init__(self, name: str, pos: tuple):
        self._name = name
        self._pos = pos

    def __str__(self):
        return "{}({}, {})".format(self._name, self._pos[0], self._pos[1])

    @property
    def name(self) -> str:
        return self._name

    @property
    def pos(self) -> tuple:
        return self._pos

    @pos.setter
    def pos(self, pos: tuple):
        self._pos = pos
```

Path Class

路徑類

In [ ]:

```
class Path:

    @staticmethod
    def getTwoPosDistance(pos1: tuple, pos2: tuple) -> float:
        """
```

```
        計算兩點的曼哈頓距離

        Args:
            pos1 (tuple): 點1
            pos2 (tuple): 點2

        Returns:
            float: 曼哈頓距離
        """
        distance = abs( pos1[0] - pos2[0] ) + abs( pos1[1] - pos2[1] )
        return distance

    @staticmethod
    def getLowestCostPath(deliveryman: Deliveryman, buyers: list[Buyer]) -> list:
        """
        計算外送最短段路徑

        Args:
            deliveryman (Deliveryman): 外送員
            buyers (list[Buyer]): 買家列表

        Returns:
            list: 路徑列表
        """
        buyers_sorted = sorted(buyers, key = lambda buyer: Path.getTwoPosDistance(deliveryman.pos, buyer.pos))
        path = list()
        last_pos = deliveryman.pos
        for buyer in buyers_sorted:
            path.append({
                "buyer": buyer,
                "distance": Path.getTwoPosDistance(last_pos, buyer.pos)
            })
            last_pos = buyer.pos
        return path
```

Money Class

錢類

In [ ]:

```
class Money:

    @staticmethod
    def getDeliveryBonus(distance: float) -> float:
        """
        計算外送距離獎勵金
```

```
        Args:
            distance (float): 距離

        Returns:
            float: 獎勵金
        """
        return distance * 0.01 * 5

    @staticmethod
    def saveMoney(account: Account, money: float):
        """
        為指定帳戶存款

        Args:
            account (Account): 帳戶
            money (float): 金額
        """
        account.money += money

    @staticmethod
    def withdrawMoney(account: Account, money: float):
        """
        為指定帳戶提腳

        Args:
            account (Account): 帳戶
            money (float): 金額

        Raises:
            ValueError: 餘額不足
        """
        if account.money >= money:
            account.money -= money
        else:
            raise ValueError("存戶餘額不足！")
```

Main

主程式

In [ ]:
```
import random
import math
```

In [ ]:
```
deliveryman_name = ["甲外送員", "乙外送員"]
```

```
deliverymans = [Deliveryman(name, (random.randint(-1000, 1000), random.randint(-1000, 1000))) for name in deliveryman_name]

print("外送員資訊")
print("名稱\t座標")
for deliveryman in deliverymans:
    print("{}\t{}".format(deliveryman.name, deliveryman.pos))
```

```
外送員資訊
名稱    座標
甲外送員  (-117, 19)
乙外送員  (185, -932)
```

In [ ]:

```
buyer_name = ["買家A", "買家B", "買家C"]
buyers = [Buyer(name, (random.randint(-1000, 1000), random.randint(-1000, 1000))) for name in buyer_name]

print("買家資訊")
print("名稱\t座標")
for buyer in buyers:
    print("{}\t{}".format(buyer.name, buyer.pos))
```

```
買家資訊
名稱    座標
買家A   (-522, -384)
買家B   (-140, -394)
買家C   (363, 555)
```

In [ ]:

```
minDis = math.inf
minDeliveryman = deliverymans[0]
minBuyer = buyers[0]
for deliveryman in deliverymans:
    for buyer in buyers:
        dis = Path.getTwoPosDistance(deliveryman.pos, buyer.pos)
        if dis < minDis:
            minDis = dis
            minDeliveryman = deliveryman
            minBuyer = buyer
print("{} 離 {} 最近，獲得整筆訂單".format(minDeliveryman, minBuyer))
```

```
甲外送員(-117, 19) 離 買家B(-140, -394) 最近，獲得整筆訂單
```

In [ ]:

```
path = Path.getLowestCostPath(minDeliveryman, buyers)
print("最佳運送路徑")
print("{} -> ".format(minDeliveryman), end='')
for p in path:
    print("{}({}) -> ".format(p['buyer'], p['distance']), end='')
print("完成")
```

```
最佳運送路徑
甲外送員(-117, 19) -> 買家B(-140, -394)(436) -> 買家A(-522, -384)(392) -> 買家C(363, 555)(1824) -> 完成
```

In [ ]:

```
max_distance = sorted(path, key = lambda p: p['distance'], reverse = True)[0]
base_salary = 100
bonus = Money.getDeliveryBonus(max_distance['distance'])
salary = base_salary + bonus
Money.saveMoney(minDeliveryman, salary)
print("本次訂單 {} 獲得 基本費${} + 距離獎勵${}".format(minDeliveryman.name, base_salary, bonus))
```

```
本次訂單 甲外送員 獲得 基本費$100 + 距離獎勵$91.20000000000002
```

In [ ]:

```
print("外送員存款")
print("編號\t名稱\t餘額")
i = 0
for deliveryman in deliverymans:
    print("{}\t{}\t${}".format(i + 1, deliveryman.name, deliveryman.money))
    i += 1
```

```
外送員存款
編號	名稱	餘額
1	甲外送員	$10191.2
2	乙外送員	$10000
```

In [ ]:

```
print("請選擇要提領的存戶編號")
try:
    id = int(input()) - 1
    money = abs( float(input()) )
except ValueError:
    print("格式錯誤！")
if id != None:
    if id >= 0 and id < len(deliverymans):
        Money.withdrawMoney(deliverymans[id], money)
        print("從 {} 帳戶提款 ${} 完成，餘額 ${}".format(deliverymans[id].name, money, deliverymans[id].money))
    else:
        print("編號不存在！")
```

```
請選擇要提領的存戶編號
從 甲外送員 帳戶提款 $191.0 完成，餘額 $10000.2
```