Name_____**StudentID**_____

Basic requirements for working with object-defining classes

- Function __str__ is expected to be added to every class although it is not specified in the writing.
- **Attributes should be declared as private unless stated otherwise.**
- To access the value of each attribute, include getter (or accessor) method or set up property of function with the same as the attribute name accordingly.
- Limit the outside class NOT to change the value of each attribute unless it is necessary.

Level 1

One factory produces clothes. The owner decides to write a program to help out handling clothing sales.

In this level 1, declare 3 classes to represent clothing items, stock of clothing items at the factory, and orders from customer.

1.1 Define class **Item** such that there are 3 attributes:

- *id:* clothing ID. Each clothing item will not have the same clothing ID.
- *type*: clothing type. For example, T-shirt, Polo-shirt, Shirt, Pants.
- *color:* clothing color

In addition, add a method __eq__ Note that two items are equal if they have the same clothing ID.

See output to run Item class at the end of 1.4

1.2 Define class **Stock** such that there are 3 attributes:

- *item*: clothing Item object (from class Item)
- *amount:* number of this clothing item in the factory stock
- *price:* price of one clothing item

Example: Let the clothing item be white T-shirt. If the factory stocks 100 white T-shirts, where each white T-shirt costs 60 Baht, then *amount* for stock is 100, and *price* is 60 Baht (per one white T-shirt).

See output to run Stock class at the end of 1.4

1.3 Define class **Order** such that there are 4 attributes:

- *item*: clothing Item object (from class Item)
- *status:* status of this order. There are 3 statuses: To process, Delivered, Insufficient.

Order status value	Meaning
To process	Status when an order first comes to factory
Delivered	When factory has enough clothing item to deliver for customer, the order is successfully delivered.
Insufficient	When factory has NOT enough clothing item for customer, the order is marked as insufficient.

- *amount*: pieces of this clothing item that customer would like to buy
- cost: cost of this clothing item. If customer would like to buy n pieces of this clothing item, cost will be the sum of prices for n pieces. The default value for cost is zero Baht.

Example: Let the clothing item be white T-shirt. If the customer would like to buy 10 white T-shirts, where the factory sells one white T-shirt for 60 Baht, then *amount* for order is 10, and *cost* is 60x10 = 600 Baht (for 10 white T-shirts).
Note that cost should be updated after determining whether the factory has enough stock to deliver for customer.

See output to run Order class at the end of 1.4

1.4 In run_level1.py, add 4 more functions as specified below.

Function Name	Inputs	Outputs	Functionality
print_stock_list	A list of stock objects	-	Print information of stock objects.
print_item_list	A list of item objects	-	Print information of item objects.
print_order_list	A list of order objects	-	Print information of order objects.
process_order_list	A list of order objects and a list of order objects	-	For each order in the list of order objects, If status of order is not 'Delivered', and - if amount of order is not more than amount of stock, then (1) deduct the stock amount with amount of order, (2) update cost and update status of order as 'Delivered' - if amount of order is more than amount of stock, then update status of order as 'Insufficient'. Note that if stock is not enough, nothing will be delivered

See output to run 1.4 below.

Output after running_level1.py

run_level1.py	Output

`from item import *`
`from stock import *`
`from order import *`
 	
`item1 = Item(1,'T-shirt', 'White')`
`item2 = Item(2,'T-shirt', 'Black')`
`item3 = Item(3,'Polo-shirt', 'White')`
`item4 = Item(4,'Polo-shirt', 'Green')`
`item5 = Item(5,'Shirt', 'Green')`
`item6 = Item(6,'Shirt', 'Black')`
`print(item1)`
`print(item2)`
 	
`print(item1 == item2)`
`print(item2 == item2)`
 	
`stock1 = Stock(item1, 100, 60)`
`stock2 = Stock(item2, 100, 90)`
`stock3 = Stock(item3, 100, 120)`
`stock4 = Stock(item4, 100, 140)`
`stock5 = Stock(item5, 100, 200)`
`stock6 = Stock(item6, 100, 220)`
`print(stock3)`
`print(stock5)`
	`1,T-shirt,White`
	`2,T-shirt,Black`
	
	`False`
	`True`
	
	`(3,Polo-shirt,White,100,120)`
	`(5,Shirt,Green,100,200)`

`order1 = Order(item1, 10)` `print(order1)`	`(1,T-shirt,White,10,0 Baht,To process)`
`## Add your own code to create a list of stocks here`	

`print_stock_list(stock_list)`	`Stock List:`
	`(1,T-shirt,White,100,60)`
	`(2,T-shirt,Black,100,90)`
	`(3,Polo-shirt,White,100,120)`
	`(4,Polo-shirt,Green,100,140)`
	`(5,Shirt,Green,100,200)`
	`(6,Shirt,Black,100,220)`

`## Add your own code to create a list of 6 items here`	

`print_item_list(item_list)`	`Item List:`
	`1,T-shirt,White`
	`2,T-shirt,Black`
	`3,Polo-shirt,White`
	`4,Polo-shirt,Green`
	`5,Shirt,Green`
	`6,Shirt,Black`

`## Add your own code to read orders from user and use them to create a list of orders`	`Read order list from user…`
	`Enter item id (negative to quit): 1`
	`Enter amount: 10`
	`Enter item id (negative to quit): 2`
	`Enter amount: 20`
	`Enter item id (negative to quit): 3`
	`Enter amount: 300`
	`Enter item id (negative to quit): -1`

`print_order_list(order_list)`	`Order List:`
	`(1,T-shirt,White,10,0 Baht,To process)`
	`(2,T-shirt,Black,20,0 Baht,To process)`
	`(3,Polo-shirt,White,300,0 Baht,To process)`

`process_order_list(order_list,stock_list)`	

`print_order_list(order_list)`	`Order List:`
	`(1,T-shirt,White,10,600 Baht,Delivered)`
	`(2,T-shirt,Black,20,1800 Baht,Delivered)`
	`(3,Polo-shirt,White,300,0 Baht,Insufficient)`

`print_stock_list(stock_list)`	`Stock List:`
	`(1,T-shirt,White,90,60)`
	`(2,T-shirt,Black,80,90)`
	`(3,Polo-shirt,White,100,120)`
	`(4,Polo-shirt,Green,100,140)`
	`(5,Shirt,Green,100,200)`
	`(6,Shirt,Black,100,220)`

Level 2

In this level 2, extend from level 1 by adding another class called Customer.

Define class Customer such that there are 3 attributes:

- name: name of customer
- order_list: list of order of this customer
- total_cost: total cost from every order in the order_list

Inside Customer class, add method specified below:

Method	Functionality
compute_total_cost()	Compute summation of cost from every order in the order_list. Then, use this summation to update value of total_cost attribute.

Output after running_level2.py

run_level2.py

```
from item import *
from stock import *
from order import *
from customer import *

# Copy 4 functions from Level 1 here:
# print_stock_list, print_item_list,
print_order_list, process_order_list

item1 = Item(1,'T-shirt', 'White')
item2 = Item(2,'T-shirt', 'Black')
item3 = Item(3,'Polo-shirt', 'White')
item4 = Item(4,'Polo-shirt', 'Green')
item5 = Item(5,'Shirt', 'Green')
item6 = Item(6,'Shirt', 'Black')

stock1 = Stock(item1, 100, 60)
stock2 = Stock(item2, 100, 90)
stock3 = Stock(item3, 100, 120)
stock4 = Stock(item4, 100, 140)
stock5 = Stock(item5, 100, 200)
stock6 = Stock(item6, 100, 220)

## Add your own code to create a list of
stocks here
print_stock_list(stock_list)




## Add your own code to create a list of 6
items here
print_item_list(item_list)
```

Output

```
Stock List:
(1,T-shirt,White,100,60)
(2,T-shirt,Black,100,90)
(3,Polo-shirt,White,100,120)
(4,Polo-shirt,Green,100,140)
(5,Shirt,Green,100,200)
(6,Shirt,Black,100,220)


Item List:
1,T-shirt,White
2,T-shirt,Black
3,Polo-shirt,White
```

```
## Add your own code to create a list of
customers

## Hint: For each customer, you need to read
two things from user: 1) name and 2)
order_list

## After reading information of each
customer, create and print this customer,
append each customer to a list of customers
```

```
## In the list of customers, process order
list of each customer

## After processing order list of all
customers, print information of each
customer
```

```
4,Polo-shirt,Green
5,Shirt,Green
6,Shirt,Black


Customer #1
Enter name of customer: Ann
Enter item id (negative to quit): 1
Enter amount: 10
Enter item id (negative to quit): 2
Enter amount: 20
Enter item id (negative to quit): 3
Enter amount: 300
Enter item id (negative to quit): -1
Customer: Ann
Total_cost = 0
(1,T-shirt,White,10,0 Baht,To process)
(2,T-shirt,Black,20,0 Baht,To process)
(3,Polo-shirt,White,15,0 Baht,To process)

Continue to read new customer (y/n): y

Customer #2
Enter name of customer: Bob
Enter item id (negative to quit): 2
Enter amount: 20
Enter item id (negative to quit): 4
Enter amount: 30
Enter item id (negative to quit): 5
Enter amount: 40
Enter item id (negative to quit): 6
Enter amount: 25
Enter item id (negative to quit): -2
Customer: Bob
Total_cost = 0
(2,T-shirt,Black,20,0 Baht,To process)
(4,Polo-shirt,Green,30,0 Baht,To process)
(5,Shirt,Green,40,0 Baht,To process)
(6,Shirt,Black,25,0 Baht,To process)

Continue to read new customer (y/n): n


Customer: Ann
Total_cost = 2400
(1,T-shirt,White,10,600 Baht,Delivered)
(2,T-shirt,Black,20,1800 Baht,Delivered)
(3,Polo-shirt,White,300,0
Baht,Insufficient)

Customer: Bob
Total_cost = 19500
(2,T-shirt,Black,20,1800 Baht,Delivered)
(4,Polo-shirt,Green,30,4200
Baht,Delivered)
(5,Shirt,Green,40,8000 Baht,Delivered)
(6,Shirt,Black,25,5500 Baht,Delivered)
```

`print_stock_list(stock_list)`	`Stock List:`
	`(1,T-shirt,White,90,60)`
	`(2,T-shirt,Black,60,90)`
	`(3,Polo-shirt,White,100,120)`
	`(4,Polo-shirt,Green,70,140)`
	`(5,Shirt,Green,60,200)`
	`(6,Shirt,Black,75,220)`

Level 3

In this level 3, extend from levels 1,2 by adding another class called Factory.

Define class Factory such that there are 3 attributes:

- stock_list: list of clothing item stocks at the factory
- item_list: list of clothing items
- customer_list: list of customers

Inside Factory class, add methods specified below:

Method	Functionality
create_customer_list(*filename*)	Read information of customers from specific *filename*.
print_stock_list()	Print information of stock objects.
print_item_list()	Print information of item objects.
print_customer_list()	Print information of customer objects.
process_all_customers()	For each customer in the customer_list, process order_list of the customer and update total_cost
process _one_customer(*name*)	For customer with specific *name*, process order_list of such customer
update_stock(*item_id*, *amount*)	Increase stock that has specific *item_id* with specific *amount*

Output after running_level3.py

run_level3.py	Output

`from item import *`
`from stock import *`
`from order import *`
`from customer import *`
`from factory import *`
 	
`item1 = Item(1,'T-shirt', 'White')`
`item2 = Item(2,'T-shirt', 'Black')`
`item3 = Item(3,'Polo-shirt', 'White')`
`item4 = Item(4,'Polo-shirt', 'Green')`
`item5 = Item(5,'Shirt', 'Green')`
`item6 = Item(6,'Shirt', 'Black')`
 	
`stock1 = Stock(item1, 100, 60)`
`stock2 = Stock(item2, 100, 90)`
`stock3 = Stock(item3, 100, 120)`
`stock4 = Stock(item4, 100, 140)`
`stock5 = Stock(item5, 100, 200)`
`stock6 = Stock(item6, 100, 220)`	

```
## Add your own code to create a list
of stocks here

## Add your own code to create a list
of 6 items here

factory = Factory(stock_list,
item_list, 'customer_orders.txt')

factory.print_stock_list()
```

```
Stock List:
(1,T-shirt,White,100,60)
(2,T-shirt,Black,100,90)
(3,Polo-shirt,White,100,120)
(4,Polo-shirt,Green,100,140)
(5,Shirt,Green,100,200)
(6,Shirt,Black,100,220)
```

```
factory.print_item_list()
```

```
Item List:
1,T-shirt,White
2,T-shirt,Black
3,Polo-shirt,White
4,Polo-shirt,Green
5,Shirt,Green
6,Shirt,Black
```

```
factory.print_customer_list()
```

```
Customer List:
Customer: Ann
Total_cost = 0
(1,T-shirt,White,10,0 Baht,To process)
(2,T-shirt,Black,20,0 Baht,To process)
(3,Polo-shirt,White,300,0 Baht,To process)

Customer: Bob
Total_cost = 0
(2,T-shirt,Black,20,0 Baht,To process)
(4,Polo-shirt,Green,30,0 Baht,To process)
(5,Shirt,Green,40,0 Baht,To process)
(6,Shirt,Black,25,0 Baht,To process)

Customer: Charlie
Total_cost = 0
(1,T-shirt,White,50,0 Baht,To process)
(2,T-shirt,Black,30,0 Baht,To process)
(3,Polo-shirt,White,35,0 Baht,To process)
(4,Polo-shirt,Green,25,0 Baht,To process)

Customer: Dave
Total_cost = 0
(6,Shirt,Black,15,0 Baht,To process)
(5,Shirt,Green,45,0 Baht,To process)
(4,Polo-shirt,Green,50,0 Baht,To process)
(3,Polo-shirt,White,40,0 Baht,To process)
(2,T-shirt,Black,30,0 Baht,To process)
(1,T-shirt,White,20,0 Baht,To process)

Customer: Eric
Total_cost = 0
(2,T-shirt,Black,10,0 Baht,To process)
(4,Polo-shirt,Green,30,0 Baht,To process)
```

```
## Add your own code from here
```

```
(6,Shirt,Black,30,0 Baht,To process)

Customer: Francis
Total_cost = 0
(3,Polo-shirt,White,20,0 Baht,To process)
(2,T-shirt,Black,10,0 Baht,To process)
(5,Shirt,Green,20,0 Baht,To process)
(1,T-shirt,White,15,0 Baht,To process)

------------------------
1. Process all customers
2. Process one customer
3. Add stock
4. Quit
Enter your choice: 1

Customer: Ann
Total_cost = 2400
(1,T-shirt,White,10,600 Baht,Delivered)
(2,T-shirt,Black,20,1800 Baht,Delivered)
(3,Polo-shirt,White,300,0 Baht,Insufficient)

Stock List:
(1,T-shirt,White,90,60)
(2,T-shirt,Black,80,90)
(3,Polo-shirt,White,100,120)
(4,Polo-shirt,Green,100,140)
(5,Shirt,Green,100,200)
(6,Shirt,Black,100,220)

Customer: Bob
Total_cost = 19500
(2,T-shirt,Black,20,1800 Baht,Delivered)
(4,Polo-shirt,Green,30,4200 Baht,Delivered)
(5,Shirt,Green,40,8000 Baht,Delivered)
(6,Shirt,Black,25,5500 Baht,Delivered)

Stock List:
(1,T-shirt,White,90,60)
(2,T-shirt,Black,60,90)
(3,Polo-shirt,White,100,120)
(4,Polo-shirt,Green,70,140)
(5,Shirt,Green,60,200)
(6,Shirt,Black,75,220)

Customer: Charlie
Total_cost = 13400
(1,T-shirt,White,50,3000 Baht,Delivered)
(2,T-shirt,Black,30,2700 Baht,Delivered)
(3,Polo-shirt,White,35,4200 Baht,Delivered)
(4,Polo-shirt,Green,25,3500 Baht,Delivered)

Stock List:
(1,T-shirt,White,40,60)
(2,T-shirt,Black,30,90)
(3,Polo-shirt,White,65,120)
(4,Polo-shirt,Green,45,140)
(5,Shirt,Green,60,200)
(6,Shirt,Black,75,220)
```

```
Customer: Dave
Total_cost = 21000
(6,Shirt,Black,15,3300 Baht,Delivered)
(5,Shirt,Green,45,9000 Baht,Delivered)
(4,Polo-shirt,Green,50,0 Baht,Insufficient)
(3,Polo-shirt,White,40,4800 Baht,Delivered)
(2,T-shirt,Black,30,2700 Baht,Delivered)
(1,T-shirt,White,20,1200 Baht,Delivered)

Stock List:
(1,T-shirt,White,20,60)
(2,T-shirt,Black,0,90)
(3,Polo-shirt,White,25,120)
(4,Polo-shirt,Green,45,140)
(5,Shirt,Green,15,200)
(6,Shirt,Black,60,220)

Customer: Eric
Total_cost = 10800
(2,T-shirt,Black,10,0 Baht,Insufficient)
(4,Polo-shirt,Green,30,4200 Baht,Delivered)
(6,Shirt,Black,30,6600 Baht,Delivered)

Stock List:
(1,T-shirt,White,20,60)
(2,T-shirt,Black,0,90)
(3,Polo-shirt,White,25,120)
(4,Polo-shirt,Green,15,140)
(5,Shirt,Green,15,200)
(6,Shirt,Black,30,220)

Customer: Francis
Total_cost = 3300
(3,Polo-shirt,White,20,2400 Baht,Delivered)
(2,T-shirt,Black,10,0 Baht,Insufficient)
(5,Shirt,Green,20,0 Baht,Insufficient)
(1,T-shirt,White,15,900 Baht,Delivered)

Stock List:
(1,T-shirt,White,5,60)
(2,T-shirt,Black,0,90)
(3,Polo-shirt,White,5,120)
(4,Polo-shirt,Green,15,140)
(5,Shirt,Green,15,200)
(6,Shirt,Black,30,220)

------------------------
1. Process all customers
2. Process one customer
3. Add stock
4. Quit
Enter your choice: 3
Enter item id: 2
Enter amount: 50
Stock List:
(1,T-shirt,White,5,60)
(2,T-shirt,Black,50,90)
(3,Polo-shirt,White,5,120)
```

```
(4,Polo-shirt,Green,15,140)
(5,Shirt,Green,15,200)
(6,Shirt,Black,30,220)

------------------------
1. Process all customers
2. Process one customer
3. Add stock
4. Quit
Enter your choice: 3
Enter item id: 5
Enter amount: 100
Stock List:
(1,T-shirt,White,5,60)
(2,T-shirt,Black,50,90)
(3,Polo-shirt,White,5,120)
(4,Polo-shirt,Green,15,140)
(5,Shirt,Green,115,200)
(6,Shirt,Black,30,220)

------------------------
1. Process all customers
2. Process one customer
3. Add stock
4. Quit
Enter your choice: 2
Enter customer name: Francis
Customer: Francis
Total_cost = 8200
(3,Polo-shirt,White,20,2400 Baht,Delivered)
(2,T-shirt,Black,10,900 Baht,Delivered)
(5,Shirt,Green,20,4000 Baht,Delivered)
(1,T-shirt,White,15,900 Baht,Delivered)

Stock List:
(1,T-shirt,White,5,60)
(2,T-shirt,Black,40,90)
(3,Polo-shirt,White,5,120)
(4,Polo-shirt,Green,15,140)
(5,Shirt,Green,95,200)
(6,Shirt,Black,30,220)

------------------------
1. Process all customers
2. Process one customer
3. Add stock
4. Quit
Enter your choice: 2
Enter customer name: Eric
Customer: Eric
Total_cost = 11700
(2,T-shirt,Black,10,900 Baht,Delivered)
(4,Polo-shirt,Green,30,4200 Baht,Delivered)
(6,Shirt,Black,30,6600 Baht,Delivered)

Stock List:
(1,T-shirt,White,5,60)
(2,T-shirt,Black,30,90)
(3,Polo-shirt,White,5,120)
```

	(4,Polo-shirt,Green,15,140) (5,Shirt,Green,95,200) (6,Shirt,Black,30,220) ---------- 1. Process all customers 2. Process one customer 3. Add stock 4. Quit Enter your choice: 4