


7.5/10

	National University of Computer and Emerging Sciences (Lahore)		
	Course:	OOP	Course code: CS217
	Section:	BSCS-2B	Semester: Spring 2024
	Duration:	40 minutes	Total Marks: 10
	Date:		ID: B
	Name:	Abdul Azeed	Roll no: 231-0831

Question 1:

NOTE: Read the entire question first before attempting.

A retail store needs to develop a customer management system to handle different types of customers. Implement a system that categorizes customers into three classes: Regular, VIP, and Thieves. Make sure that all customers must belong to one of the specialized types and no generic customer is created in the system at any point and all types of customers are able to purchase. Each type of customer derived from a base class has some shared and some unique attributes and/or behaviors as described below:

1. Regular Customer:

- Attributes: name (string), age (int), membershipPoints (int), limit (double)
- Behaviors: double purchase(double bill) - is only allowed to purchase up to a value of limit and the final bill returned is equal to "bill - membership points".

2. VIP Customer:

- Attributes: name (string), age (int), rank (int).
- Behaviors: ^{int} purchase(double bill) - final bill is equal to $\text{bill} - (\text{bill} * \text{rank} / 100)$

$$100 \times 20 / 100 = \frac{1}{8} \times 100 = 12.5$$

3. Thief:

- Attributes: name (string), age (int).
- Behaviors: double purchase(double bill) - final bill is always zero.

$$100 - 20 = 80$$

1. Implement default and parameterised constructors, destructors and a purchase() method in each derived class to calculate the salary of the respective employee type based on the provided attributes.
2. Give output of the main given on the next page.

```
int main() {  
    // Create customer objects  
    vector<Customer*> customers;  
  
    customers.push_back(new RegularCustomer("Muzammil Aleem", 20, 50, 1000.0));  
    customers.push_back(new VIPCustomer("Manal Asif", 20, 20));  
    customers.push_back(new Thief("Abdul Muhaimin", 20));  
  
    // Simulate purchases  
    double bill = 100.0;  
  
    for (const auto& customer : customers) {  
        double finalBill = customer->purchase(bill);  
        cout << customer->name << "'s Final Bill: $" << finalBill << endl;  
    }  
  
    // Free memory  
    for (const auto& customer : customers) {  
        delete customer;  
    }  
  
    return 0;  
}
```

Output:

Muzammil Aleem's Final Bill: \$ 50.0

Manal Asif's Final Bill: \$ 80.0

Abdul Muhaimin's Final Bill: \$ 0.0



class ~~customer~~ customer

{

~~customer~~ ()

Private;

string name;

int age;

Public;

customer ()

~~string~~ name = "default"

age = 0;

customer (string n, int a)

name = n;

virtual age = a;

virtual customer ()

} -2

virtual double

purchase (double bill) = 0;

pure virtual
-0.5

class Regularcustomer : public customer

{

Private;

int membership_points = 0;

double limit;

Public;

Regularcustomer () : call parent constructor

{

membership_points = 0

limit = 0;

}


```
RegularCustomer (string n, int a, int m, double  
l) : customer(n, a)
```

```
{
```

```
    membership_points = m;
```

```
    limit = l;
```

```
}
```

```
~RegularCustomer() { };
```

```
override double  
purchase (double bill)
```

```
{
```

```
    return (bill - membership_points);
```

```
}
```

```
class VIPCustomer : public Customer
```

```
{
```

```
    int stock
```

```
    VIPCustomer() : ?
```

```
{
```

```
    stock = 0;
```

```
}
```

```
VIPCustomer (string n, int a, int s)
```

```
: Customer(n, a)
```

```
{
```

```
    stock = s
```

```
}
```

```
~VIPCustomer() { }
```

```
override int  
purchase (double bill)
```

```
{
```



```
return (bill - (bill * 100 / 100));  
}
```

```
class Thief : public customer  
{  
    Thief() : ?  
    { };  
    Thief(string n, int a) : customer(n, a);  
    ~Thief() { };  
    override double  
    purchase(double bill)  
    {  
        return 0;  
    }  
}
```

~~customer :: purchase(double a)~~