MPP Midterm January 27, 2018

Please make sure that you write **VERY CLEARLY**! If I can not read your writing you will Not get credit for the question.

For All code below, if you need to create another helper method that is fine. Just write the code for the method.

1) (one point). What is your name?

2) (18 points). **Inside the ACustomer class**, write a method called 'largestOrderPrice' which returns the largest order price of all the orders for that customer. **The class diagram is below**. .

The first line of the method is : double largestOrderPrice ()

3) (18 points). **Inside the ACustomer class**, write a method called 'largestQuantityInsideAnOrderline' which returns the data member 'quantity' inside an orderLine object, which has the largest value for that customer, considering All of the Orders and All of the OrderLines for that customer.

The first line of the method is : double largestQuantityInsideAnOrderline ()

4) (18 points). Add the following to the class diagram below. A MainApp class that has main in it. The MainApp class has a List of 'ICustomer' objects inside it.

Let's add another class, a child to the 'ACustomer' class called MilitaryCustomer. A MilitaryCustomer is for example, the Army, or a department inside the Navy, etc.

A MilitaryCustomer object has a data member inside of it called 'uniform', which is a String that describes the uniform of that MilitaryCustomer. MilitaryCustomer also has a data member called 'name' which is a String and describes the MilitaryCustomer(for example, Army, Navy, etc.)

Write a method inside the MainApp class that prints out the name and the uniform for all of the MilitaryCustomer objects.

5) (18 points). Add the following to the class diagram below. A MainApp class that has main in it. The MainApp class has a List of 'Order' objects inside it. It also has a List of 'ICustomer' objects inside it.

In addition, we have a class called Book which has the isbn, title, **cost of the book**, and author of the book. We assume there is only one author.

An OrderLine object has a List of 'Book' objects inside it, which describe the OrderLine Product from different points of view. **An OrderLine has only one IProduct inside it**. The 'n' in the diagram is a mistake.

Write a method in the MainApp class called 'totalCostOfAllBooks' which returns the total cost of all the books for all of the OrderLine objects in our system.

6) (20 points). Design a UML class diagram for a real estate agency. We can rent and sell both apartments and houses. We have staff working and we keep track of their name, address and phone. Some staff are supervisors and they can have up to ten people working for them. A staff person can manage up to 100 properties. We keep track of which properties they manage.

Each client (customer) works with one staff person. For clients we keep track of their name, address and phone. Clients have a preference which is either an apartment or a house. They also have a maximum rent that they can afford.

Clients can view a property to see if they like it. They can see many properties, and with each viewing we keep track of the viewing date and a comment the client has about the viewing (for example, the apartment was nice but is dirty).

For the properties we keep track of the address, type (apartment or house), number of rooms, and the rent or selling price.

If the client rents an apartment, a legal lease is signed by the client. For the lease we keep track of the deposit paid, rent start date, rent finish date, and the duration (length) of the lease.

We also keep track of the owners of the house or apartment. An owner can be either a private person or a business owner. For a private owner we keep track of the first name, last name, address, and phone. For a business owner we keep track of the address, phone, business name, the type of business, and the contact name.

Make sure you specify all multiplicities, relationship types and direction, attributes and methods, interfaces and abstract classes. You do not have to use interfaces unless they are required, but make sure to indicate classes that should be abstract. You do not need to show getter and setter methods. You do not need to show attribute types, but you should show method signatures. Your design should follow the open-closed principle—in particular, allow for polymorphism where appropriate.

7) (5 points). Write a paragraph or two relating SCI to either UML, OR, to Threads. You should have "Clear Connections, Stern Logic, Utter Simplicity".

8) Extra Credit : Write the constructors for all of the classes in the diagram. Then in main, create two Order objects, each one having two Orderline objects. Then create one Corporate customer, and one Personal customer. Write all of the code necessary to put the above objects into the appropriate ArrayLists.

Good Luck!

NAME: Binyam Heyi, ID = 986164

Q1)

Name: Binyam Shiferaw Heyi
ID: 986164

Q2)

Assumption

1) Order class has a getter function defined below.

```
Class Order
        Public double getOrderPrice()
          {
            return this.Orderprice()
          }
```

2) their is a list data meenember in A customer class defined as
List < Order > OrderLists = new ArrayList < Order >();

Solution

```
Public double largestOrderprice()
  {
    double largestPrice = 0
    for (Order order : OrderLists) // → Assuming the size > 0.
      {
        if( largestPrice < Order.getOrderPrice())
            largestPrice = Order.getOrderprice();
      }
    return largestprice;
  }
```

Q3)

Assumptions 3) 99½ + XC

There are list data members in A customer & Order class defines as follows.

1) List <Order> OrderLists = new ArrayList <Order> (); → A custome class
2) List <OrderLine> orderLines = new ArrayList <OrderLine> (); → Order Class.
3) There is a getter function in OrderLine class for Quantity defin as

```
Public int getQuantity()
    {
     return this.getQuantity;
    }
```

4) There is a getter function for get OrderLine list in class Order

```
Public List <OrderLines> getOrderLineList()
    {
     return this.OrderLines;
    }
```

Solution

```
Public double largestQuantityInsideAnOrderLine()
    {
     double largestQuantity = 0;
     for (Order order : OrderLists) // → Assumption the size > 0
        {
         for (OrderLine orderline : order.getOrderLineList())
                                    // the size is > 0 // →
            {
             if (largestQuantity < orderline.getQuantity())
                {
                 largestQuantity = orderline.getQuantity()
                }
            }
        }
     return largestQuantity
    }
```

Q4) Assumption 4/

1) The list in mainApp is List<ICustomers> customerIntList...
2) Their is a getter function getUniform defined as below in MilitaryCustomer class.

```
Public String getUniform()
    {
    return this.Uniform;
    }
```

3) Their is a getter function for name either in Acustomer/MilitaryCustomer classes. defined as

```
Public String getName()
    {
    return this.name;
    }
```

Solution

```
Public void Print()
    {
    if( CustomerIntlist.Size()==0 )
        System.Out.println("Nothing On the List");

    else
      {
        for(ICustomer customer : CustomerIntList)
          { if( customer instanceOf MilitaryCustomer)
            {
            MilitaryCustomer miliCustomer=(MilitaryCustomer)Customer;
        System.Out.println("Name :" + miliCustomer.getname()
                    +" Uniform:" + mil.customer.getUniform());

          }

    }
```

Q5) Assumption

1) In Main App class the list of Order is List<Order> orderLists
2) In Order class the list of OrderLine is List<OrderLine> OrderLineLists;
3) In OrderLine class the list of Books is List<Books> bookLists.
4) There is a getter function defined below in class Order.

```
Public List<OrderLine> getOrderLists()
    {
    return this.getOrderLists()
    }
```

---

5) There is a getter function define below in class OrderLine

```
Public List<Book> getBookLists()
    }
    return this.getBookList.
    }
```

6) There is a getter function defin below in class Book

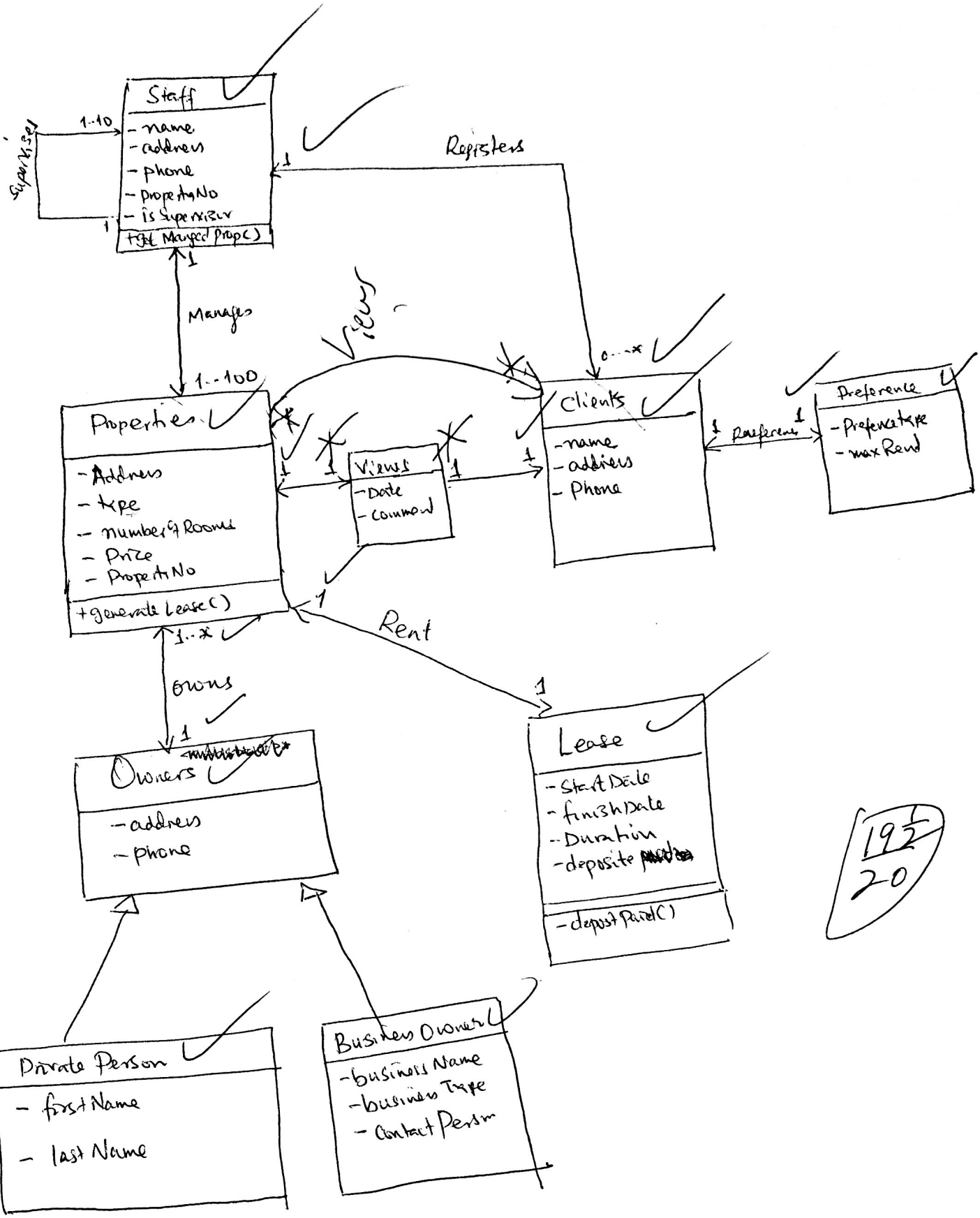
```
Public double getCost()
    {
    return this.cost;
    }
```

Solution

```
Public double totalCostOfAllBooks()
{   double cost = 0;
 if (orderLists.size == 0)
    return cost;

else
  { for (Order order : OrderList)
      { for (OrderLine orderLine : order.getOrderLists())
          { for (Book book : orderLine.getBookLists())
              cost += book.getCost()
          }
      }
  }
return cost;
}
```

Q6)


Staff
- name
- address
- phone
- propertyNo
- isSupervisor
+ getManagedProp()
Supervises
1..10
Registers
Manages
1..100
Views
0..*
Properties
- Address
- type
- numberOfRooms
- Price
- PropertyNo
+ generateLease()
Views
- Date
- comment
Clients
- name
- address
- Phone
Preference
- PreferenceType
- maxRent
Prefers
Rent
Owns
1..*
Owners
- address
- phone
Lease
- StartDate
- finishDate
- Duration
- depositPaid
- depositPaid()
Private Person
- firstName
- lastName
Business Owner
- businessName
- businessType
- ContactPerson

19/20

Unified Modelling Language (UML) enables us to model the Complex Software development, which in turn makes the coding and the implementation easier, this is analogous to ~~[illegible]~~ Capturing the Gold principle of SCI, i-e, if we are able to do Transcedental Meditation twice a day, we will get into pure awarness, which will help us to be Successful, happy and blissfull in this Complex life that we leave these days. Good!

Q8)

Public ~~Interface~~ Icustomer

{

}

(XC)

abstract

Public class Acustomer implement Icustomer

{

Acustomer (String name, String phone, String address, String point)

{

super ( )

this.name = name;

this.phone = phone;

this.address = address;

this.point = point;

}

}

Public class Corporate Custom extends Acustom

{ CorporateCustomer (String name, String phone, String address, String point,

String point, String credit rating, String limit)

{

super (name, phone, address, point);

this.credit rating = credit rating

this.credit limit = limit;

}

add order (order Order)

{ orderList.add (orders)

}

Public class Personal Customer extends Acustom

{

Personal customer (String name, String phone, String address, this point

String credit rating, String credit card)

{

super (name, phone, address, point)

this.Credit Rating = credit Rating

this.CredCard = CreditCard → Continues on page 8.

```
void add order (List<Order> orders)
{
    orderlist.add(orde
}
Public class order implements Icustomer
{
    Public Order (Strin ordernumber, Strin Prepaid, Strin orderprice
                  Strin ordrdat, Strin Status, ICustomer Icustom)
    {
        Super()
        this.Ordernumber = Order.number
        this.Prepaid = Prepaid
        this.orderprice = orderprice
        this.Status = Status
        this.Icostm = Icustom
    }

    Public void addorderline (Orderline O)
    {
        Orderlinelist.add(O)
    }

}

Public class Orderline implements IProduct
{
    Public Orderline (Strin Quantity, Strin price, Strin Status, Strin shipdat, Strin portf
                      IProduct iproduct) Order, order);
    {
        Super()
        this.Quantity = Quantity;
        this.price = price
        this.Status = Status
        this.Shipdat = Shipdat
        this.portf = portf
        this.iProduct.iProduct this.orde = order
    }
}
```

```
Public Interface IProduct
    {

    }

Public class Product implements IProduct
    {
    Product ( String descripti, Stri Productnumber, double pric, IcompPoint pri)
        {
          Super ( )
          this.descriptin = descriptin
          this. productnumbe = productnumb
          thi. price = price
          this. icompu = icompon
          }
    }

Public class
Public Interface IComponints
    {

    }

Public abstract class AcomppointsImplements IcompPoints
    { Public Acomponints }
      Super ( )
        {

    }

public class Computer points extends AcompPoints
    {
      public CompnterPoints (double points)
          } Super ( )
            this. points = points
          }

    }
```

```
Public class Audio-Videopoints extens Acomppoints
{
    Public Audio-Videopoints (double points)
    {
        Super()
        this.points = points;
    }
}
Public class Health_points extends Acomp Point
{
    Public Health-Point (double points)
    {
        Super()
        this.point = point
    }
}
Public class Other point extends Acomppoints
{
    Public Otherpoints (double points)
    {
        Super()
        this.points = points
    }
}
Public class Main{
    Public Static void main (String[] args)
    {
Order order1 = new order ("1", true, 400, "2017-1-1", "Shipp");
order order2 = new orde ("8", false, 90, '2018-1-1', "NotShipped");
Orderline orderline1 = new Orderline (2, 500.3, "shipped", '2017-1-1', 2, order1)
Orderline orderline2 = new orderline (2, 500.3, "Shipp", '2018-1-1', 3, order2)
Orderline orderli3 = new Orderline (2, 500.3, "NotShipped", '2018-2-2', 4, order1)
OrderLine orderline4 = new orderline (2, 500.4, "Shipped", '2018-3-2', 3, order2)
```

Customer Customer = new Customer ("Bimpe", "Iowa", "641", 2, "poor", 0);

Customer Mum = new CorporateCustomer ("Mum", "Iowa", "642", 31, "High", 3100)

Mum.addOrder(order1);

Customer.addOrder(order2);

}

}