

Group 22 - Sparepart

Anggota:

2440030241 - Reynald Slamet Putra

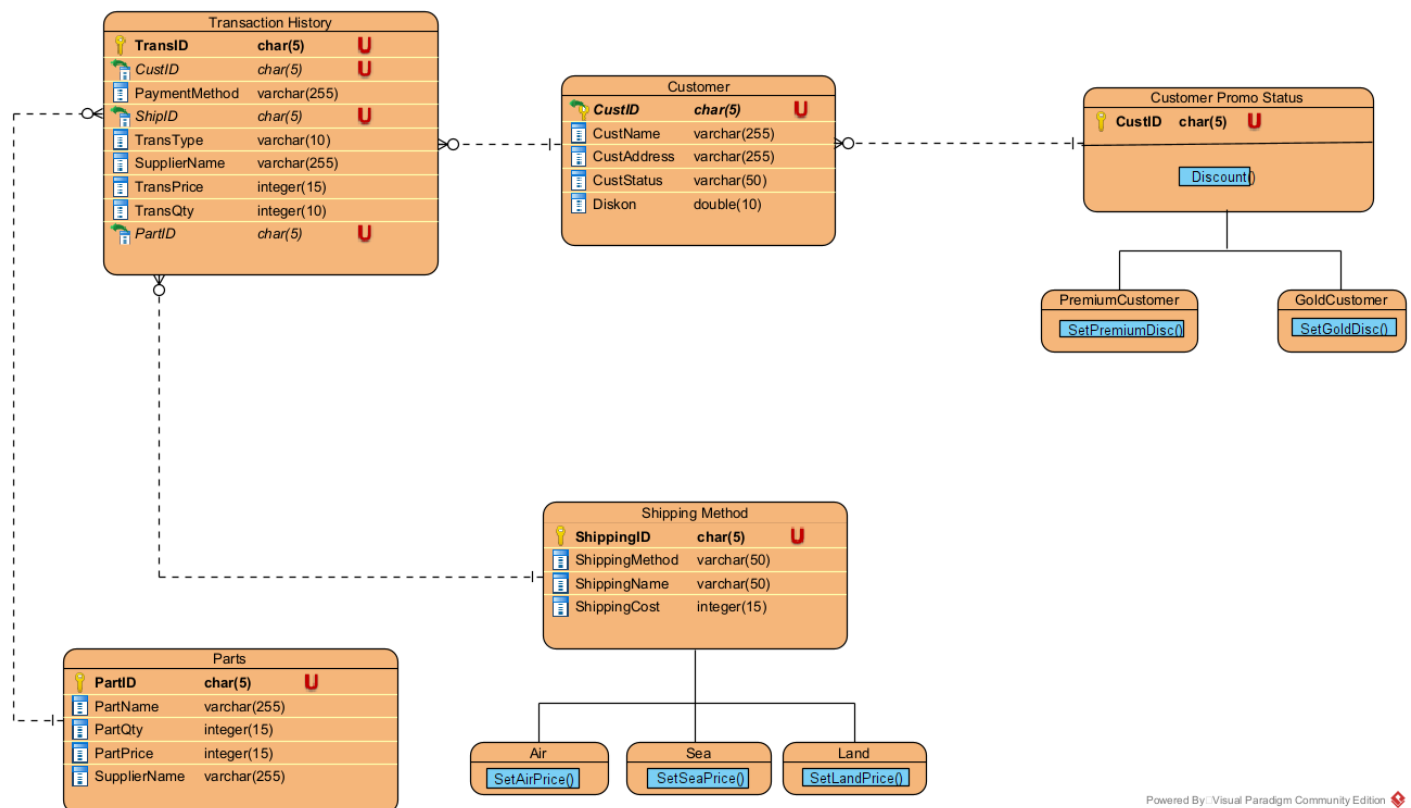
2440062924 - Charles Christopher

2440046984 - Elliot Lie Arifin

Penjelasan Kasus:

Zaman sekarang masih banyak toko yang menggunakan buku catatan untuk mencatat semua transaksi dan stok. Hal ini sangat tidak efisien dan mempunyai resiko tinggi data hilang atau salah. Maka dari itu kami membuat sebuah aplikasi untuk melakukan pencatatan secara transparan terhadap transaksi penjualan ke pembeli dan transaksi pembelian ke supplier. Hal ini memudahkan pemilik toko untuk melihat stok, melihat data transaksi, melihat harga, melihat data member, dan memberikan potongan harga khusus untuk member.

Class Diagram:



Class Analysis:

Seperti yang tertera pada diagram, Transaction History dipecah menjadi 2 bagian (header dan detail), hal ini ditujukan hanya untuk merapikan diagram dan bukannya membagi class Transaction History menjadi 2 bagian. Sehingga akan terdapat class: Transaction History, Parts, Customer, Shipping, Payment Method, dan Customer Promo Status.

Berikut paparan methods-methods yang akan digunakan pada Main sistem toko sparepart:

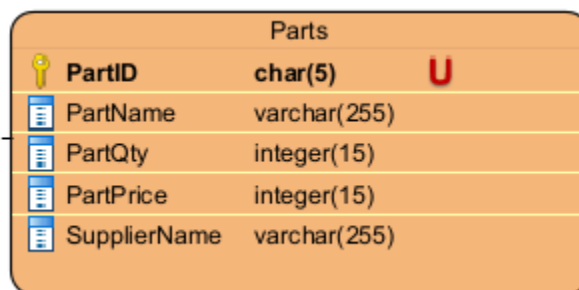
USER: Pemilik Toko

1. Buy Parts
2. Sell Parts
 - 2a. Check Member (pengecekan status member terkait promo)
3. View Member
 - 3a. Add New Member
 - 3b. Update Member
 - 3c. Remove Member
4. View Available Parts
5. View Transaction History
6. View Shipping Method

Terkait datafield sudah tertera pada diagram yang dilampirkan.

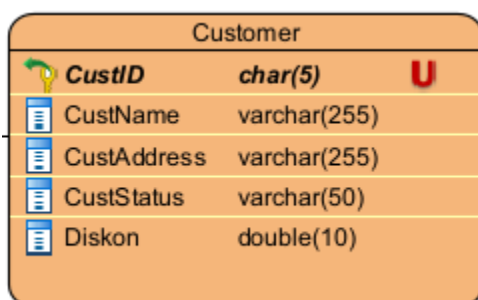
- **Parts**

= Pada class parts hanya terdapat method setter and getter dengan atribut dan datafield seperti dibawah ini:



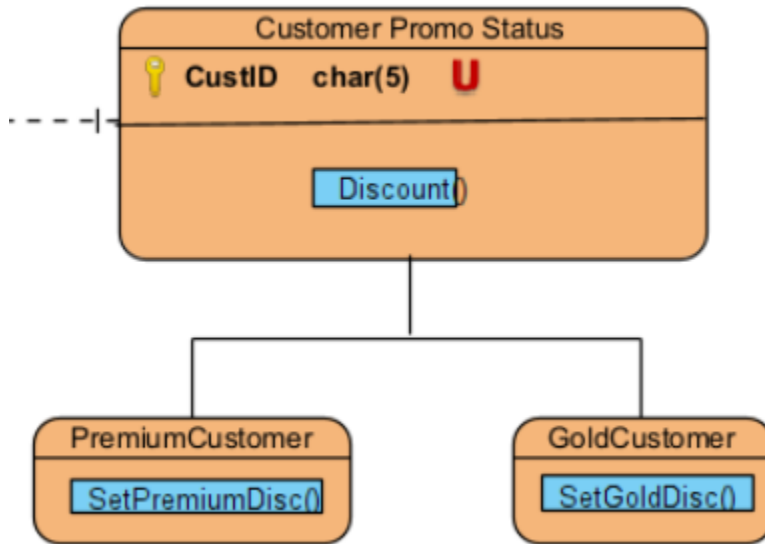
- **Customer**

= Pada class customer hanya terdapat method setter and getter dengan atribut dan datafield seperti dibawah ini:



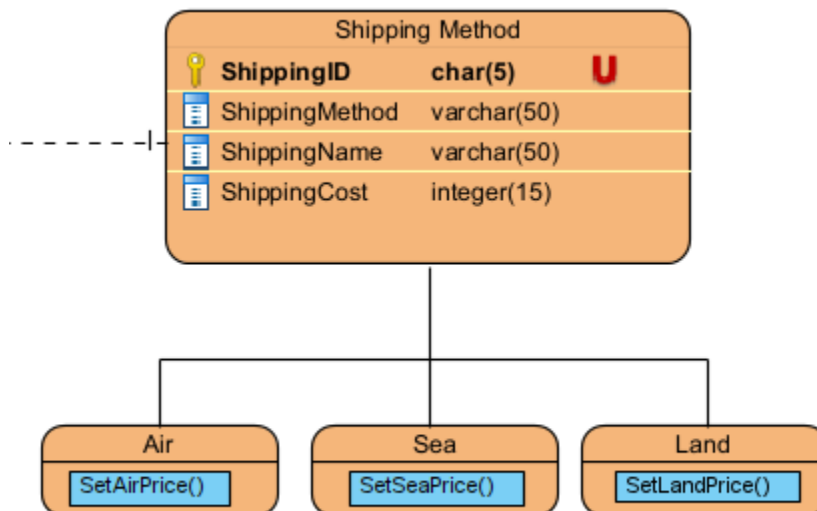
- **CustomerStatus**

= class CustomerStatus merupakan abstract class yang memiliki child class berupa: premiumCust dan goldCust yang hanya memiliki method bernama diskon() untuk me-return nilai diskon tergantung pada status membership dari customer (premiumCust atau goldCust)



- **Shipping**










= class Shipping memiliki atribut dan datafield seperti dibawah ini:



Class Shipping mengaplikasikan konsep inheritance dengan adanya pembagian 3 child class (class Land, class Sea, dan class Air) dan polymorphism dengan adanya perubahan value shippingCost di masing-masing child class dari Shipping.

- **TransactionHistory**

= Pada class **TransactionHistory** hanya terdapat method setter and getter dengan atribut dan datafield seperti dibawah ini:

Transaction History			
	TransID	char(5)	U
	<i>CustID</i>	<i>char(5)</i>	U
	PaymentMethod	varchar(255)	
	<i>ShipID</i>	<i>char(5)</i>	U
	TransType	varchar(10)	
	SupplierName	varchar(255)	
	TransPrice	integer(15)	
	TransQty	integer(10)	
	<i>PartID</i>	<i>char(5)</i>	U

PEMBAGIAN TUGAS PER INDIVIDU

Penjelasan dan perancangan kasus:

Elliot

Charles

Reynald

Class Diagram:

Elliot

Charles

Reynald

Class Analysis:

Elliot

Charles

Reynald

Design Console Output & Smell Code Check:

Reynald

Create Repo and maintain git repo:

Charles

Coding:

("1. Buy Parts"); Elliot

("2. Sell Parts"); Elliot

("3. View Member"); Charles

 ("1. Add Member"); Charles

 ("2. Update Member"); Charles

 ("3. Remove Member"); Charles

 ("4. Return to Main Menu"); Charles

("4. View Available Parts"); Elliot

("5. View Transaction History"); Reynald

("6. View Shipping List"); Reynald

("7. Exit"); Reynald

Penerapan

Constructor : Method khusus yang akan dieksekusi pada saat pembuatan objek. Biasanya method ini digunakan untuk inialisasi atau mempersiapkan data untuk objek.

```
public class Customer {
    private String custID;
    private String custName;
    private String custStatus;
    private double diskon;

    public Customer(String custID, String custName, String custStatus, double diskon) {
        super();
        this.custID = custID;
        this.custName = custName;
        this.custStatus = custStatus;
        this.diskon = diskon;
    }
}
```

Pemanggilan class : Suatu blueprint atau cetakan untuk menciptakan suatu instant dari object. class juga merupakan grup suatu object dengan kemiripan attributes/properties, behaviour dan relasi ke object lain.

```
Scanner sc = new Scanner(System.in);
Vector<Parts> parts = new Vector<Parts>();
Vector<Customer> customers = new Vector<Customer>();
Vector<TransactionHistory> transHistory = new Vector<TransactionHistory>();
Vector<Air> airs = new Vector<Air>();
Vector<Sea> seas = new Vector<Sea>();
Vector<Land> lands = new Vector<Land>();

public void addTransHistory(String TransType, int qty, String partID, String custID, String supplierName, int totalPrice, String paymentMethod)
{
    String id = "TR" + String.format("%03d", transHistory.size()+1);
    TransType.toUpperCase();
    transHistory.add(new TransactionHistory(id, TransType, qty, partID, custID, supplierName, totalPrice, paymentMethod)); ✓
}
```

Method: Method adalah kumpulan blok kode yang bisa dijalankan ketika dipanggil, method bisa menerima dan mengirim data melalui parameter. Berikut contoh dari program kami:

```
public class Customer {
    private String custID;
    private String custName;
    private String custStatus;
    private double diskon;
    public Customer(String custID, String custName, String custStatus, double diskon) {
        super();
        this.custID = custID;
        this.custName = custName;
        this.custStatus = custStatus;
        this.diskon = diskon;
    }
    public String getCustID() {
        return custID;
    }
    public void setCustID(String custID) {
        this.custID = custID;
    }
    public String getCustName() {
        return custName;
    }
    public void setCustName(String custName) {
        this.custName = custName;
    }
    public String getCustStatus() {
        return custStatus;
    }
    public void setCustStatus(String custStatus) {
        this.custStatus = custStatus;
    }
}

public void clearScreen()
{
    for (int i=0; i<30; i++) System.out.println("");
}
public int validateIntInput(int input, int type)
{
    int value = 1;
    while(value == 1)
    {
        //variabel type digunakan untuk mengetahui pertanyaan (sebelum input) apa yang harus di print
        if (type == 1) System.out.print("Choose Menu: ");
        else if (type == 2) System.out.print("Input part's price to buy [at least 100 or more]: ");
        else if (type == 3) System.out.print("Input part's quantity to buy [at least 1 or more]: ");
        else if (type == 4) System.out.print("Quantity to sell: ");
        else if (type == 5) System.out.print(">> ");
        try {
            input = sc.nextInt();
            if (input == (int) input)
            {
                value = 0;
                break;
            }
        }
        catch (Exception e) {
            System.out.println("Your input must be integer!");
            value = 1;
        }
        finally {
            sc.nextLine();
        }
    }
    return input;
}
```

Data field sesuai dengan class diagram : Atribut data yang bisa kita sisipkan di objek.
Sebelum memasukkan data apa saja yang ada di objek, terlebih dahulu kita deklarasikan data tersebut di dalam class.

Data Field Transaction History

```
private String transID;  
private String transType;  
private int transQty;  
private String custID;  
private String partID;  
private String supplierName;  
private int transPrice;  
private String paymentMethod;
```

Data Field Customer

```
private String custID;  
private String custName;  
private String custStatus;  
private double diskon;
```

Data Field Parts

```
String partID;  
String partName;  
int partPrice;  
int partQty;  
String supplierName;
```

Data Field Shipping

```
private String shipID;  
private String shippingName;  
private String shippingMethodName;  
private int shippingCost = 0;
```


Encapsulation: Menjaga data customerID, name, status dan diskon dengan sebuah class agar tidak bisa diakses secara terbuka

```
public class Customer {
    private String custID;
    private String custName;
    private String custStatus;
    private double diskon;
    public Customer(String custID, String custName, String custStatus, double diskon) {
        super();
        this.custID = custID;
        this.custName = custName;
        this.custStatus = custStatus;
        this.diskon = diskon;
    }
    public String getCustID() {
        return custID;
    }
    public void setCustID(String custID) {
        this.custID = custID;
    }
    public String getCustName() {
        return custName;
    }
    public void setCustName(String custName) {
        this.custName = custName;
    }
    public String getCustStatus() {
        return custStatus;
    }
    public void setCustStatus(String custStatus) {
        this.custStatus = custStatus;
    }
}
```

Penggunaan Inheritance: Mengelompokkan Shipping class menjadi 3 bagian (3 child class) yaitu: Sea, Land, dan Air untuk memudahkan dalam mengupdate codingan di kemudian hari. Parent disini adalah class Shipping dengan Child adalah class Sea, Land dan Air

```
public class Shipping {  
  
    private String shipID;  
    private String shippingName;  
    private String shippingMethodName;  
    private int shippingCost = 0;  
    public Shipping(String shipID, String shippingName, String shippingMethodName) {  
        super();  
        this.shipID = shipID;  
        this.shippingName = shippingName;  
        this.shippingMethodName = shippingMethodName;  
    }  
    public int getShippingCost() {  
        return shippingCost;  
    }  
    public void setShippingCost(int shippingCost) {  
        this.shippingCost = shippingCost;  
    }  
    public String getShipID() {  
        return shipID;  
    }  
    public void setShipID(String shipID) {  
        this.shipID = shipID;  
    }  
    public String getShippingName() {  
        return shippingName;  
    }  
    public void setShippingName(String shippingName) {  
        this.shippingName = shippingName;  
    }  
    public String getShippingMethodName() {  
        return shippingMethodName;  
    }  
    public void setShippingMethodName(String shippingMethodName) {  
        this.shippingMethodName = shippingMethodName;  
    }  
}
```

```

class Land extends Shipping {
    private int shippingCost;
    public Land(String shipID, String shippingName, String shippingMethodName) {
        super(shipID, shippingName, shippingMethodName);
        this.setShippingCost(20000);
    }
    public int getShippingCost() {
        return shippingCost;
    }
    public void setShippingCost(int shippingCost) {
        this.shippingCost = shippingCost;
    }
}
class Sea extends Shipping {
    private int shippingCost;
    public Sea(String shipID, String shippingName, String shippingMethodName) {
        super(shipID, shippingName, shippingMethodName);
        this.setShippingCost(50000);
    }
    public int getShippingCost() {
        return shippingCost;
    }
    public void setShippingCost(int shippingCost) {
        this.shippingCost = shippingCost;
    }
}
class Air extends Shipping {
    private int shippingCost;
    public Air(String shipID, String shippingName, String shippingMethodName) {
        super(shipID, shippingName, shippingMethodName);
        this.setShippingCost(100000);
    }
    public int getShippingCost() {
        return shippingCost;
    }
    public void setShippingCost(int shippingCost) {
        this.shippingCost = shippingCost;
    }
}

```

Penggunaan Polymorphism: Pada gambar codingan class diatas, terimplementasi konsep polymorphism yang berguna untuk menentukan nilai shippingCost untuk masing-masing pengiriman.

Penggunaan Abstract Class: Penggunaan abstract class terdapat pada class CustomerStatus yang dapat mempermudah kita dalam mengecek kesalahan dan melakukan perubahan pada metode pemberian diskon kedepannya.

```
public abstract class CustomerStatus {  
  
    public abstract double diskon();  
}  
  
class premiumCust extends CustomerStatus {  
    public double diskon()  
    {  
        return 0.1;  
    }  
}  
class goldCust extends CustomerStatus {  
    public double diskon()  
    {  
        return 0.05;  
    }  
}
```