PATTERN SOFTWARE DESIGN ASSIGNMENT

Kelompok 4

Anggota Kelompok :

- Reihan Anggriawan - 2440051555
- Christoper Lim - 2440026591
- Martin - 2440032663

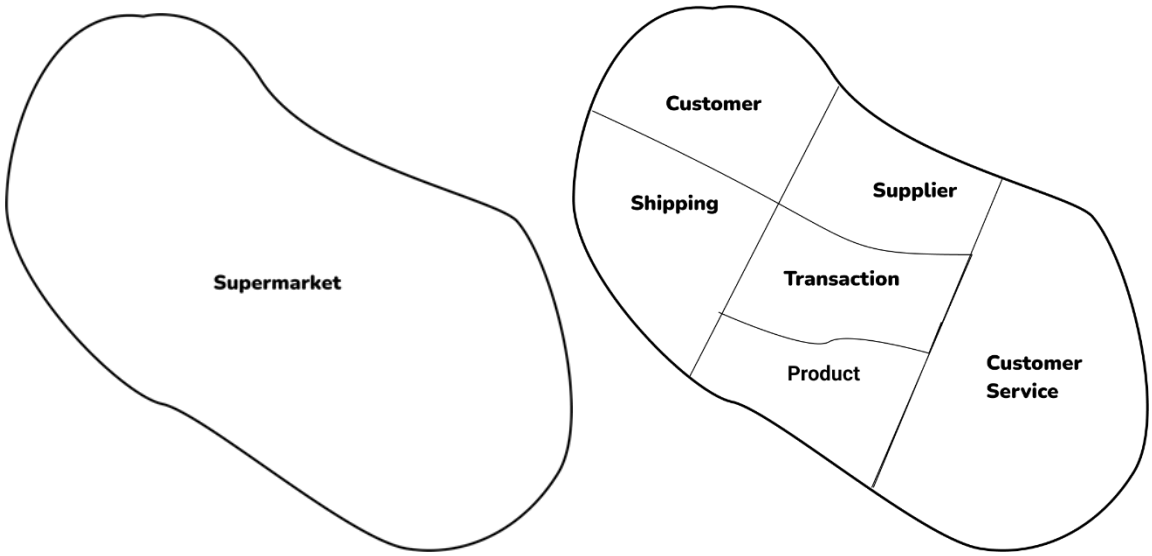
ASSIGNMENT 1

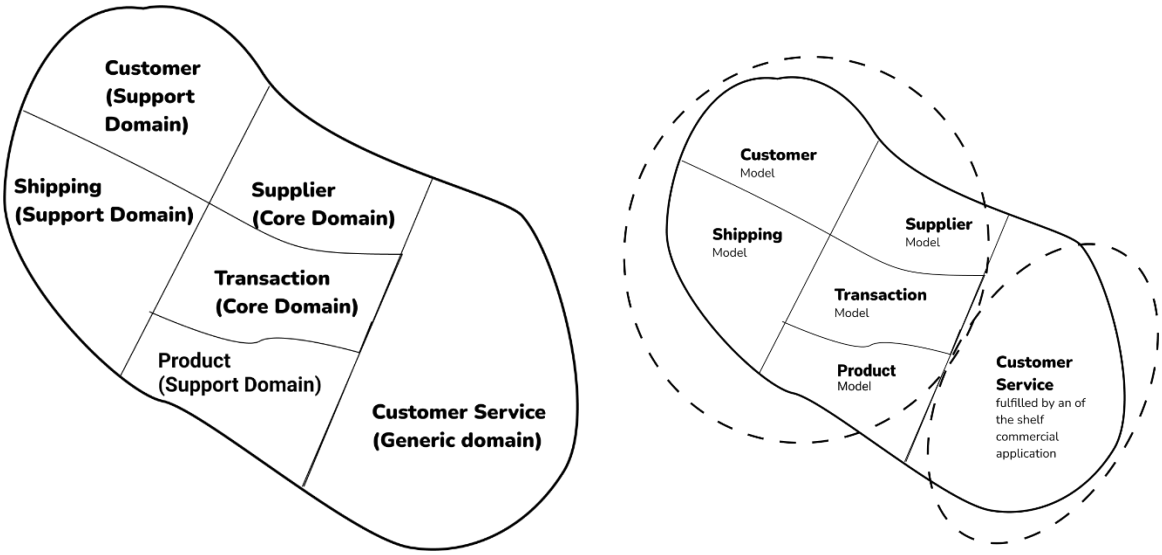
Topik Domain Driven Design = Supermarket

Domain Problem :

1. Customer
2. Shipping
3. Supplier
4. Transaction
5. Product
6. Customer Service

Distilling our problem into problem domain:





subdomain pada Domain Supermarket adalah sebagai berikut:

- Transaction (core domain)

Berfokus pada transaksi yang dilakukan oleh pelanggan agar dapat membeli produk yang diinginkan melalui metode pembayaran yang disediakan dari website.

- Customer (support domain)

Mengenai pelanggan yang telah mendaftarkan diri di website yang mendapatkan akses untuk melakukan transaksi melalui website.

- Product (support domain)

Mengenai produk yang dijual dan ditampilkan di website kepada pelanggan/

- Supplier (core domain)

Mengenai supplier atau produsen produk yang dijual di supermarket untuk mengisi stok produk yang dijual serta keanekaragaman produk yang dijual.

- Shipping (support domain)

Mengenai pengiriman barang ke pelanggan yang telah memesan produk. pengiriman ini akan mengantarkan barang ke alamat yang diberikan pelanggan.

- Customer Service (generic domain)

mengenai pemberian layanan dan bantuan jika terdapat suatu masalah yang dihadapi oleh customer Hal ini bertujuan agar meningkatkan pelayanan supermarket kepada pelanggan.

Skenario yang terjadi pada domain supermarket.

- supermarket membeli produk dari berbagai supplier dalam jumlah yang telah ditentukan.
- Supermarket membeli produk dari supplier untuk mengisi stok produk yang perlu ditambahkan.
- Jumlah Stok dari produk menjadi salah hal yang harus diperhatikan oleh supermarket karena bila kekurangan stok produk tersebut saat pelanggan mencari produknya, akan meninggalkan kerugian bagi kedua belah pihak dimana pihak supermarket akan kehilangan transaksi dan melakukan kesalahan serta pihak pelanggan yang merasa apa yang dijual oleh supermarket tidak lengkap yang tidak menutup kemungkinan pelanggan untuk berpindah ke supermarket lain.
- supplier dapat menawarkan produk baru agar dijual oleh supermarket.
- supermarket menampilkan produk beserta informasinya di halaman website dengan menyusunnya secara beraturan yang bergantung seperti pada kategori produk tersebut.
- Agar dapat membeli produk yang dijual oleh supermarket melalui website, pelanggan perlu terlebih dahulu sign up dengan mengisi informasi-informasi yang wajib diisi agar menjadi member yang mendapatkan akses transaksi jika belum memiliki akun. Sedangkan jika sudah memiliki akun dan sudah terhubung dengan website, dapat memilih produk
- Pelanggan dapat mencari produk dengan men-search nama atau kategori produk di website. Produk akan ditampilkan sesuai dengan pencarian pelanggan.
- Pelanggan dapat membaca informasi produk agar dapat meyakinkan diri dalam memilih produk yang ingin dibeli.
- Pihak supermarket juga akan biasanya membuat promo yang menarik perhatian pelanggan dan menguntungkan pelanggan. Promo ini dapat berupa diskon, obral, dan voucher potongan harga dan sejenisnya.
- Jika pelanggan ingin membeli produk tersebut, dapat memasukkan ke dalam list keranjang /list pilihan. List keranjang dapat menampung semua pilihan produk yang diinginkan pelanggan.
- Jika pelanggan sudah selesai memilih produk yang diinginkan, pelanggan perlu menyesuaikan alamat pengiriman beserta memilih cara pengiriman yang disediakan website.
- setelah itu, pelanggan melanjutkan ke langkah pembayaran atau transaksi.
- pada tahap pembayaran, pelanggan akan memilih satu dari pilihan-pilihan metode pembayaran yang disediakan oleh website.
- Pada tahap ini juga, pelanggan akan memilih kurir pengiriman serta jangka hari pengiriman sesuai harga. Harga tersebut akan ditambahkan pada harga akhir transaksi tersebut.
- Pelanggan mengikuti instruksi sesuai pada tahap pembayaran agar dapat sukses.

- dengan pembayaran yang telah disetujui, transaksi pelanggan akan selesai dan menunggu pengirimannya.
- Bila terjadi hal yang kurang memuaskan pelanggan atau kesalahan dalam penggunaan dalam transaksi atau penggunaan website, pelanggan dapat menghubungi pihak supermarket untuk mendapatkan penyelesaian masalah yang terjadi. Pihak supermarket akan memberikan tanggapan yang diharapkan memuaskan pelanggan. Serta dengan menghubungi pihak supermarket terkait kesalahan yang terjadi, pihak supermarket dapat semakin meningkatkan kualitas pelayanan serta kekurangan yang terjadi pada website ataupun transaksi.

Ubiquitous language:

Supplier Domain:

- Category Product
- Supplier

Transaction Domain:

- Payment
- Order

Customer Domain:

- Member
- Order

Product Domain:

- Category product
- Stock Management

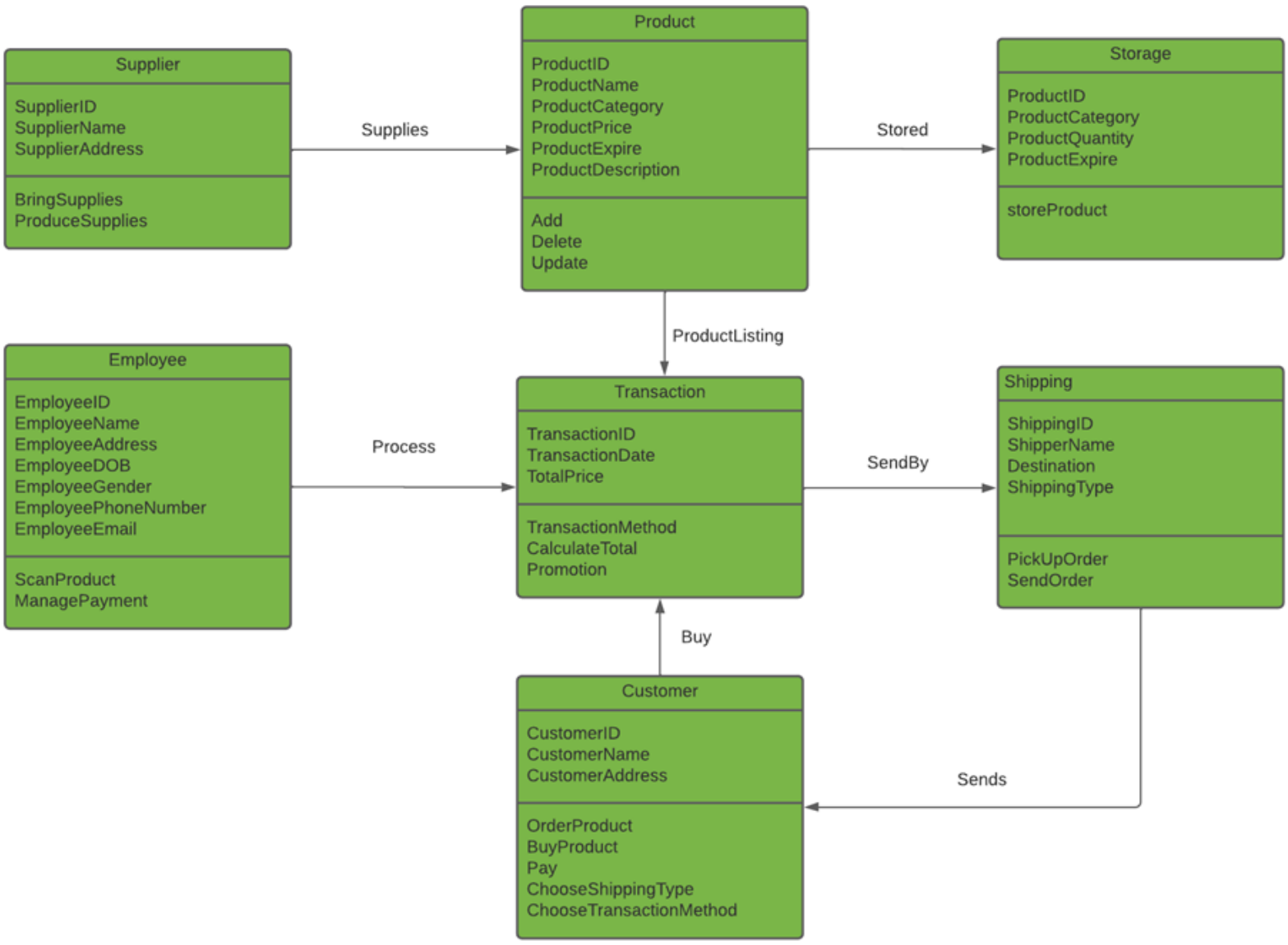
Shipping Domain:

- Order
- Shipment

Customer Service Domain:

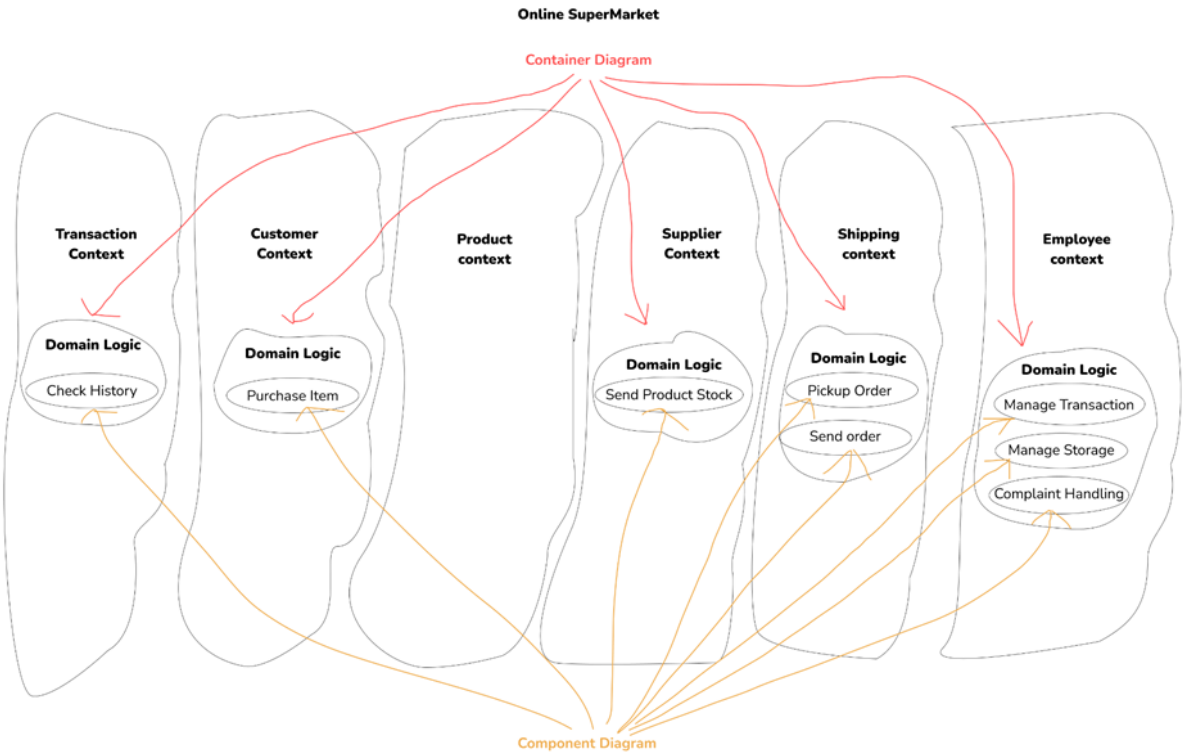
- Member
- Customer Service

ASSIGNMENT 2

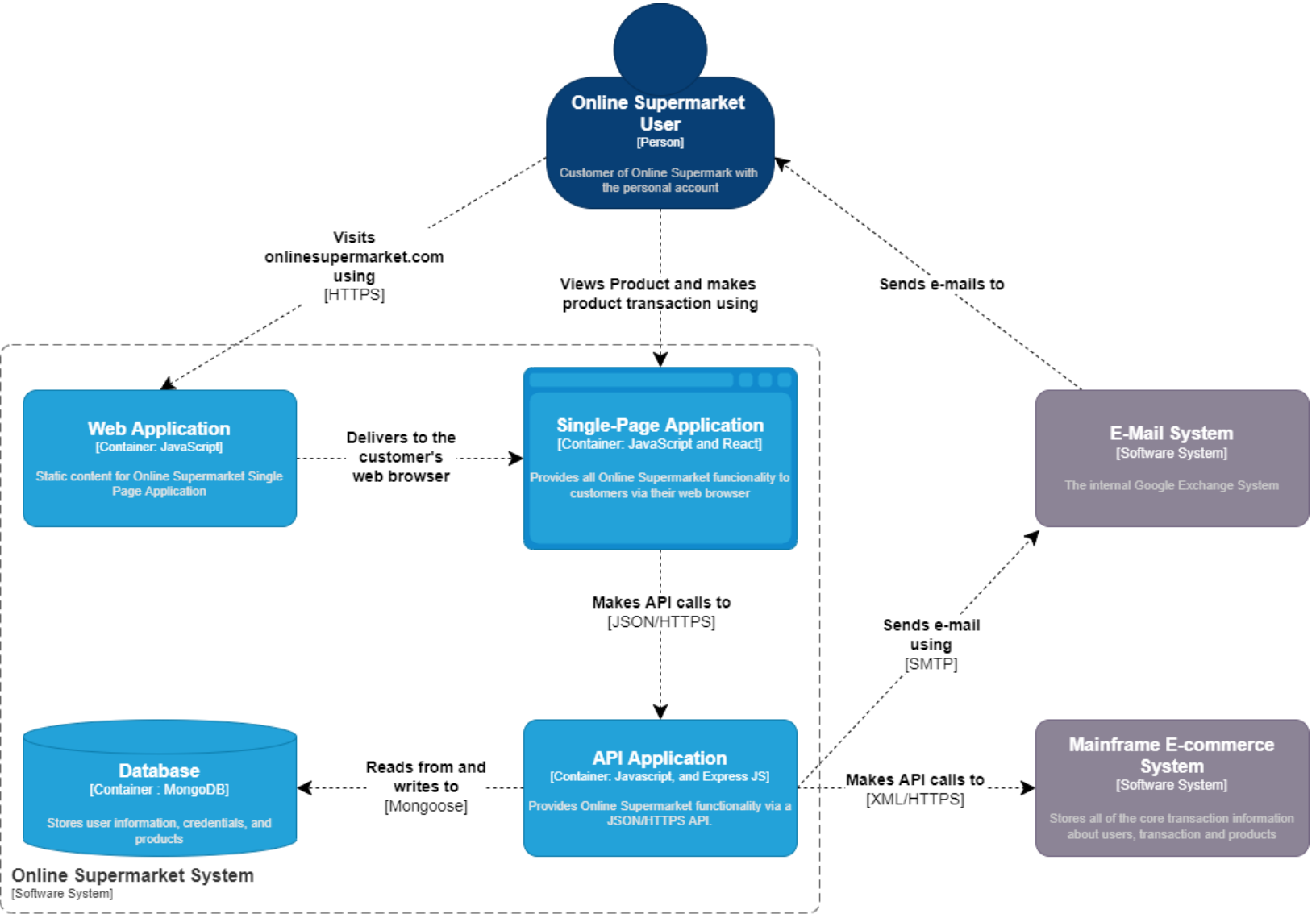


ASSIGNMENT 3

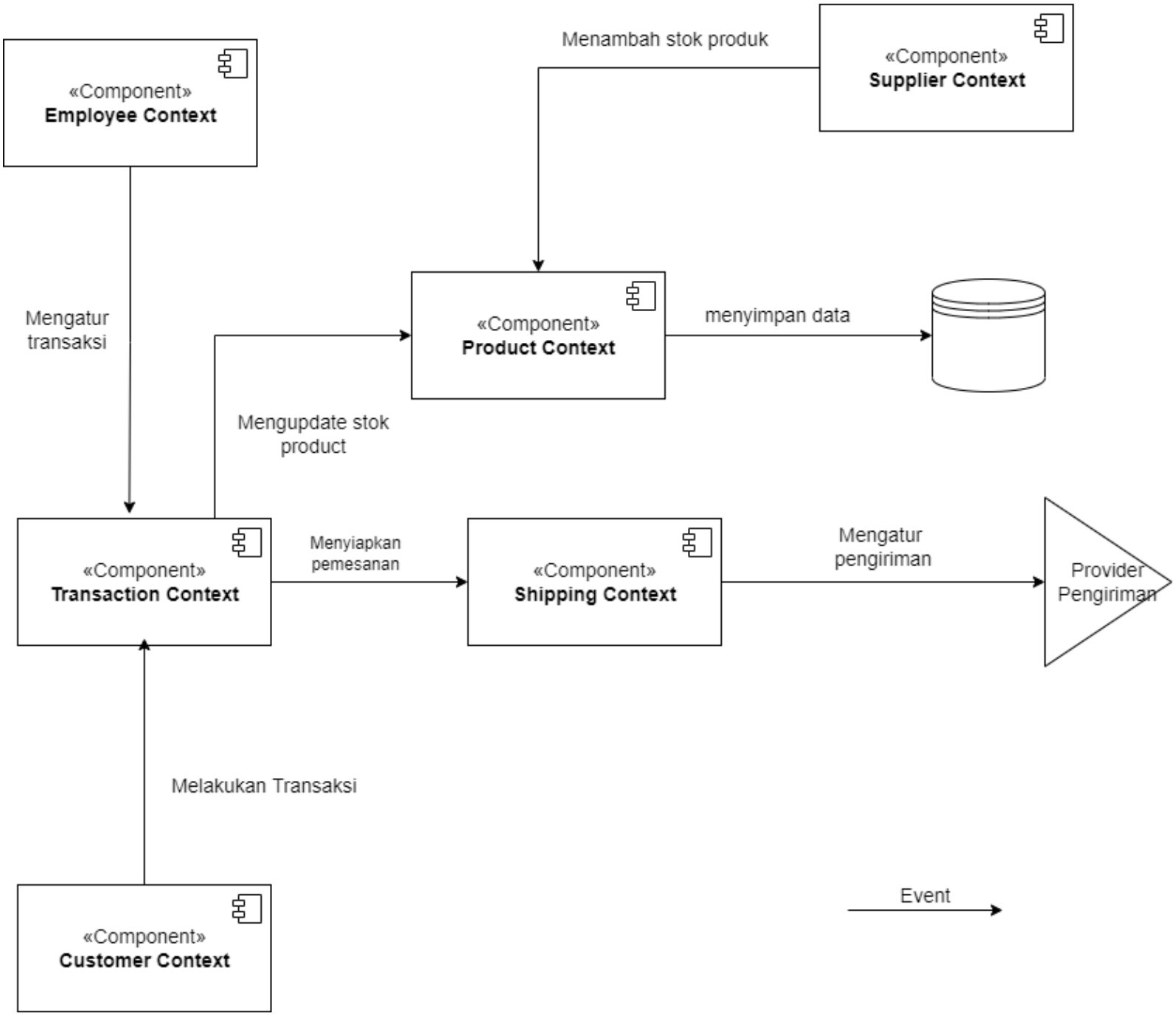
Bounded Context



Container Diagram

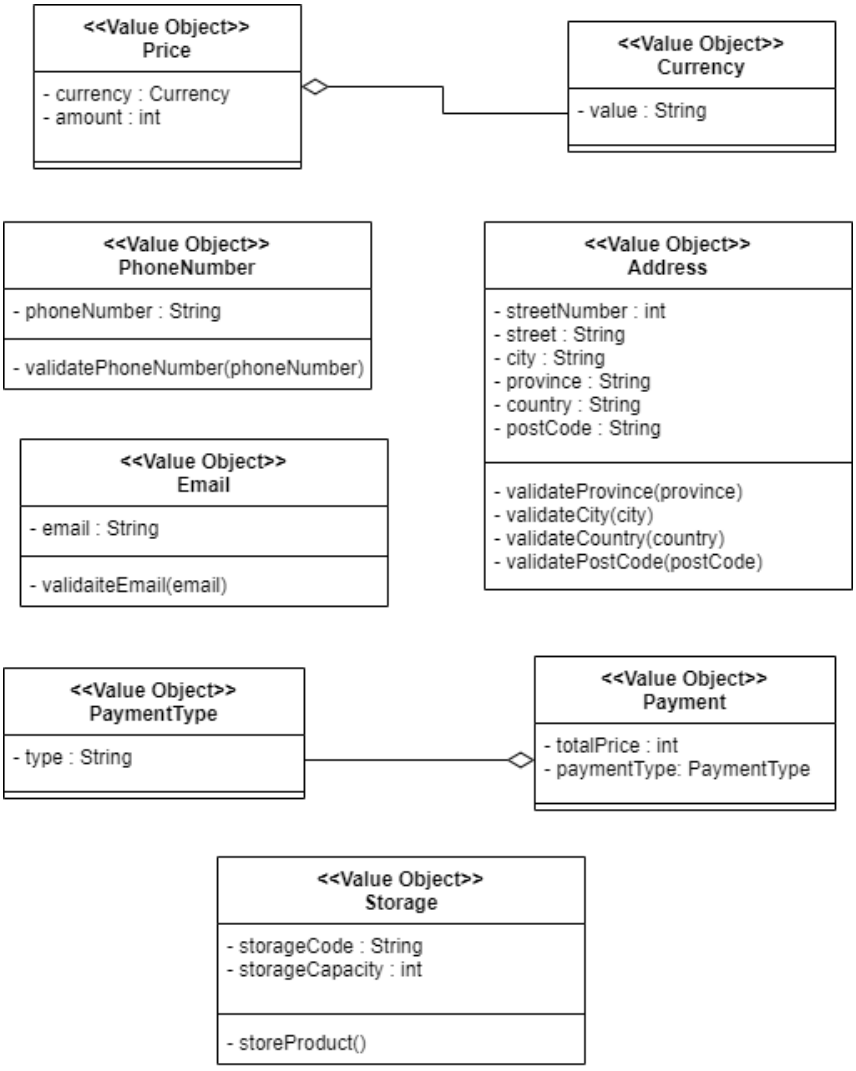


Component Diagram

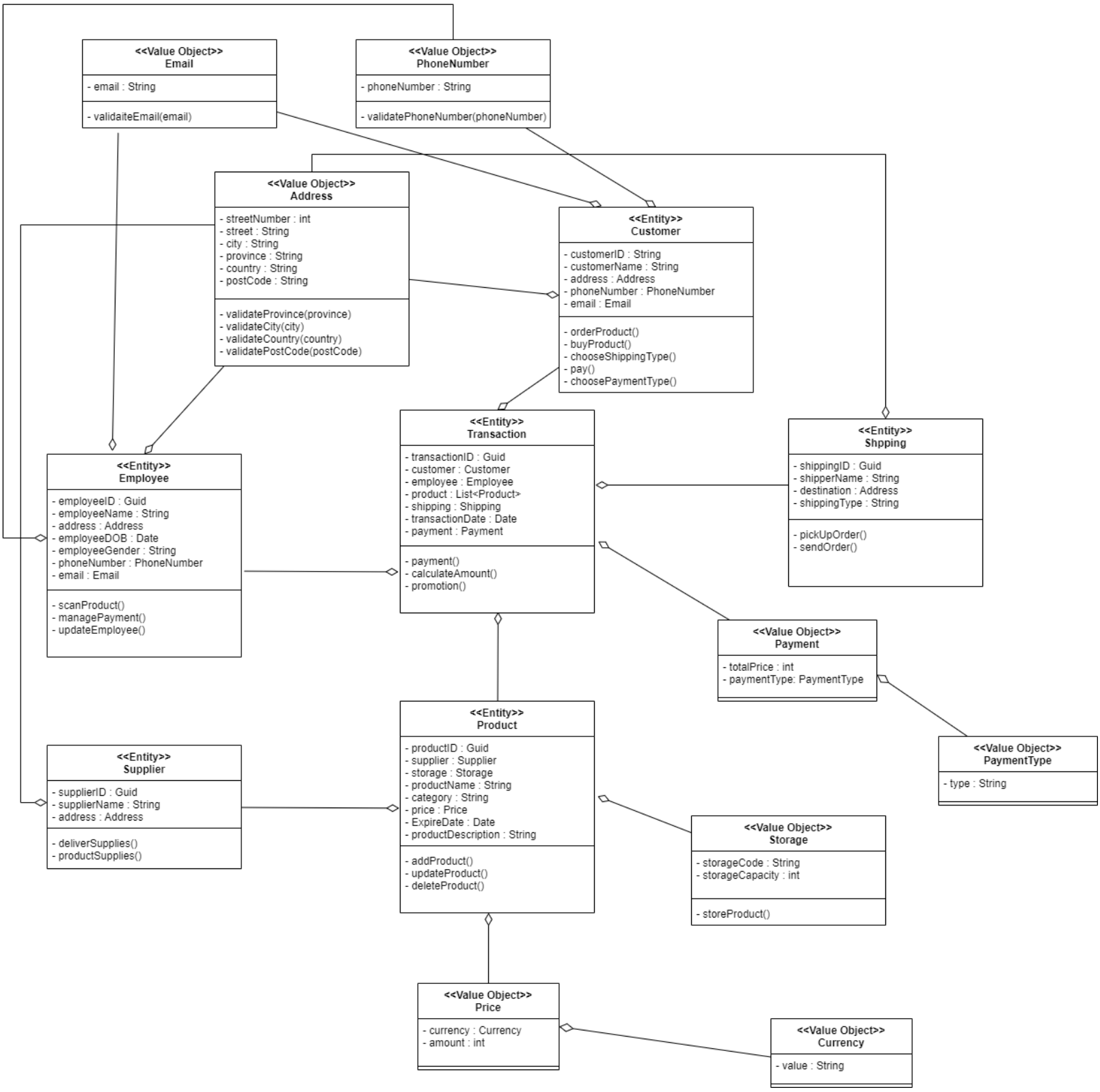


ASSIGNMENT 4

Value Object Model



Entity Model



Code

```
//value object
public class Currency
{
    public Currency(string value)
    {
        this.value = value;
    }
    private string value { get; set; }
}

public class Price
{
    public Price(Currency currency, int amount)
    {
        this.currency = currency;
        this.amount = amount;
    }
    private Currency currency { get; set; }
    private int amount { get; set; }
}

public class PhoneNumber
{
    public PhoneNumber(string phoneNumber)
    {
        this.phoneNumber = phoneNumber;
    }
```

```
    private string phoneNumber { get; set; }
    private void validatePhoneNumber(string phoneNumber)
    {
        //...
    }
}

public class Email
{
    public Email(string email)
    {
        this.email = email;
    }
    private string email { get; set; }
    private void validateEmail(string email)
    {
        //...
    }
}

public class PaymentType
{
    public PaymentType(string type)
    {
        this.type = type;
    }
    private string type { get; set; }
}
```

```
public class Payment
{
    public Payment(int totalPrice, PaymentType paymentType)
    {
        this.totalPrice = totalPrice;
        this.paymentType = paymentType;
    }
    private int totalPrice { get; set; }
    private PaymentType paymentType { get; set; }
}

public class Address
{
    public Address(int streetNumber, string street, string city, string province, string country,
string postCode)
    {
        this.streetNumber = streetNumber;
        this.street = street;
        this.city = city;
        this.province = province;
        this.country = country;
        this.postCode = postCode;
    }
    private int streetNumber { get; set; }
    private string street { get; set; }
    private string city { get; set; }
    private string province { get; set; }
    private string country { get; set; }
```

```
    private string postCode { get; set; }

    private void validateProvince(string province)
    {
        //...
    }
    private void validateCity(string city)
    {
        //...
    }
    private void validateCountry(string country)
    {
        //...
    }
    private void validatePostCode(string postCode)
    {
        //...
    }
}

public class Storage
{
    public Storage(int StorageCapacity)
    {
        this.StorageID = new Guid();
        this.StorageCapacity = StorageCapacity;
    }
```

```
    public Guid StorageID { get; set; }
    public int StorageCapacity { get; set; }

    public void StoreProduct()
    {
        //...
    }
}

//Entity
public class Product
{
    public Product(Guid ProductId ,Guid SupplierId ,Storage StorageID ,string ProductName
,string ProductCategory ,int ProductPrice ,DateTime productExpire ,string productDescription)
    {
        this.ProductId = new Guid();
        this.SupplierId = SupplierId;
        this.StorageID=StorageID;
        this.ProductName = ProductName;
        this.ProductCategory = ProductCategory;
        this.ProductPrice = ProductPrice;
        this.productExpire = productExpire;
        this.productDescription = productDescription;
    }

    public Guid ProductId { get; set; }
    public Guid SupplierId { get; set; }
    public Storage StorageID { get; set; }
    public string ProductName { get; set; }
```

```
    public string ProductCategory { get; set; }
    public int ProductPrice { get; set; }
    public DateTime productExpire { get; set; }
    public string productDescription { get; set; }

    public void AddProduct()
    {
        //...
    }
    public void DeleteProduct()
    {
        //...
    }
    public void UpdateProduct()
    {
        //...
    }
}

public class Supplier
{
    public Supplier( string SupplierName, Address address)
    {
        this.SupplierId = new Guid();
        this.SupplierName = SupplierName;
        this.SupplierAddress = address;
    }
```

```
    public Guid SupplierId { get; set; }
    public string SupplierName { get; set; }
    public Address SupplierAddress { get; set; }

    public void BringSupplies()
    {
        //...
    }
    public void ProduceSupplies()
    {
        //...
    }
}

public class Employee
{
    public Employee(string EmployeeName, Address EmployeeAddress, DateTime
EmployeeDOB, string EmployeeGender, PhoneNumber EmployeePhoneNumber, Email
EmployeeEmail)
    {
        this.EmployeeID = new Guid();
        this.EmployeeName = EmployeeName;
        this.EmployeeAddress = EmployeeAddress;
        this.EmployeeDOB = EmployeeDOB;
        this.EmployeeGender = EmployeeGender;
        this.EmployeePhoneNumber = EmployeePhoneNumber;
        this.EmployeeEmail = EmployeeEmail;
    }
```

```
    public Guid EmployeeID { get; set; }
    public string EmployeeName { get; set; }
    public Address EmployeeAddress { get; set; }
    public DateTime EmployeeDOB { get; set; }
    public string EmployeeGender { get; set; }
    public PhoneNumber EmployeePhoneNumber { get; set; }
    public Email EmployeeEmail { get; set; }

    public void ScanProduct()
    {
        //...
    }
    public void ManagePayment()
    {
        //...
    }
}

public class Shipping
{
    public Shipping(string ShipperName, Address Destination, string ShippingType)
    {
        this.ShippingId = new Guid();
        this.ShipperName = ShipperName;
        this.Destination = Destination;
        this.ShippingType = ShippingType;
    }
    public Guid ShippingId { get; set;}
```

```
    public string ShipperName { get; set;}
    public Address Destination { get; set; }
    public string ShippingType { get; set; }

    public void PickUpOrder()
    {
        //...
    }

    public void SendOrder()
    {
        //...
    }
}

public class Customer
{
    public Customer(string CustomerName, Address CustomerAddress, PhoneNumber
PhoneNumber, Email Email)
    {
        this.CustomerId = new Guid();
        this.CustomerName = CustomerName;
        this.CustomerAddress = CustomerAddress;
        this.Email = Email;
        this.PhoneNumber = PhoneNumber;
    }
    public Guid CustomerId { get; set; }
    public string CustomerName { get; set; }
    public Address CustomerAddress { get; set; }
```

```
    public PhoneNumber PhoneNumber { get; set; }
    public Email Email { get; set; }

    public void OrderProduct()
    {
        //...
    }
    public void BuyProduct()
    {
        //...
    }
    public void Pay()
    {
        //...
    }
    public void ChooseShippingType()
    {
        //...
    }
    public void ChooseTransactionMethod()
    {
        //...
    }
}

public class Transaction
{
    public Transaction(Customer CustomerID, Product ProductID, Employee EmployeeID,
Shipping ShippingID, DateTime TransactionDate, Payment payment)
```

```
{
    this.TransactionId = new Guid();
    this.CustomerID = CustomerID;
    this.ProductID = ProductID;
    this.EmployeeID = EmployeeID;
    this.ShippingID = ShippingID;
    this.TransactionDate = TransactionDate;
    this.Payment = payment;
}
public Guid TransactionId { get; set; }
public Customer CustomerID { get; set; }
public Product ProductID { get; set; }
public Employee EmployeeID { get; set; }
public Shipping ShippingID { get; set; }
public DateTime TransactionDate { get; set; }
public Payment Payment { get; set; }

public void TransactionMethod()
{
    //...
}

public void CalculateTotal()
{
    //...
}

public void Promotion()
```

```
    {
        //...
    }
}
```

ASSIGNMENT 5

Domain Services

- Promotion

Promotion adalah layanan domain promosi atau diskon yang digunakan untuk meningkatkan perhatian Customer agar meningkatkan penjualan produk serta meningkatkan minat Customer pada online shop.

```
public class Promotion
{
    public Guid PromotionId { get; set;}
    public string PromotionName { get; set;}
    public DateTime PromotionExpiredDate { get; set; }
    private List<Product> productList;
    public void createPromotion(List<Product> productList, string PromotionName, DateTime PromotionExpiredDate)
    {
        this.PromotionId = new Guid();
        this.PromotionName = PromotionName;
        this.PromotionExpiredDate = PromotionExpiredDate;
        this.productList = productList;
    }
```

}

- Tier

Tier adalah layanan domain yang berupa pangkat atau status dari Customer terkait keaktifan dalam transaksi yang dilakukan dalam jangka waktu tertentu.

```
public class Tier
{
    public Guid TierId { get; set; }
    public string TierName { get; set; }
    public int totalPoint { get; set; }
    public void createTier()
    {
        this.TierId = new Guid();
        this.TierName = "Bronze";
        this.totalPoint = 0;
    }
    public void upgradeTier()
    {
        if (this.totalPoint > 3500)
        {
            this.TierName = "Diamond";
        }
        else if (this.totalPoint > 1500)
        {
            this.TierName = "Platinum";
        }
        else if (this.totalPoint > 500)
        {
```

```
            this.TierName = "Gold";
        }
    }
    public void addPoint(int points)
    {
        this.totalPoint = this.totalPoint + points;
    }
}
```

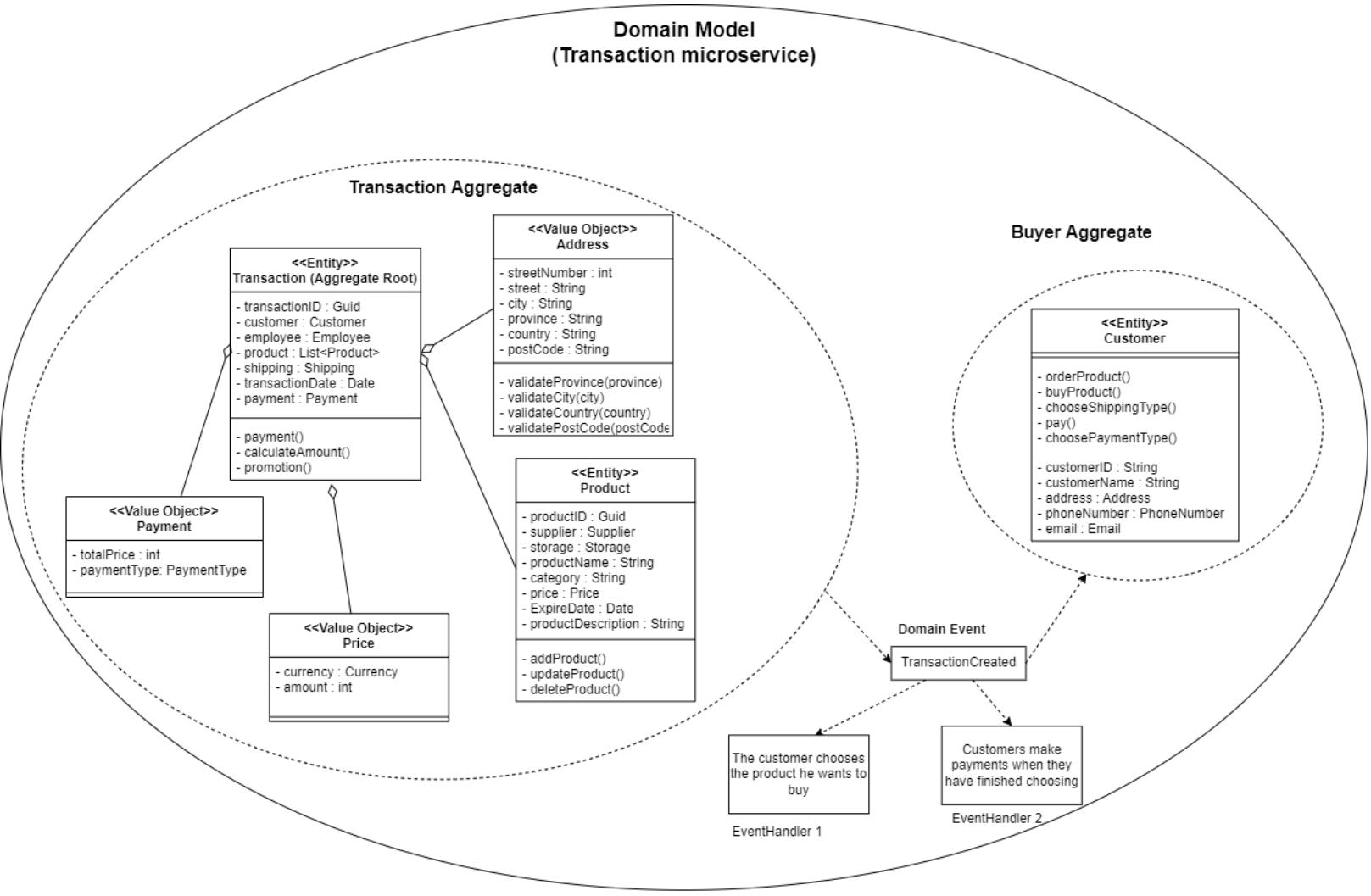
- Quest

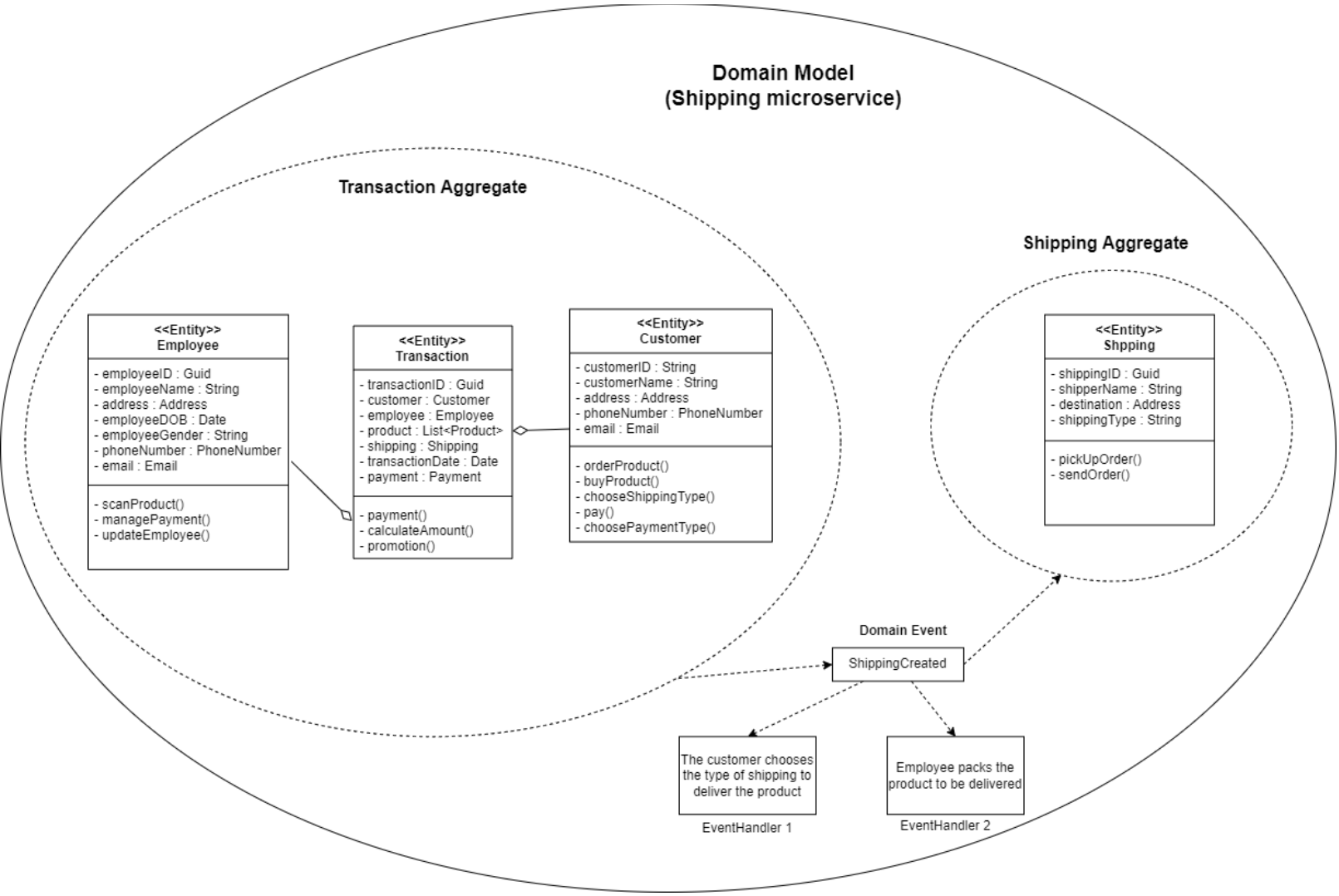
Quest adalah layanan domain yang berupa sebuah misi menarik dimana Customer mendapatkan tantangan berupa target transaksi dalam jangka waktu tertentu dimana bila berhasil diselesaikan, maka Customer akan mendapatkan hadiah yang bisa berupa voucher, diskon, dan lain-lain.

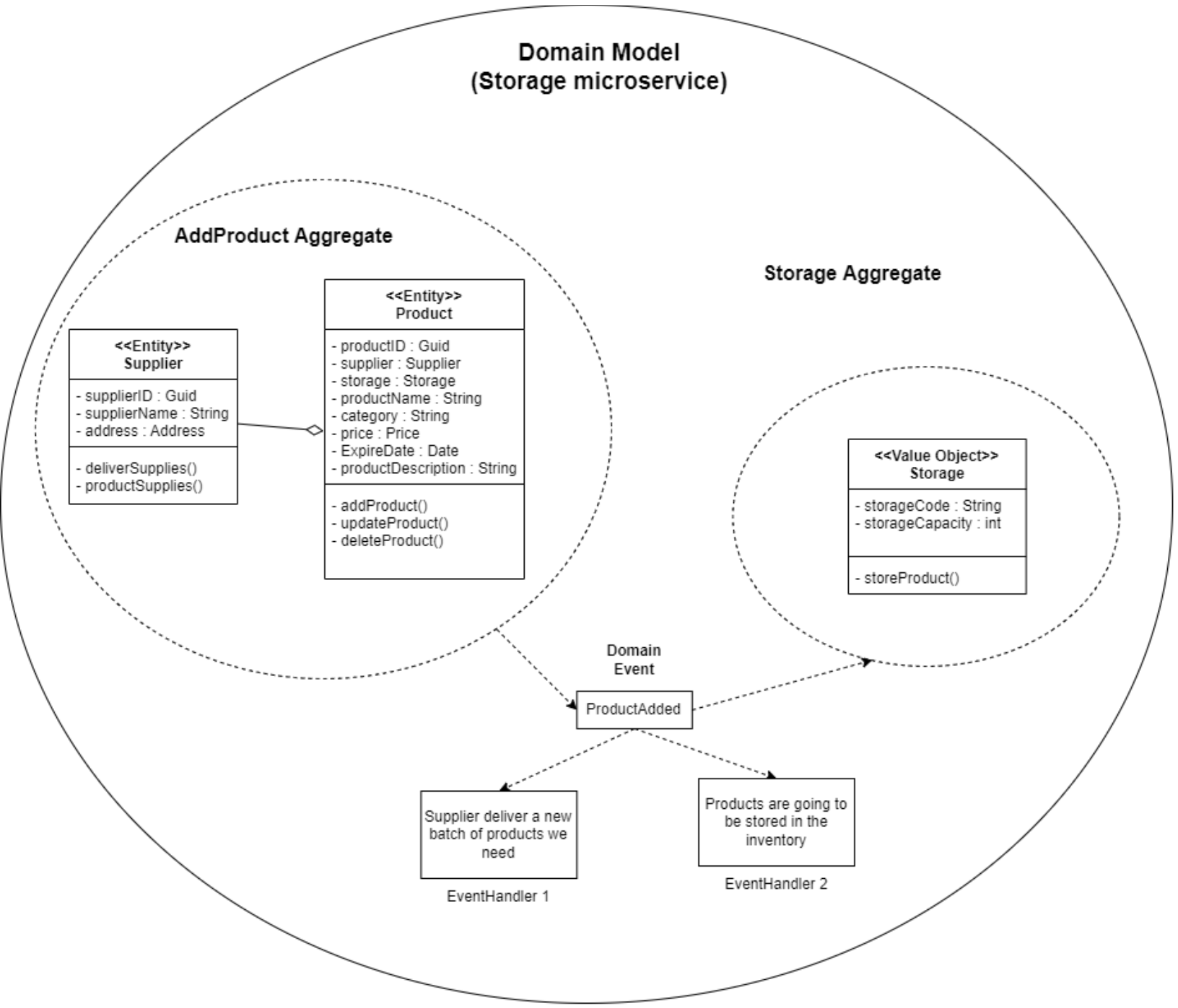
```
public class Quest
{
    public Guid QuestId { get; set; }
    public string QuestName { get; set; }
    public DateTime startDate { get; set; }
    public DateTime endDate { get; set; }
    public bool isDone { get; set; }

    public void createQuest(string QuestName, DateTime startDate, DateTime endDate)
    {
        this.QuestId = new Guid();
        this.QuestName = QuestName;
        this.startDate = startDate;
        this.endDate = endDate;
        this.isDone = false;
```

```
    }
    public void questCompleted()
    {
        this.isDone = true;
        // Give user the promotion
        new Promotion();
    }
}
```

Domain Event



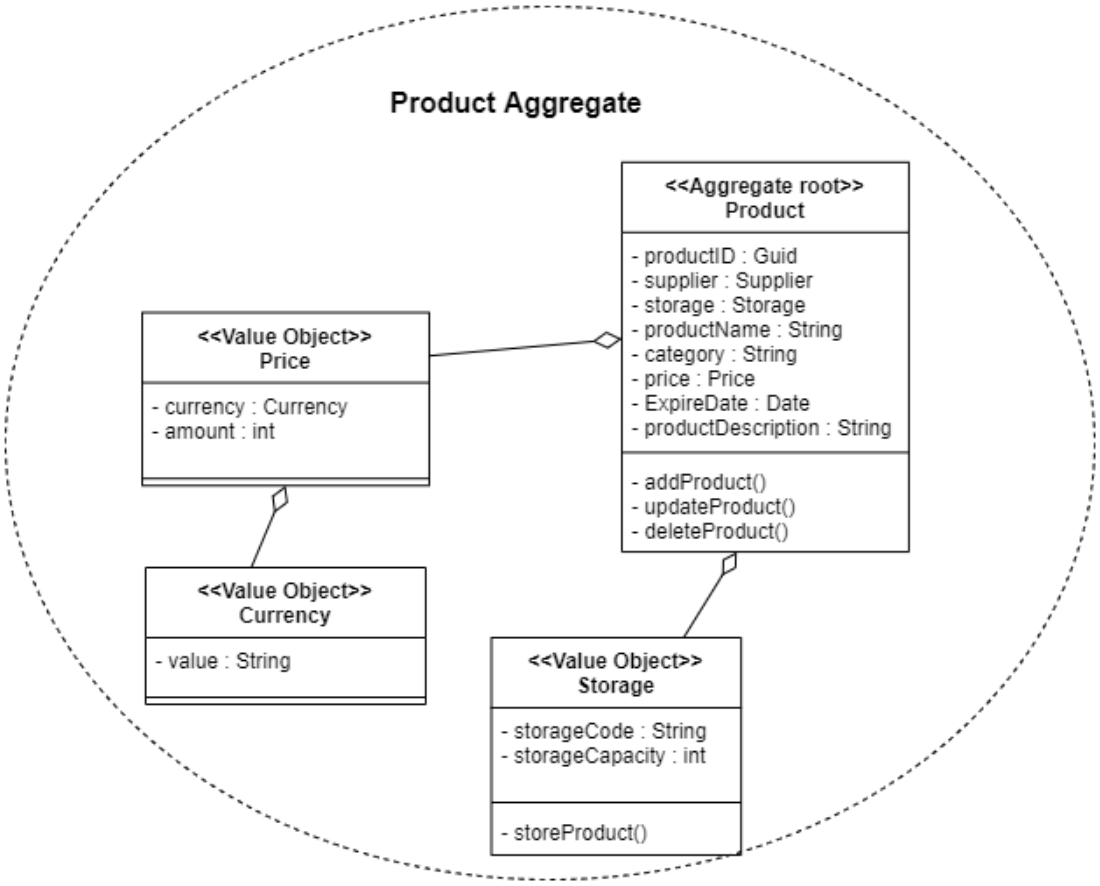

Domain Model
(Shipping microservice)
Transaction Aggregate
Shipping Aggregate
<<Entity>>
Employee
- employeeID : Guid
- employeeName : String
- address : Address
- employeeDOB : Date
- employeeGender : String
- phoneNumber : PhoneNumber
- email : Email
- scanProduct()
- managePayment()
- updateEmployee()
<<Entity>>
Transaction
- transactionID : Guid
- customer : Customer
- employee : Employee
- product : List<Product>
- shipping : Shipping
- transactionDate : Date
- payment : Payment
- payment()
- calculateAmount()
- promotion()
<<Entity>>
Customer
- customerID : String
- customerName : String
- address : Address
- phoneNumber : PhoneNumber
- email : Email
- orderProduct()
- buyProduct()
- chooseShippingType()
- pay()
- choosePaymentType()
<<Entity>>
Shpping
- shippingID : Guid
- shipperName : String
- destination : Address
- shippingType : String
- pickUpOrder()
- sendOrder()
Domain Event
ShippingCreated
The customer chooses the type of shipping to deliver the product
EventHandler 1
Employee packs the product to be delivered
EventHandler 2


Domain Model
(Storage microservice)
AddProduct Aggregate
<<Entity>>
Supplier
- supplierID : Guid
- supplierName : String
- address : Address
- deliverSupplies()
- productSupplies()
<<Entity>>
Product
- productID : Guid
- supplier : Supplier
- storage : Storage
- productName : String
- category : String
- price : Price
- ExpireDate : Date
- productDescription : String
- addProduct()
- updateProduct()
- deleteProduct()
Storage Aggregate
<<Value Object>>
Storage
- storageCode : String
- storageCapacity : int
- storeProduct()
Domain
Event
ProductAdded
Supplier deliver a new batch of products we need
EventHandler 1
Products are going to be stored in the inventory
EventHandler 2

ASSIGNMENT 6

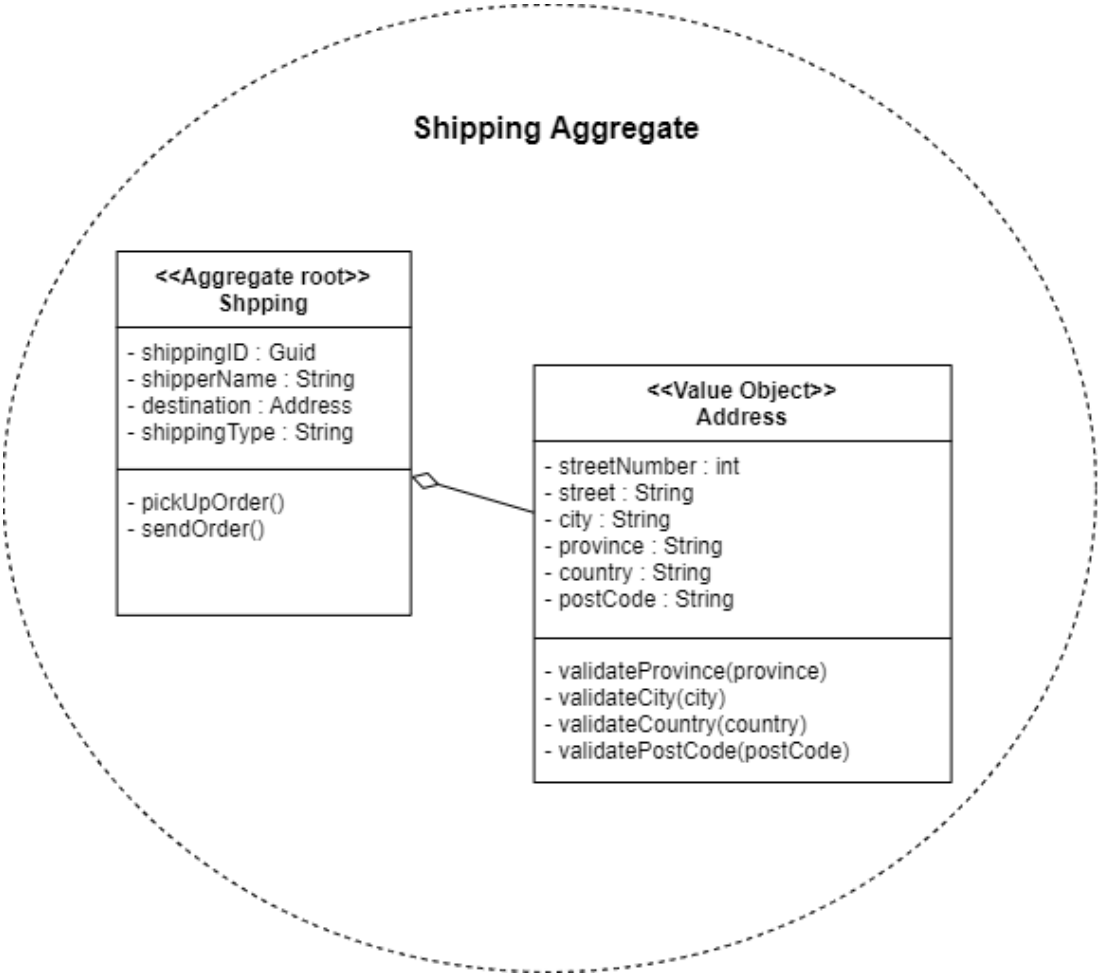
Aggregate merupakan sebuah kesatuan antara Entity dan Value Object yang merepresentasi konsep domain dan dibatasi oleh aggregate boundary. Setiap aggregate memiliki aggregate root yang bertugas sebagai referensi untuk entry ke aggregate tersebut.

Product Aggregate



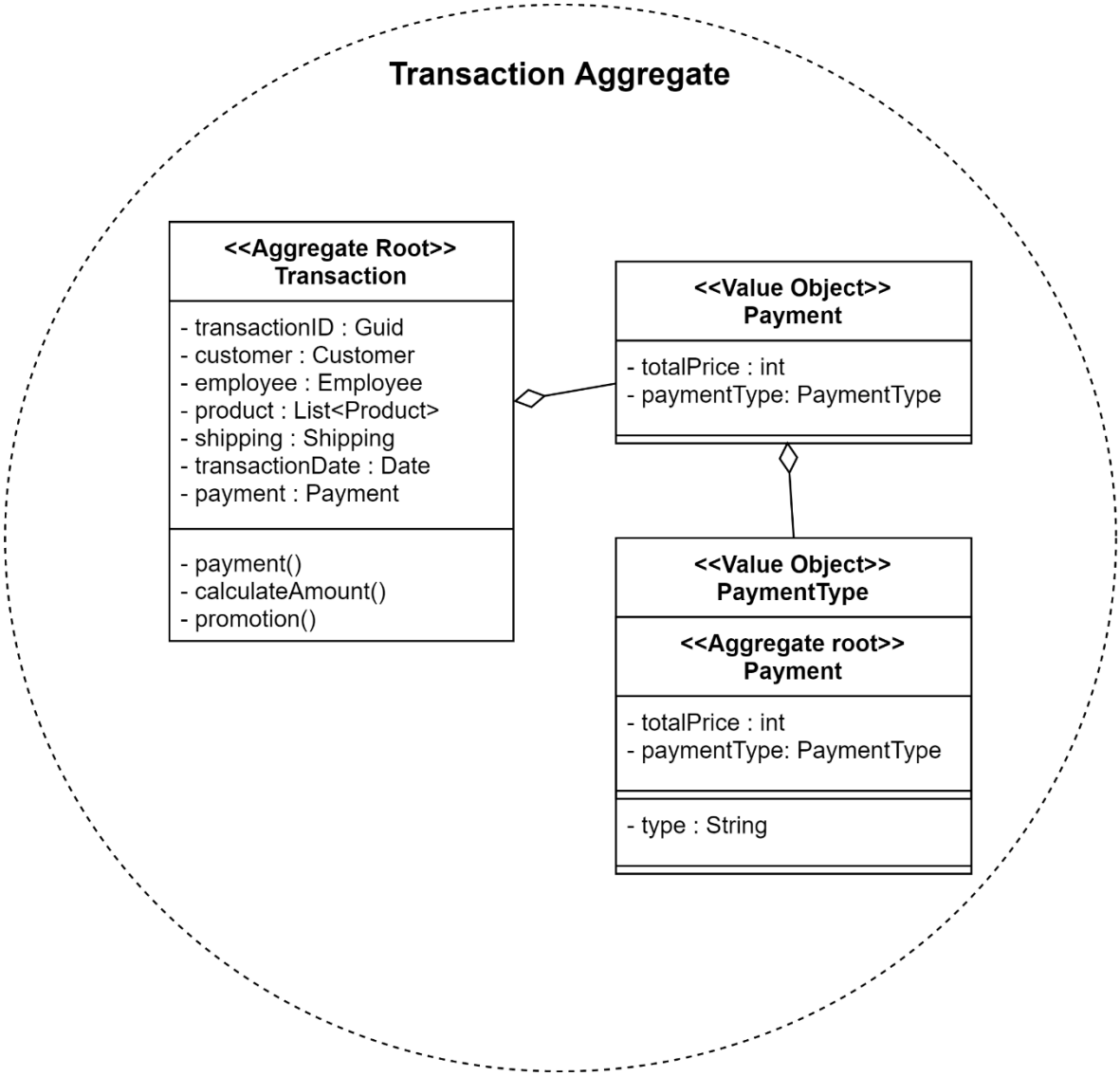
Aggregate product bertugas atau mengatur pada bagian manajemen produk dari supermarket. Entity product diatur sebagai aggregate root yang menjadi referensi untuk mengakses ke product aggregate.

Shipping Aggregate



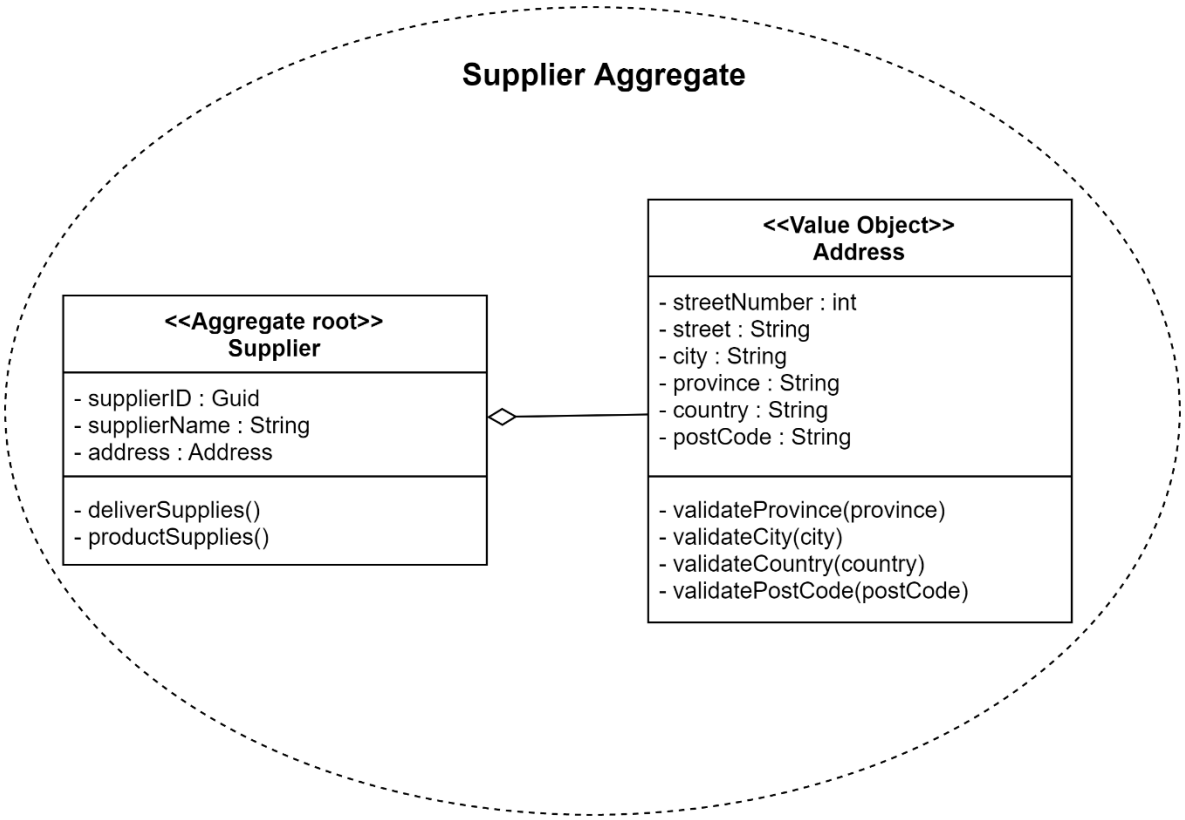
Aggregate shipping bertugas atau mengatur pada bagian manajemen pengiriman produk dari yang diatur oleh transaksi supermarket. Entity Shipping diatur sebagai aggregate root yang menjadi referensi untuk mengakses ke shipping aggregate

Transaction Aggregate



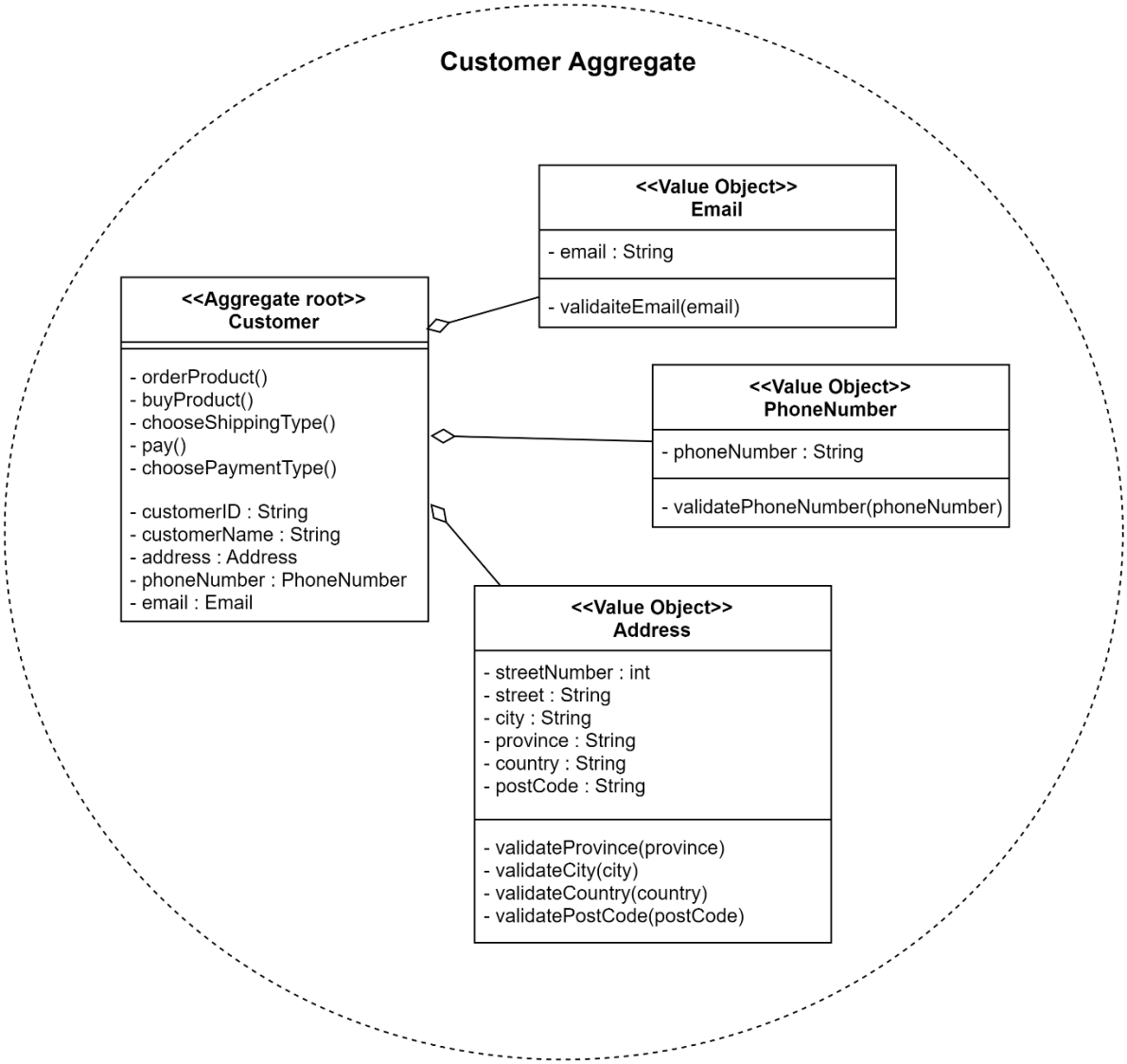
Aggregate Transaction bertugas mengatur pada bagian manajemen transaksi dari supermarket. Entity Transaction diatur sebagai aggregate root yang menjadi referensi untuk mengakses ke Transaction aggregate.

Supplier Aggregate



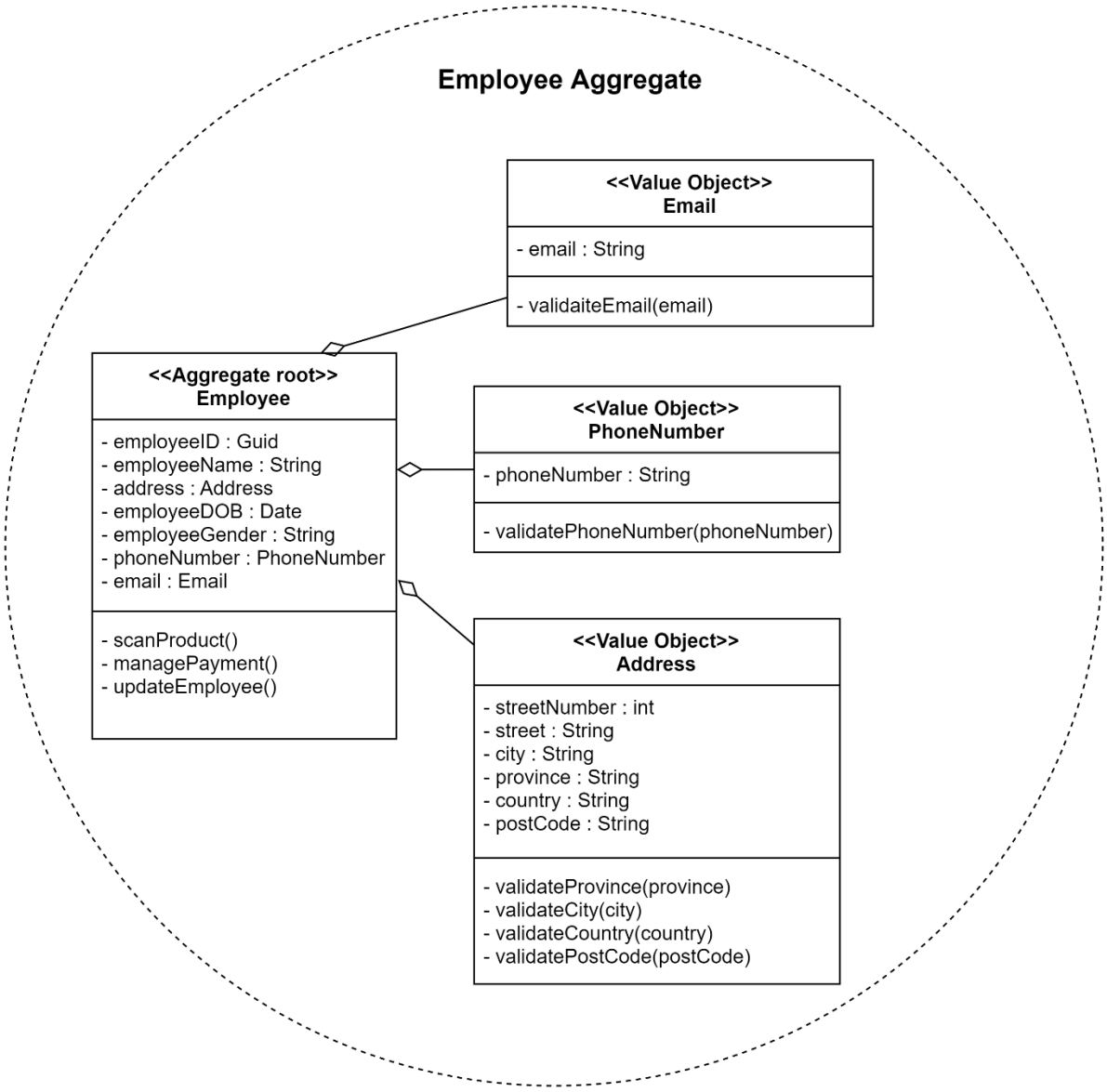
Aggregate supplier bertugas atau mengatur pada bagian penyedia dan pemasok stok produk untuk supermarket. Entity Supplier diatur sebagai aggregate root yang menjadi referensi untuk mengakses ke Supplier aggregate

Customer Aggregate



Aggregate product bertugas atau mengatur pada bagian data dari customer supermarket. Entity Customer diatur sebagai aggregate root yang menjadi referensi untuk mengakses ke Customer aggregate.

Employee Aggregate



Aggregate employee bertugas atau mengatur pada bagian karyawan dari supermarket. Entity Employee diatur sebagai aggregate root yang menjadi referensi untuk mengakses ke employee aggregate.

ASSIGNMENT 7

Factory

Implementasi Factory pada Domain Driven Design dari kelompok kami ialah berbentuk method di aggregate untuk membuat object baru. Sesuai dengan tugas dan fungsi dari Factory ialah untuk membuat object baru. Sehingga setiap aggregate baik itu Customer aggregate, Employee aggregate, Product aggregate, Shipping aggregate, Supplier aggregate, dan Transaction aggregate memiliki factory method masing-masing untuk membuat sebuah object.

Customer Aggregate and Factory method

```
namespace Domain.Customer{
  public partial class Customer
  {
    public Customer(string CustomerName, Address CustomerAddress, PhoneNumber
PhoneNumber, Email Email)
      {
        this.update(CustomerName,  CustomerAddress,  PhoneNumber,  Email)
      }

    public void update(string CustomerName, Address CustomerAddress, PhoneNumber
PhoneNumber, Email Email){
      this.CustomerId = new Guid();
      this.CustomerName = CustomerName;
      this.CustomerAddress = CustomerAddress;
      this.Email = Email;
      this.PhoneNumber = PhoneNumber;
    }
    //Factory Method for Creating customer object
    public Customer createCustomerFactory(string CustomerName, Address CustomerAddress,
PhoneNumber PhoneNumber, Email Email) {
```

```
    return new Customer(CustomerName,CustomerAddress,PhoneNumber,Email);
   }
  }
}
```

Employee Aggregate and Factory method

```
namespace Domain.Employee
{
  public partial class Employee
  {
    public Employee(string EmployeeName, Address EmployeeAddress, DateTime
EmployeeDOB, string EmployeeGender, PhoneNumber EmployeePhoneNumber, Email
EmployeeEmail)
    {
     this.update(EmployeeName, EmployeeAddress, EmployeeDOB, EmployeeGender,
EmployeePhoneNumber, EmployeeEmail);
    }

     public void update(string EmployeeName, Address EmployeeAddress, DateTime
EmployeeDOB, string EmployeeGender, PhoneNumber EmployeePhoneNumber, Email
EmployeeEmail){
       this.EmployeeID = new Guid();
     this.EmployeeName = EmployeeName;
     this.EmployeeAddress = EmployeeAddress;
     this.EmployeeDOB = EmployeeDOB;
     this.EmployeeGender = EmployeeGender;
     this.EmployeePhoneNumber = EmployeePhoneNumber;
     this.EmployeeEmail = EmployeeEmail;
     }
    //factory method for creating employee object
```

```
   public Employee createEmployeeFactory(string EmployeeName, Address EmployeeAddress, DateTime EmployeeDOB, string EmployeeGender, PhoneNumber EmployeePhoneNumber, Email EmployeeEmail)
       {
          return new Employee(EmployeeName, EmployeeAddress, EmployeeDOB, EmployeeGender, EmployeePhoneNumber, EmployeeEmail);
       }
  }
}
```

Product Aggregate and Factory method

```
namespace Domain.Product
{
  public partial class Product
  {
     public Product( Supplier supplier, Storage StorageID, string ProductName, string ProductCategory, int ProductPrice, DateTime productExpire, string productDescription)
     {
        this.update(ProductId, supplier, StorageID, ProductName, ProductCategory, ProductPrice, productExpire, productDescription)
     }

     public void update( Supplier supplier, Storage StorageID, string ProductName, string ProductCategory, int ProductPrice, DateTime productExpire, string productDescription){
       this.ProductId = new Guid();
       this.SupplierId = supplier;
       this.StorageID = StorageID;
       this.ProductName = ProductName;
       this.ProductCategory = ProductCategory;
       this.ProductPrice = ProductPrice;
```

```
        this.productExpire = productExpire;
        this.productDescription = productDescription;
      }
      //factory method for creating product object
      public Product createProductFactory(Supplier supplier, Storage StorageID, string ProductName, string ProductCategory, int ProductPrice, DateTime productExpire, string productDescription){
        return new Product(new Guid(), supplier, StorageID, ProductName, ProductCategory, ProductPrice, productExpire, productDescription);
      }
  }
}
```

Shipping Aggregate and Factory method

```
namespace Domain.Shipping
{
  public partial class Shipping
  {
    public Shipping(string ShipperName, Address Destination, string ShippingType){
      this.update(ShipperName, Destination, ShippingType)
    }

    public void update(string ShipperName, Address Destination, string ShippingType){
        this.ShippingId = new Guid();
        this.ShipperName = ShipperName;
        this.Destination = Destination;
        this.ShippingType = ShippingType;
    }
    //factory method for creating shipping object
```

```
   public Shipping createShippingFactory(string ShipperName, Address Destination, string ShippingType){
     return new Shipping(ShipperName, Destination, ShippingType);
   }
  }
}
```

Supplier Aggregate and Factory method

```
namespace Domain.Supplier
{
   public partial class Supplier
   {
      public Supplier(string SupplierName, Address address)
      {
         this.update(SupplierName, address);
      }

      public void update(string SupplierName, Address address)
      {
         this.SupplierId = new Guid();
         this.SupplierName = SupplierName;
         this.SupplierAddress = address;
      }
      //Factory method for creating supplier object
      public Supplier createSupplierFactory(string SupplierName, Address address)
      {
         return new Supplier(SupplierName, address);
      }
   }
```

```
}
```

Transaction Aggregate and Factory method

```
namespace Domain.Transaction
{
   public partial class Transaction
   {
      public Transaction(Customer customer, Product product, Employee employee, Shipping shipping, DateTime TransactionDate, Payment payment)
      {
         this.Update(customer, product, employee, shipping, TransactionDate, payment);
      }

      public void Update(Customer customer, Product product, Employee employee, Shipping shipping, DateTime TransactionDate, Payment payment)
      {
         this.TransactionId = new Guid();
         this.customer = customer;
         this.product = product;
         this.product = employee;
         this.shipping = shipping;
         this.TransactionDate = TransactionDate;
         this.Payment = payment;
      }
      //Factory Method for creating Transaction object
      public Transaction createTransactionFactory(Customer customer, Product product, Employee employee, Shipping shipping, DateTime TransactionDate, Payment payment)
      {
         return new Transaction(customer, product, employee, shipping, TransactionDate, payment);
```

```
    }
  }
}
```

Repository

Customer Repositor

```
namespace Infrastructure.Repositories.CustomerRepository
{
  internal class CustomerRepository
  {
    public void Add(Guid CustomerID, string CustomerName, Address CustomerAddress,
PhoneNumber PhoneNumber, Email Email)
    {
      (INSERT INTO VALUES (CustomerID, CustomerName, CustomerAddress,
PhoneNumber, Email));
    }

    public void GetAll()
    {
      (SELECT * FROM Customer);
    }

    public void Get(Guid CustomerID)
    {
      (SELECT * FROM Customer WHERE customerID LIKE(CustomerID));
    }

    public void Update(Guid CustomerID, string CustomerName, Address CustomerAddress,
PhoneNumber PhoneNumber, Email Email)
    {
      (UPDATE Customer
      SET
        customerName = CustomerName,
        customerAddress = CustomerAddress,
        phoneNumber = PhoneNumber,
        email = Email
      WHERE
          customerID = CustomerID)
    }
```

```
        public void Delete(Guid CustomerID)
        {
          (DELETE FROM Customer WHERE customerID LIKE(CustomerID))
        }
    }
}
```

Employee Repository

```
namespace Infrastructure.Repositories.EmployeeRepository
{
    internal class EmployeeRepository
    {
        public void Add(Guid EmployeeID, string EmployeeName, Address EmployeeAddress,
DateTime EmployeeDOB, string EmployeeGender, PhoneNumber EmployeePhoneNumber,
Email EmployeeEmail)
        {
          (INSERT INTO VALUES (EmployeeID, EmployeeName, EmployeeAddress,
EmployeeDOB, EmployeeGender, EmployeePhoneNumber, EmployeeEmail));
        }

        public void GetAll()
        {
          (SELECT * FROM Employee);
        }

        public void Get(Guid EmployeeID)
        {
          (SELECT * FROM Employee WHERE EmployeeID LIKE(EmployeeID));
        }

        public void Update(Guid employeeid,string EmployeeName, Address EmployeeAddress,
DateTime EmployeeDOB, string EmployeeGender, PhoneNumber EmployeePhoneNumber,
Email EmployeeEmail)
        {
          (UPDATE employee
          SET
            employeeName = EmployeeName,
            employeeAddress = EmployeeAddress,
            employeeDOB = EmployeeDOB,
            employeeGender = EmployeeGender,
            employeePhoneNumber = EmployeePhoneNumber,
            employeeEmail = EmployeeEmail
```

```
        WHERE employeeid = employeeid
       )
      }

      public void Delete(Guid EmployeeID)
      {
       (DELETE FROM  Employee WHERE employeeID LIKE(EmployeeID))
      }
   }
}
```

Product Repository

```
namespace Infrastructure.Repositories.ProductRepository
{
   internal class ProductRepository
   {
      public void Add(Guid ProductID, Supplier supplier, Storage storage, string ProductName,
string ProductCategory, int ProductPrice, DateTime productExpire, string productDescription)
      {
       (INSERT INTO VALUES (ProductID, supplier, storage, ProductName, ProductCategory,
ProductPrice, productExpire, productDescription));
      }

      public void GetAll()
      {
       (SELECT * FROM Product);
      }

      public void Get(Guid ProductID)
      {
       (SELECT * FROM Product WHERE productID LIKE(ProductID));
      }

      public void Update(Guid ProductID, Supplier supplier, Storage storage, string
ProductName, string ProductCategory, int ProductPrice, DateTime productExpire, string
productDescription)
      {
       UPDATE Product
       SET
         supplier = Supplier,
         storage = storage,
         productName= ProductName,
```

```
            productPrice = ProductPrice,
            productExpire = productExpire,
            productDescription = productDescription
          WHERE
            productID LIKE(ProductID)
        }

        public void Delete(Guid ProductID)
        {
          (DELETE FROM Product WHERE productID LIKE(ProductID))
        }
    }
}
```

Shipping Repository

```
namespace Infrastructure.Repositories.ShippingRepository
{
    internal class ShippingRepository
    {
        public void Add(Guid ShippingID, string ShipperName, Address Destination, string
ShippingType)
        {
          (INSERT INTO VALUES (ShippingID, ShipperName, Destination, ShippingType));
        }

        public void GetAll()
        {
          (SELECT * FROM Shipping);
        }

        public void Get(Guid ShippingID)
        {
          (SELECT * FROM Shipping WHERE Shippingid LIKE(ShippingID));
        }

        public void Update(Guid ShippingID, string ShipperName, Address Destination, string
ShippingType)
        {
          (UPDATE Shipping
          SET
            shipperName = ShipperName,
            destination = Destination,
```

```
        shippingType = ShippingType
       WHERE
        shippingID LIKE(ShippingID))
     }

     public void Delete(Guid ShippingID)
     {
      (DELETE FROM Shipping WHERE shippingID LIKE(ShippingID))
     }
   }
}
```

Supplier Repository

```
namespace Infrastructure.Repositories.SupplierRepository
{
   internal class SupplierRepository
   {
      public void Add(Guid SupplierID, string SupplierName, Address address)
      {
       (INSERT INTO VALUES (SupplierID, SupplierName, address));
      }

      public void GetAll()
      {
       (SELECT * FROM Supplier);
      }

      public void Get(Guid SupplierID)
      {
       (SELECT * FROM Supplier WHERE supplierID LIKE(SupplierID));
      }

      public void Update(Guid SupplierID, string SupplierName, Address address)
      {
       (UPDATE Supplier
       SET
        supplierName = SupplierName;
        supplierAddress = address;
       WHERE
        supplierID = SupplierID)
      }

      public void Delete(Guid SupplierID)
```

```
        {
          (DELETE FROM Supplier WHERE supplierID LIKE(SupplierID))
        }
    }
}
```

Transaction Repository

```
namespace Infrastructure.Repositories.TransactionRepository
{
    internal class TransactionRepository
    {
        public void Add(Guid TransactionID, Customer customer, Product product, Employee
employee, Shipping shipping, DateTime TransactionDate, Payment payment)
        {
          (INSERT INTO VALUES (TransactionID, customer, product, employee, shipping,
TransactionDate, payment));
        }
        public void GetAll()
        {
          (SELECT * FROM Transaction);
        }
        public void Get(Guid TransactionID)
        {
          (SELECT * FROM Transaction WHERE transactionID LIKE(TransactionID));
        }
        public void Update(Guid TransactionID, Customer customer, Product product, Employee
employee, Shipping shipping, DateTime TransactionDate, Payment payment)
        {
          (UPDATE Transaction
          SET
            customer = customer;
            product = product;
            product = employee;
            shipping = shipping;
            transactionDate = TransactionDate;
            payment = payment;
          WHERE
              transactionID = TransactionID)
        }
        public void Delete(Guid TransactionID)
        {
          (DELETE FROM Transaction WHERE transactionID like(TransactionID))
```

```
        }
    }
}
```

Event Sourcing

```
namespace Infrastructure.EventSourcing
{
    internal class EventSourcing
    {
        private DateTime _recorded, _occured;

        internal EventSourcing(DateTime occured)
        {
            this._occured = occured;
            this._recorded = DateTime.Now;
        }
    }
}
```