Michał Nowaczyk 263971

Michał Bernacki-Janson 264021

Inżynieria oprogramowania – etapy 5-7 Program wystawiający rachunki

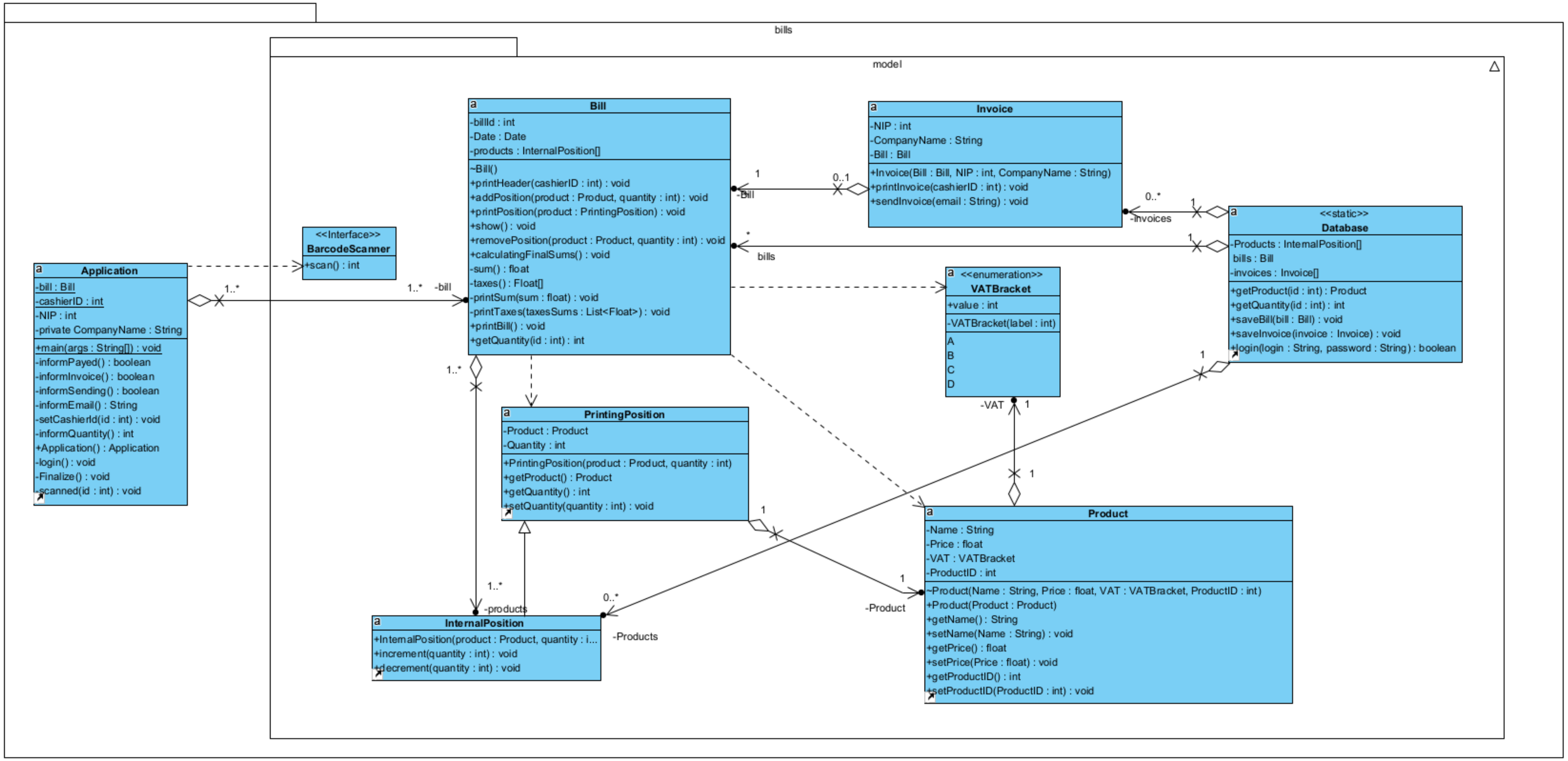
Spis treści

Diagram klas 2
- Struktura klas 3
  - Application 3
  - BarcodeScanner 3
  - VATBracket 3
  - PrintingPosition 3
  - Product 3
  - InternalPosition 4
  - Database 4
  - Invoice 4
  - Bill 5

Diagramy sekwencji 6
- Diagram PU adding/removing a product 6
  - Subdiagram getProduct() 7
  - Subdiagram addPosition() 8
  - Subdiagram removePosition() 9
  - Subdiagram printPosition() 10
- Diagram PU Finalzaing the bill 11

Uzyskany kod 13
- BarcodeScanner 13
- VATBracket 13
- Invoice 14
- PrintingPosition 14
- Database 14
- Bill 14
- InternalPosition 18
- Application 18
- Product 19

Diagram klas


bills
model
Bill
-billId : int
-Date : Date
-products : InternalPosition[]
~Bill()
+printHeader(cashierID : int) : void
+addPosition(product : Product, quantity : int) : void
+printPosition(product : PrintingPosition) : void
+show() : void
+removePosition(product : Product, quantity : int) : void
+calculatingFinalSums() : void
-sum() : float
-taxes() : Float[]
-printSum(sum : float) : void
-printTaxes(taxesSums : List<Float>) : void
+printBill() : void
+getQuantity(id : int) : int
Invoice
-NIP : int
-CompanyName : String
-Bill : Bill
+Invoice(Bill : Bill, NIP : int, CompanyName : String)
+printInvoice(cashierID : int) : void
+sendInvoice(email : String) : void
<<static>>
Database
-Products : InternalPosition[]
bills : Bill
-invoices : Invoice[]
+getProduct(id : int) : Product
+getQuantity(id : int) : int
+saveBill(bill : Bill) : void
+saveInvoice(invoice : Invoice) : void
+login(login : String, password : String) : boolean
<<Interface>>
BarcodeScanner
+scan() : int
Application
-bill : Bill
-cashierID : int
-NIP : int
-private CompanyName : String
+main(args : String[]) : void
-informPayed() : boolean
-informInvoice() : boolean
-informSending() : boolean
-informEmail() : String
-setCashierId(id : int) : void
-informQuantity() : int
+Application() : Application
-login() : void
-Finalize() : void
-scanned(id : int) : void
<<enumeration>>
VATBracket
+value : int
-VATBracket(label : int)
A
B
C
D
PrintingPosition
-Product : Product
-Quantity : int
+PrintingPosition(product : Product, quantity : int)
+getProduct() : Product
+getQuantity() : int
+setQuantity(quantity : int) : void
InternalPosition
+InternalPosition(product : Product, quantity : i...
+increment(quantity : int) : void
+decrement(quantity : int) : void
Product
-Name : String
-Price : float
-VAT : VATBracket
-ProductID : int
~Product(Name : String, Price : float, VAT : VATBracket, ProductID : int)
+Product(Product : Product)
+getName() : String
+setName(Name : String) : void
+getPrice() : float
+setPrice(Price : float) : void
+getProductID() : int
+setProductID(ProductID : int) : void
1
0..1
-Bill
0..*
1
-invoices
*
1
bills
1..*
1..*
-bill
1..*
1..*
-products
1
-VAT
1
1
1
-Product
0..*
-Products
1

Struktura klas

Application

```
public class Application {
       static Bill;
       private static int cashierID;
       private int NIP;
       private String CompanyName;

       Application(){}
        private int informQuantity(){return 0;}
        private void setCashierId(int id){}
        private boolean informPayed(){return false;}
        private boolean informInvoice(){return false;}
        private boolean informSending(){return false;}
        private String informEmail(){return null;}
         private void login(){}
        private void Finalize(){}
       private void scanned(int id){}
        public static void main(String[] args){}
}
```

BarcodeScanner

```
public interface BarcodeScanner {
    public int scan();
}
```

VATBracket

```
public enum VATBracket {
   A(23),
   B(8),
   C(5),
   D(0);

   public final int value;
   private VATBracket(int label) {this.value = label;}
}
```

PrintingPosition

```
public class PrintingPosition {

   private Product Product;
   private int Quantity;

   public PrintingPosition(Product product, int quantity) {}
   public Product getProduct(){return Product;}
   public int getQuantity(){return Quantity;}
   public void setQuantity(int quantity){}

}
```

Product

```

public class Product {
   private String Name;
   private float Price;
   private VATBracket VAT;
   private int ProductID;

   Product(String Name, float Price, VATBracket VAT, int ProductID){}

   public Product(Product Product) {}

   public String getName() {
      return Name;
   }

   public float getPrice() {
      return Price;
   }

   public VATBracket getVAT() {
      return VAT;
   }

   public int getProductID() {
      return ProductID;
   }

}
```

InternalPosition

```
public class InternalPosition extends PrintingPosition {

   public InternalPosition(Product product, int quantity) {
      super(product, quantity);
   }

   public void increment(int quantity) {}
   public void decrement(int quantity) {}
}
```

Database

```
public class Database {
    private static ArrayList<InternalPosition> Products;
    private ArrayList<Bill> bills;
    private ArrayList<Invoice> invoices;
    public static Product getProduct(int id){null;}
    public static int getQuantity(int id){return 0;}

    public void saveBill(Bill bill){}
    public void saveInvoice(Invoice invoice){}
    public boolean login(String login, String password){ return false;}

}
```

Invoice

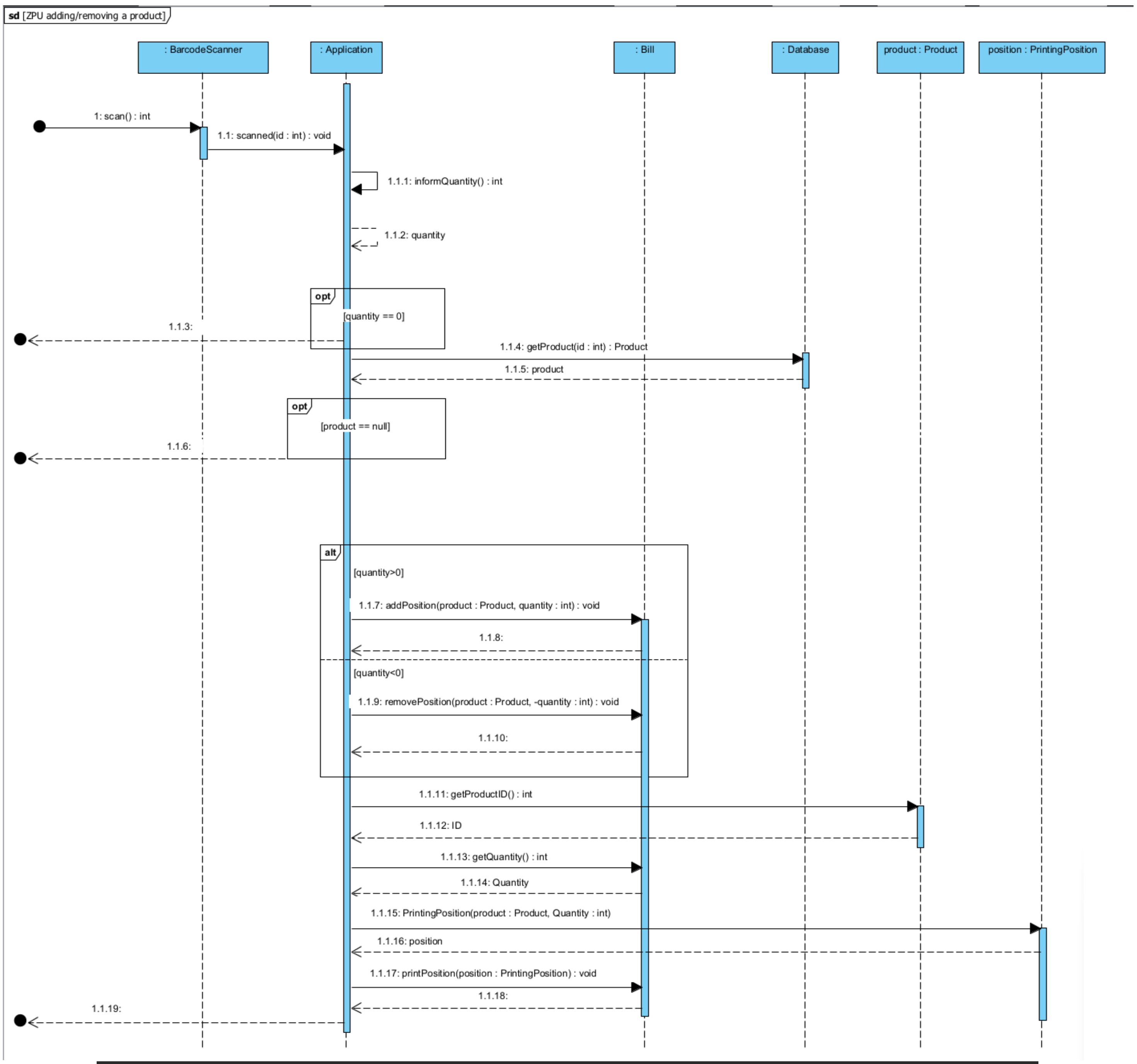
```
public class Invoice{
    Bill Bill;
    int NIP;
    String CompanyName;

    public Invoice(Bill Bill,int NIP,String CompanyName){
        this.Bill=Bill;
        this.NIP = NIP;
        this.CompanyName = CompanyName;
    }

    public void printInvoice(int cashierID){}
    public void sendInvoice(String email){}

}
```

Bill

```
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
public class Bill {

   private ArrayList<InternalPosition> products;
   private int billId;
   private Date Date;

   Bill(){}

   public void printHeader(int cashierID){}
   public void addPosition(Product product, int quantity) {}

   public void printPosition(PrintingPosition product) {}

   public void show() {}

   public void removePosition(Product product, int quantity) {}

   public void calculatingFinalSums(){}

   private float sum(){return 0.0f}
   private ArrayList<Float> taxes() { return null;}
   private void printSum(float sum){}
   private void printTaxes(ArrayList<Float> taxesSums){}
   public void printBill(){}
   public int getQuantity(int id){ return 0;}


}
```

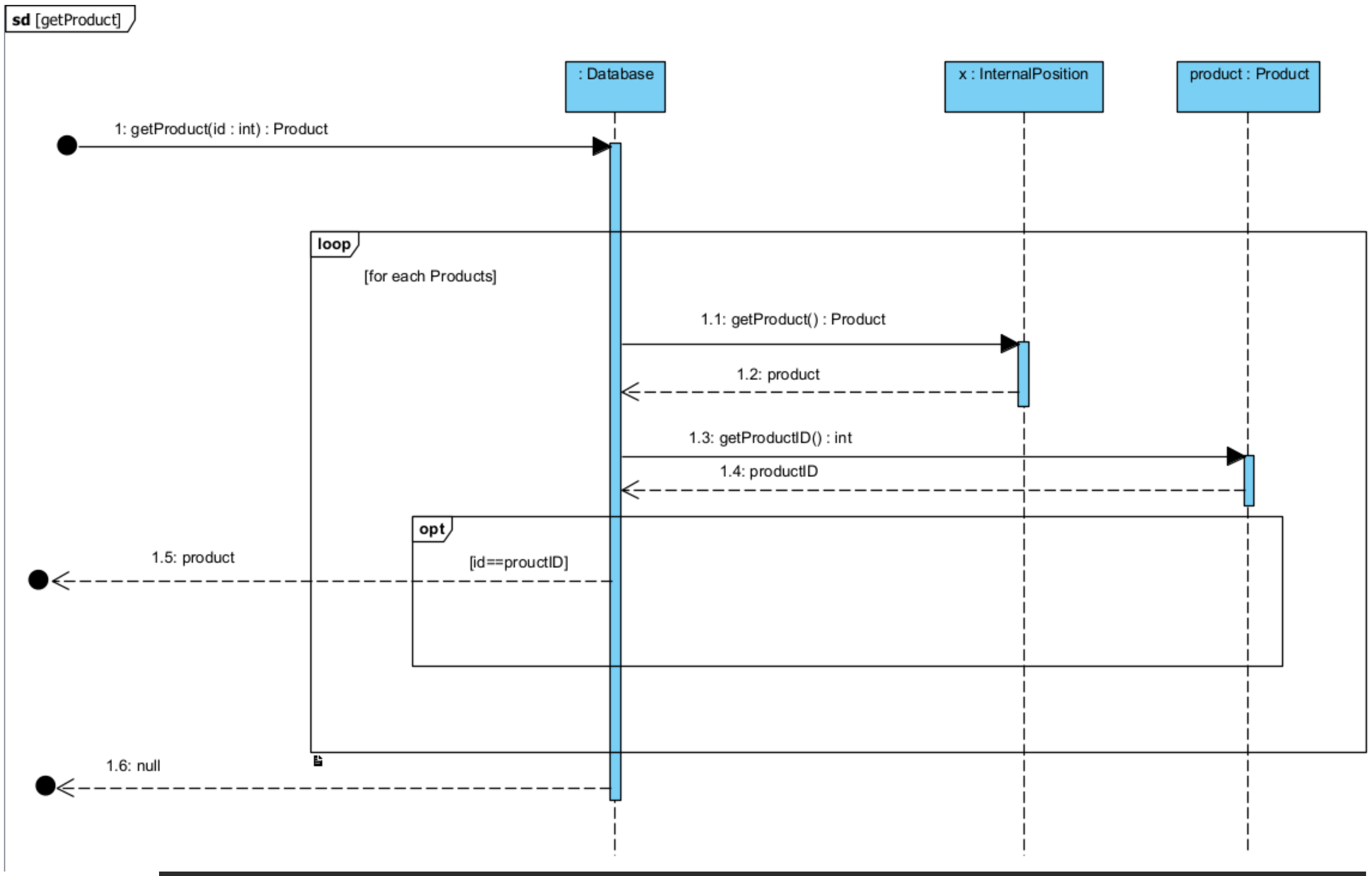
Diagramy sekwencji

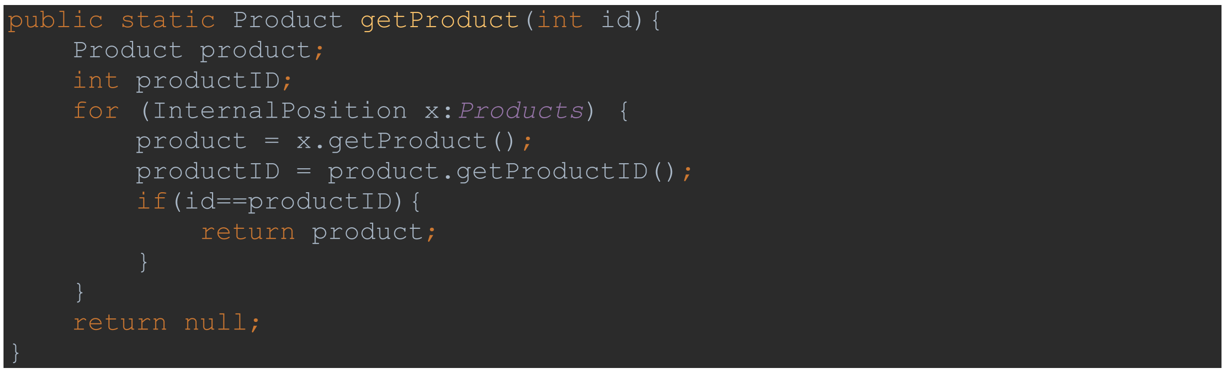
Diagram PU adding/removing a product


sd [ZPU adding/removing a product]
: BarcodeScanner
: Application
: Bill
: Database
product : Product
position : PrintingPosition
1: scan() : int
1.1: scanned(id : int) : void
1.1.1: informQuantity() : int
1.1.2: quantity
opt
[quantity == 0]
1.1.3:
1.1.4: getProduct(id : int) : Product
1.1.5: product
opt
[product == null]
1.1.6:
alt
[quantity>0]
1.1.7: addPosition(product : Product, quantity : int) : void
1.1.8:
[quantity<0]
1.1.9: removePosition(product : Product, -quantity : int) : void
1.1.10:
1.1.11: getProductID() : int
1.1.12: ID
1.1.13: getQuantity() : int
1.1.14: Quantity
1.1.15: PrintingPosition(product : Product, Quantity : int)
1.1.16: position
1.1.17: printPosition(position : PrintingPosition) : void
1.1.18:
1.1.19:

```
private void scanned(int id){
        int quantity;
        quantity = informQuantity();
        if(quantity==0) return;
        Product product=Database.getProduct(id);
        if(product==null) return;
        if(quantity>0)bill.addPosition(product,quantity);
        else if(quantity<0)bill.removePosition(product,quantity);
        int ID = product.getProductID();
        int Quantity = bill.getQuantity(ID);
        PrintingPosition position=new PrintingPosition(product,Quantity );
        bill.printPosition(position);
}
```

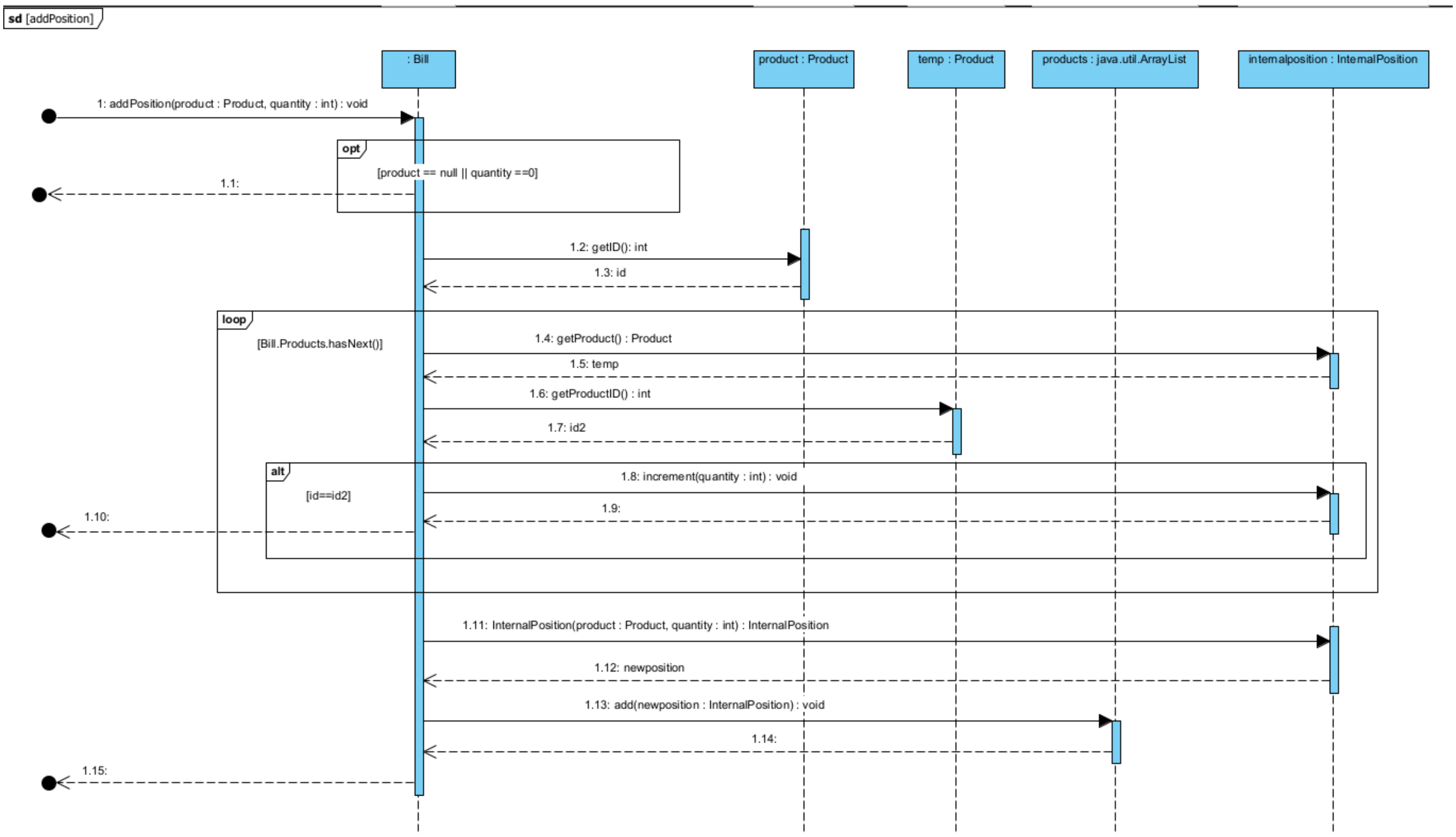
Subdiagram getProduct()





```
public static Product getProduct(int id){
    Product product;
    int productID;
    for (InternalPosition x:Products) {
        product = x.getProduct();
        productID = product.getProductID();
        if(id==productID){
            return product;
        }
    }
    return null;
}
```

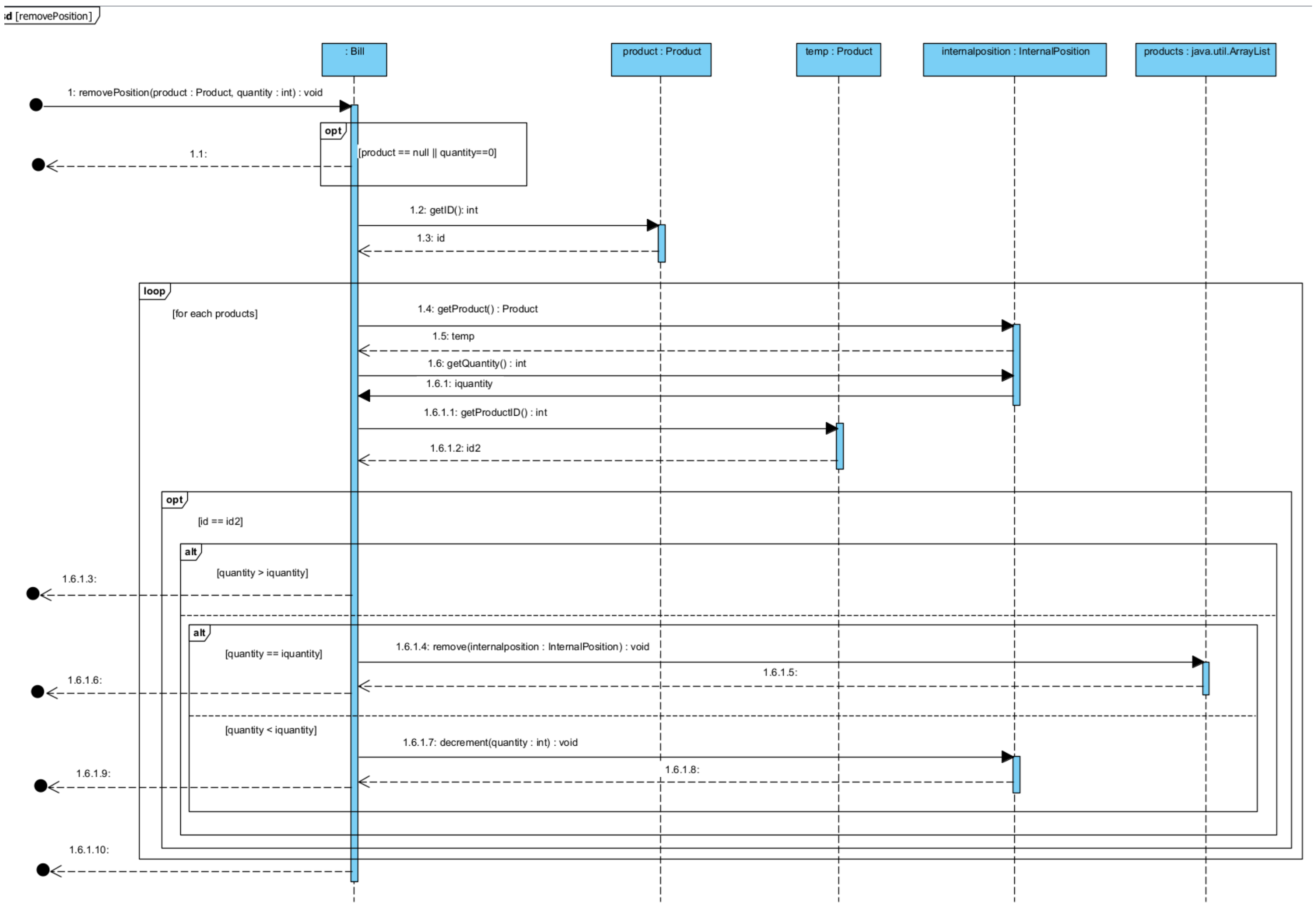
Subdiagram addPosition()

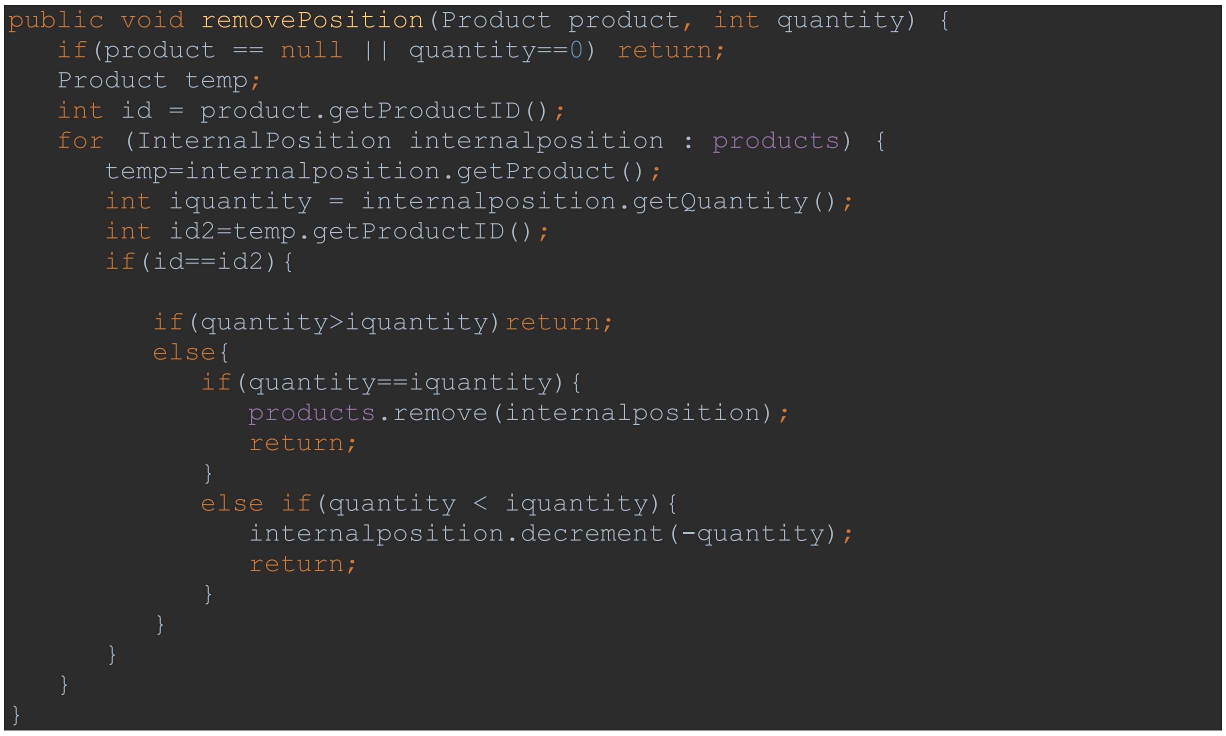

sd [addPosition]
: Bill
product : Product
temp : Product
products : java.util.ArrayList
internalposition : InternalPosition
1: addPosition(product : Product, quantity : int) : void
opt
[product == null || quantity ==0]
1.1:
1.2: getID(): int
1.3: id
loop
[Bill.Products.hasNext()]
1.4: getProduct() : Product
1.5: temp
1.6: getProductID() : int
1.7: id2
alt
[id==id2]
1.8: increment(quantity : int) : void
1.9:
1.10:
1.11: InternalPosition(product : Product, quantity : int) : InternalPosition
1.12: newposition
1.13: add(newposition : InternalPosition) : void
1.14:
1.15:

```
public void addPosition(Product product, int quantity) {
   if(product == null || quantity==0) return;
   Product temp;
   int id = product.getProductID();
   for (InternalPosition internalposition : products) {
      temp=internalposition.getProduct();
      int id2=temp.getProductID();
      if(id==id2){
         internalposition.increment(quantity);
         return;
      }
   }
   products.add(new InternalPosition(product,quantity));
}
```

```
public void increment(int quantity) {
   this.Quantity+=quantity;
}
```

Subdiagram removePosition()

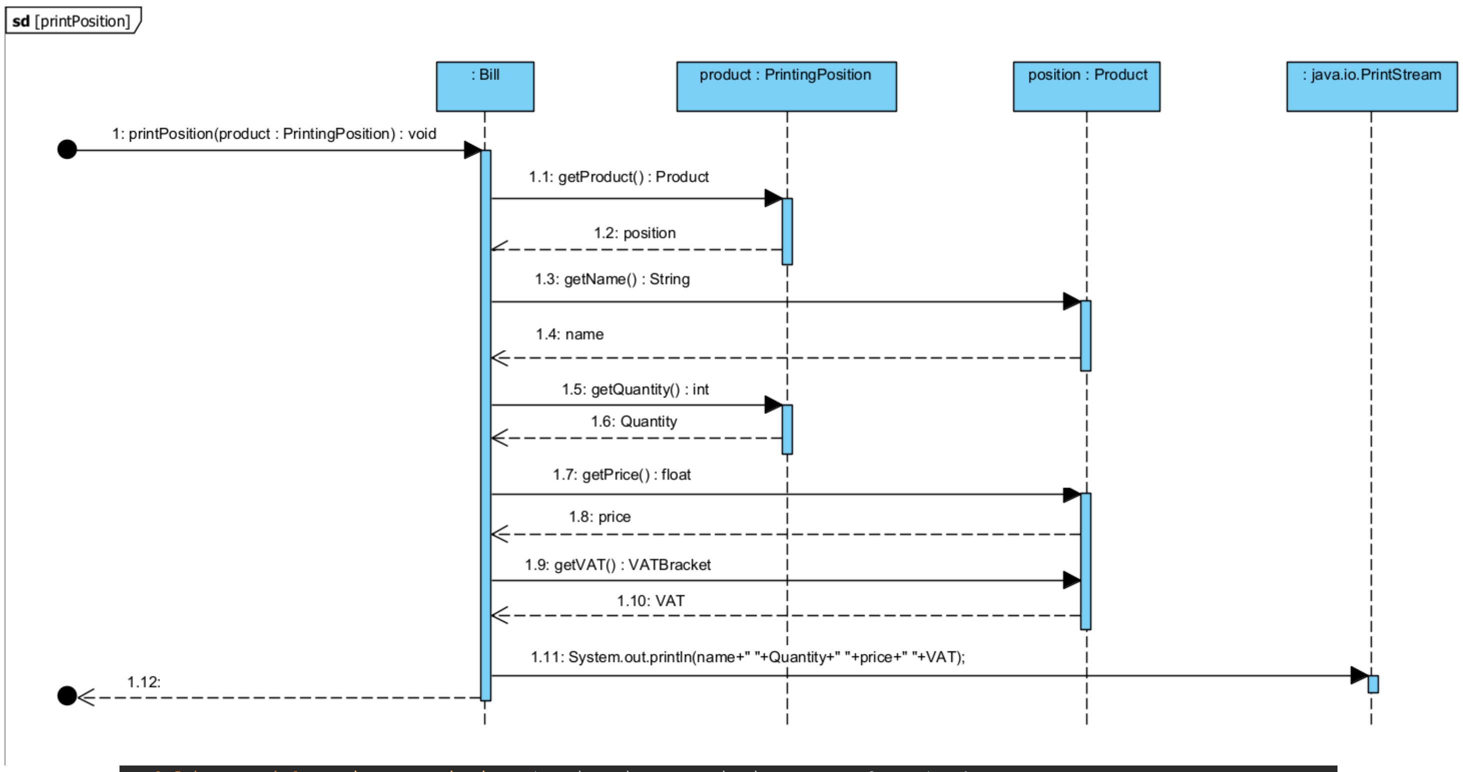

sd [removePosition]
: Bill
product : Product
temp : Product
internalposition : InternalPosition
products : java.util.ArrayList
1: removePosition(product : Product, quantity : int) : void
opt
[product == null || quantity==0]
1.1:
1.2: getID(): int
1.3: id
loop
[for each products]
1.4: getProduct() : Product
1.5: temp
1.6: getQuantity() : int
1.6.1: iquantity
1.6.1.1: getProductID() : int
1.6.1.2: id2
opt
[id == id2]
alt
[quantity > iquantity]
1.6.1.3:
alt
[quantity == iquantity]
1.6.1.4: remove(internalposition : InternalPosition) : void
1.6.1.5:
1.6.1.6:
[quantity < iquantity]
1.6.1.7: decrement(quantity : int) : void
1.6.1.8:
1.6.1.9:
1.6.1.10:



```java
public void removePosition(Product product, int quantity) {
    if(product == null || quantity==0) return;
    Product temp;
    int id = product.getProductID();
    for (InternalPosition internalposition : products) {
        temp=internalposition.getProduct();
        int iquantity = internalposition.getQuantity();
        int id2=temp.getProductID();
        if(id==id2){

            if(quantity>iquantity)return;
            else{
                if(quantity==iquantity){
                    products.remove(internalposition);
                    return;
                }
                else if(quantity < iquantity){
                    internalposition.decrement(-quantity);
                    return;
                }
            }
        }
    }
}
```

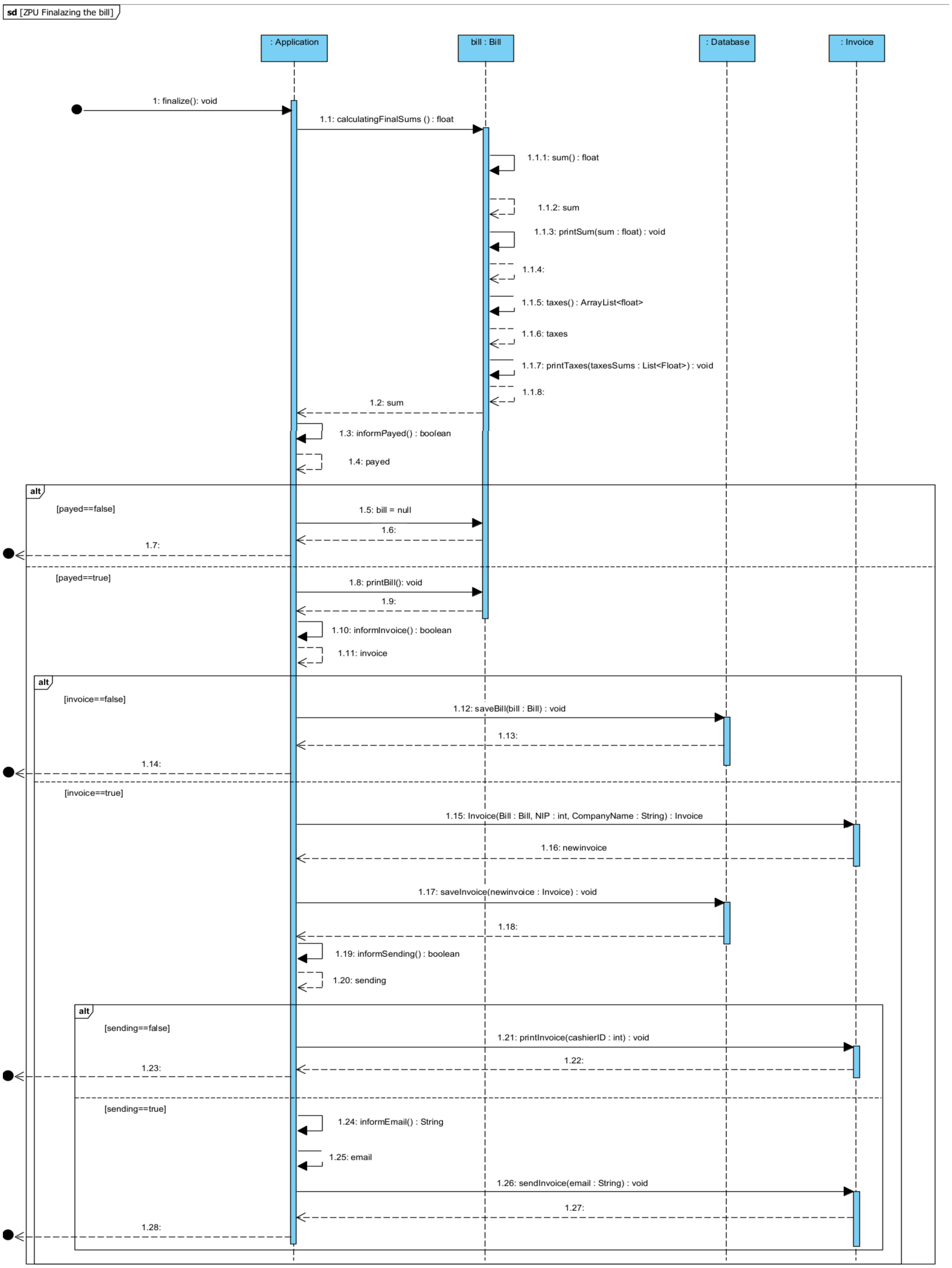
```
public void decrement(int quantity) {
   this.Quantity-=quantity;
}
```

Subdiagram printPosition()


sd [printPosition]
: Bill
product : PrintingPosition
position : Product
: java.io.PrintStream
1: printPosition(product : PrintingPosition) : void
1.1: getProduct() : Product
1.2: position
1.3: getName() : String
1.4: name
1.5: getQuantity() : int
1.6: Quantity
1.7: getPrice() : float
1.8: price
1.9: getVAT() : VATBracket
1.10: VAT
1.11: System.out.println(name+" "+Quantity+" "+price+" "+VAT);
1.12:

```
public void printPosition(PrintingPosition product) {
   Product position = product.getProduct();
   int Quantity = product.getQuantity();
   String name = position.getName();
   float price = position.getPrice();
   VATBracket VAT = position.getVAT();
   System.out.println(name+"  "+Quantity+"   "+price+"  "+VAT);
}
```

Diagram PU Finalzaing the bill


sd [ZPU Finalazing the bill]
: Application
bill : Bill
: Database
: Invoice
1: finalize(): void
1.1: calculatingFinalSums () : float
1.1.1: sum() : float
1.1.2: sum
1.1.3: printSum(sum : float) : void
1.1.4:
1.1.5: taxes() : ArrayList<float>
1.1.6: taxes
1.1.7: printTaxes(taxesSums : List<Float>) : void
1.1.8:
1.2: sum
1.3: informPayed() : boolean
1.4: payed
alt
[payed==false]
1.5: bill = null
1.6:
1.7:
[payed==true]
1.8: printBill(): void
1.9:
1.10: informInvoice() : boolean
1.11: invoice
alt
[invoice==false]
1.12: saveBill(bill : Bill) : void
1.13:
1.14:
[invoice==true]
1.15: Invoice(Bill : Bill, NIP : int, CompanyName : String) : Invoice
1.16: newinvoice
1.17: saveInvoice(newinvoice : Invoice) : void
1.18:
1.19: informSending() : boolean
1.20: sending
alt
[sending==false]
1.21: printInvoice(cashierID : int) : void
1.22:
1.23:
[sending==true]
1.24: informEmail() : String
1.25: email
1.26: sendInvoice(email : String) : void
1.27:
1.28:

```
private void Finalize(){
       bill.calculatingFinalSums();
       boolean payed = informPayed();
       if(payed == false){
           bill=null;
           return;
       }
       else{
           bill.printBill();
           boolean invoice = informInvoice();
           if(invoice == false){
               Database.saveBill(bill);
               return;
           }
           else{
               Invoice newinvoice = new Invoice(bill,NIP,CompanyName);
               Database.saveInvoice(newinvoice);
               boolean sending;
               sending=informSending();
               if(sending==false){
                    newinvoice.printInvoice(cashierID);
               }
               else{
                   String email;
                   email=informEmail();
                   newinvoice.sendInvoice(email);
               }
           }
       }
}
```

```
public void calculatingFinalSums(){
   printSum(sum());;
   printTaxes(taxes());

}
private float sum(){
   float sum = 0;
   for (var product : products){
      sum += product.Product.getPrice() * product.Quantity;
   }
   return sum;
}


private void printSum(float sum){
   System.out.println("Suma: "+sum);
}
private void printTaxes(ArrayList<Float> taxesSums){

   for(int i=0;i<4;i++)
   System.out.println(VATBracket.getBracketForValue(i)+"
"+taxesSums.get(i));


}
public void printBill(){
   System.out.println("Dziekujemy za zakupy");
}
```

```java
public void saveBill(Bill bill){
    bills.add(bill);
}
public void saveInvoice(Invoice invoice){
    invoices.add(invoice);
}
```

```java
public void printInvoice(int cashierID){
    System.out.print("Data : ");
    Bill.printHeader(cashierID);
    System.out.println("NAZWA FIRMY : " + CompanyName);
    System.out.println("NIP : " + NIP);
    Bill.show();
    Bill.calculatingFinalSums();
}
public void sendInvoice(String email){
    System.out.println("Wysylanie faktury na adres email: " + email);
}
```

```java
private ArrayList<Float> taxes() {
   ArrayList<Float> taxesSums = new ArrayList<Float>();
   taxesSums.add(0.0f);
   taxesSums.add(0.0f);
   taxesSums.add(0.0f);
   taxesSums.add(0.0f);

   for (var product : products) {
      int index = switch (product.Product.getVAT()) {
         case A -> 0;
         case B -> 1;
         case C -> 2;
         case D -> 3;
      };

      taxesSums.set(index, taxesSums.get(index) +
product.Product.getPrice() * product.Quantity *
VATBracket.valueOf(product.Product.getVAT().name()).value / 100);
   }

   return taxesSums;
}
```

Uzyskany kod

BarcodeScanner

```java
public interface BarcodeScanner {
    public int scan();
}
```

VATBracket

```java
public enum VATBracket {
   A(23),
   B(8),
   C(5),
   D(0);

   public final int value;
   private VATBracket(int label) {this.value = label;}
}
```

Invoice

```
public class Invoice{
    Bill Bill;
    int NIP;
    String CompanyName;

    public Invoice(Bill Bill,int NIP,String CompanyName){
        this.Bill=Bill;
        this.NIP = NIP;
        this.CompanyName = CompanyName;

    }

    public void printInvoice(int cashierID){
        Bill.printHeader(cashierID);
        System.out.println("NAZWA FIRMY : " + CompanyName);
        System.out.println("NIP : " + NIP);
        Bill.show();
        Bill.calculatingFinalSums();
    }
    public void sendInvoice(String email){
        System.out.println("Wysylanie faktury na adres email: " + email);
    }

}
```

PrintingPosition

```
public class PrintingPosition {

   private Product Product;
   private int Quantity;

   public PrintingPosition(Product product, int quantity) {

      Product=product;
      Quantity=quantity;
   }
   public Product getProduct(){
      return Product;
   }
   public int getQuantity(){
      return Quantity;
   }
   public void setQuantity(int quantity){
      Quantity=quantity;
   }
}
```

Database

```
public class Database {
    private static ArrayList<InternalPosition> Products;
    static{
        Products=new ArrayList<InternalPosition>();
        Products.add(new InternalPosition(new Product("Chleb 
pszenny",3.49f, VATBracket.B,1),36));
        Products.add(new InternalPosition(new Product("Mleko muuu",3.99f, 
VATBracket.B,2),314));
        Products.add(new InternalPosition(new Product("Telewizor 
32''",1299.99f, VATBracket.A,3),4));
        Products.add(new InternalPosition(new Product("Radioodbiornik 
Rydzunio",333.33f, VATBracket.A,4),12));
    }
    private static ArrayList<Bill> bills = new ArrayList<Bill>();
    private static ArrayList<Invoice> invoices= new ArrayList<Invoice>();

    public static Product getProduct(int id){
        Product product;
        int productID;
        for (InternalPosition x:Products) {
            product = x.getProduct();
            productID = product.getProductID();
            if(id==productID){
                return product;
            }
        }
        return null;
    }
    public static void saveBill(Bill bill){
        bills.add(bill);
    }
    public static void saveInvoice(Invoice invoice){
        invoices.add(invoice);
    }
}
```

Bill

```
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
public class Bill {

   private ArrayList<InternalPosition> products;
   private int billId;
   private Date Date;

   Bill(){
      products=new ArrayList<InternalPosition>();
      billId  = (int) new Date().getTime();
      Date = new Date();
   }

   public void printHeader(int cashierID){
      SimpleDateFormat f = new SimpleDateFormat("yyyy-MM-dd hh:mm");
   System.out.println("Sklep Fajny");
   System.out.println("NIP 328957834275");
   System.out.println("Kasjer : "+cashierID);
   System.out.println(f.format(Date));

   }
   public void addPosition(Product product, int quantity) {
      if(product == null || quantity==0) return;
      Product temp;
      int id = product.getProductID();
      for (InternalPosition internalposition : products) {
         temp=internalposition.getProduct();
         int id2=temp.getProductID();
         if(id==id2){
            internalposition.increment(quantity);
            return;
         }
      }
      products.add(new InternalPosition(product,quantity));
   }

   public void printPosition(PrintingPosition product) {
      Product position = product.getProduct();
      int Quantity = product.getQuantity();
      String name = position.getName();

      float price = position.getPrice();
      VATBracket VAT = position.getVAT();
      System.out.println(name+"  "+Quantity+"   "+price+"  "+VAT);
   }

   public void show() {
      for (InternalPosition product:products) {
         Product Product = product.getProduct();
         System.out.println(Product.getName()+" "+product.getQuantity()+"
"+Product.getPrice()+" "+Product.getVAT());
      }
   }

   public void removePosition(Product product, int quantity) {
      if(product == null || quantity==0) return;
      Product temp;
      int id = product.getProductID();
      for (InternalPosition internalposition : products) {
         temp=internalposition.getProduct();
         int iquantity = internalposition.getQuantity();
         int id2=temp.getProductID();
         if(id==id2){

            if(quantity>iquantity)return;
            else{
               if(quantity==iquantity){
                  products.remove(internalposition);
                  return;
               }
               else if(quantity < iquantity){
                  internalposition.decrement(-quantity);
                  return;
               }
            }
         }
      }
   }

   public void calculatingFinalSums(){
      printSum(sum());;
      printTaxes(taxes());

   }

   private float sum(){
      float sum = 0;
      for (var product : products){
         sum += product.getProduct().getPrice() * product.getQuantity();
      }
      return sum;
   }
   private ArrayList<Float> taxes() {
      ArrayList<Float> taxesSums = new ArrayList<Float>();
      taxesSums.add(0.0f);
      taxesSums.add(0.0f);
      taxesSums.add(0.0f);
      taxesSums.add(0.0f);

      for (var product : products) {
         int index = switch (product.getProduct().getVAT()) {
            case A -> 0;
            case B -> 1;
            case C -> 2;
            case D -> 3;
         };

         taxesSums.set(index, taxesSums.get(index) +
product.getProduct().getPrice() * product.getQuantity() *
VATBracket.valueOf(product.getProduct().getVAT().name()).value / 100);
      }

      return taxesSums;
   }

   private void printSum(float sum){
      System.out.println("Suma: "+sum);
   }
   private void printTaxes(ArrayList<Float> taxesSums){
      char v = 'A';
      for(int i=0;i<4;i++){
         System.out.println(v+" "+taxesSums.get(i));
         v++;
      }

   }

   public void printBill(){
      System.out.println("Dziekujemy za zakupy");
   }

   public int getQuantity(int id){
         Product product;
         int productID;
         for (InternalPosition x:products) {
            product = x.getProduct();
            productID = product.getProductID();
            if(id==productID){
               return x.getQuantity();
            }
         }

      return 0;
   }
}
```

InternalPosition

```
public class InternalPosition extends PrintingPosition {

   public InternalPosition(Product product, int quantity) {
      super(product, quantity);
   }

   public void increment(int quantity) {
      this.setQuantity(this.getQuantity()+quantity);
   }
   public void decrement(int quantity) {
      this.setQuantity(this.getQuantity()-quantity);
   }
}
```

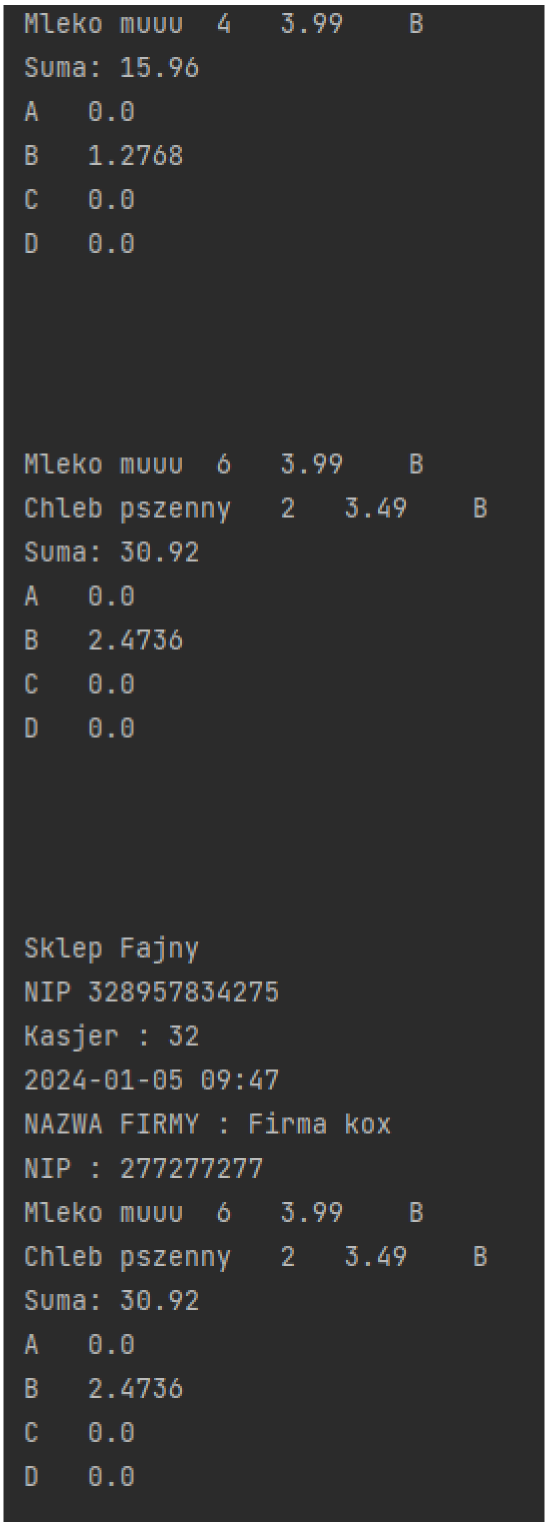
Application

```
public class Application {
      private static Bill bill;
       private static int cashierID;
       private int NIP;
       private String CompanyName;
       Application(){ //ustawione dane do przetestowania
         bill = new Bill();
           setCashierId(32);
           bill.addPosition(Database.getProduct(2),2);
           bill.addPosition(Database.getProduct(2),2);
           bill.show();
           bill.calculatingFinalSums();
           System.out.println("\n\n\n");
           bill.addPosition(Database.getProduct(1),2);
           bill.removePosition(Database.getProduct(2),2);
           bill.show();
           bill.calculatingFinalSums();
           System.out.println("\n\n\n");
           Invoice x = new Invoice(bill,277277277,"Firma kox");
           x.printInvoice(cashierID);
       }
       private void setCashierId(int id){
            cashierID = id;
       }
    private int informQuantity(){return 0;}
    private boolean informPayed(){return false;}
    private boolean informInvoice(){return false;}
    private boolean informSending(){return false;}
    private String informEmail(){return null;}
    private void Finalize(){
           bill.calculatingFinalSums();
           boolean payed = informPayed();
           if(payed == false){
               bill=null;
               return;
           }
           else{
               bill.printBill();
               boolean invoice = informInvoice();
               if(invoice == false){
                   Database.saveBill(bill);
                   return;
               }
               else{
                   Invoice newinvoice = new Invoice(bill,NIP,CompanyName);
                   Database.saveInvoice(newinvoice);
                   boolean sending;
                   sending=informSending();
                   if(sending==false){
                        newinvoice.printInvoice(cashierID);
                   }
                   else{
                       String email;
                       email=informEmail();
                       newinvoice.sendInvoice(email);
                   }
               }
               scanned(2);
           }
    }
    private void scanned(int id){
           int quantity;
           quantity = informQuantity();
           if(quantity==0)return;
           Product product=Database.getProduct(id);
           if(product==null)return;
           if(quantity>0)bill.addPosition(product,quantity);
           else if(quantity<0)bill.removePosition(product,quantity);
           int ID = product.getProductID();
           int Quantity = bill.getQuantity(ID);
           PrintingPosition position=new PrintingPosition(product,Quantity
);

           bill.printPosition(position);
    }
    public static void main(String[] args){
           new Application();
    }
}
```

Product

```
public class Product {
   private String Name;
   private float Price;
   private VATBracket VAT;
   private int ProductID;

   Product(String Name, float Price, VATBracket VAT, int ProductID){
      this.Name=Name;
      this.Price=Price;
      this.VAT=VAT;
      this.ProductID=ProductID;
   }

   public String getName() {
      return Name;
   }

   public float getPrice() {
      return Price;
   }

   public VATBracket getVAT() {
        return VAT;
    }

    public int getProductID() {
        return ProductID;
    }
}
```

Wynik działania testowego



```
Mleko muuu  4   3.99    B
Suma: 15.96
A   0.0
B   1.2768
C   0.0
D   0.0





Mleko muuu  6   3.99    B
Chleb pszenny   2   3.49    B
Suma: 30.92
A   0.0
B   2.4736
C   0.0
D   0.0





Sklep Fajny
NIP 328957834275
Kasjer : 32
2024-01-05 09:47
NAZWA FIRMY : Firma kox
NIP : 277277277
Mleko muuu  6   3.99    B
Chleb pszenny   2   3.49    B
Suma: 30.92
A   0.0
B   2.4736
C   0.0
D   0.0
```