Michał Nowaczyk 263971

Michał Bernacki-Janson 264021

**Inżynieria oprogramowania – etapy 5-7 Program wystawiający rachunki**

Spis treści

[Diagram klas do poprawy 2](#_Toc153043201)

[Struktura klas 3](#_Toc153043202)

[Application 3](#_Toc153043203)

[BarcodeScanner 3](#_Toc153043204)

[VATBracket 3](#_Toc153043205)

[PrintingPosition 3](#_Toc153043206)

[Product 3](#_Toc153043207)

[InternalPosition 4](#_Toc153043208)

[Database 4](#_Toc153043209)

[Invoice 4](#_Toc153043210)

[Bill 6](#_Toc153043211)

[Diagramy sekwencji 7](#_Toc153043212)

[Diagram PU adding/removing a product 7](#_Toc153043213)

[Subdiagram getProduct() 8](#_Toc153043214)

[Subdiagram addPosition() 9](#_Toc153043215)

[Subdiagram removePosition() 10](#_Toc153043216)

[Subdiagram printPosition() 11](#_Toc153043217)

[Diagram PU Finalzaing the bill 12](#_Toc153043218)

[Subdiagram taxes() 14](#_Toc153043219)

[Uzyskany kod 15](#_Toc153043220)

[BarcodeScanner 15](#_Toc153043221)

[VATBracket 15](#_Toc153043222)

[Invoice 15](#_Toc153043223)

[PrintingPosition 15](#_Toc153043224)

[Database 16](#_Toc153043225)

[Bill 16](#_Toc153043226)

[InternalPosition 19](#_Toc153043227)

[Application 19](#_Toc153043228)

[Product 19](#_Toc153043229)

[Wyniki testów 20](#_Toc153043230)

# Diagram klas do poprawy

Obraz zawierający tekst, zrzut ekranu, diagram, Równolegle

Opis wygenerowany automatycznie

## Struktura klas

### Application

public class Application {  
 static Bill;  
 static int *cashierID*;  
 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 {  
  
 protected Product Product;  
 protected int Quantity;  
  
 public PrintingPosition(Product product, int quantity) {  
 Product=product;  
 Quantity=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){  
 this.Name=Name;  
 this.Price=Price;  
 this.VAT=VAT;  
 this.ProductID=ProductID;  
 }  
  
 public Product(Product Product) {  
 this.ProductID=Product.ProductID;  
 this.Name=Product.Name;  
 this.VAT=Product.VAT;  
 this.Price=Product.Price;  
 }  
  
 public String getName() {  
 return Name;  
 }  
 public float getPrice() {  
 return Price;  
 }  
  
 public VATBracket getVAT() {  
 return VAT;  
 }  
  
 public int getProductID() {  
 return ProductID;  
 }  
  
 public void setProductID(int ProductID) {  
 this.ProductID=ProductID;  
 }  
  
}

### InternalPosition

public class InternalPosition extends PrintingPosition {  
  
 public InternalPosition(Product product, int quantity) {  
 super(product, quantity);  
 }  
  
 public void increment(int quantity) {  
 this.Quantity+=quantity;  
 }  
 public void decrement(int quantity) {  
 this.Quantity-=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

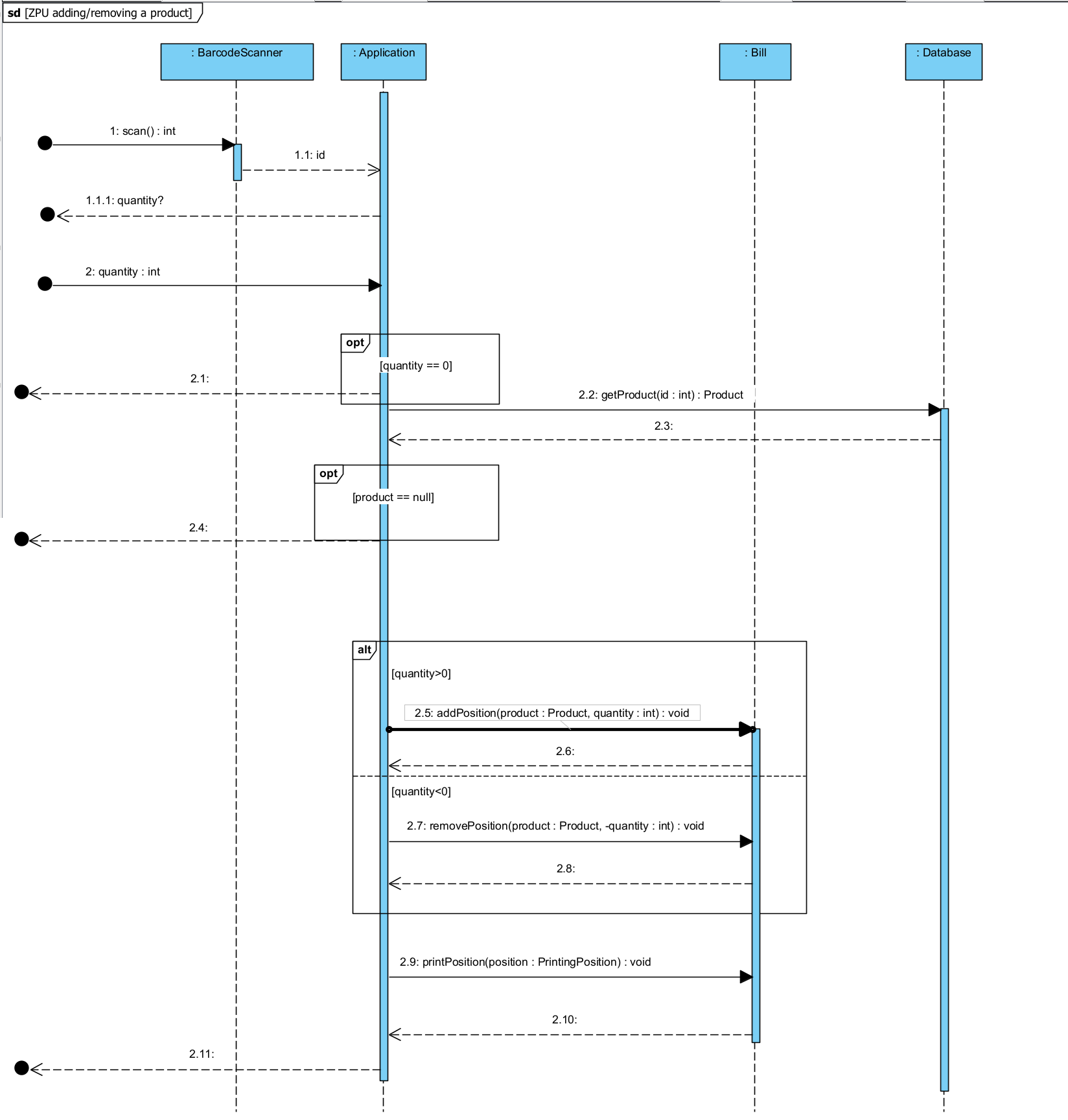
public class Invoice extends Bill {  
 Bill Bill;  
 int NIP;  
 String CompanyName;  
  
 public Invoice(Bill Bill,int NIP,String CompanyName){  
 this.Bill=Bill;  
 this.NIP = NIP;  
 this.CompanyName = CompanyName;  
  
 }  
  
 public void printInvoice(int cashierID){}  
 public void sendInvoice(String 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(){}  
  
}

# Diagramy sekwencji

## Diagram PU adding/removing a product



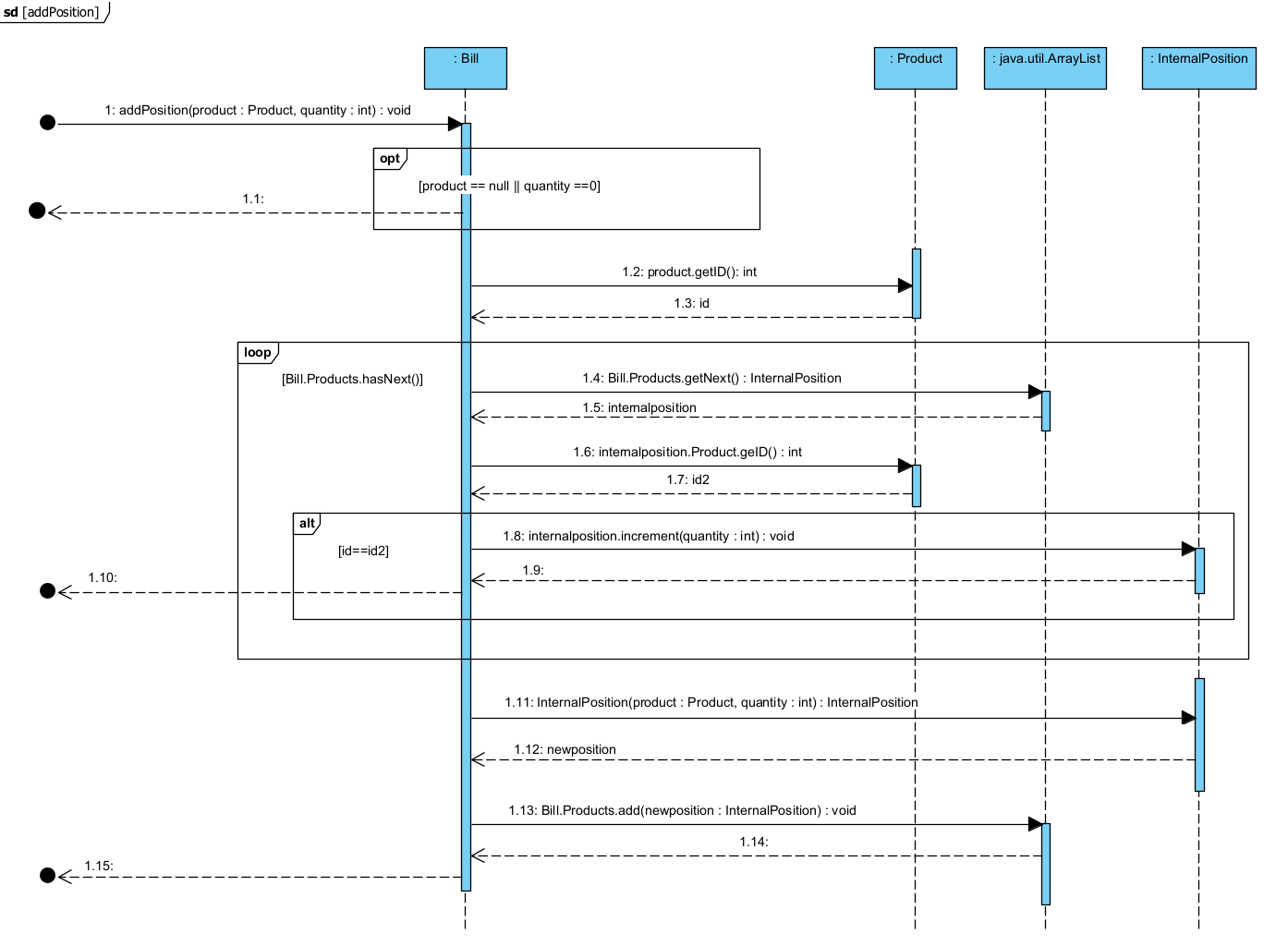
### Subdiagram getProduct()

Obraz zawierający tekst, zrzut ekranu, diagram, linia

Opis wygenerowany automatycznie

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

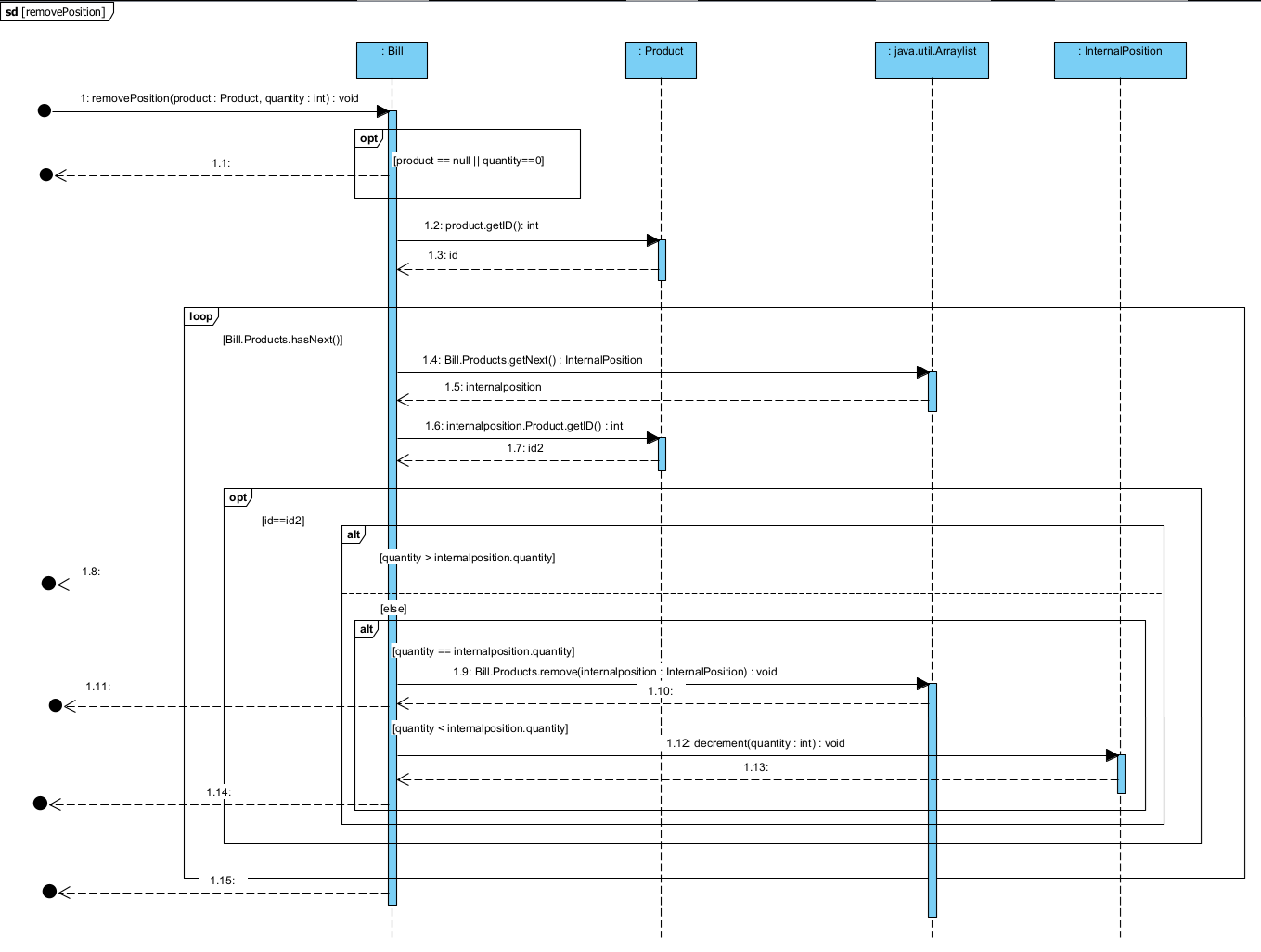
### Subdiagram addPosition()



public void addPosition(Product product, int quantity) {  
  
 for (var produt : products) {  
  
 if(produt.Product.getProductID()==product.getProductID()){  
 produt.increment(quantity);  
 return;  
 }  
 }  
 products.add(new InternalPosition(product,quantity));  
}

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

### Subdiagram removePosition()



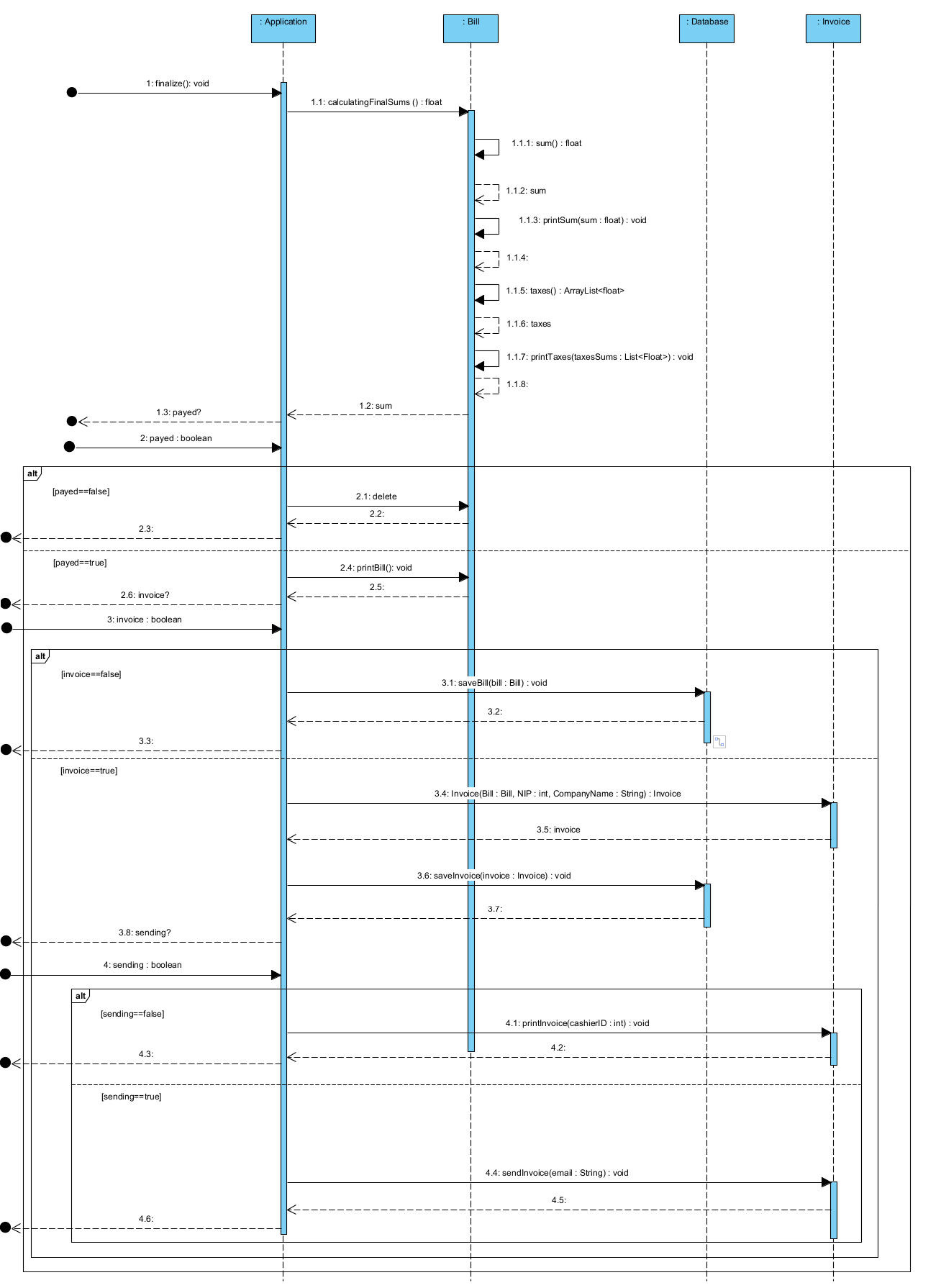
public void removePosition(Product product, int quantity) {  
 for (var produt : products) {  
 if (produt.Product.getProductID() == product.getProductID()) {  
 if(quantity> produt.Quantity) return;  
 else {  
 if (produt.Quantity > quantity)  
 produt.decrement(quantity);  
 else {  
 products.remove(produt);  
 }  
 }  
 return;  
 }  
 }  
}

### Subdiagram printPosition()

Obraz zawierający tekst, zrzut ekranu, diagram, linia

Opis wygenerowany automatyczniepublic void printPosition(PrintingPosition product) {  
System.*out*.println(product.Product.getName()+" "+product.Quantity+" "+product.Product.getPrice()+" "+product.Product.getVAT());  
}

## Diagram PU Finalzaing the bill



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");  
}

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

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);  
}

### Subdiagram taxes()

Obraz zawierający tekst, zrzut ekranu, Równolegle, diagram

Opis wygenerowany automatycznie

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

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;}  
}

## Invoice

import java.util.ArrayList;

public class Invoice extends sklep.Bill {  
 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 {  
  
 protected Product Product;  
 protected int Quantity;  
  
 public PrintingPosition(Product product, int quantity) {  
  
 Product=product;  
 Quantity=quantity;  
 }  
}

## Database

import java.util.ArrayList;  
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 ArrayList<Bill> bills = new ArrayList<Bill>();  
 private ArrayList<Invoice> invoices= new ArrayList<Invoice>();  
 public static Product getProduct(int id){  
 Product product;  
 for (InternalPosition x:*Products*) {  
 if(x.Product.getProductID()==id){  
 product=x.Product;  
 return product;  
 }  
 }  
 return null;  
 }  
 public static int getQuantity(int id){  
  
 int quantity = -1;  
 for (InternalPosition x:*Products*) {  
 if(x.Product.getProductID()==id){  
 quantity=x.Quantity;  
 }  
 }  
 return quantity;  
 }  
  
public void saveBill(Bill bill){  
 bills.add(bill);  
}  
public void saveInvoice(Invoice invoice){  
 invoices.add(invoice);  
}  
  
public boolean login(String login, String password){  
 if(login.compareTo(Login) == 0 && password.compareTo(Passowrd)==0)  
 return true;  
 return false;  
}

}

## 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) {  
  
 for (var produt : products) {  
  
 if(produt.Product.getProductID()==product.getProductID() && produt.Quantity<quantity){  
 produt.increment(quantity);  
 return;  
 }  
 }  
 products.add(new InternalPosition(product,quantity));  
 }  
  
 public void printPosition(PrintingPosition product) {  
 System.*out*.println(product.Product.getName()+" "+product.Quantity+" "+product.Product.getPrice()+" "+product.Product.getVAT());  
 }  
  
 public void show() {  
 for (InternalPosition product:products) {  
 System.*out*.println(product.Product.getName()+" "+product.Quantity+" "+product.Product.getPrice()+" "+product.Product.getVAT());  
 }  
 }  
  
 public void removePosition(Product product, int quantity) {  
 for (var produt : products) {  
 if (produt.Product.getProductID() == product.getProductID()) {  
 if(quantity> produt.Quantity) return;  
 else {  
 if (produt.Quantity > quantity)  
 produt.decrement(quantity);  
 else {  
 products.remove(produt);  
 }  
 }  
 return;  
 }  
 }  
 }  
  
 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 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;  
 }  
  
 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");  
 }  
  
}

## InternalPosition

public class InternalPosition extends PrintingPosition {  
  
 public InternalPosition(Product product, int quantity) {  
 super(product, quantity);  
 }  
  
 public void increment(int quantity) {  
 this.Quantity+=quantity;  
 }  
 public void decrement(int quantity) {  
 this.Quantity-=quantity;  
 }  
}

## Application

public class Application {  
 static Bill *bill*;  
 static int *cashierID*;  
 Application(){  
 *bill* = new Bill();  
 *cashierID*=32;  
  
 }  
  
 public static void main(String[] args){  
 Application app = new Application();  
 *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*);  
 }  
  
}

## 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 this.Name;  
 }  
  
 public float getPrice() {  
 return this.Price;  
 }  
  
 public VATBracket getVAT() {  
 return this.VAT;  
 }  
  
 public int getProductID() {  
 return this.ProductID;  
 }  
}

# Wyniki testów

*bill*.addPosition(Database.*getProduct*(2),2);  
*bill*.addPosition(Database.*getProduct*(2),2);  
*bill*.show();  
*bill*.calculatingFinalSums();

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

*bill*.addPosition(Database.*getProduct*(1),2);  
*bill*.removePosition(Database.*getProduct*(2),2);  
*bill*.show();  
*bill*.calculatingFinalSums();

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Invoice x = new Invoice(*bill*,277277277,"Firma kox");  
x.printInvoice(*cashierID*);

Sklep Fajny  
NIP 328957834275  
Kasjer : 32  
2023-12-08 05:43  
NAZWA FIRMY : Firma kox  
NIP : 277277277  
Mleko muuu 2 3.99 B  
Chleb pszenny 2 3.49 B  
Suma: 14.96  
A 0.0  
B 1.1968  
C 0.0  
D 0.0