Michał Nowaczyk 263971

Michał Bernacki-Janson 264021

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

Spis treści

[Diagram klas 2](#_Toc155383220)

[Struktura klas 3](#_Toc155383221)

[Application 3](#_Toc155383222)

[BarcodeScanner 3](#_Toc155383223)

[VATBracket 3](#_Toc155383224)

[PrintingPosition 3](#_Toc155383225)

[Product 3](#_Toc155383226)

[InternalPosition 4](#_Toc155383227)

[Database 4](#_Toc155383228)

[Invoice 4](#_Toc155383229)

[Bill 5](#_Toc155383230)

[Diagramy sekwencji 6](#_Toc155383231)

[Diagram PU adding/removing a product 6](#_Toc155383232)

[Subdiagram getProduct() 7](#_Toc155383233)

[Subdiagram addPosition() 8](#_Toc155383234)

[Subdiagram removePosition() 9](#_Toc155383235)

[Subdiagram printPosition() 10](#_Toc155383236)

[Diagram PU Finalzaing the bill 11](#_Toc155383237)

[Uzyskany kod 13](#_Toc155383238)

[BarcodeScanner 13](#_Toc155383239)

[VATBracket 13](#_Toc155383240)

[Invoice 14](#_Toc155383241)

[PrintingPosition 14](#_Toc155383242)

[Database 14](#_Toc155383243)

[Bill 14](#_Toc155383244)

[InternalPosition 18](#_Toc155383245)

[Application 18](#_Toc155383246)

[Product 19](#_Toc155383247)

# Obraz zawierający tekst, diagram, zrzut ekranu, linia Opis wygenerowany automatycznieDiagram klas

## 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

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);  
}

### Obraz zawierający tekst, diagram, Równolegle, linia Opis wygenerowany automatycznieSubdiagram 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;  
}

### Obraz zawierający tekst, zrzut ekranu, Równolegle, diagram Opis wygenerowany automatycznieSubdiagram addPosition()

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()

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;  
}

### Obraz zawierający tekst, zrzut ekranu, diagram, Równolegle Opis wygenerowany automatycznieSubdiagram printPosition()

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

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");  
}

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);  
}

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

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

Obraz zawierający tekst, zrzut ekranu, menu

Opis wygenerowany automatycznie