Hibernate, JPA – laboratorium

Julia Smerdel

Zadanie Domowe

1. Wprowadzenie pojęcia produktu.

Klasa Product:

```java
package org.example;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int productID;
    1 usage
    private String ProductName;
    1 usage
    private int UnitsOnStock;

    public Product(){
    }

    public Product(String name, int number){
        this.ProductName = name;
        this.UnitsOnStock = number;
    }
}
```

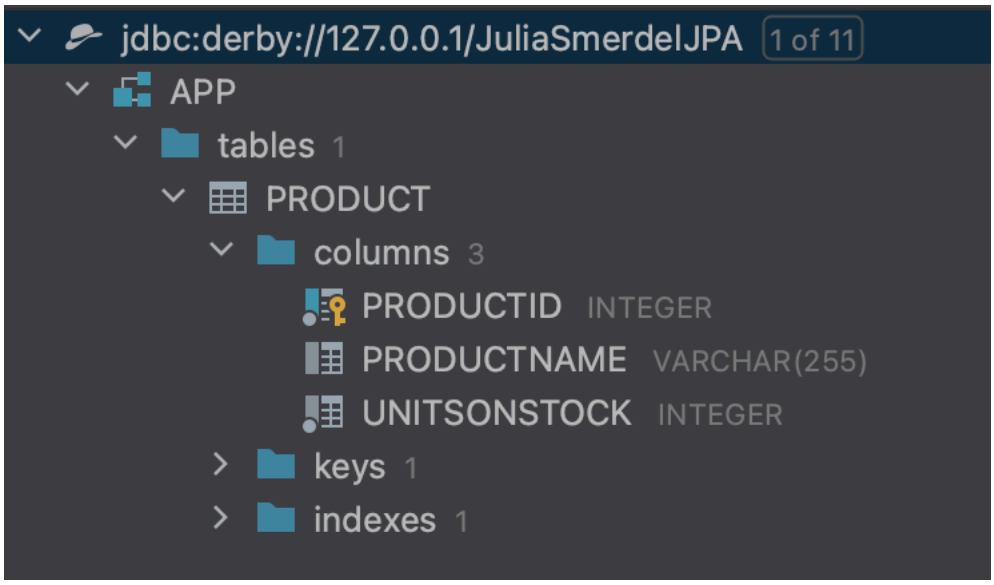
Klasa Main:

```java
package org.example;
import org.hibernate.HibernateException;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;

public class Main {
    2 usages
    private static final SessionFactory ourSessionFactory;

    static {
        try {
            Configuration configuration = new Configuration();
            configuration.configure();

            ourSessionFactory = configuration.buildSessionFactory();
        } catch (Throwable ex) {
            throw new ExceptionInInitializerError(ex);
        }
    }

    1 usage
    public static Session getSession() throws HibernateException {
        return ourSessionFactory.openSession();
    }
    public static void main(final String[] args) throws Exception {
        final Session session = getSession();
        Product product = new Product( name: "Krzesło",  number: 111);
        try {
            Transaction tx = session.beginTransaction();
            session.save(product);
            tx.commit();
        } finally {
            session.close();
        }
    }
}
```

Plik konfiguracyjny:

```xml
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD//EN"
    "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <property name="connection.url">jdbc:derby://127.0.0.1/JuliaSmerdelJPA;create=true;</property>
    <property name="connection.driver_class">org.apache.derby.jdbc.ClientDriver</property>

    <!-- DB schema will be updated if needed -->
    <property name="hibernate.hbm2ddl.auto">update</property>
    <property name="show_sql">true</property>
    <property name="format_sql">true</property>
    <mapping class = "org.example.Product"></mapping>
  </session-factory>
</hibernate-configuration>
```

Wywołania:



	PRODUCTID	PRODUCTNAME	UNITSONSTOCK
1	1	Krzesło	111

2. Wprowadzenie pojęcia dostawcy.

Klasa Supplier:

```java
package org.example;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int supplierID;

    1 usage
    private String CompanyName;
    1 usage
    private String Street;
    1 usage
    private String City;

    public Supplier(){

    }
    public Supplier(String name, String street, String city){
        this.CompanyName = name;
        this.Street = street;
        this.City = city;
    }
}
```

Klasa Product:

```java
@Entity
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int productID;
    1 usage
    private String ProductName;
    1 usage
    private int UnitsOnStock;

    2 usages
    @ManyToOne
    @JoinColumn(name = "supplierID")
    private Supplier supplier;


    public Product(){
    }

    public Product(String name, int number){
        this.ProductName = name;
        this.UnitsOnStock = number;
    }

1 usage
public void setSupplier(Supplier supplier){
    this.supplier = supplier;
}

public Supplier getSupplier(){
    return this.supplier;
}
```

Klasa Main:

Względem punktu 1 zmiana nastąpiła wyłącznie w kodzie zamieszczonym poniżej.

```
public static void main(final String[] args) throws Exception {
    try (Session session = getSession()){
        Transaction tx = session.beginTransaction();

        Product product = session.get(Product.class, serializable: 1);
        Supplier supplier = new Supplier( name: "FikuMiku", street: "Miła", city: "Wrocław");

        product.setSupplier(supplier);

        session.save(product);
        session.save(supplier);
        tx.commit();
        session.close();
    }
}
```

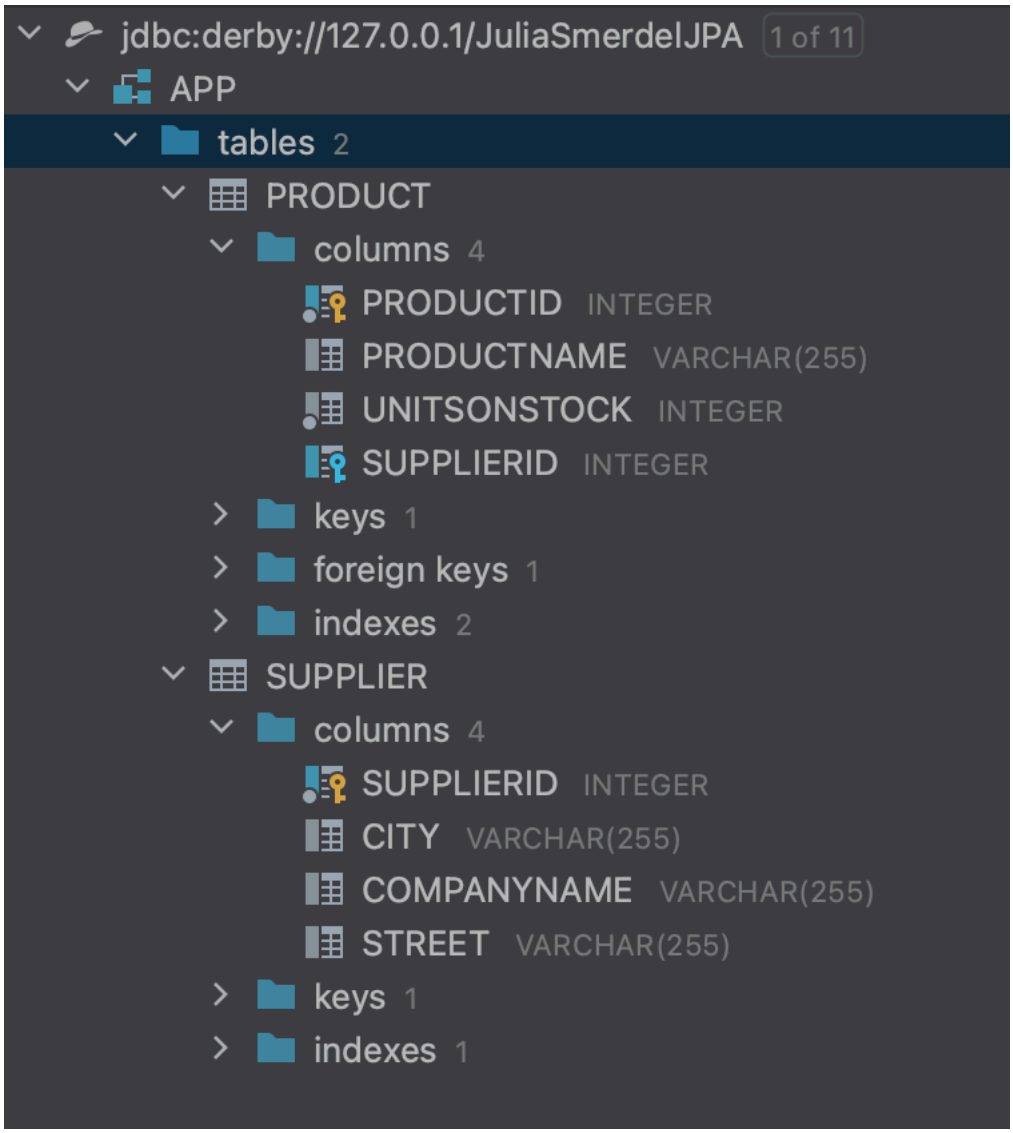
Plik konfiguracyjny hibernate:

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD//EN"
    "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <property name="connection.url">jdbc:derby://127.0.0.1/JuliaSmerdelJPA;create=true;</property>
    <property name="connection.driver_class">org.apache.derby.jdbc.ClientDriver</property>

    <!-- DB schema will be updated if needed -->
    <property name="hibernate.hbm2ddl.auto">update</property>
    <property name="show_sql">true</property>
    <property name="format_sql">true</property>
    <mapping class = "org.example.Product"></mapping>
    <mapping class = "org.example.Supplier"></mapping>
  </session-factory>
</hibernate-configuration>
```

Wywołanie:

	PRODUCTID	PRODUCTNAME	UNITSONSTOCK	SUPPLIERID
1	1	Krzesło	111	2

	SUPPLIERID	CITY	COMPANYNAME	STREET
1	2	Wrocław	FikuMiku	Miła



3. Odwrócenie relacji.

3a) z tabelą łącznikową

Klasa Product:

```
13 usages
@Entity
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int productID;
    1 usage
    private String ProductName;
    1 usage
    private int UnitsOnStock;

    public Product(){
    }

    4 usages
    public Product(String name, int number){
        this.ProductName = name;
        this.UnitsOnStock = number;
    }

}
```

Klasa Supplier:

```java
@Entity
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int supplierID;

    1 usage
    private String CompanyName;
    1 usage
    private String Street;
    1 usage
    private String City;

    2 usages
    @OneToMany
    private Collection<Product> products = new ArrayList<>();


    public Supplier(){

    }
    3 usages
    public Supplier(String name, String street, String city){
        this.CompanyName = name;
        this.Street = street;
        this.City = city;
    }

    3 usages
    public void setProducts(Product ...products){
        this.products.addAll(Arrays.asList(products));
    }

    public Collection<Product> getProducts(){
        return this.products;
    }
}
```

W pliku konfiguracyjnym hibernate zostało zmienione wyłącznie

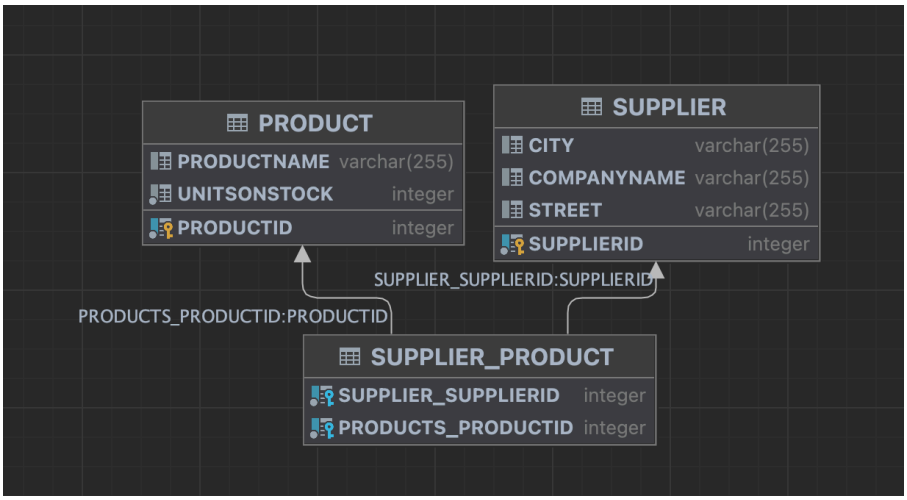
```
<property name="hibernate.hbm2ddl.auto">create-drop</property>
```

Na create-drop, aby ułatwić wpisywanie danych.

Klasa Main:

```
public static void main(final String[] args) throws Exception {
    try (Session session = getSession()){
        Transaction tx = session.beginTransaction();

        Product product1 = new Product( name: "Pomidor",  number: 10);
        Product product2 = new Product( name: "Gruszka",  number: 1);
        Product product3 = new Product( name: "Sałata",  number: 12);
        Product product4 = new Product( name: "Jabłko",  number: 4);

        Supplier supplier1 = new Supplier( name: "FikuMiku",  street: "Miła",  city: "Wrocław");
        Supplier supplier2 = new Supplier( name: "Frutki",  street: "Owocowa",  city: "Kraków");
        Supplier supplier3 = new Supplier( name: "PoraNaPomidora",  street: "Pomidorowa",  city: "Warszawa");

        supplier1.setProducts(product4);
        supplier2.setProducts(product2);
        supplier3.setProducts(product1, product3);
        session.save(product1);
        session.save(product2);
        session.save(product3);
        session.save(product4);
        session.save(supplier1);
        session.save(supplier2);
        session.save(supplier3);
        tx.commit();
        session.close();
    }
}
```

Wywołania:



	PRODUCTID	PRODUCTNAME	UNITSONSTOCK
1	1	Pomidor	10
2	2	Gruszka	1
3	3	Sałata	12
4	4	Jabłko	4

	SUPPLIERID	CITY	COMPANYNAME	STREET
1	5	Wrocław	FikuMiku	Miła
2	6	Kraków	Frutki	Owocowa
3	7	Warszawa	PoraNaPomidora	Pomidorowa

	SUPPLIER_SUPPLIERID	PRODUCTS_PRODUCTID
1	5	4
2	6	2
3	7	1
4	7	3

3b) bez tabeli łącznikowej

Klasa Supplier:

```java
@Entity
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long supplierID;

    1 usage
    private String CompanyName;
    1 usage
    private String Street;
    1 usage
    private String City;

    2 usages
    @OneToMany
    @JoinColumn(name="SUPPLIER_FK")
    private Collection<Product> products = new ArrayList<>();


    public Supplier(){

    }
    3 usages
    public Supplier(String name, String street, String city){
        this.CompanyName = name;
        this.Street = street;
        this.City = city;
    }
    3 usages
    public void setProducts(Product ...products) { this.products.addAll(Arrays.asList(products)); }

    public Collection<Product> getProducts() { return this.products; }
}
```

Klasa Product:

```java
@Entity
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int productID;
    1 usage
    private String ProductName;
    1 usage
    private int UnitsOnStock;

    @Column(name="SUPPLIER_FK")
    private int supplierFK;

    public Product(){
    }

    4 usages
    public Product(String name, int number){
        this.ProductName = name;
        this.UnitsOnStock = number;
    }

}
```

Klasa Main identyczna jak w 3a)

Wywołania:

	PRODUCTID	PRODUCTNAME	UNITSONSTOCK	SUPPLIER_FK
1	1	Pomidor	10	3
2	2	Gruszka	1	2
3	3	Sałata	12	3
4	4	Jabłko	4	1

	SUPPLIERID	CITY	COMPANYNAME	STREET
1	1	Wrocław	FikuMiku	Miła
2	2	Kraków	Frutki	Owocowa
3	3	Warszawa	PoraNaPomidora	Pomidorowa

4. Relacja dwustronna.

Klasa Supplier

```java
@Entity
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long supplierID;

    1 usage
    private String CompanyName;
    1 usage
    private String Street;
    1 usage
    private String City;

    2 usages
    @OneToMany
    private Collection<Product> products = new ArrayList<>();


    public Supplier(){

    }
    3 usages
    public Supplier(String name, String street, String city){
        this.CompanyName = name;
        this.Street = street;
        this.City = city;
    }
```

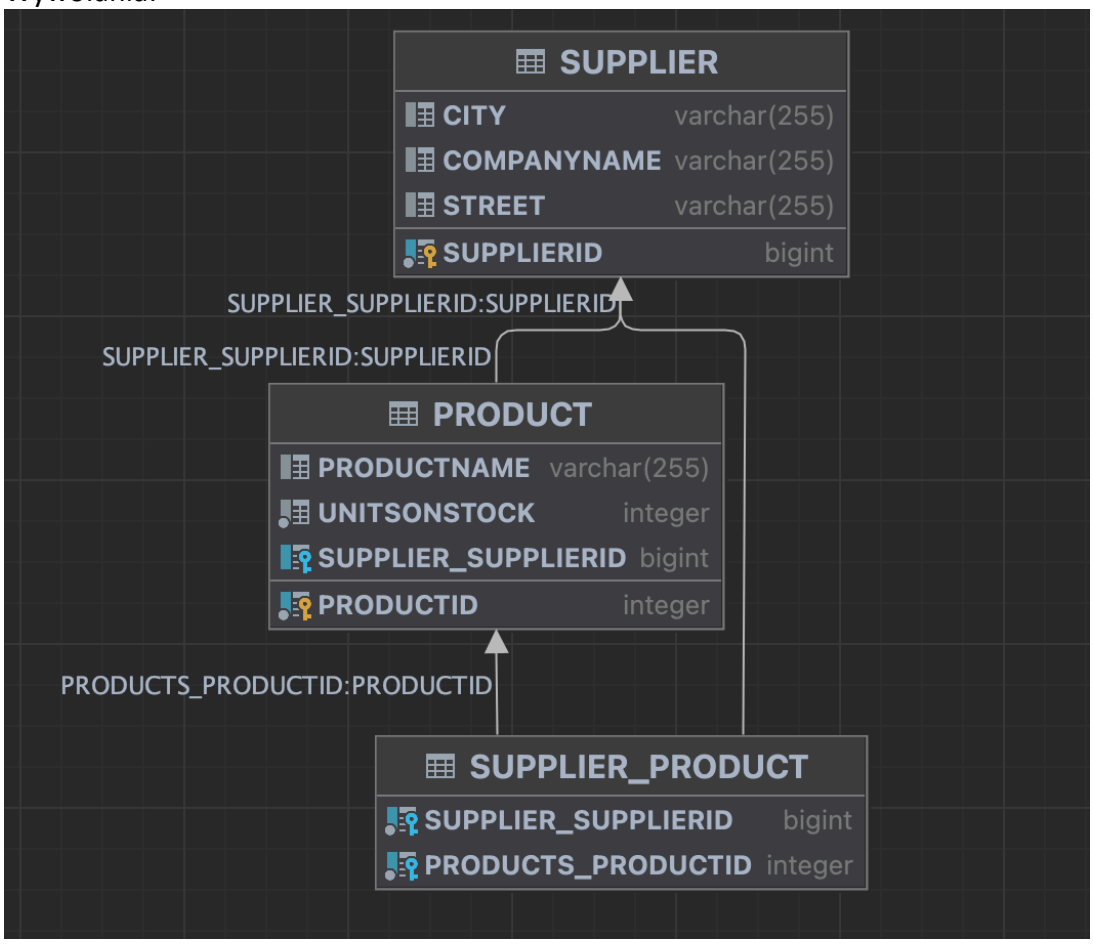
Klasa Product:

```java
@Entity
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int productID;
    1 usage
    private String ProductName;
    1 usage
    private int UnitsOnStock;

    1 usage
    @ManyToOne
    private Supplier supplier;

    public Product(){
    }

    4 usages
    public Product(String name, int number){
        this.ProductName = name;
        this.UnitsOnStock = number;
    }

    4 usages
    public void setSupplier(Supplier supplier){
        this.supplier = supplier;
    }
```

Klasa Main:

```
public static void main(final String[] args) throws Exception {
    try (Session session = getSession()){
        Transaction tx = session.beginTransaction();

        Product product1 = new Product( name: "Pomidor",  number: 10);
        Product product2 = new Product( name: "Gruszka",  number: 1);
        Product product3 = new Product( name: "Sałata",  number: 12);
        Product product4 = new Product( name: "Jabłko",  number: 4);

        Supplier supplier1 = new Supplier( name: "FikuMiku",  street: "Miła",  city: "Wrocław");
        Supplier supplier2 = new Supplier( name: "Frutki",  street: "Owocowa",  city: "Kraków");
        Supplier supplier3 = new Supplier( name: "PoraNaPomidora",  street: "Pomidorowa",  city: "Warszawa");


        session.save(product1);
        session.save(product2);
        session.save(product3);
        session.save(product4);
        session.save(supplier1);
        session.save(supplier2);
        session.save(supplier3);

        supplier1.setProducts(product4);
        supplier2.setProducts(product2);
        supplier3.setProducts(product1, product3);
       supplier1.setProducts(product4);
       supplier2.setProducts(product2);
       supplier3.setProducts(product1, product3);

       product4.setSupplier(supplier1);
       product2.setSupplier(supplier2);
       product1.setSupplier(supplier3);
       product3.setSupplier(supplier3);


       tx.commit();
       session.close();
    }
}
```

Wywołania:



	SUPPLIER_SUPPLIERID	PRODUCTS_PRODUCTID
1	1	4
2	2	2
3	3	1
4	3	3

	SUPPLIERID	CITY	COMPANYNAME	STREET
1	1	Wrocław	FikuMiku	Miła
2	2	Kraków	Frutki	Owocowa
3	3	Warszawa	PoraNaPomidora	Pomidorowa

	PRODUCTID	PRODUCTNAME	UNITSONSTOCK	SUPPLIER_SUPPLIERID
1	1	Pomidor	10	3
2	2	Gruszka	1	2
3	3	Sałata	12	3
4	4	Jabłko	4	1

5. Dodanie Category.

W pliku hibernate.cfg.xml zostało dodane

```
<mapping class = "org.example.Category"></mapping>
```

Klasa Category:

```
@Entity
public class Category {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int categoryID;

    1 usage
    private String name;

    1 usage
    @OneToMany
    private Collection<Product> products = new ArrayList<>();

    public Category(){

    }

    2 usages
    public Category(String name){
        this.name = name;
    }
    public void addProduct(Product product) { products.add(product); }

    1 usage
    public ArrayList<String> getProducts(){
        ArrayList<String> names = new ArrayList<>();
        for (Product prod: products){
            names.add(prod.getName());
        }
        return names;
    }
    2 usages
    public String getName(){
        return this.name;
    }
```

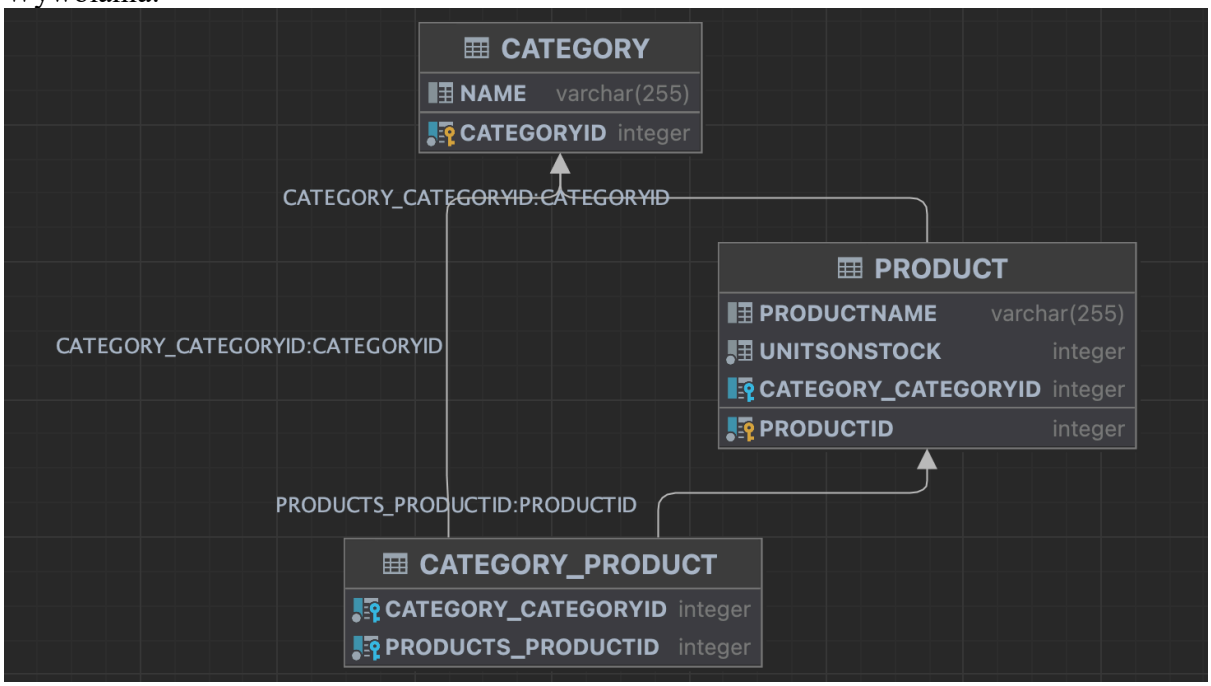
Klasa Product:

```java
@Entity
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int productID;
    1 usage
    private String ProductName;
    1 usage
    private int UnitsOnStock;

    1 usage
    @ManyToOne
    private Category category;

    public Product(){
    }

    public Product(String name, int number){
        this.ProductName = name;
        this.UnitsOnStock = number;
    }
public void setCategory(Category category){
    this.category = category;
}

1 usage
public Category getCategory(){
    return this.category;
}
1 usage
public String getName(){
    return this.ProductName;
}
```

Klasa Main:

```
public static void main(final String[] args) throws Exception {
    try (Session session = getSession()){
        Transaction tx = session.beginTransaction();

        Category category1 = new Category( name: "Warzywa");
        Category category2 = new Category( name: "Owoce");


        Product p1 = session.get(Product.class,  serializable: 5);
        Product p2 = session.get(Product.class,  serializable: 6);
        Product p3 = session.get(Product.class,  serializable: 7);
        Product p4 = session.get(Product.class,  serializable: 8);

        category1.addProduct(p1);
        category1.addProduct(p3);
        category2.addProduct(p2);
        category2.addProduct(p4);
            p1.setCategory(category1);
            p2.setCategory(category2);
            p3.setCategory(category1);
            p4.setCategory(category2);

            session.save(category1);
            session.save(category2);
Category category = session.get(Category.class,  serializable: 1);
System.out.println("Category: " + category.getName() + ", Products in category: " + category.getProducts());

Product product = session.get(Product.class,  serializable: 5);
System.out.println("Product: " + product.getName() + ", Category: " + product.getCategory().getName());
```

Wywołania:



	CATEGORY_CATEGORYID	PRODUCTS_PRODUCTID
1	1	5
2	1	7
3	2	6
4	2	8

	CATEGORYID	NAME
1	1	Warzywa
2	2	Owoce

	PRODUCTID	PRODUCTNAME	UNITSONSTOCK	CATEGORY_CATEGORYID
1	5	Pomidor	10	1
2	6	Gruszka	1	2
3	7	Sałata	12	1
4	8	Jabłko	4	2

```
Category: Warzywa, Products in category: [Pomidor, Sałata]
Product: Pomidor, Category: Warzywa
```

6. Relacja wiele-do-wielu.

W pliku hibernate.cfg.xml zostało dodane

```
<mapping class = "org.example.Invoice"></mapping>
```

Klasa Product:

```
@Entity
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int productID;
    2 usages
    private String ProductName;
    4 usages
    private int UnitsOnStock;

    2 usages
    @ManyToMany(mappedBy = "products")
    private Collection<Invoice> invoices = new HashSet<>();

    public Product(){
    }

    4 usages
    public Product(String name, int number){
        this.ProductName = name;
        this.UnitsOnStock = number;
    }
3 usages
public String getName() { return this.ProductName; }

1 usage
public Collection<Invoice> getInvoice(){
    return this.invoices;
}
public int getUnitsOnStock(){
    return this.UnitsOnStock;
}

4 usages
public void sellProduct(int wantsToBuy, Invoice invoice) throws InvalidAttributesException {
    if (wantsToBuy > this.UnitsOnStock){
        throw new InvalidAttributesException("Cannot sell more than it is available");
    }
    this.UnitsOnStock -= wantsToBuy;
    invoice.addProduct( product: this, wantsToBuy);
    invoices.add(invoice);
}
```

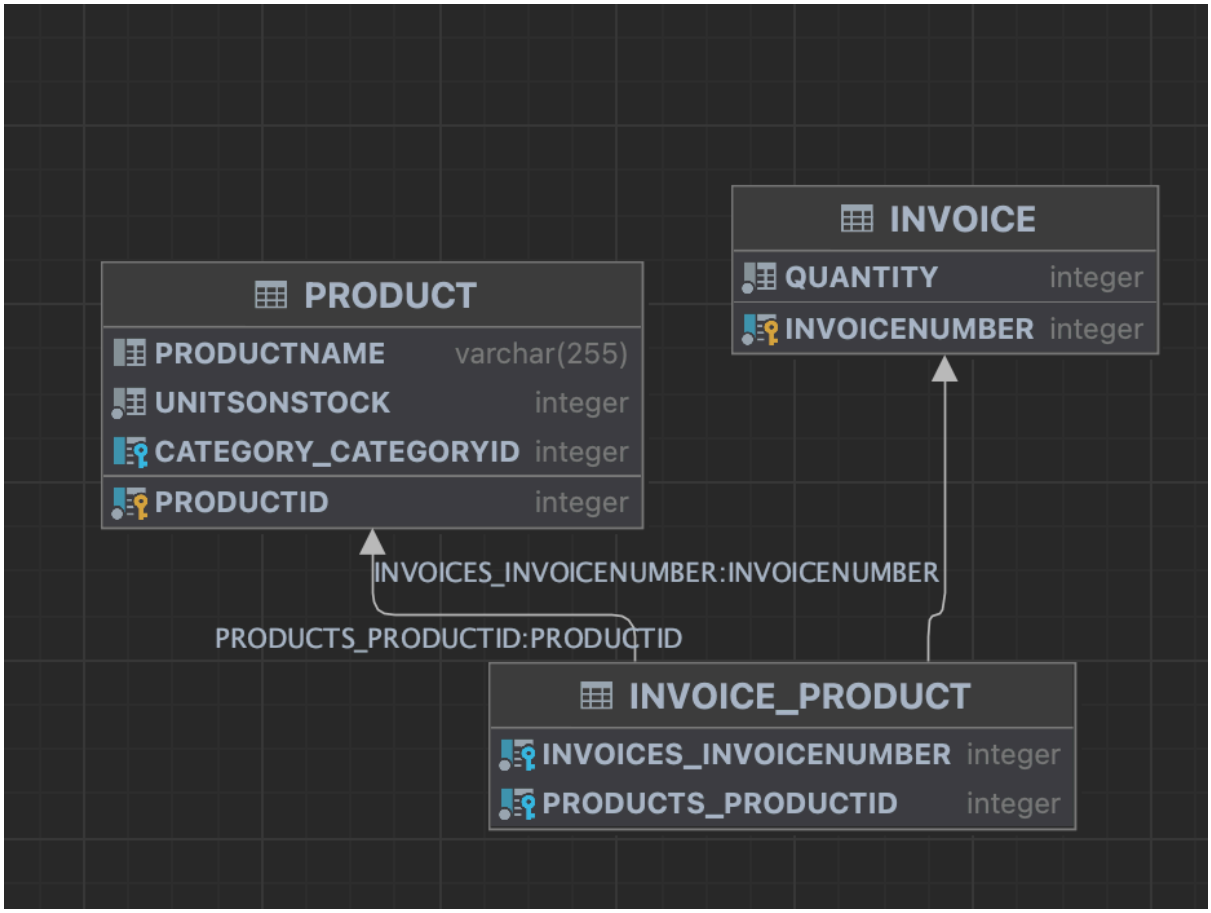
Klasa Main:

```
public static void main(final String[] args) throws Exception {
    try (Session session = getSession()){
        Transaction tx = session.beginTransaction();

        Product prod1 = new Product( name: "Zegarek",  number: 34);
        Product prod2 = new Product( name: "Łódka",  number: 1);
        Product prod3 = new Product( name: "Myszka",  number: 20);
        Product prod4 = new Product( name: "Butelka",  number: 100);

        Invoice invoice1 = new Invoice();
        Invoice invoice2 = new Invoice();

        prod1.sellProduct( wantsToBuy: 14, invoice1);
        prod2.sellProduct( wantsToBuy: 1, invoice1);
        prod1.sellProduct( wantsToBuy: 15, invoice2);
        prod4.sellProduct( wantsToBuy: 33, invoice2);


        session.save(prod1);
        session.save(prod2);
        session.save(prod3);
        session.save(prod4);
        session.save(invoice1);
        session.save(invoice2);
        Invoice invoice = session.get(Invoice.class,  serializable: 1);
        System.out.println("Invoice number: " + invoice.getInvoiceNumber() + ", items: " + invoice.getProducts());
        Product product = session.get(Product.class,  serializable: 1);
        System.out.println("Product: " + product.getName() + ", invoices: ");
        Collection<Invoice> invoices = product.getInvoice();
        for (Invoice inv : invoices){
            System.out.println(inv.getInvoiceNumber());
        }

        tx.commit();
        session.close();
    }
}
```

Klasa Invoice:

```java
@Entity
public class Invoice {
    1 usage
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int invoiceNumber;

    2 usages
    private int quantity;

    2 usages
    @ManyToMany
    private Collection<Product> products = new HashSet<>();

    2 usages
    public Invoice(){
    }

    1 usage
    public void addProduct(Product product, int quantity){
        products.add(product);
        this.quantity += quantity;
    }

    2 usages
    public int getInvoiceNumber(){
        return this.invoiceNumber;
    }
    public int getQuantity(){
        return this.quantity;
    }

    1 usage
    public Collection<String> getProducts(){
        ArrayList<String> names = new ArrayList<>();
        for (Product prod: products){
            names.add(prod.getName());
        }
        return names;
    }
```

Wywołania:



	INVOICENUMBER	QUANTITY
1	1	15
2	2	48

	PRODUCTID	PRODUCTNAME	UNITSONSTOCK
1	1	Zegarek	5
2	2	Łódka	0
3	3	Myszka	20
4	4	Butelka	67

	INVOICES_INVOICENUMBER	PRODUCTS_PRODUCTID
1	1	1
2	1	2
3	2	4
4	2	1

```
Invoice number: 1, items: [Zegarek, Łódka]
Product: Zegarek, invoices:
1
2
```

7. JPA.

Klasa Main:

```java
public static void main(final String[] args) throws InvalidAttributesException {
    EntityManagerFactory emf = Persistence.createEntityManagerFactory( persistenceUnitName: "JuliaSmerdel");
    EntityManager em = emf.createEntityManager();
    EntityTransaction etx = em.getTransaction();
    etx.begin();

    Product prod1 = new Product( name: "Zegarek",  number: 34);
    Product prod2 = new Product( name: "Łódka",  number: 1);
    Product prod3 = new Product( name: "Myszka",  number: 20);
    Product prod4 = new Product( name: "Butelka",  number: 100);

    Invoice invoice1 = new Invoice();
    Invoice invoice2 = new Invoice();

    prod1.sellProduct( wantsToBuy: 14, invoice1);
    prod2.sellProduct( wantsToBuy: 1, invoice1);
    prod1.sellProduct( wantsToBuy: 15, invoice2);
    prod4.sellProduct( wantsToBuy: 33, invoice2);

  em.persist(prod1);
  em.persist(prod2);
  em.persist(prod3);
  em.persist(prod4);
  em.persist(invoice1);
  em.persist(invoice2);

  Invoice invoice = em.find(Invoice.class,  o: 1);
  System.out.println("Invoice number: " + invoice.getInvoiceNumber() + ", items: " + invoice.getProducts());
  Product product = em.find(Product.class,  o: 1);
  System.out.println("Product: " + product.getName() + ", invoices: ");
  Collection<Invoice> invoices = product.getInvoice();
  for (Invoice inv : invoices){
      System.out.println(inv.getInvoiceNumber());
  }

  etx.commit();
  em.close();
```

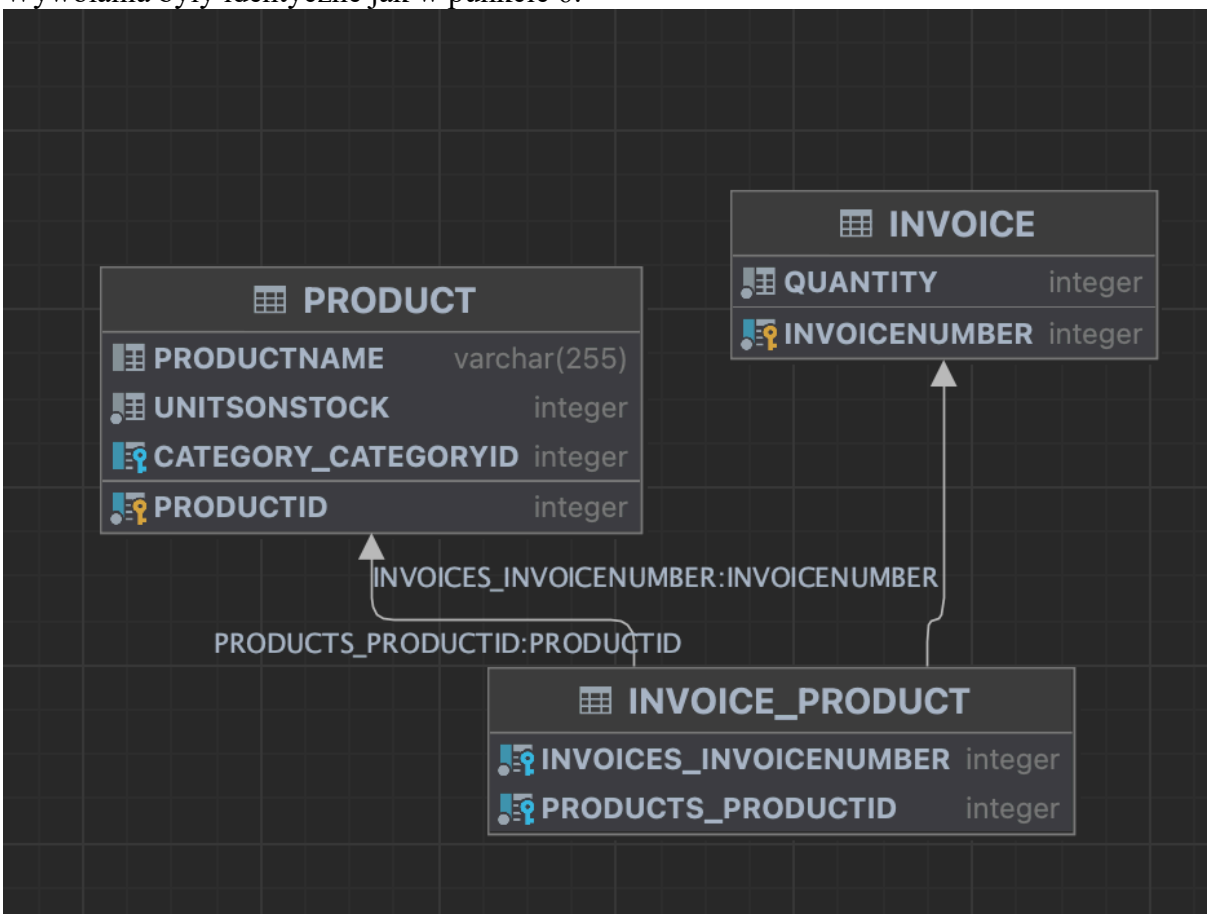
Plik persistence.xml:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd"
             version="2.0">
  <persistence-unit name="JuliaSmerdel">
    <properties>
      <property name="hibernate.connection.driver_class"
                value="org.apache.derby.jdbc.ClientDriver"/>
      <property name="hibernate.connection.url"
                value="jdbc:derby://127.0.0.1/JuliaSmerdelJPA"/>
      <property name="hibernate.show_sql" value="true"/>
      <property name="hibernate.format_sql" value="true"/>
      <property name="hibernate.hbm2ddl.auto" value="create-drop"/>
    </properties>

  </persistence-unit>
</persistence>
```

Plik pom.xml:

```xml
<dependencies>
    <dependency>
        <groupId>javax.persistence</groupId>
        <artifactId>javax.persistence-api</artifactId>
        <version>2.2</version>
    </dependency>
</dependencies>
```

Wywołania były identyczne jak w punkcie 6:




```
Invoice number: 1, items: [Zegarek, Łódka]
Product: Zegarek, invoices:
1
2
```

	INVOICENUMBER	QUANTITY
1	1	15
2	2	48

	INVOICES_INVOICENUMBER	PRODUCTS_PRODUCTID
1	1	1
2	1	2
3	2	4
4	2	1

	PRODUCTID	PRODUCTNAME	UNITSONSTOCK
1	1	Zegarek	5
2	2	Łódka	0
3	3	Myszka	20
4	4	Butelka	67

8. Kaskady.

W klasie Product zmiana nastąpiła wyłącznie tutaj:

```
@ManyToMany(mappedBy = "products", cascade = {CascadeType.PERSIST})
private Collection<Invoice> invoices = new HashSet<>();

public Product(){
}
```

Analogicznie w klasie Invoice:

```
@ManyToMany(cascade = {CascadeType.PERSIST})
private Collection<Product> products = new HashSet<>();
```

Klasa Main:

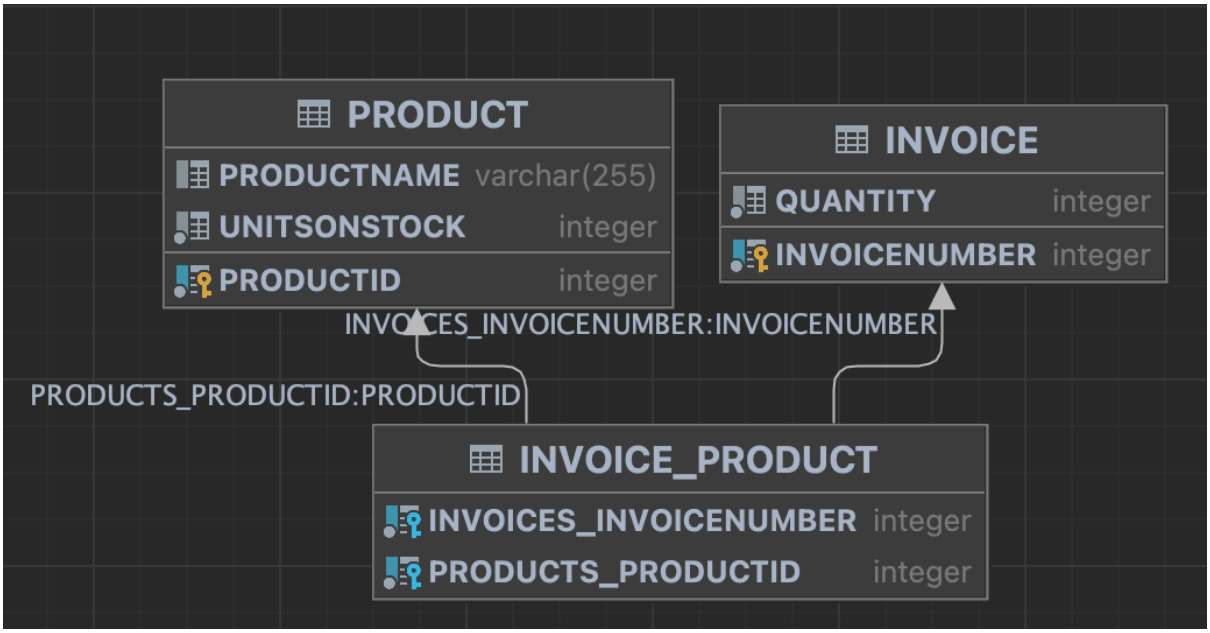
```
public static void main(final String[] args) throws InvalidAttributesException {
    EntityManagerFactory emf = Persistence.createEntityManagerFactory( persistenceUnitName: "JuliaSmerdel");
    EntityManager em = emf.createEntityManager();
    EntityTransaction etx = em.getTransaction();
    etx.begin();

    Product prod1 = new Product( name: "Zegarek",  number: 34);
    Product prod2 = new Product( name: "Łódka",  number: 1);
    Product prod3 = new Product( name: "Myszka",  number: 20);
    Product prod4 = new Product( name: "Butelka",  number: 100);

    Invoice invoice1 = new Invoice();
    Invoice invoice2 = new Invoice();

    prod1.sellProduct( wantsToBuy: 14, invoice1);
    prod2.sellProduct( wantsToBuy: 1, invoice1);
    prod1.sellProduct( wantsToBuy: 15, invoice2);
    prod4.sellProduct( wantsToBuy: 33, invoice2);

  em.persist(prod1);
  em.persist(prod2);
  em.persist(prod3);
  em.persist(prod4);
  em.persist(invoice1);
  em.persist(invoice2);

  Invoice invoice = em.find(Invoice.class,  o: 1);
  System.out.println("Invoice number: " + invoice.getInvoiceNumber() + ", items: " + invoice.getProducts());
  Product product = em.find(Product.class,  o: 1);
  System.out.println("Product: " + product.getName() + ", invoices: ");
  Collection<Invoice> invoices = product.getInvoice();
  for (Invoice inv : invoices){
      System.out.println(inv.getInvoiceNumber());
  }

  etx.commit();
  em.close();
```

Wywołania:

	INVOICES_INVOICENUMBER	PRODUCTS_PRODUCTID
1	1	1
2	1	2
3	2	1
4	2	3

	PRODUCTID	PRODUCTNAME	UNITSONSTOCK
1	1	Zegarek	5
2	2	Butelka	67
3	3	Łódka	0
4	4	Myszka	20

	INVOICENUMBER	QUANTITY
1	1	48
2	2	15



9. Embedded class.

9a) Wbudowanie klasy do tabeli

Klasa Address:

```java
@Embeddable
public class Address {
    3 usages
    private String city;
    3 usages
    private String street;

    public Address(){}

    2 usages
    public Address(String city, String street){
        this.city = city;
        this.street = street;
    }

    public void setCity(String city) {
        this.city = city;
    }
public String getCity() {
    return city;
}

public void setStreet(String street) {
    this.street = street;
}

public String getStreet() {
    return street;
}
```

Klasa Supplier:

```java
@Entity
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int supplierID;

    1 usage
    private String CompanyName;
    3 usages
    @Embedded
    private Address address;

    1 usage
    public Supplier(){

    }
  public Supplier(String name, Address address){
      this.CompanyName = name;
      this.address = address;
  }

  public Address getAddress() {
      return address;
  }

  public void setAddress(Address address) {
      this.address = address;
  }
```

Klasa Main:

```java
public static void main(final String[] args) throws InvalidAttributesException {
    EntityManagerFactory emf = Persistence.createEntityManagerFactory( persistenceUnitName: "JuliaSmerdel");
    EntityManager em = emf.createEntityManager();
    EntityTransaction etx = em.getTransaction();
    etx.begin();

    Supplier supplier1 = new Supplier( name: "FikuMiku", new Address( city: "Warszawa",  street: "Miła 12"));
    Supplier supplier2 = new Supplier( name: "Chałka", new Address( city: "Kraków",  street: "Całka"));

    em.persist(supplier1);
    em.persist(supplier2);


    etx.commit();
    em.close();

}
```

Wywołania:

	SUPPLIERID	COMPANYNAME	CITY	STREET
1	1	FikuMiku	Warszawa	Miła 12
2	2	Chałka	Kraków	Całka

9a) Zmapowanie do osobnych tabel

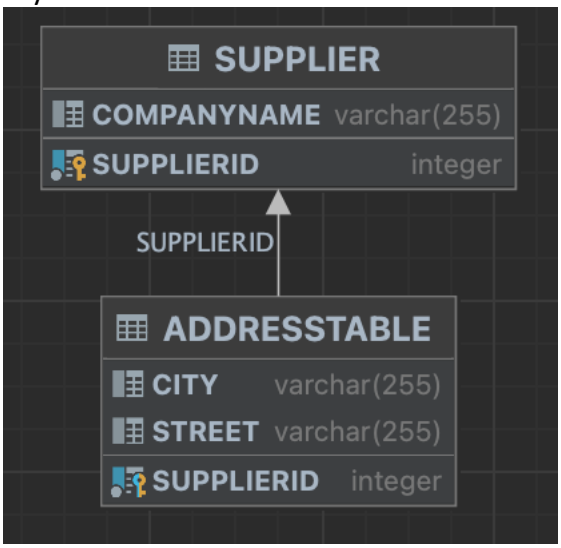
Klasa Supplier:

```java
@Entity
@SecondaryTable(name="AddressTable")
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int supplierID;

    1 usage
    private String CompanyName;

    3 usages
    @Column(table = "AddressTable")
    private String street;
    3 usages
    @Column(table = "AddressTable")
    private String city;

public Supplier(){

}
3 usages
public Supplier(String name, String city, String street){
    this.CompanyName = name;
    this.city = city;
    this.street = street;
}

public String getStreet() {
    return street;
}

public void setStreet(String street) {
    this.street = street;
}
```

```java
public void setStreet(String street) {
    this.street = street;
}

public String getCity() {
    return city;
}

public void setCity(String city) {
    this.city = city;
}
```

Klasa Main identyczna jak w podpunkcie 9a).

Wywołania:



	CITY	STREET	SUPPLIERID
1	Warszawa	Miła	1
2	Kraków	Całka	2

	SUPPLIERID	COMPANYNAME
1	1	FikuMiku
2	2	Chałka

10. Dziedziczenie.

10a) Type per Class

Klasa Company:

```java
@Entity
@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)
public abstract class Company {

    1 usage
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;

    2 usages
    private String companyName;
    2 usages
    private String street;
    2 usages
    private String city;
    2 usages
    private String zipCode;

    2 usages
    public Company(){}

    2 usages
    public Company(String name, String street, String city, String zipCode){
        this.companyName = name;
        this.street = street;
        this.city = city;
        this.zipCode = zipCode;
    }
```

Klasa Supplier:

```java
@Entity

public class Supplier extends Company{
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;

    1 usage
    private String bankAccountNumber;


    1 usage
    public Supplier(){
    }


    2 usages
    public Supplier(String nameS, String street, String city, String code, String bank){
        super(nameS, street, city, code);
        this.bankAccountNumber = bank;
    }
```

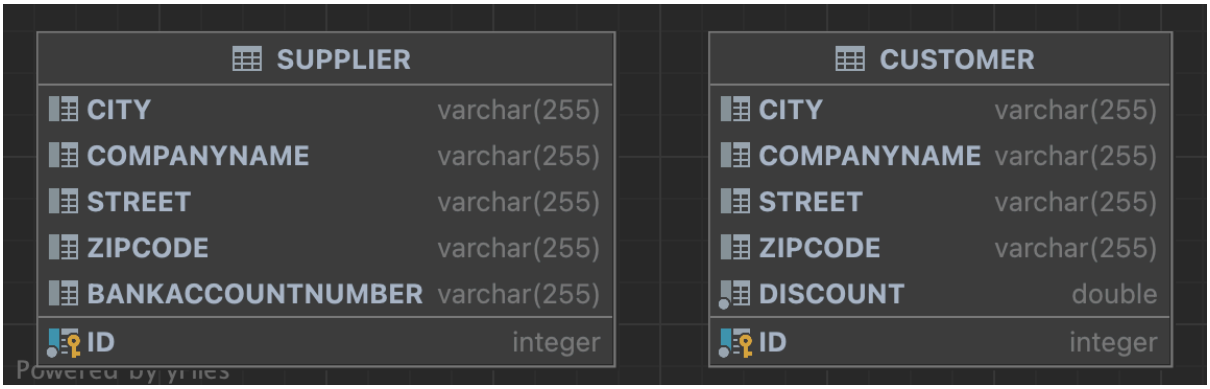
Klasa Customer:

```java
@Entity
public class Customer extends Company{
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;

    1 usage
    private double discount;

    public Customer(){}

    1 usage
    public Customer(String name, String street, String city, String code, double dis){
        super(name, street, city, code);
        this.discount = dis;
    }
}
```

Wywołania:

	ID	CITY	COMPANYNAME	STREET	ZIPCODE	DISCOUNT
1	2	Warszawa	FikuMiku	Miła	21-202	0.08

	ID	CITY	COMPANYNAME	STREET	ZIPCODE	BANKACCOUNTNUMBER
1	1	Kraków	Chałka	Całka	21-212	123456789

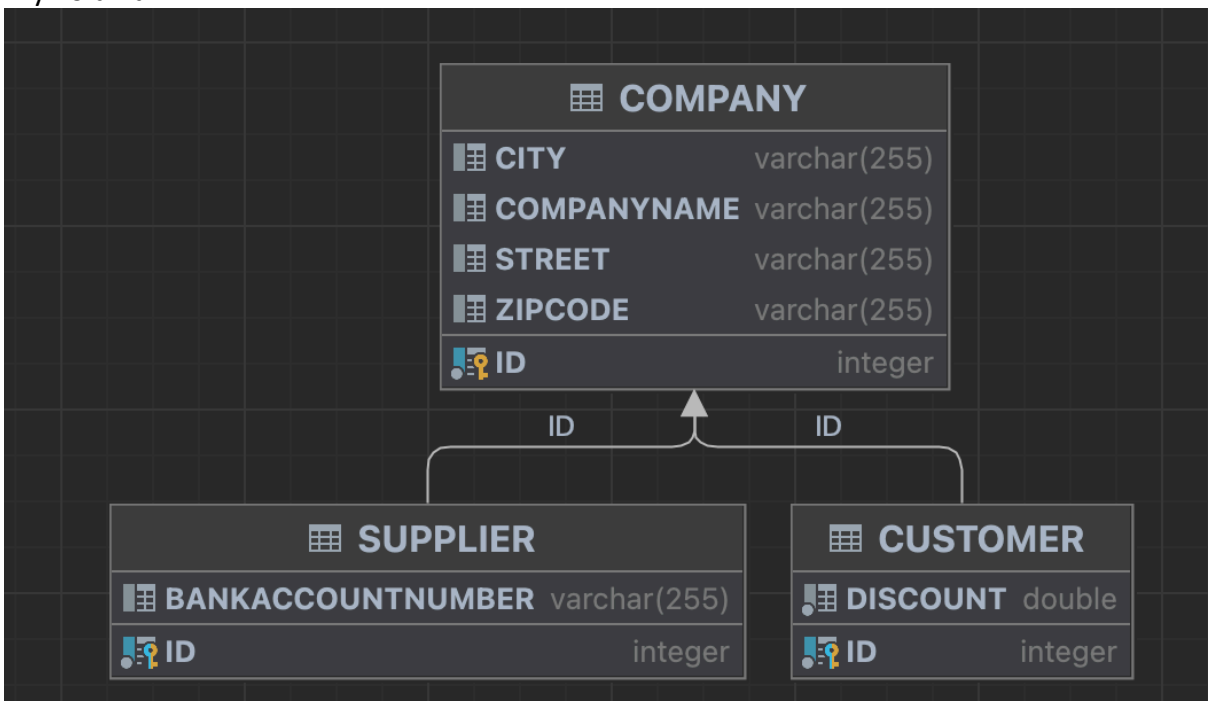


10a) Joined

Zmiana względem 10a) nastąpiła wyłącznie w klasie Company:

```
@Entity
@Inheritance(strategy = InheritanceType.JOINED)
public abstract class Company {
```

Wywołania:



	ID	CITY	COMPANYNAME	STREET	ZIPCODE
1	1	Kraków	Chałka	Całka	21-212
2	2	Warszawa	FikuMiku	Miła	21-202

	BANKACCOUNTNUMBER	ID
1	123456789	1

	DISCOUNT	ID
1	0.08	2

10c) Single-Table

Zmiana w klasie Company:

```
@Entity
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name="companyType")
public abstract class Company {
```

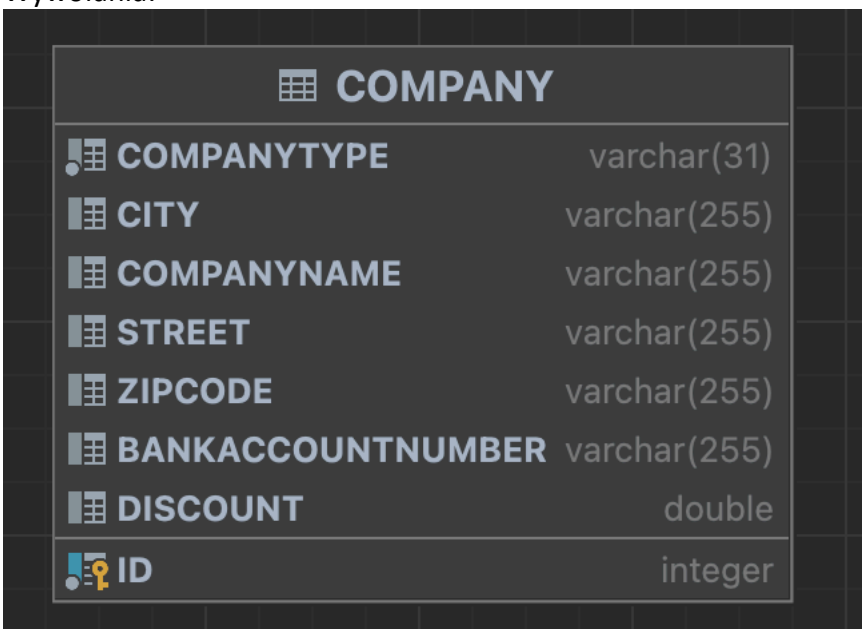
Zmiana w klasie Customer:

```
@Entity
@DiscriminatorValue(value="Customer")
public class Customer extends Company{
```

Zmiana w klasie Supplier:

```
@Entity
@DiscriminatorValue(value="Supplier")
public class Supplier extends Company{
```

Wywołania:



	COMPANYTYPE	ID	CITY	COMPANYNAME	STREET	ZIPCODE	BANKACCOUNTNUMBER	DISCOUNT
1	Supplier	1	Kraków	Chałka	Całka	21-212	123456789	<null>
2	Customer	2	Warszawa	FikuMiku	Miła	21-202	<null>	0.08