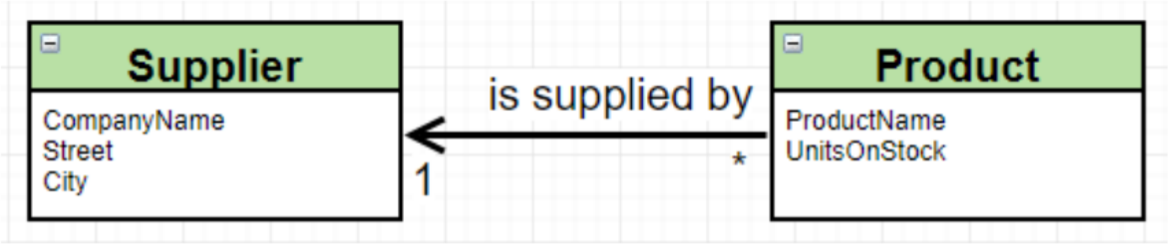
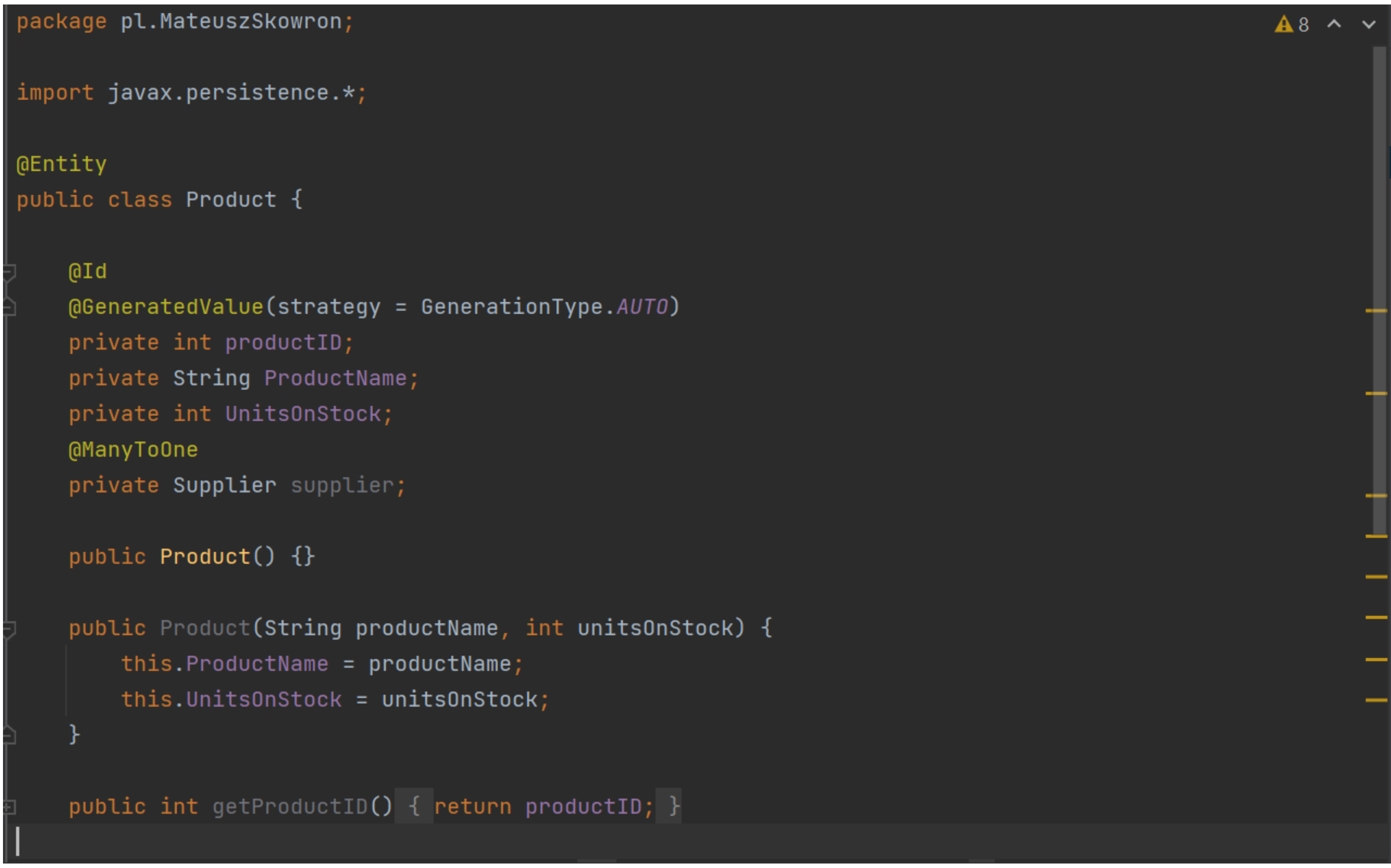
Mateusz Skowron

1. Zmodyfikuj model wprowadzając pojęcie Dostawcy jak poniżej

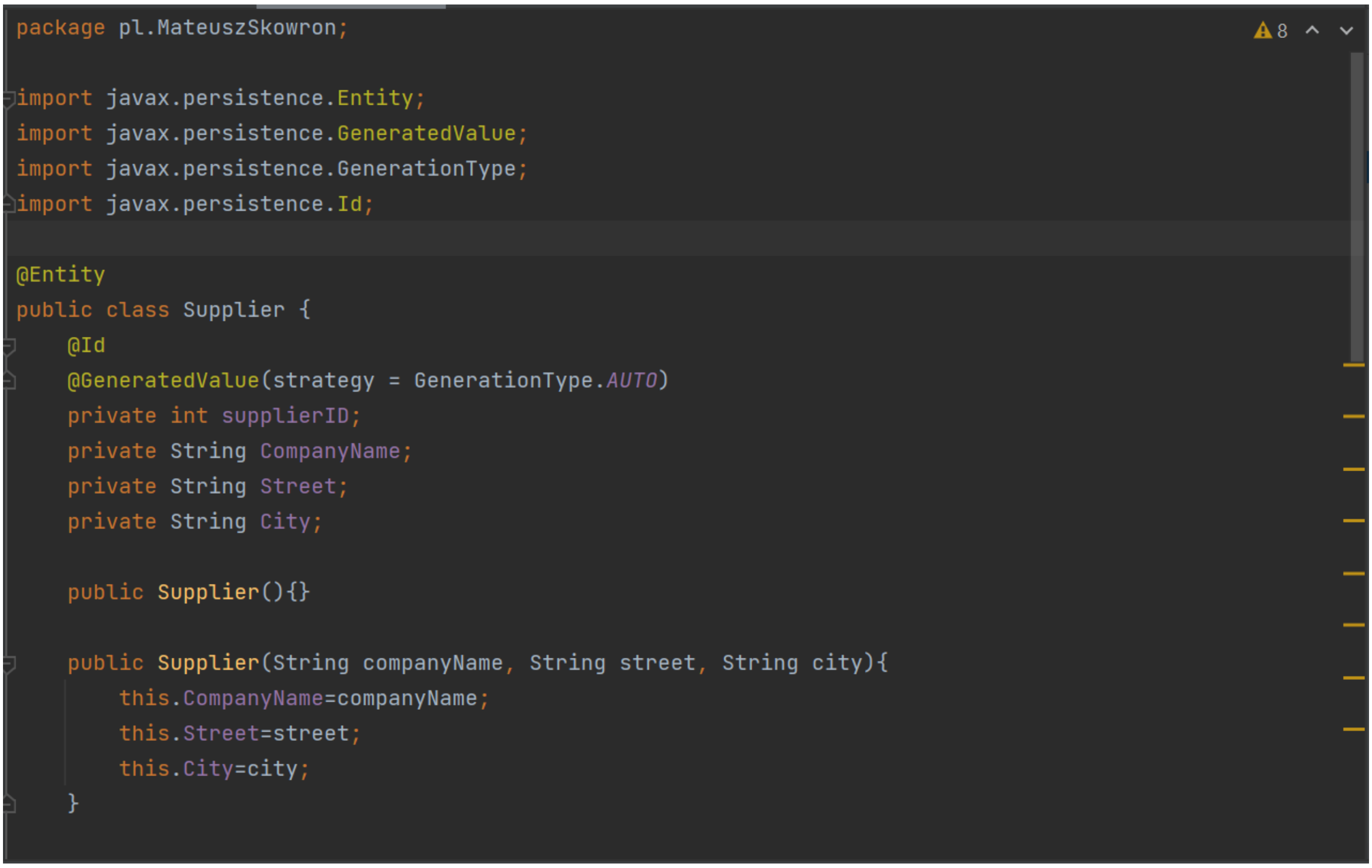


Klasa ***Product***



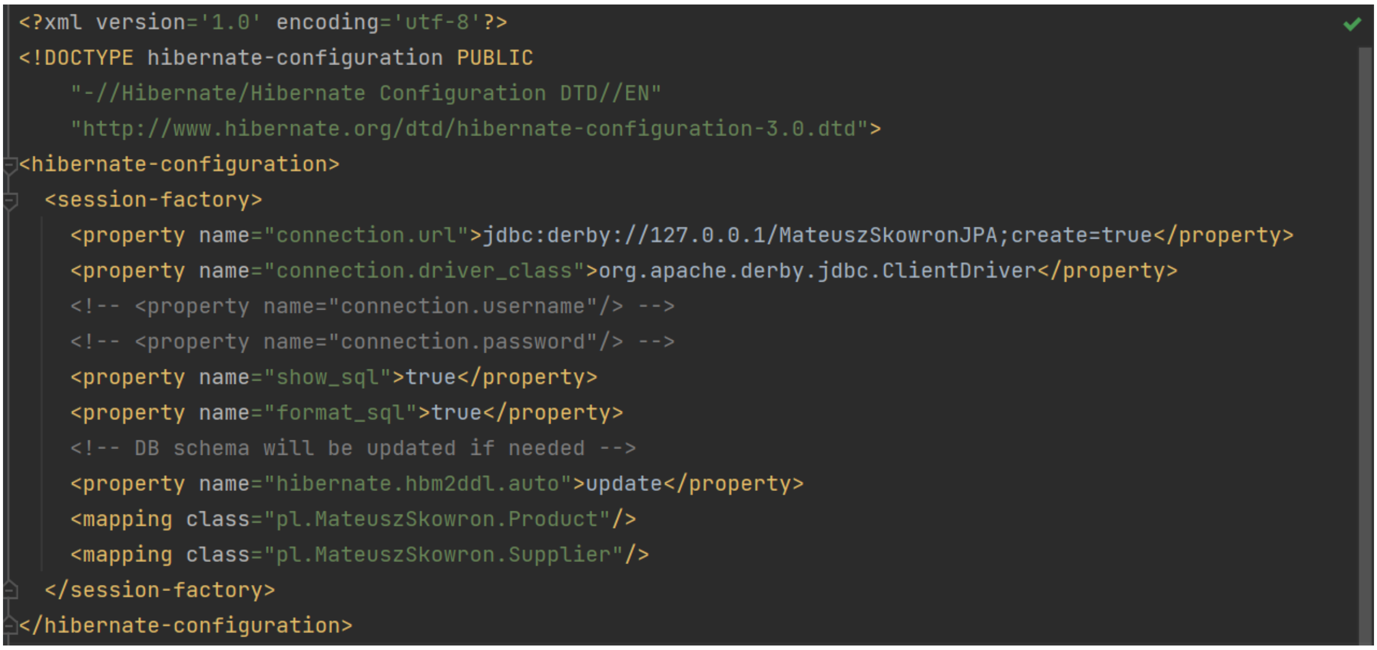
```
package pl.MateuszSkowron;

import javax.persistence.*;

@Entity
public class Product {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int productID;
    private String ProductName;
    private int UnitsOnStock;
    @ManyToOne
    private Supplier supplier;

    public Product() {}

    public Product(String productName, int unitsOnStock) {
        this.ProductName = productName;
        this.UnitsOnStock = unitsOnStock;
    }

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

Klasa *Supplier*



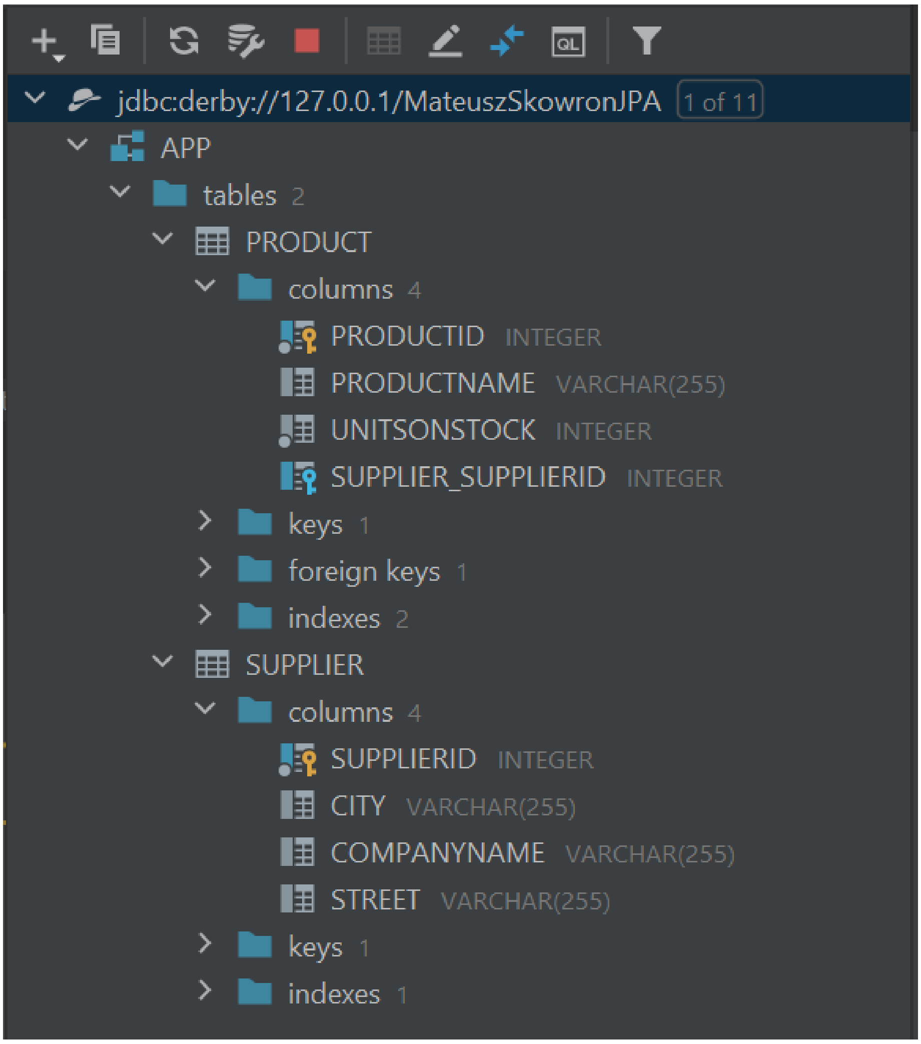
```java
package pl.MateuszSkowron;

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;
    private String CompanyName;
    private String Street;
    private String City;

    public Supplier(){}

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

Config



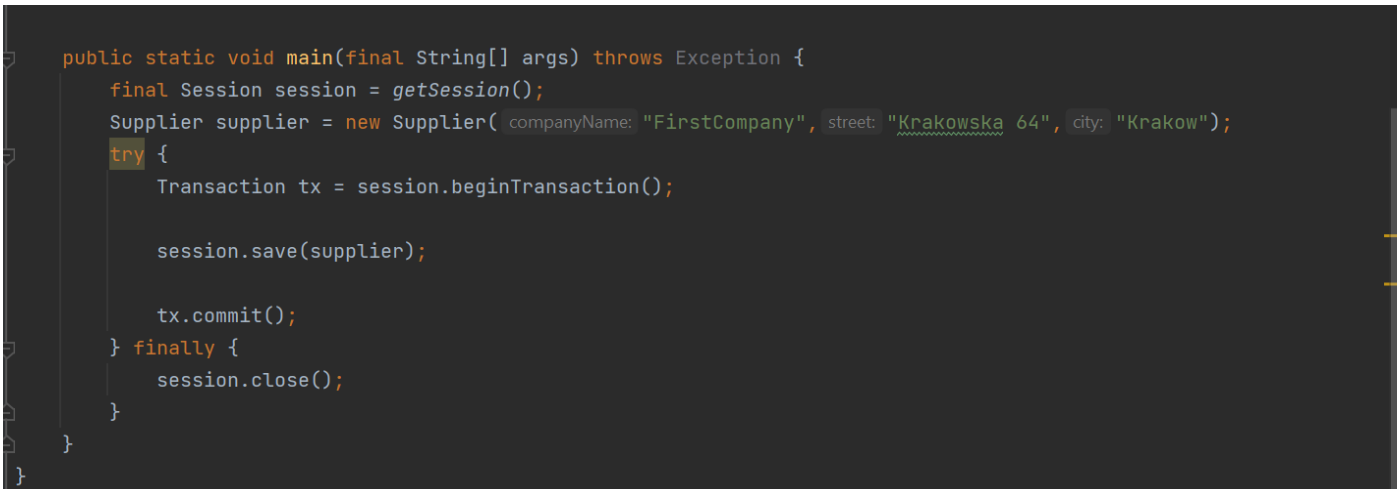
```xml
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD//EN"
    "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <property name="connection.url">jdbc:derby://127.0.0.1/MateuszSkowronJPA;create=true</property>
    <property name="connection.driver_class">org.apache.derby.jdbc.ClientDriver</property>
    <!-- <property name="connection.username"/> -->
    <!-- <property name="connection.password"/> -->
    <property name="show_sql">true</property>
    <property name="format_sql">true</property>
    <!-- DB schema will be updated if needed -->
    <property name="hibernate.hbm2ddl.auto">update</property>
    <mapping class="pl.MateuszSkowron.Product"/>
    <mapping class="pl.MateuszSkowron.Supplier"/>
  </session-factory>
</hibernate-configuration>
```

Struktura bazy danych

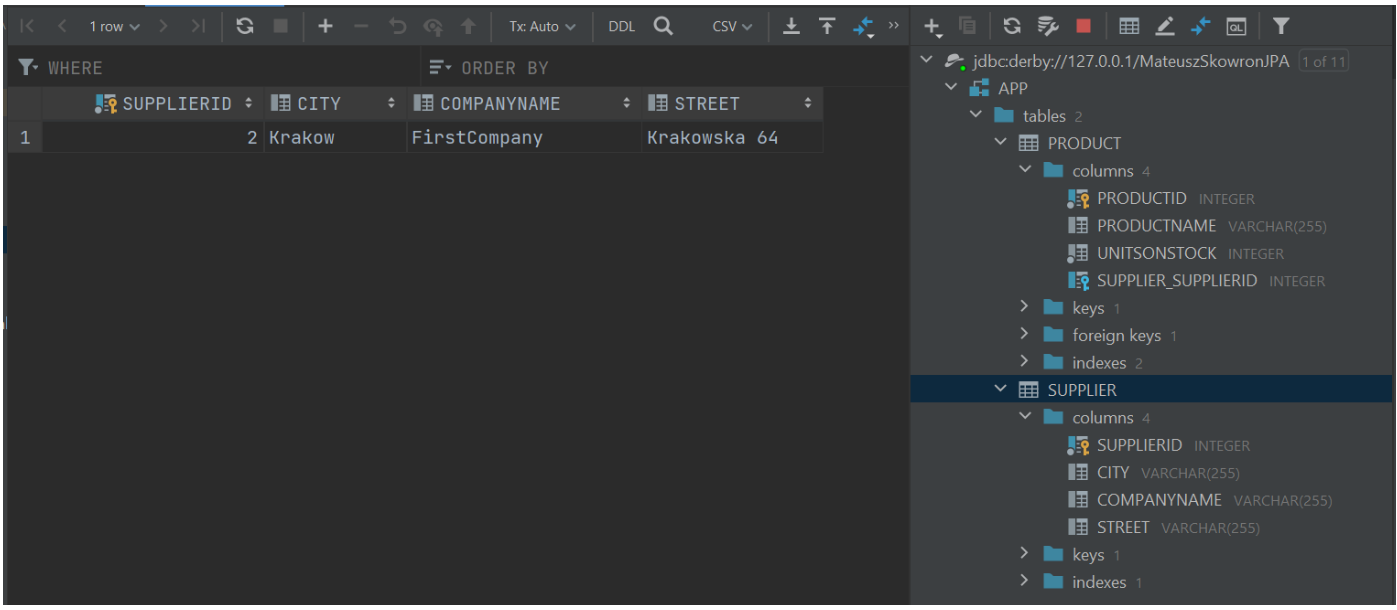


```
jdbc:derby://127.0.0.1/MateuszSkowronJPA  1 of 11
  APP
    tables 2
      PRODUCT
        columns 4
          PRODUCTID  INTEGER
          PRODUCTNAME  VARCHAR(255)
          UNITSONSTOCK  INTEGER
          SUPPLIER_SUPPLIERID  INTEGER
        keys 1
        foreign keys 1
        indexes 2
      SUPPLIER
        columns 4
          SUPPLIERID  INTEGER
          CITY  VARCHAR(255)
          COMPANYNAME  VARCHAR(255)
          STREET  VARCHAR(255)
        keys 1
        indexes 1
```

a. Stwórz nowego dostawcę

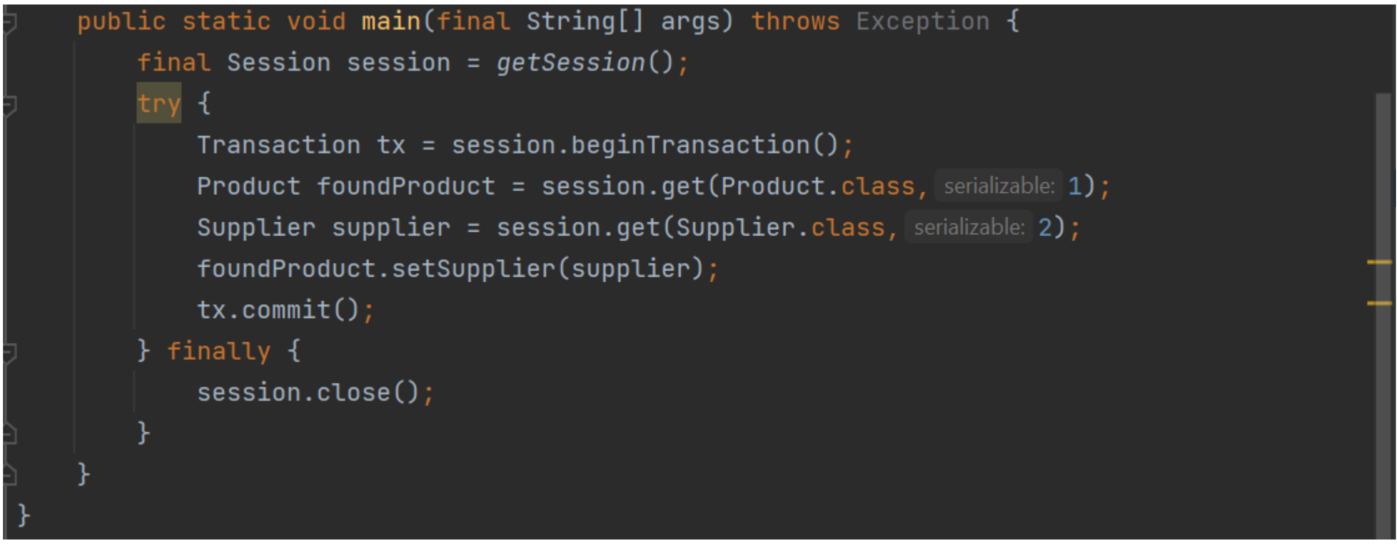


```java
    public static void main(final String[] args) throws Exception {
        final Session session = getSession();
        Supplier supplier = new Supplier( companyName: "FirstCompany", street: "Krakowska 64", city: "Krakow");
        try {
            Transaction tx = session.beginTransaction();

            session.save(supplier);

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

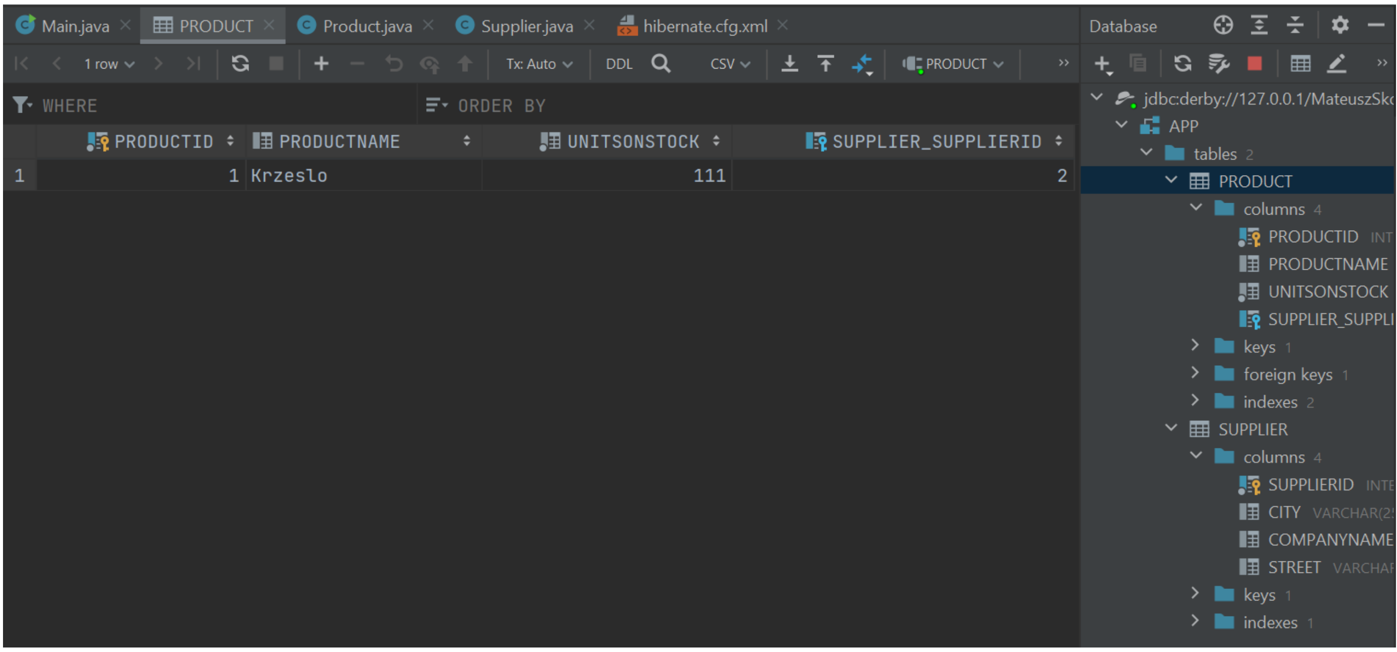


	SUPPLIERID	CITY	COMPANYNAME	STREET
1	2	Krakow	FirstCompany	Krakowska 64

b. Znajdź poprzednio wprowadzony produkt i ustaw jego dostawcę na właśnie dodanego

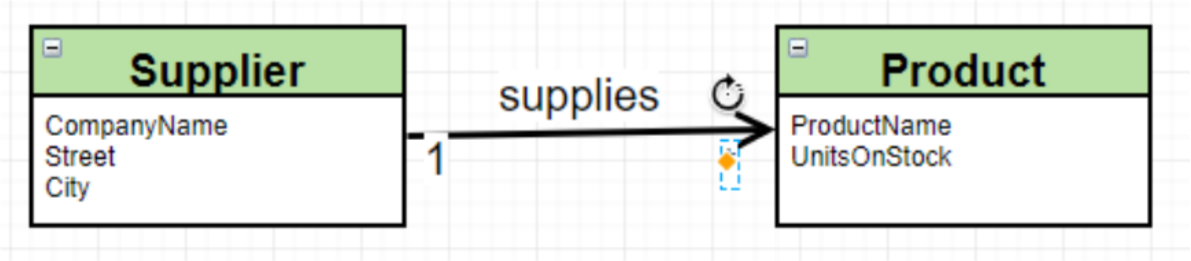


```java
    public static void main(final String[] args) throws Exception {
        final Session session = getSession();
        try {
            Transaction tx = session.beginTransaction();
            Product foundProduct = session.get(Product.class, serializable: 1);
            Supplier supplier = session.get(Supplier.class, serializable: 2);
            foundProduct.setSupplier(supplier);
            tx.commit();
        } finally {
            session.close();
        }
    }
}
```



	PRODUCTID	PRODUCTNAME	UNITSONSTOCK	SUPPLIER_SUPPLIERID
1	1	Krzeslo	111	2

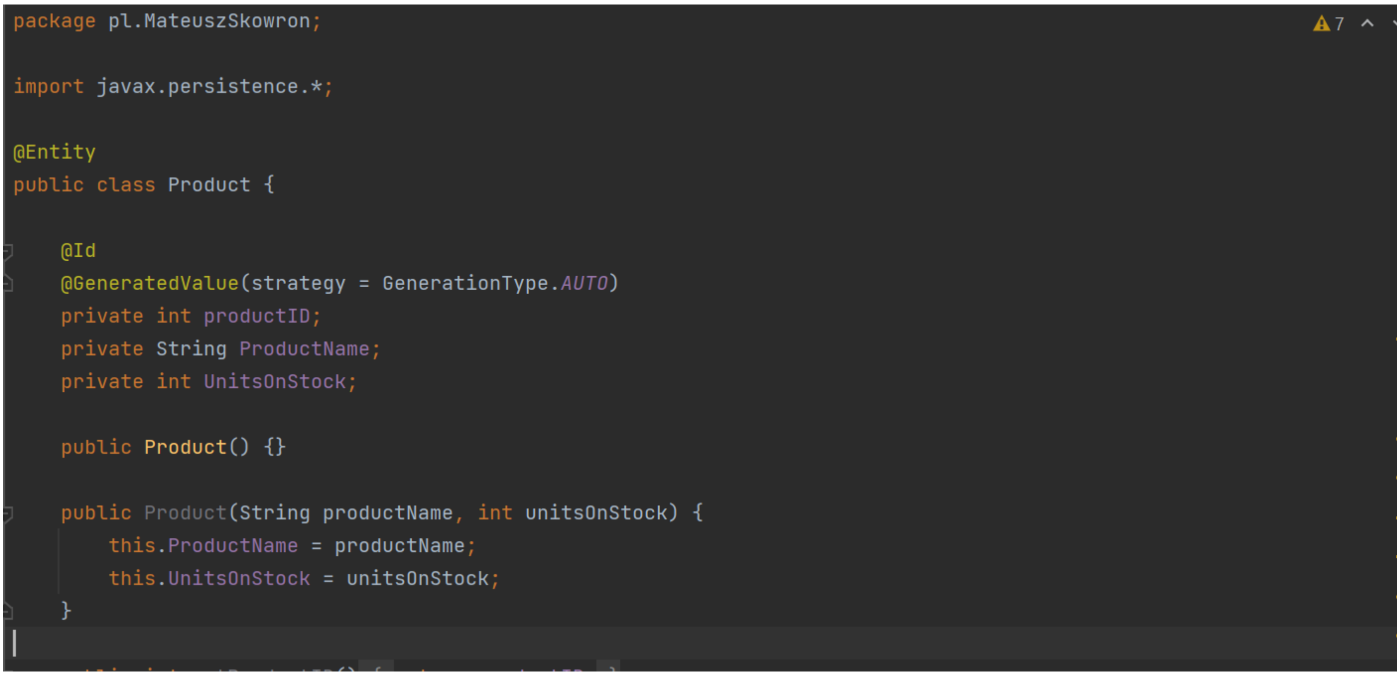
2. Odwróć relację zgodnie z poniższym schematem



   a. Zamodeluj powyższe w dwóch wariantach “z” i “bez” tabeli łącznikowej
   b. Stwórz kilka produktów
   c. Dodaj je do produktów dostarczanych przez nowo stworzonego dostawcę

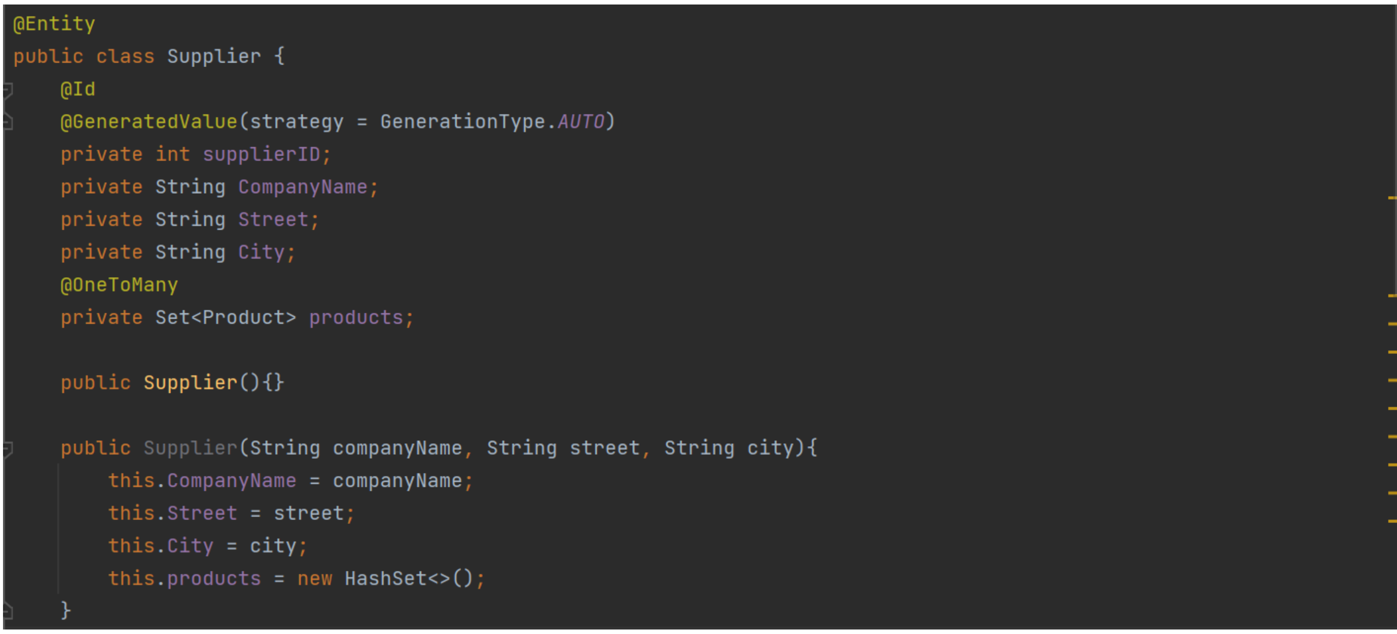
“z” tabelą łącznikową

Klasa ***Product***



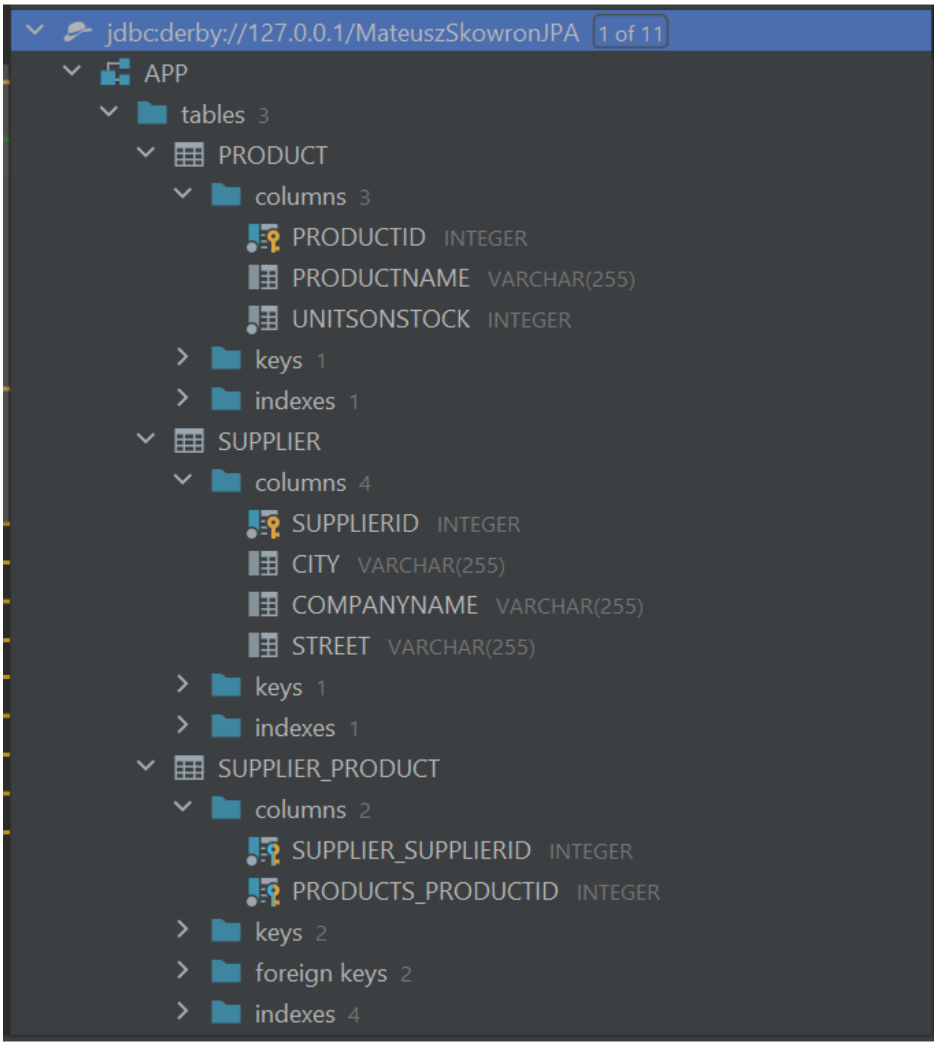
```
package pl.MateuszSkowron;

import javax.persistence.*;

@Entity
public class Product {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int productID;
    private String ProductName;
    private int UnitsOnStock;

    public Product() {}

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

Klasa ***Supplier***



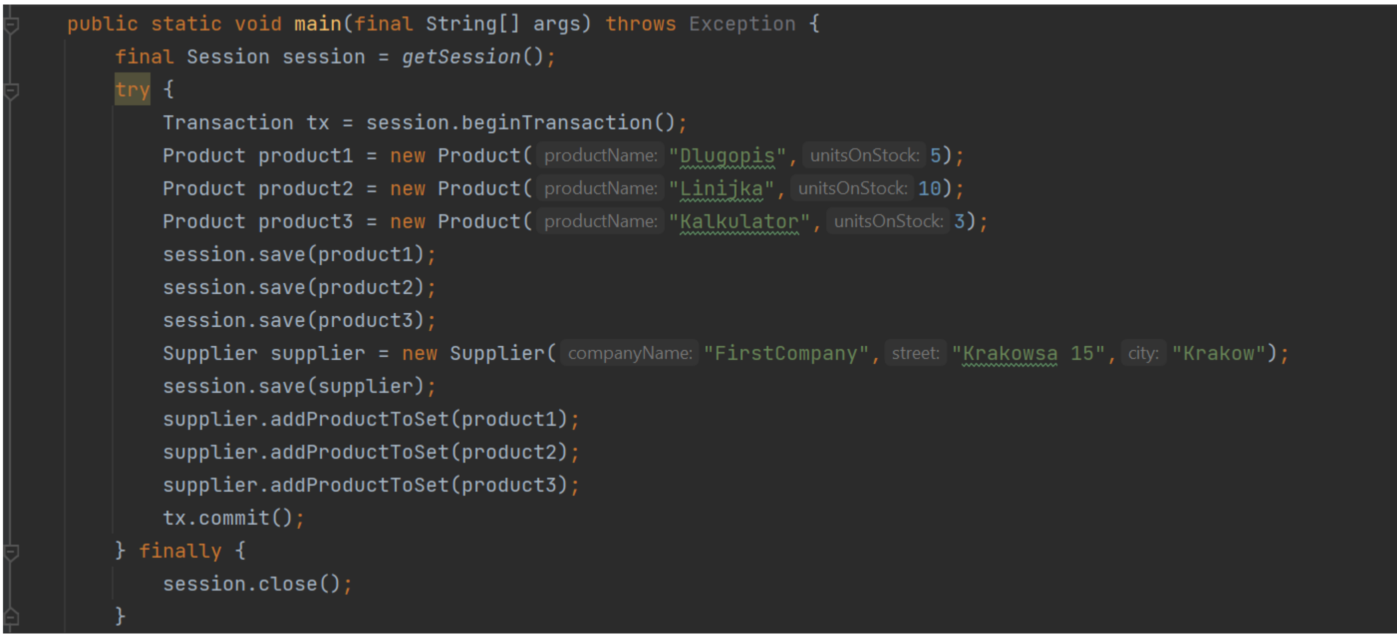
```java
@Entity
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int supplierID;
    private String CompanyName;
    private String Street;
    private String City;
    @OneToMany
    private Set<Product> products;

    public Supplier(){}

    public Supplier(String companyName, String street, String city){
        this.CompanyName = companyName;
        this.Street = street;
        this.City = city;
        this.products = new HashSet<>();
    }
```

Struktura bazy danych



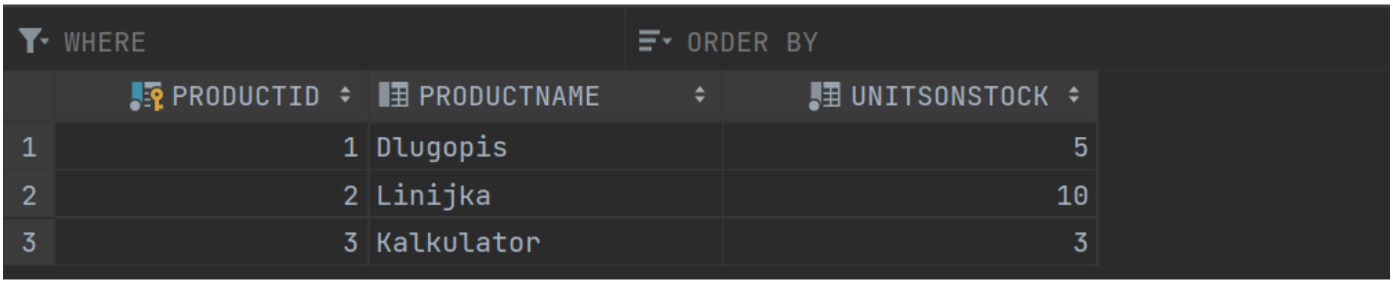
```
jdbc:derby://127.0.0.1/MateuszSkowronJPA  1 of 11
  APP
    tables 3
      PRODUCT
        columns 3
          PRODUCTID  INTEGER
          PRODUCTNAME  VARCHAR(255)
          UNITSONSTOCK  INTEGER
        keys 1
        indexes 1
      SUPPLIER
        columns 4
          SUPPLIERID  INTEGER
          CITY  VARCHAR(255)
          COMPANYNAME  VARCHAR(255)
          STREET  VARCHAR(255)
        keys 1
        indexes 1
      SUPPLIER_PRODUCT
        columns 2
          SUPPLIER_SUPPLIERID  INTEGER
          PRODUCTS_PRODUCTID  INTEGER
        keys 2
        foreign keys 2
        indexes 4
```

Wprowadzenie danych



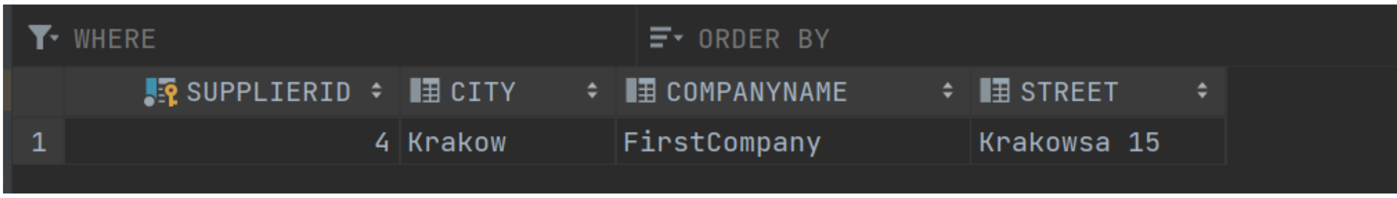
```
public static void main(final String[] args) throws Exception {
    final Session session = getSession();
    try {
        Transaction tx = session.beginTransaction();
        Product product1 = new Product( productName: "Dlugopis", unitsOnStock: 5);
        Product product2 = new Product( productName: "Linijka", unitsOnStock: 10);
        Product product3 = new Product( productName: "Kalkulator", unitsOnStock: 3);
        session.save(product1);
        session.save(product2);
        session.save(product3);
        Supplier supplier = new Supplier( companyName: "FirstCompany", street: "Krakowsa 15", city: "Krakow");
        session.save(supplier);
        supplier.addProductToSet(product1);
        supplier.addProductToSet(product2);
        supplier.addProductToSet(product3);
        tx.commit();
    } finally {
        session.close();
    }
```

Tabela ***PRODUCT***



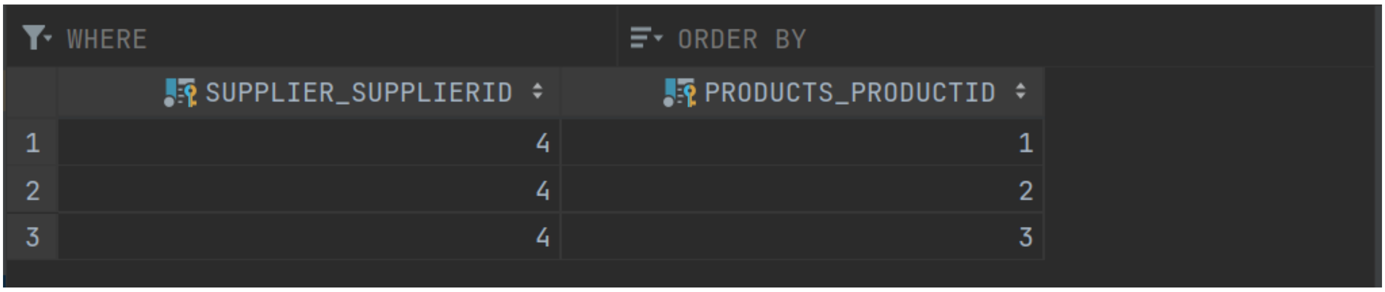
	PRODUCTID	PRODUCTNAME	UNITSONSTOCK
1	1	Dlugopis	5
2	2	Linijka	10
3	3	Kalkulator	3

Tabela ***SUPPLIER***



	SUPPLIERID	CITY	COMPANYNAME	STREET
1	4	Krakow	FirstCompany	Krakowsa 15

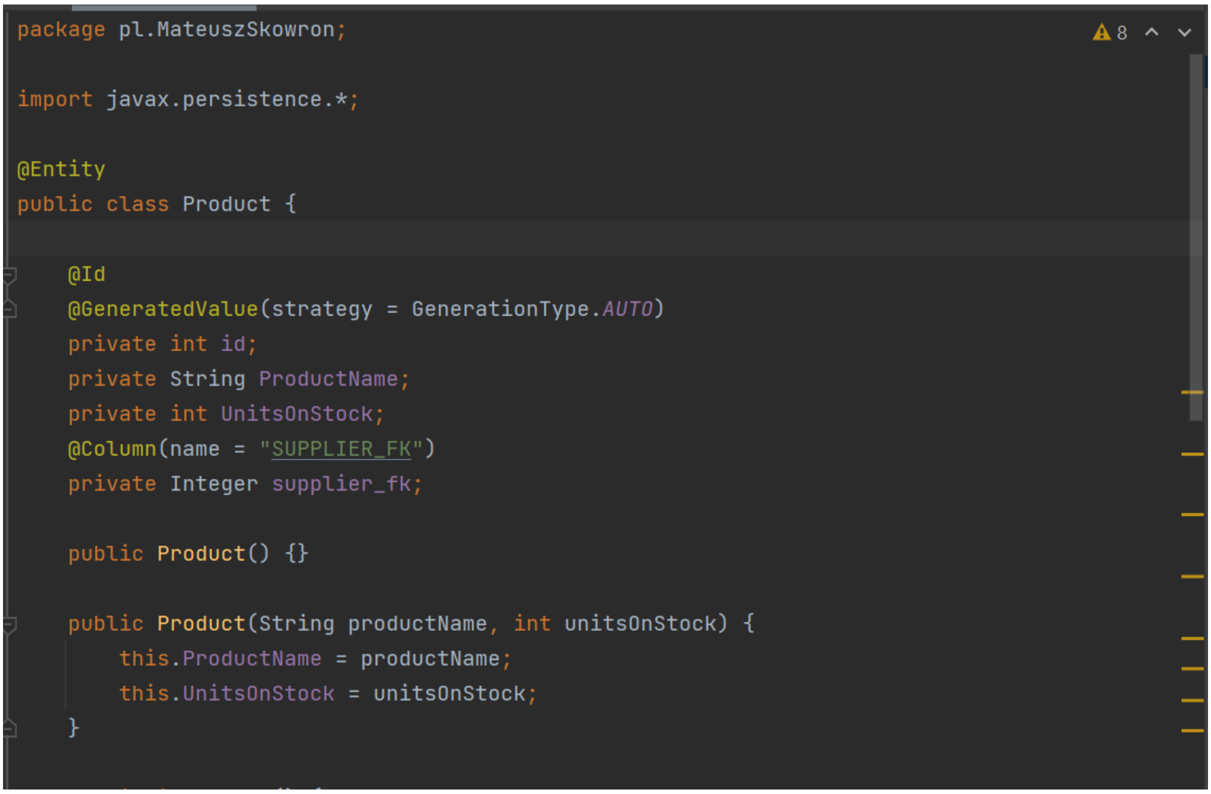
Tabela ***SUPPLIER_PRODUCT***



	SUPPLIER_SUPPLIERID	PRODUCTS_PRODUCTID
1	4	1
2	4	2
3	4	3

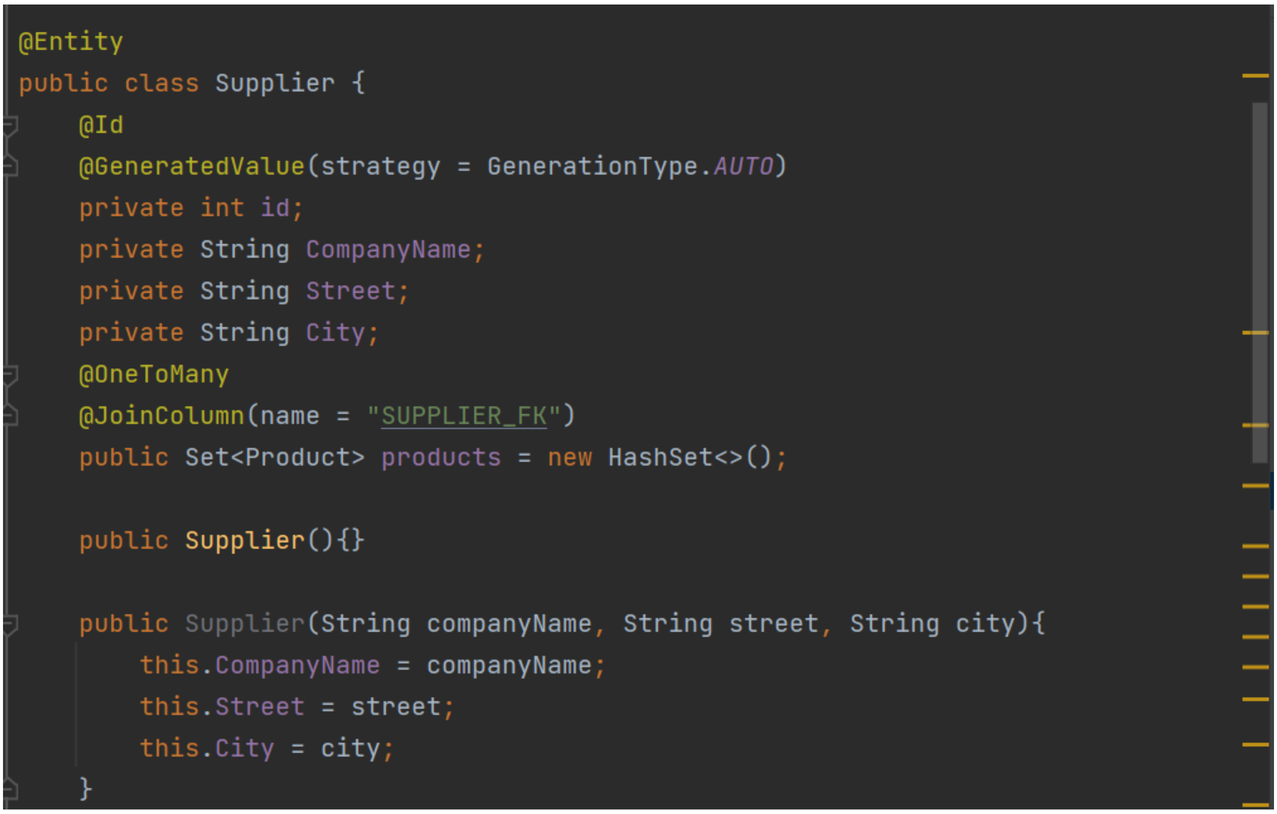
“bez” tabeli łącznikowej

Klasa ***Product***



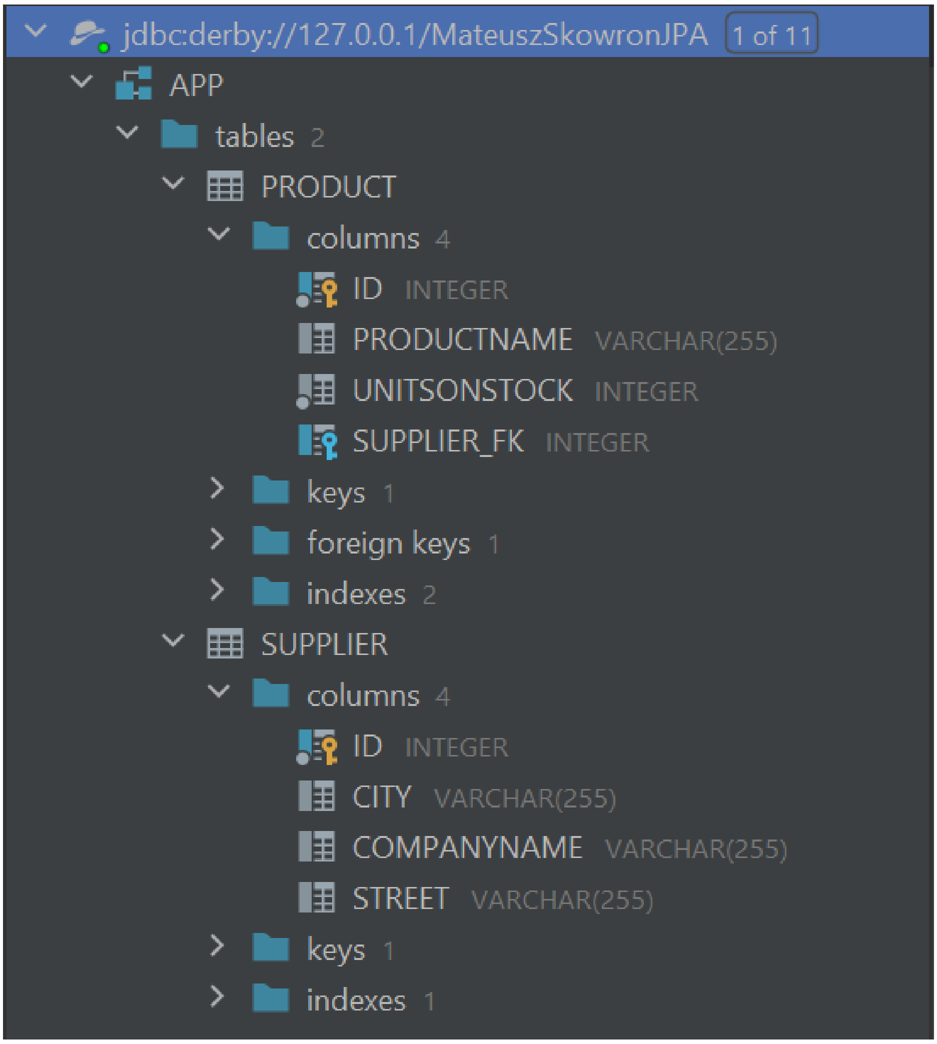
```java
package pl.MateuszSkowron;

import javax.persistence.*;

@Entity
public class Product {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;
    private String ProductName;
    private int UnitsOnStock;
    @Column(name = "SUPPLIER_FK")
    private Integer supplier_fk;

    public Product() {}

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

Klasa ***Supplier***



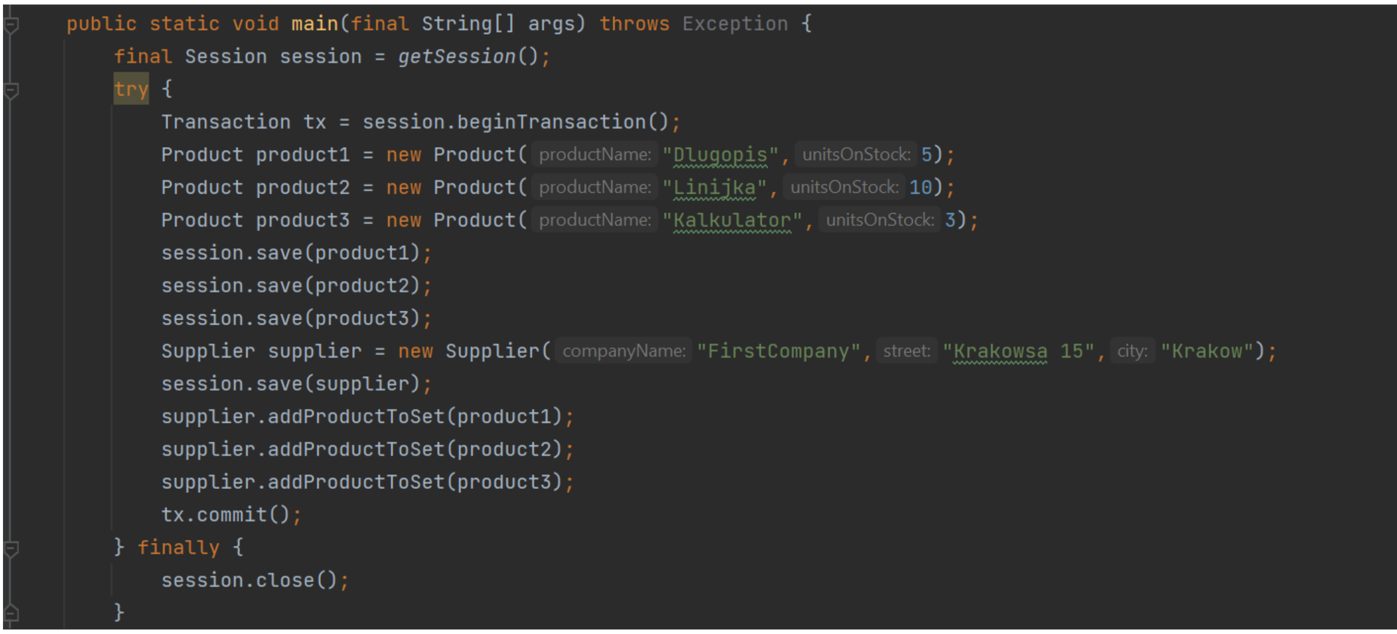
```java
@Entity
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;
    private String CompanyName;
    private String Street;
    private String City;
    @OneToMany
    @JoinColumn(name = "SUPPLIER_FK")
    public Set<Product> products = new HashSet<>();

    public Supplier(){}

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

Struktura bazy danych



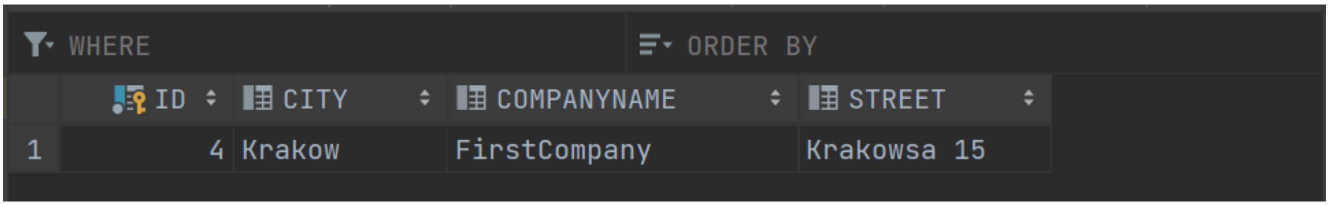
```
jdbc:derby://127.0.0.1/MateuszSkowronJPA  1 of 11
  APP
    tables 2
      PRODUCT
        columns 4
          ID  INTEGER
          PRODUCTNAME  VARCHAR(255)
          UNITSONSTOCK  INTEGER
          SUPPLIER_FK  INTEGER
        keys 1
        foreign keys 1
        indexes 2
      SUPPLIER
        columns 4
          ID  INTEGER
          CITY  VARCHAR(255)
          COMPANYNAME  VARCHAR(255)
          STREET  VARCHAR(255)
        keys 1
        indexes 1
```

Wprowadzanie danych



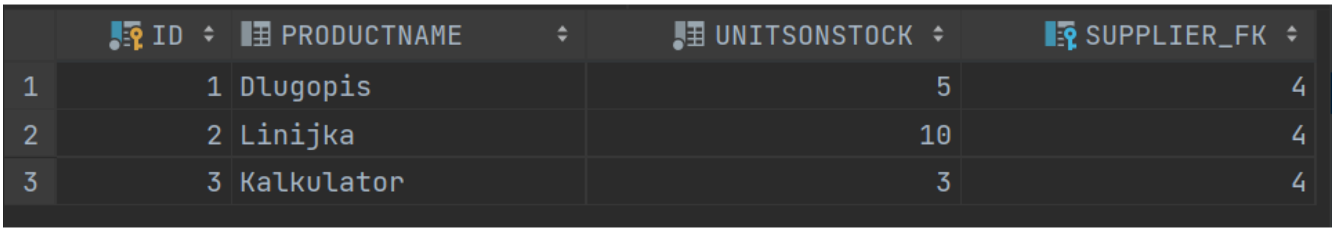
```
public static void main(final String[] args) throws Exception {
    final Session session = getSession();
    try {
        Transaction tx = session.beginTransaction();
        Product product1 = new Product( productName: "Dlugopis", unitsOnStock: 5);
        Product product2 = new Product( productName: "Linijka", unitsOnStock: 10);
        Product product3 = new Product( productName: "Kalkulator", unitsOnStock: 3);
        session.save(product1);
        session.save(product2);
        session.save(product3);
        Supplier supplier = new Supplier( companyName: "FirstCompany", street: "Krakowsa 15", city: "Krakow");
        session.save(supplier);
        supplier.addProductToSet(product1);
        supplier.addProductToSet(product2);
        supplier.addProductToSet(product3);
        tx.commit();
    } finally {
        session.close();
    }
```

Tabela ***SUPPLIER***



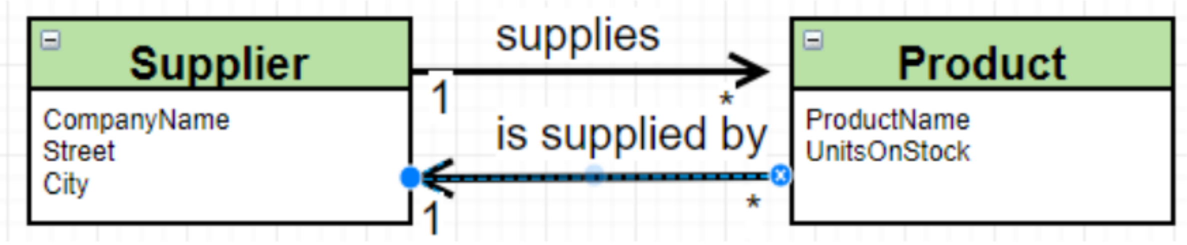
	ID	CITY	COMPANYNAME	STREET
1	4	Krakow	FirstCompany	Krakowsa 15

Table ***PRODUCT***

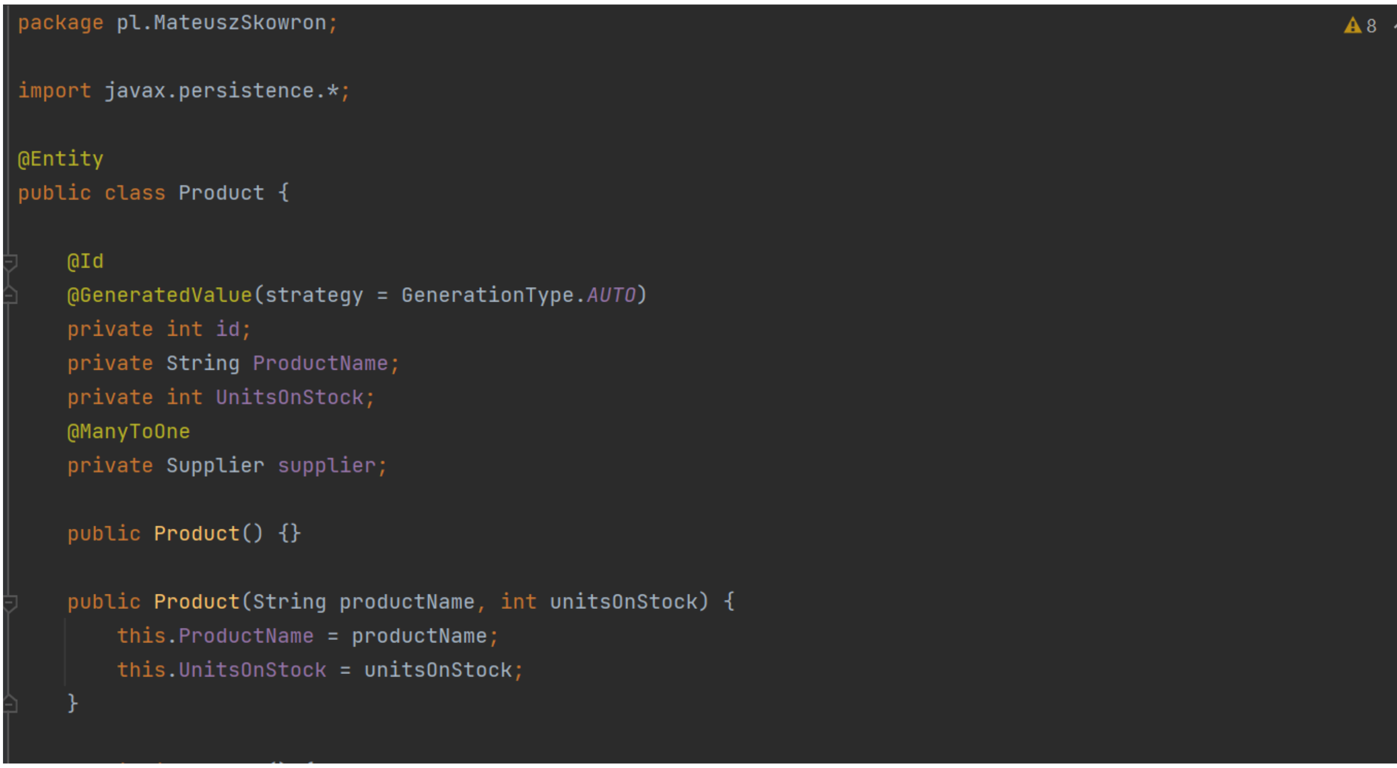


	ID	PRODUCTNAME	UNITSONSTOCK	SUPPLIER_FK
1	1	Dlugopis	5	4
2	2	Linijka	10	4
3	3	Kalkulator	3	4

3. Zamodeluj relację dwustronną jak poniżej:

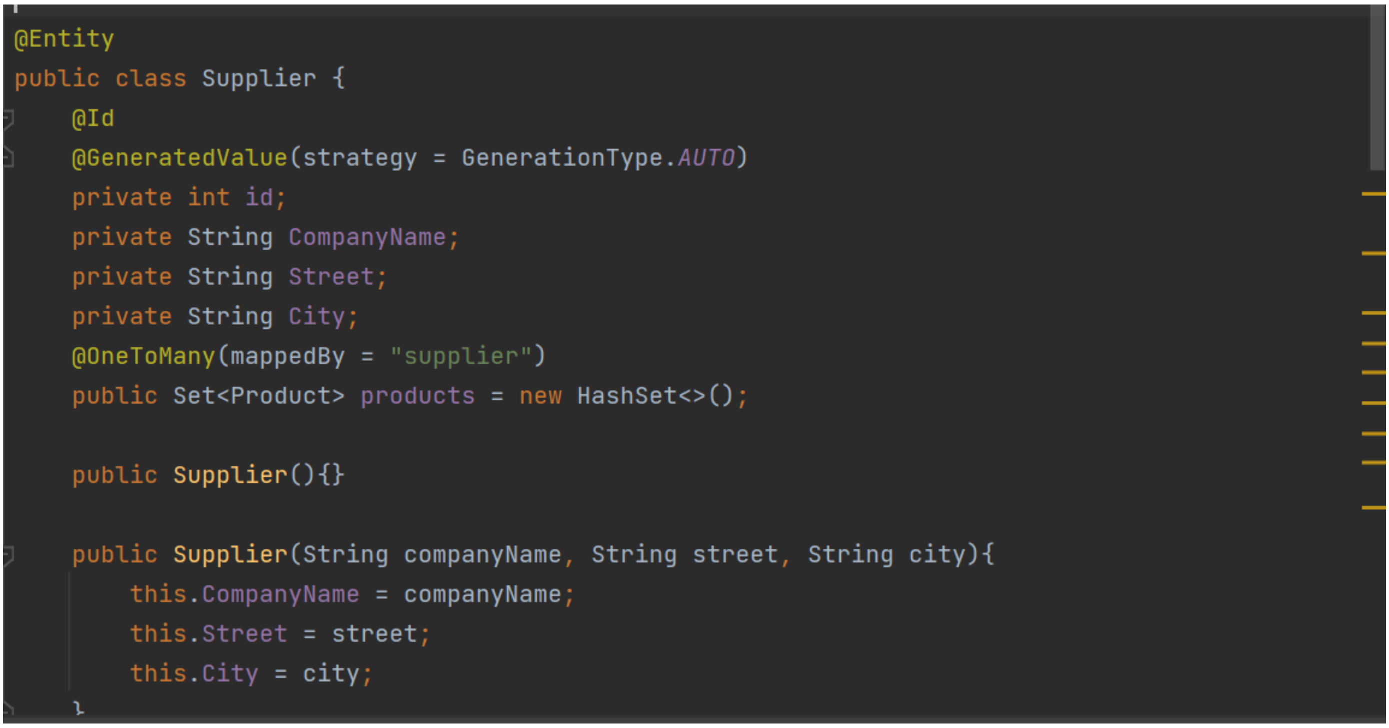


Klasa ***Product***



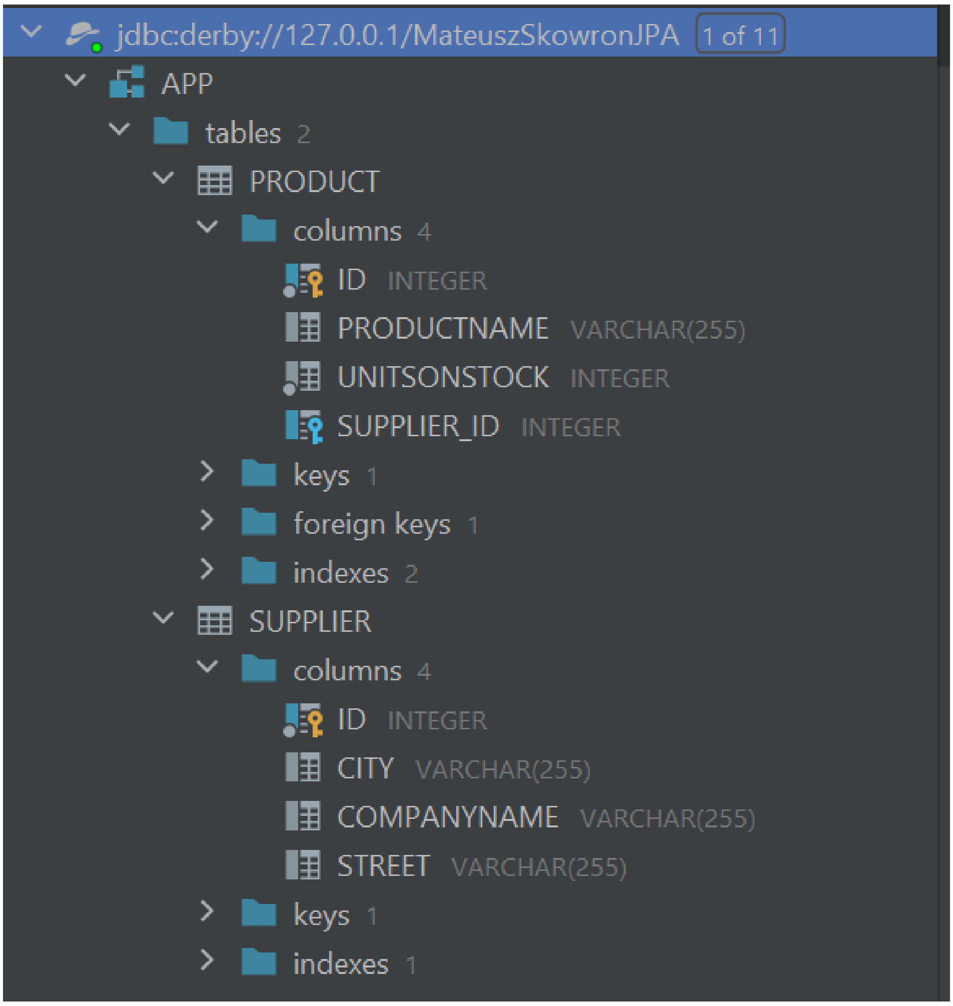
```
package pl.MateuszSkowron;

import javax.persistence.*;

@Entity
public class Product {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;
    private String ProductName;
    private int UnitsOnStock;
    @ManyToOne
    private Supplier supplier;

    public Product() {}

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

Klasa ***Supplier***



```
@Entity
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;
    private String CompanyName;
    private String Street;
    private String City;
    @OneToMany(mappedBy = "supplier")
    public Set<Product> products = new HashSet<>();

    public Supplier(){}

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

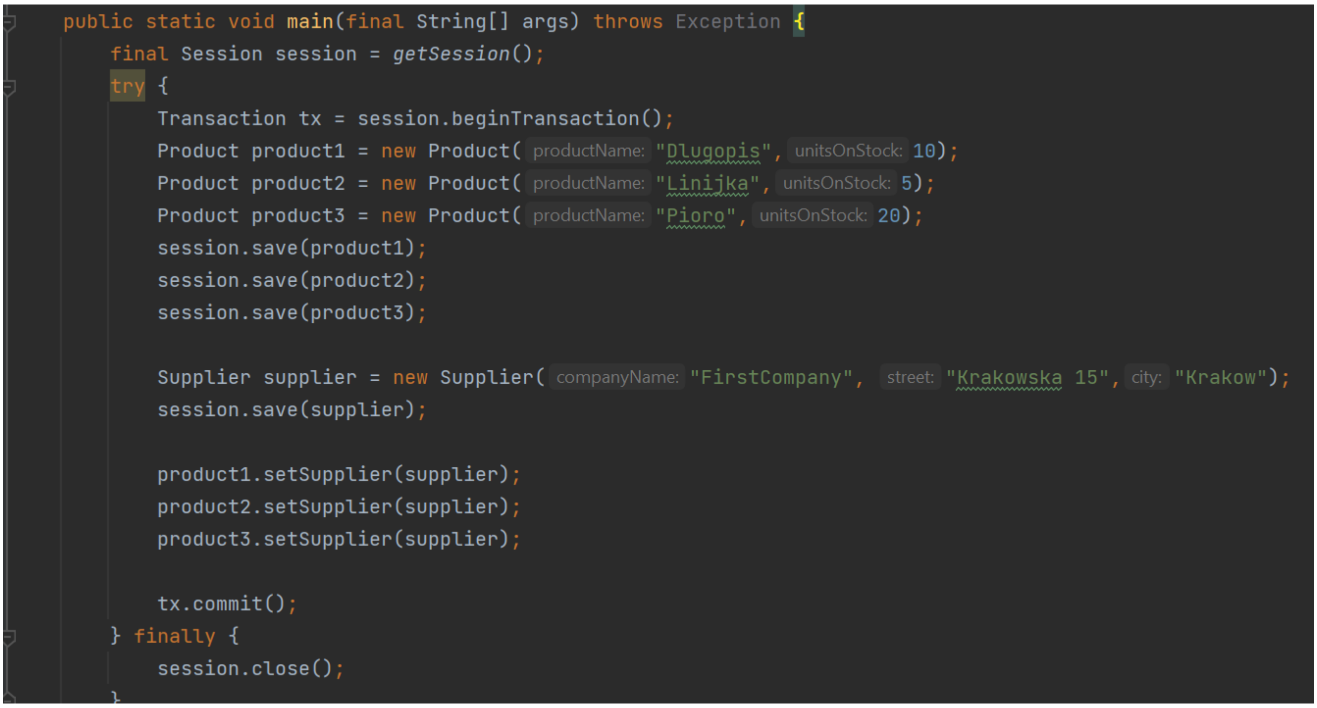
Struktura bazy danych



```
jdbc:derby://127.0.0.1/MateuszSkowronJPA  1 of 11
  APP
    tables 2
      PRODUCT
        columns 4
          ID  INTEGER
          PRODUCTNAME  VARCHAR(255)
          UNITSONSTOCK  INTEGER
          SUPPLIER_ID  INTEGER
        keys 1
        foreign keys 1
        indexes 2
      SUPPLIER
        columns 4
          ID  INTEGER
          CITY  VARCHAR(255)
          COMPANYNAME  VARCHAR(255)
          STREET  VARCHAR(255)
        keys 1
        indexes 1
```

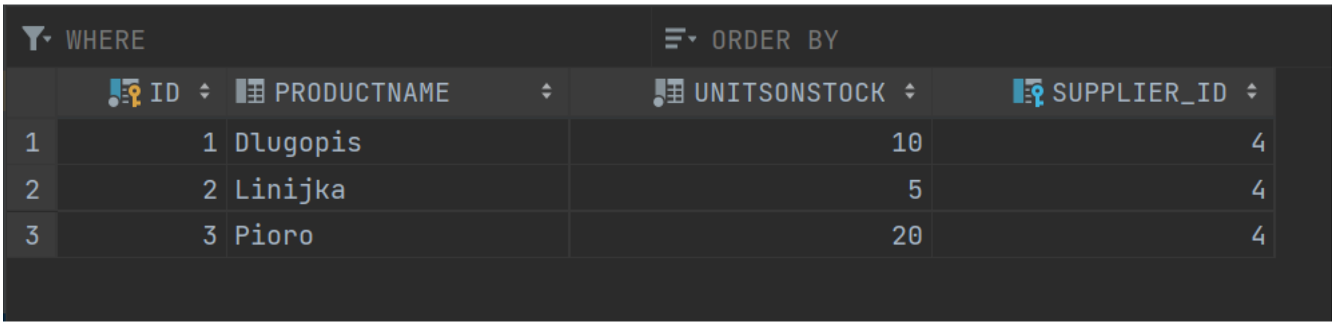
a. Stwórz kilka produktów
b. Dodaj je do produktów dostarczanych przez nowo stworzonego dostawcę (dwustronność relacji)

Wprowadzenie danych



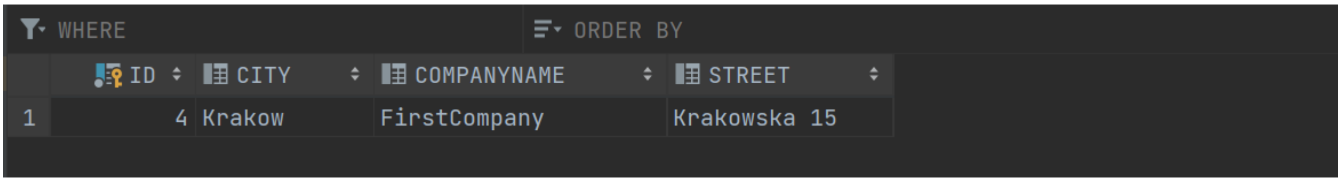
```
public static void main(final String[] args) throws Exception {
    final Session session = getSession();
    try {
        Transaction tx = session.beginTransaction();
        Product product1 = new Product( productName: "Dlugopis", unitsOnStock: 10);
        Product product2 = new Product( productName: "Linijka", unitsOnStock: 5);
        Product product3 = new Product( productName: "Pioro", unitsOnStock: 20);
        session.save(product1);
        session.save(product2);
        session.save(product3);

        Supplier supplier = new Supplier( companyName: "FirstCompany", street: "Krakowska 15", city: "Krakow");
        session.save(supplier);

        product1.setSupplier(supplier);
        product2.setSupplier(supplier);
        product3.setSupplier(supplier);

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

Tabela ***PRODUCT***



	ID	PRODUCTNAME	UNITSONSTOCK	SUPPLIER_ID
1	1	Dlugopis	10	4
2	2	Linijka	5	4
3	3	Pioro	20	4

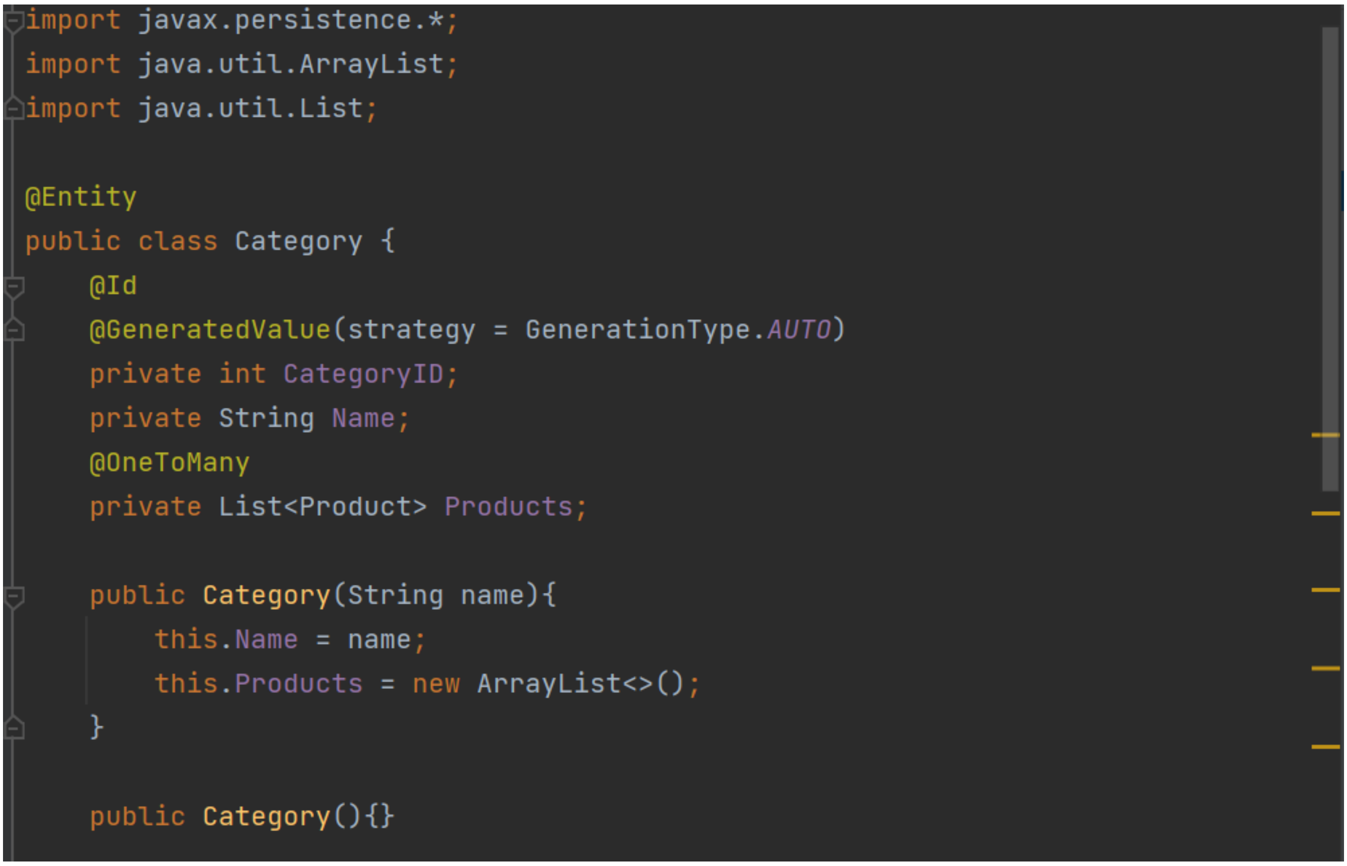
Tabela ***SUPPLIER***



	ID	CITY	COMPANYNAME	STREET
1	4	Krakow	FirstCompany	Krakowska 15

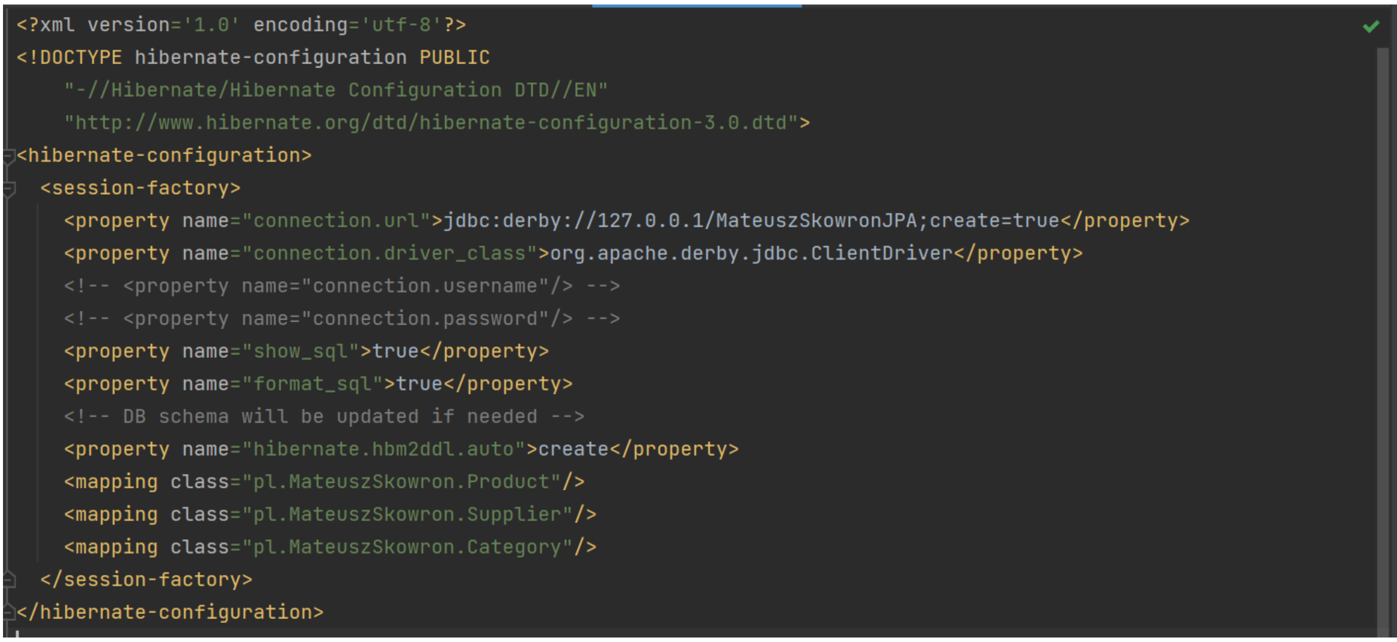
4. Dodaj klasę Category z property int CategoryID, String Name oraz listą produktów List<Product> Products

Klasa ***Category***



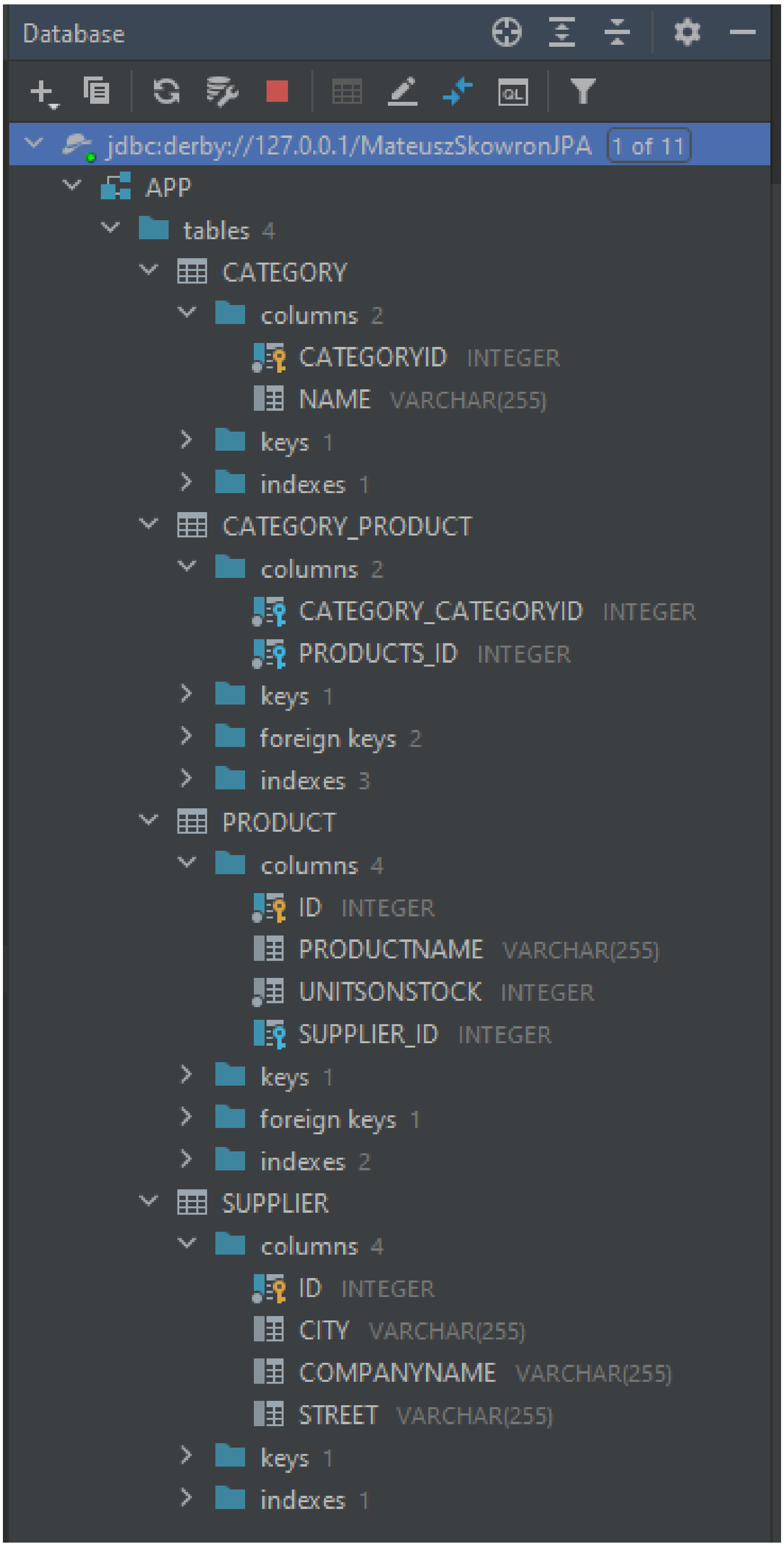
```java
import javax.persistence.*;
import java.util.ArrayList;
import java.util.List;

@Entity
public class Category {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int CategoryID;
    private String Name;
    @OneToMany
    private List<Product> Products;

    public Category(String name){
        this.Name = name;
        this.Products = new ArrayList<>();
    }

    public Category(){}
```

Config



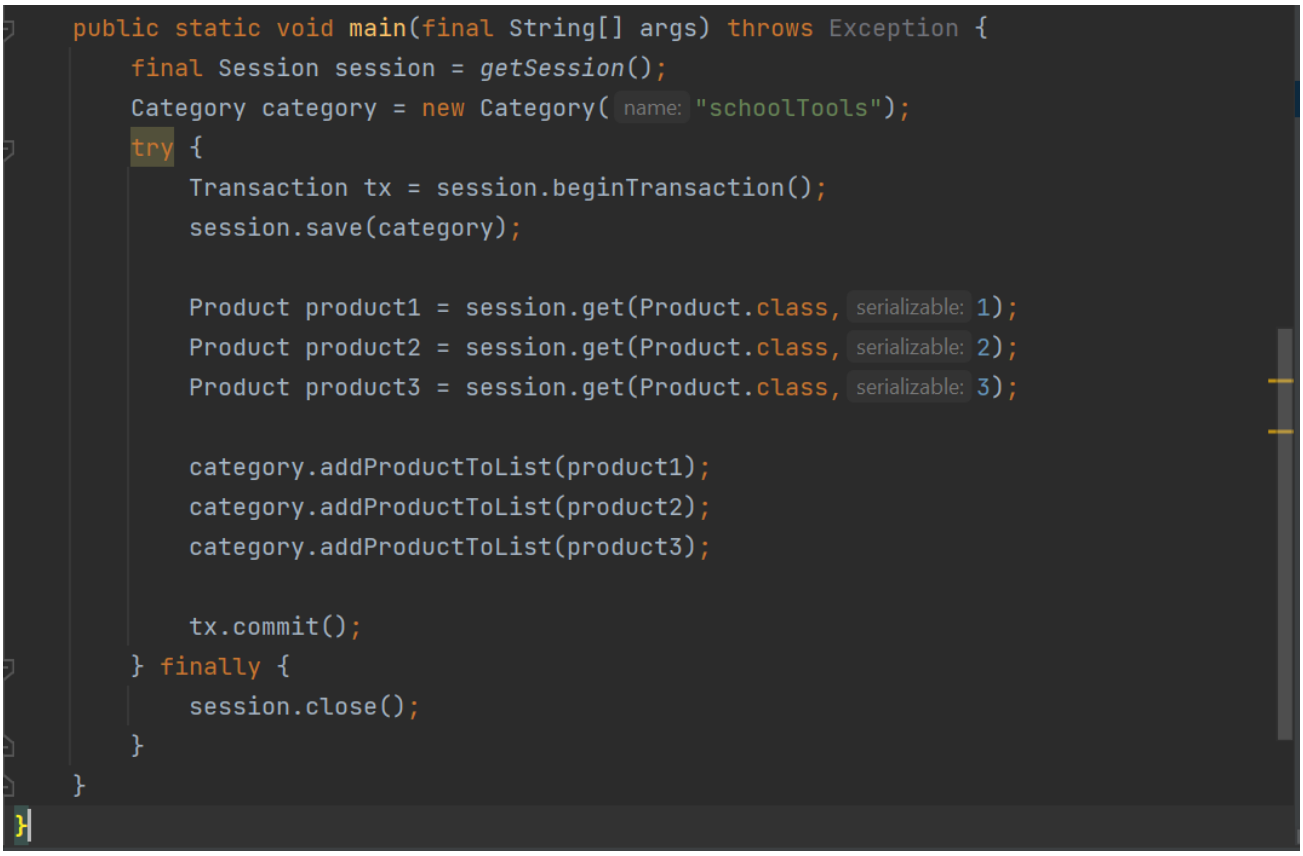
```xml
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD//EN"
    "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <property name="connection.url">jdbc:derby://127.0.0.1/MateuszSkowronJPA;create=true</property>
    <property name="connection.driver_class">org.apache.derby.jdbc.ClientDriver</property>
    <!-- <property name="connection.username"/> -->
    <!-- <property name="connection.password"/> -->
    <property name="show_sql">true</property>
    <property name="format_sql">true</property>
    <!-- DB schema will be updated if needed -->
    <property name="hibernate.hbm2ddl.auto">create</property>
    <mapping class="pl.MateuszSkowron.Product"/>
    <mapping class="pl.MateuszSkowron.Supplier"/>
    <mapping class="pl.MateuszSkowron.Category"/>
  </session-factory>
</hibernate-configuration>
```

Struktura bazy danych



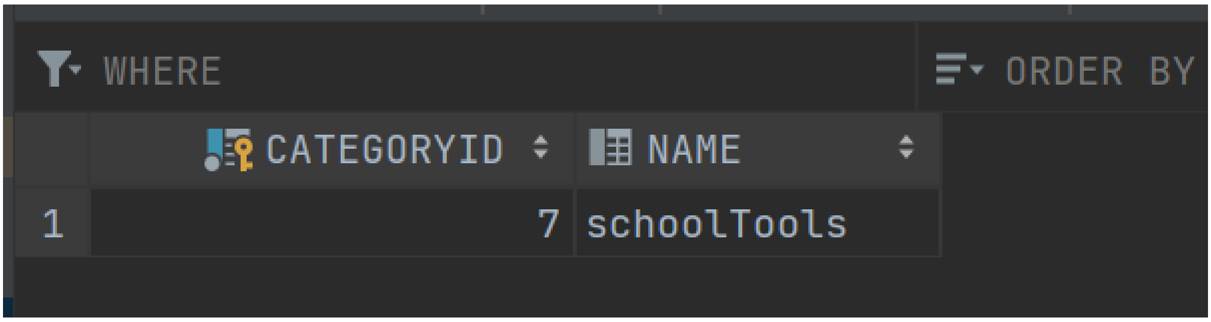
```
jdbc:derby://127.0.0.1/MateuszSkowronJPA  1 of 11
  APP
    tables 4
      CATEGORY
        columns 2
          CATEGORYID  INTEGER
          NAME  VARCHAR(255)
        keys 1
        indexes 1
      CATEGORY_PRODUCT
        columns 2
          CATEGORY_CATEGORYID  INTEGER
          PRODUCTS_ID  INTEGER
        keys 1
        foreign keys 2
        indexes 3
      PRODUCT
        columns 4
          ID  INTEGER
          PRODUCTNAME  VARCHAR(255)
          UNITSONSTOCK  INTEGER
          SUPPLIER_ID  INTEGER
        keys 1
        foreign keys 1
        indexes 2
      SUPPLIER
        columns 4
          ID  INTEGER
          CITY  VARCHAR(255)
          COMPANYNAME  VARCHAR(255)
          STREET  VARCHAR(255)
        keys 1
        indexes 1
```

a. Zmodyfikuj produkty dodając wskazanie na kategorie do której należy



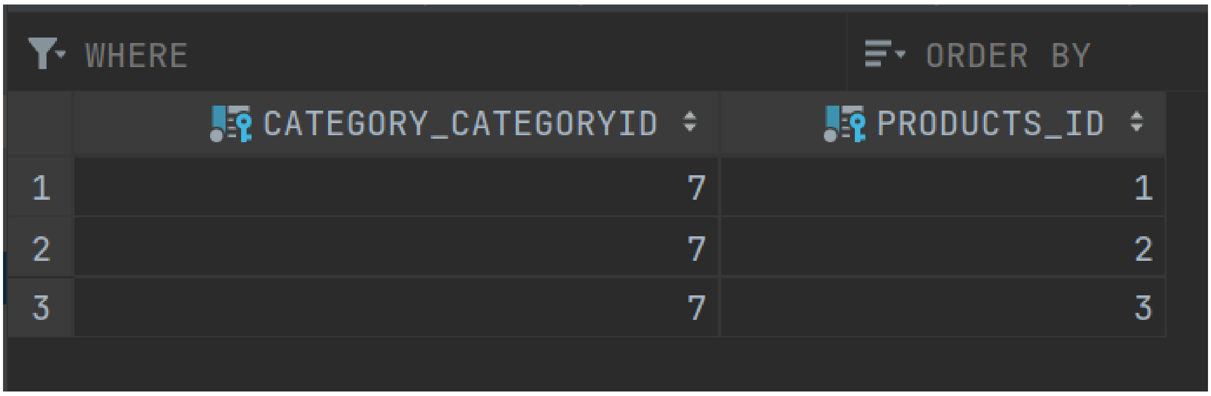
```
    public static void main(final String[] args) throws Exception {
        final Session session = getSession();
        Category category = new Category( name: "schoolTools");
        try {
            Transaction tx = session.beginTransaction();
            session.save(category);

            Product product1 = session.get(Product.class, serializable: 1);
            Product product2 = session.get(Product.class, serializable: 2);
            Product product3 = session.get(Product.class, serializable: 3);

            category.addProductToList(product1);
            category.addProductToList(product2);
            category.addProductToList(product3);

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

Tabela ***CATEGORY***



	CATEGORYID	NAME
1	7	schoolTools

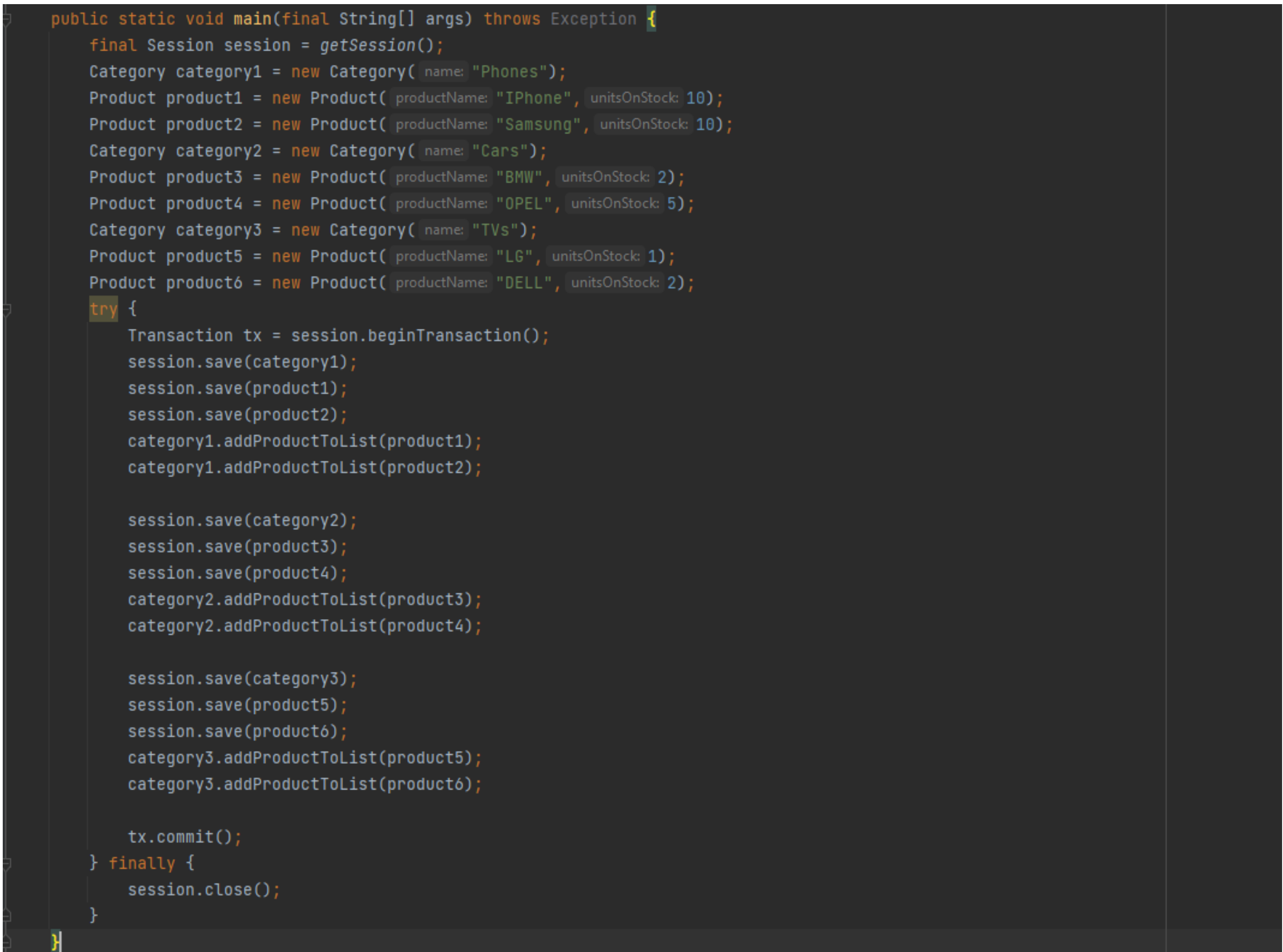
Tabela ***CATEGORY_PRODUCT***



	CATEGORY_CATEGORYID	PRODUCTS_ID
1	7	1
2	7	2
3	7	3

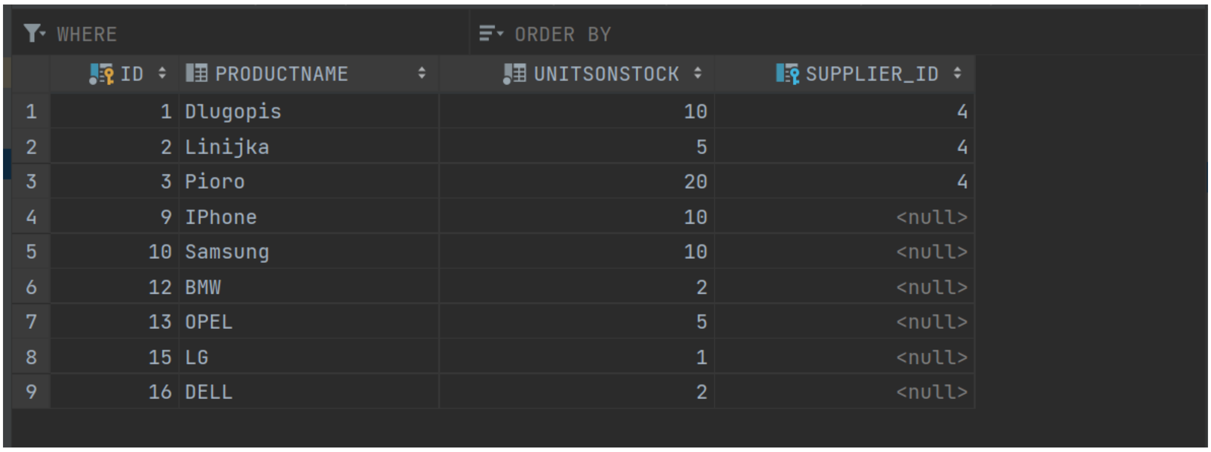
b. Stworz kilka produktow i kilka kategorii

c. Dodaj kilka produktów do wybranej kategorii



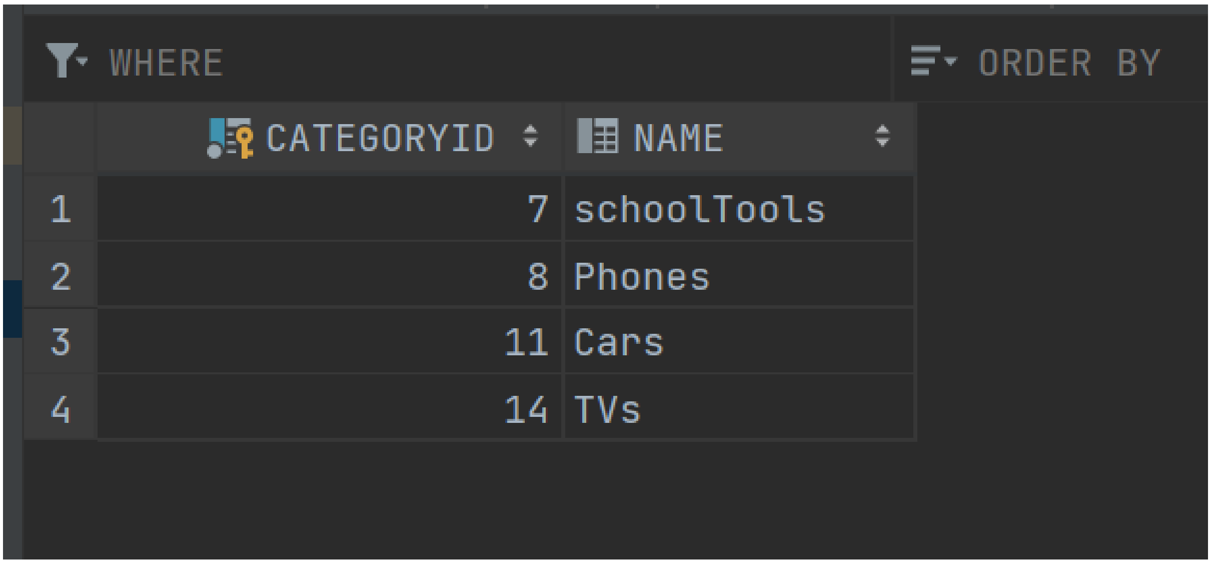
```java
public static void main(final String[] args) throws Exception {
    final Session session = getSession();
    Category category1 = new Category( name: "Phones");
    Product product1 = new Product( productName: "IPhone", unitsOnStock: 10);
    Product product2 = new Product( productName: "Samsung", unitsOnStock: 10);
    Category category2 = new Category( name: "Cars");
    Product product3 = new Product( productName: "BMW", unitsOnStock: 2);
    Product product4 = new Product( productName: "OPEL", unitsOnStock: 5);
    Category category3 = new Category( name: "TVs");
    Product product5 = new Product( productName: "LG", unitsOnStock: 1);
    Product product6 = new Product( productName: "DELL", unitsOnStock: 2);
    try {
        Transaction tx = session.beginTransaction();
        session.save(category1);
        session.save(product1);
        session.save(product2);
        category1.addProductToList(product1);
        category1.addProductToList(product2);

        session.save(category2);
        session.save(product3);
        session.save(product4);
        category2.addProductToList(product3);
        category2.addProductToList(product4);

        session.save(category3);
        session.save(product5);
        session.save(product6);
        category3.addProductToList(product5);
        category3.addProductToList(product6);

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

Tabela ***PRODUCT***



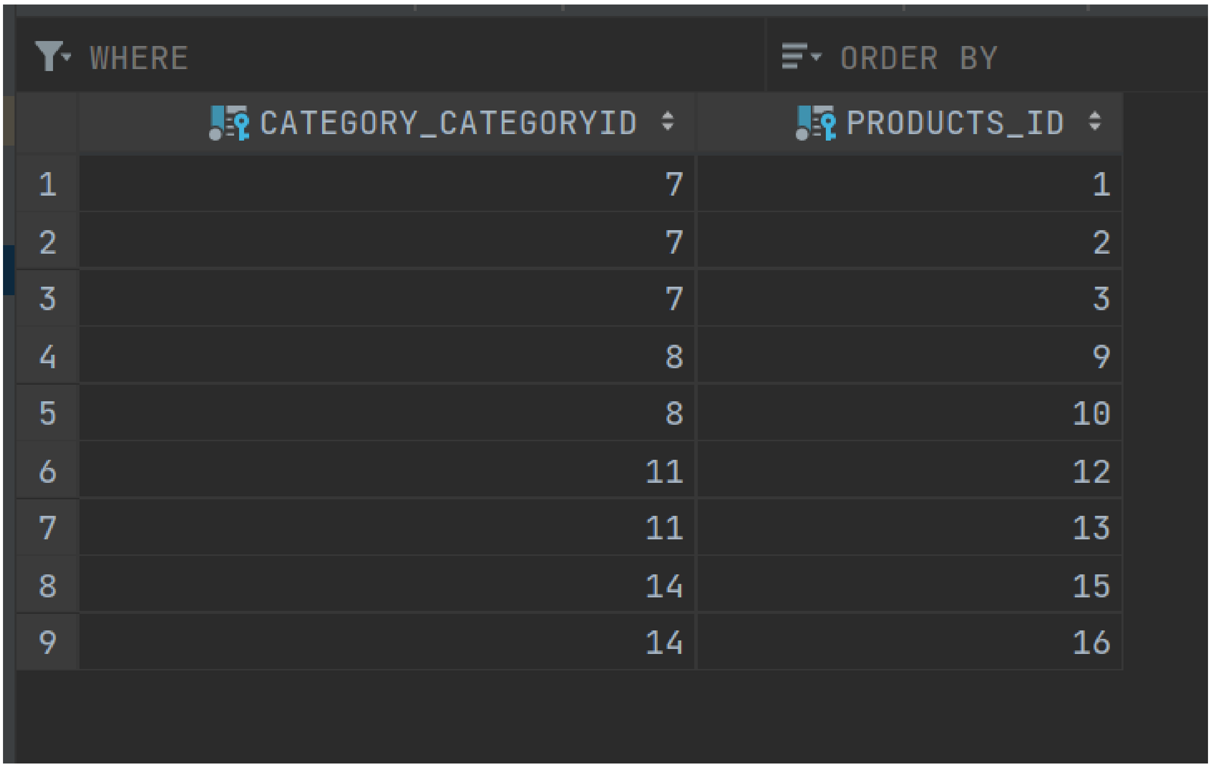
	ID	PRODUCTNAME	UNITSONSTOCK	SUPPLIER_ID
1	1	Dlugopis	10	4
2	2	Linijka	5	4
3	3	Pioro	20	4
4	9	IPhone	10	<null>
5	10	Samsung	10	<null>
6	12	BMW	2	<null>
7	13	OPEL	5	<null>
8	15	LG	1	<null>
9	16	DELL	2	<null>

Tabela ***CATEGORY***



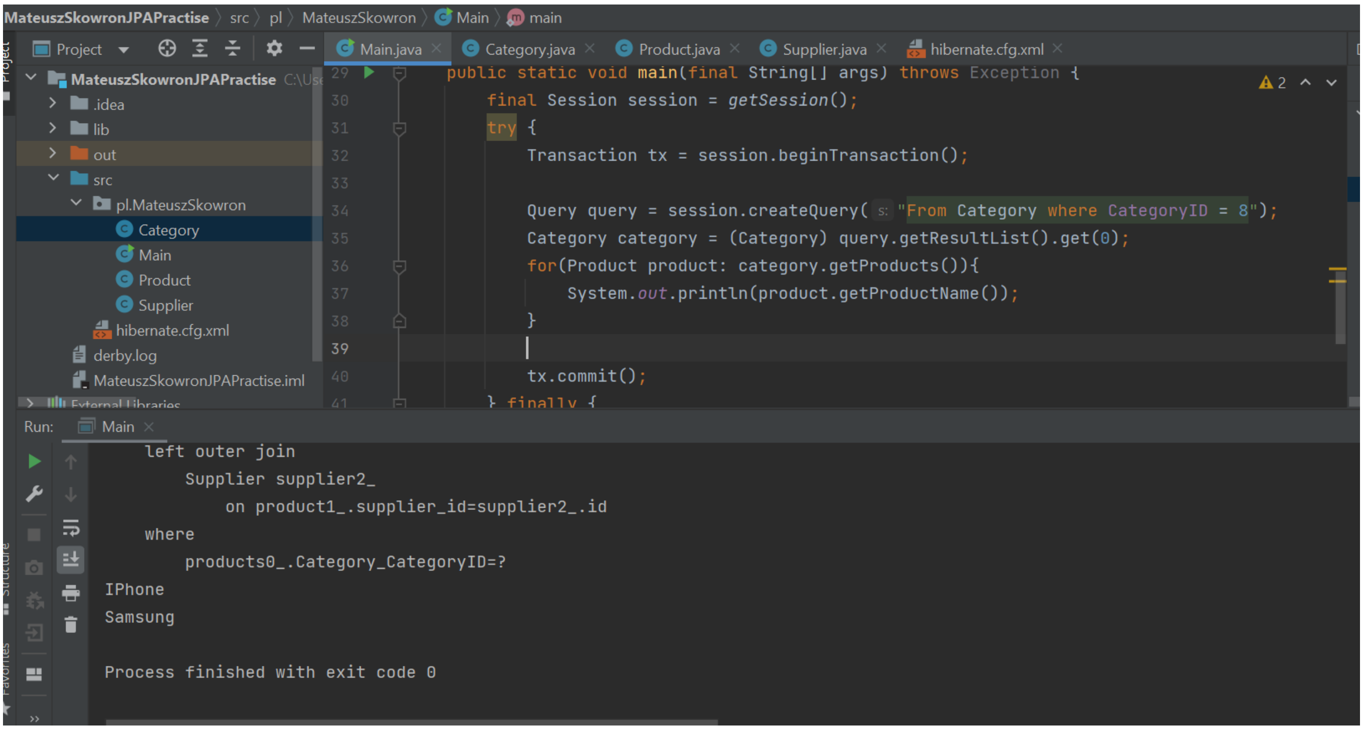
	CATEGORYID	NAME
1	7	schoolTools
2	8	Phones
3	11	Cars
4	14	TVs

Tabela ***CATEGORY_PRODUCT***



	CATEGORY_CATEGORYID	PRODUCTS_ID
1	7	1
2	7	2
3	7	3
4	8	9
5	8	10
6	11	12
7	11	13
8	14	15
9	14	16

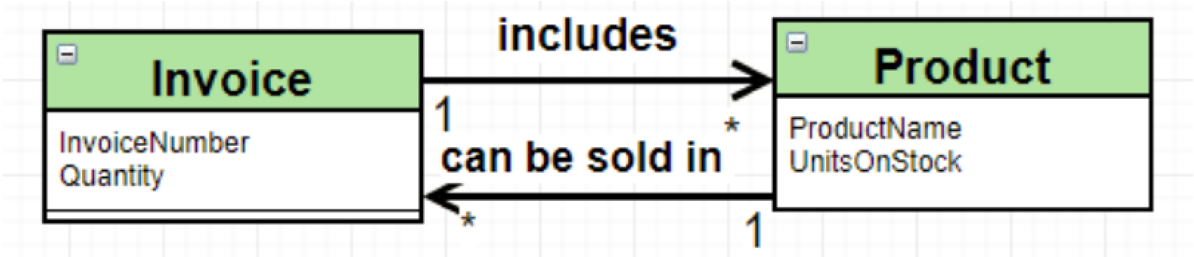
d. Wydobądź produkty z wybranej kategorii oraz kategorię do której należy wybrany produkt



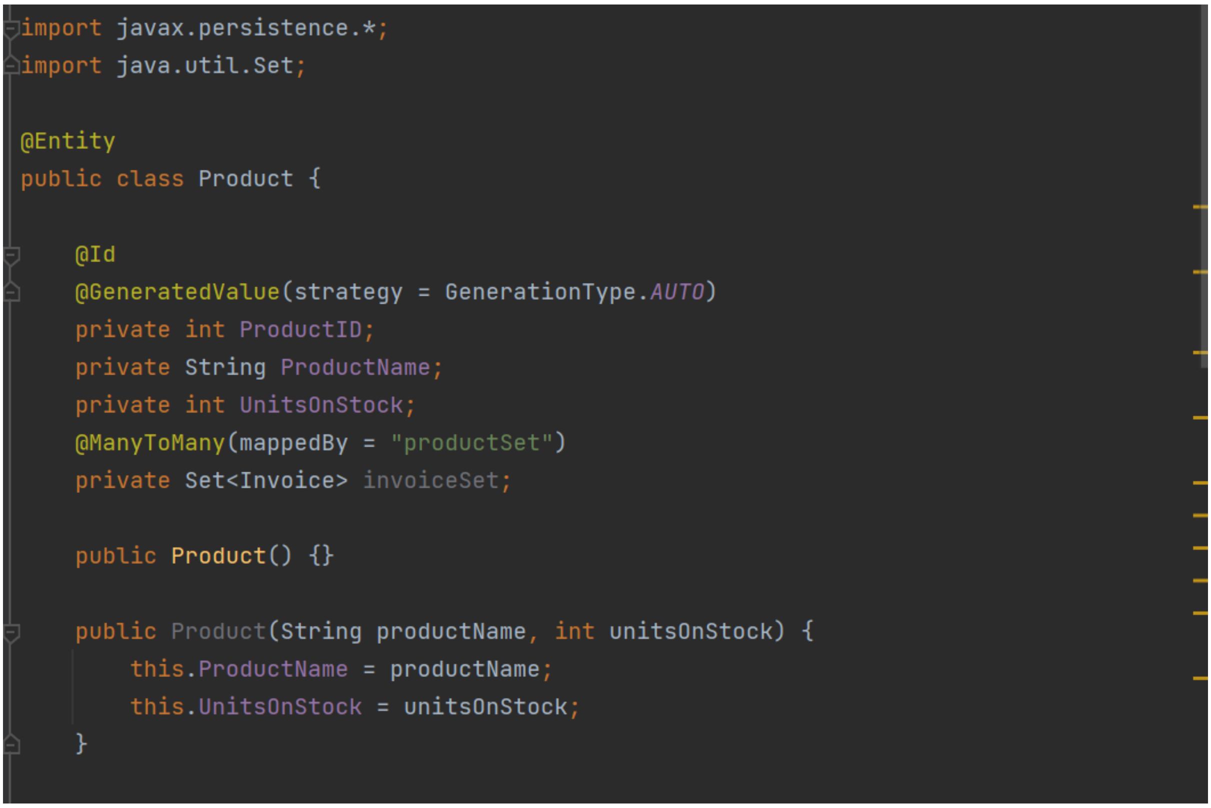
```java
public static void main(final String[] args) throws Exception {
    final Session session = getSession();
    try {
        Transaction tx = session.beginTransaction();

        Query query = session.createQuery( s: "From Category where CategoryID = 8");
        Category category = (Category) query.getResultList().get(0);
        for(Product product: category.getProducts()){
            System.out.println(product.getProductName());
        }

        tx.commit();
    } finally {
```

```
    left outer join
        Supplier supplier2_
            on product1_.supplier_id=supplier2_.id
    where
        products0_.Category_CategoryID=?
IPhone
Samsung

Process finished with exit code 0
```

5. Zamodeluj relację wiele-do-wielu jak poniżej:

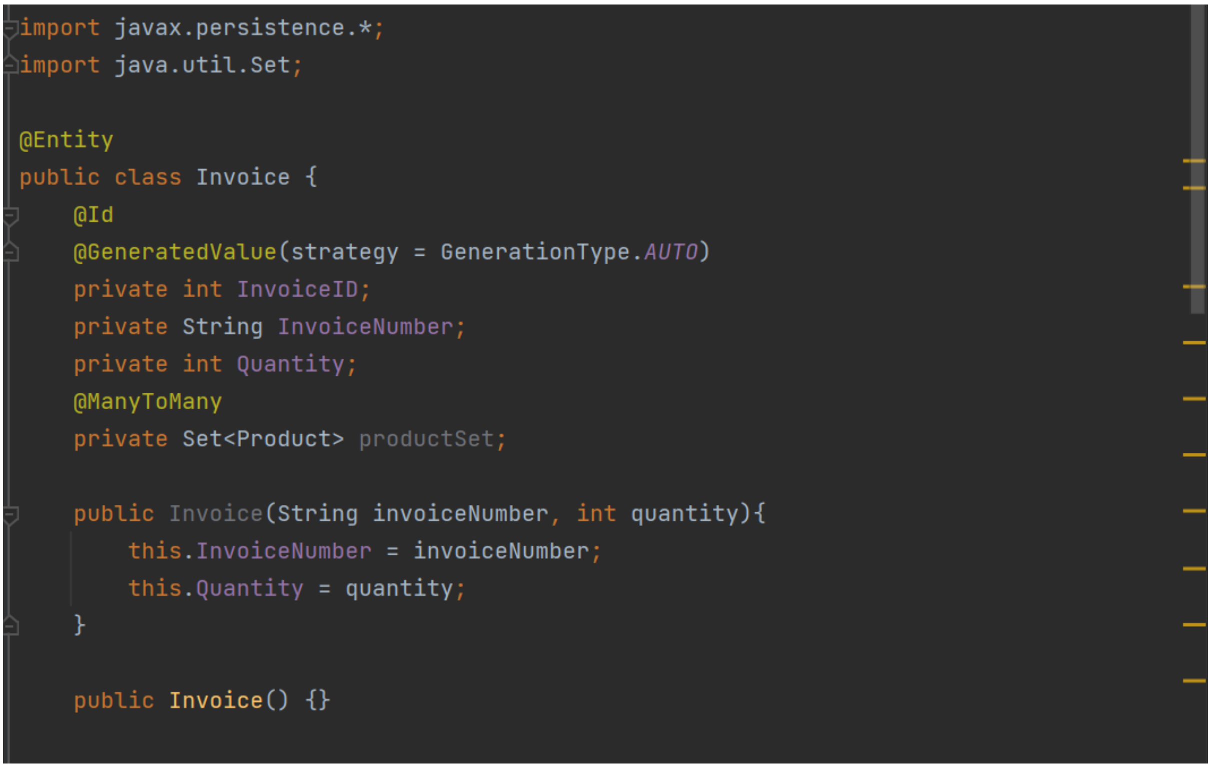


Klasa *Product*

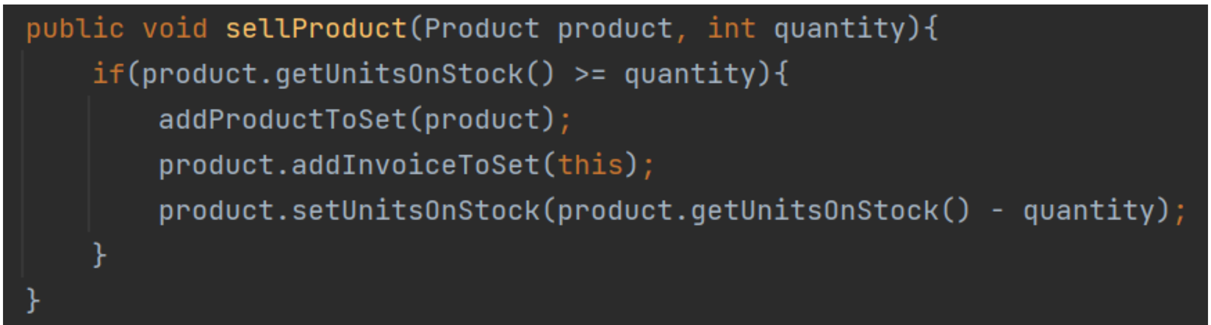


```java
import javax.persistence.*;
import java.util.Set;

@Entity
public class Product {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int ProductID;
    private String ProductName;
    private int UnitsOnStock;
    @ManyToMany(mappedBy = "productSet")
    private Set<Invoice> invoiceSet;

    public Product() {}

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

Klasa *Invoice*

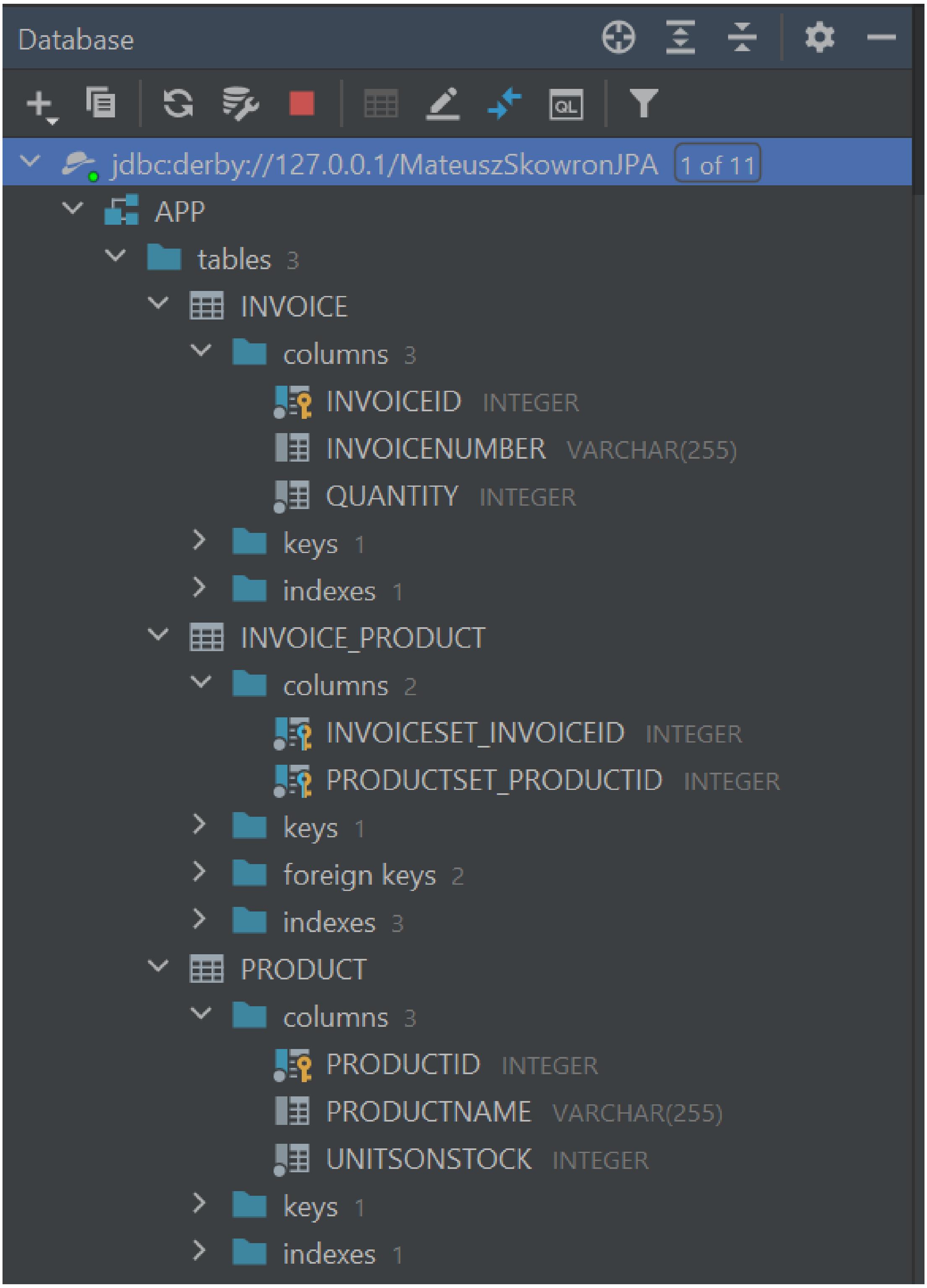


```java
import javax.persistence.*;
import java.util.Set;

@Entity
public class Invoice {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int InvoiceID;
    private String InvoiceNumber;
    private int Quantity;
    @ManyToMany
    private Set<Product> productSet;

    public Invoice(String invoiceNumber, int quantity){
        this.InvoiceNumber = invoiceNumber;
        this.Quantity = quantity;
    }

    public Invoice() {}
```



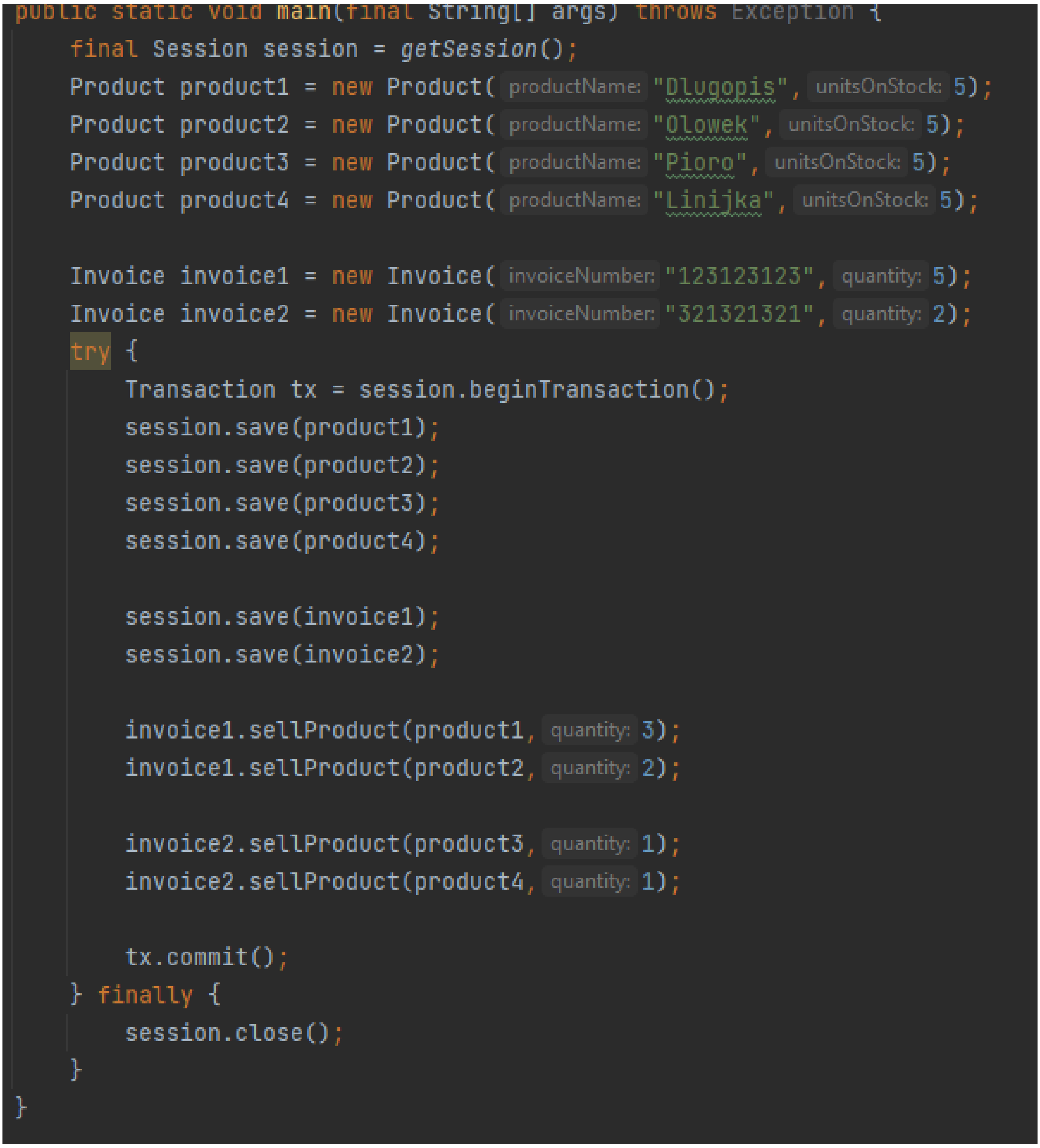
```java
public void sellProduct(Product product, int quantity){
    if(product.getUnitsOnStock() >= quantity){
        addProductToSet(product);
        product.addInvoiceToSet(this);
        product.setUnitsOnStock(product.getUnitsOnStock() - quantity);
    }
}
```

Struktura bazy danych



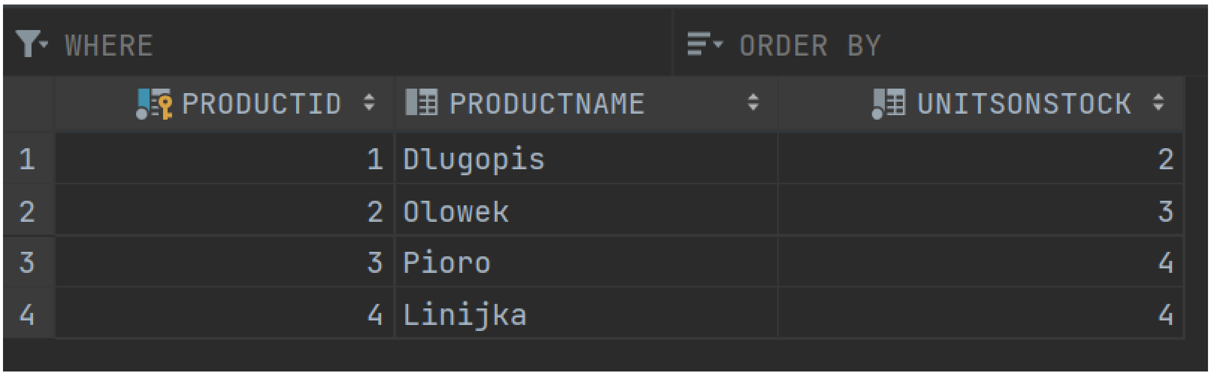
```
jdbc:derby://127.0.0.1/MateuszSkowronJPA  1 of 11
  APP
    tables 3
      INVOICE
        columns 3
          INVOICEID  INTEGER
          INVOICENUMBER  VARCHAR(255)
          QUANTITY  INTEGER
        keys 1
        indexes 1
      INVOICE_PRODUCT
        columns 2
          INVOICESET_INVOICEID  INTEGER
          PRODUCTSET_PRODUCTID  INTEGER
        keys 1
        foreign keys 2
        indexes 3
      PRODUCT
        columns 3
          PRODUCTID  INTEGER
          PRODUCTNAME  VARCHAR(255)
          UNITSONSTOCK  INTEGER
        keys 1
        indexes 1
```

a. Stwórz kilka produktów I “sprzedaj” je na kilku transakcjach



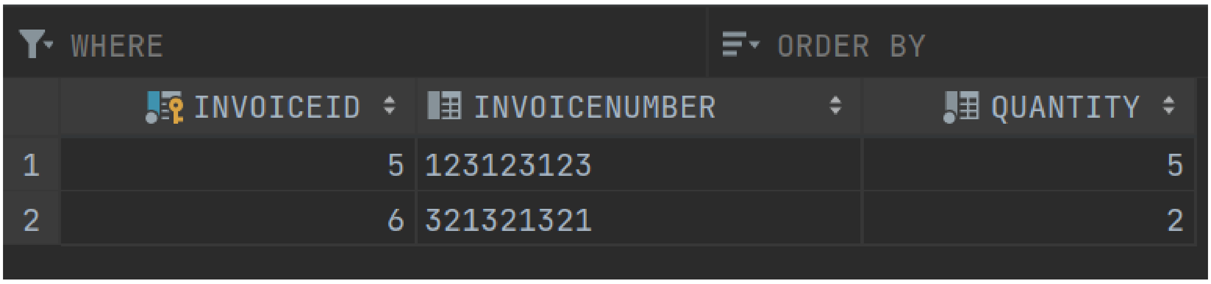
```
public static void main(final String[] args) throws Exception {
    final Session session = getSession();
    Product product1 = new Product( productName: "Dlugopis", unitsOnStock: 5);
    Product product2 = new Product( productName: "Olowek", unitsOnStock: 5);
    Product product3 = new Product( productName: "Pioro", unitsOnStock: 5);
    Product product4 = new Product( productName: "Linijka", unitsOnStock: 5);

    Invoice invoice1 = new Invoice( invoiceNumber: "123123123", quantity: 5);
    Invoice invoice2 = new Invoice( invoiceNumber: "321321321", quantity: 2);
    try {
        Transaction tx = session.beginTransaction();
        session.save(product1);
        session.save(product2);
        session.save(product3);
        session.save(product4);

        session.save(invoice1);
        session.save(invoice2);

        invoice1.sellProduct(product1, quantity: 3);
        invoice1.sellProduct(product2, quantity: 2);

        invoice2.sellProduct(product3, quantity: 1);
        invoice2.sellProduct(product4, quantity: 1);

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

Tabela ***PRODUCT***



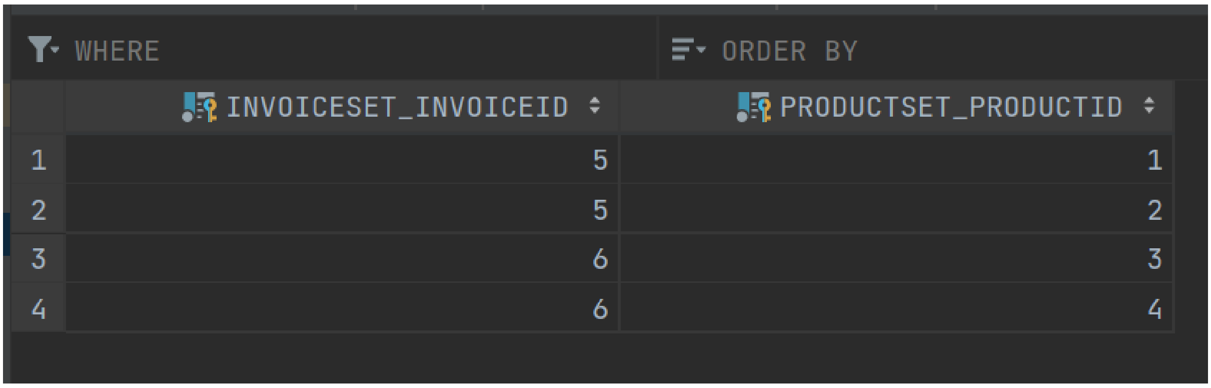
	PRODUCTID	PRODUCTNAME	UNITSONSTOCK
1	1	Dlugopis	2
2	2	Olowek	3
3	3	Pioro	4
4	4	Linijka	4

Tabela ***INVOICE***



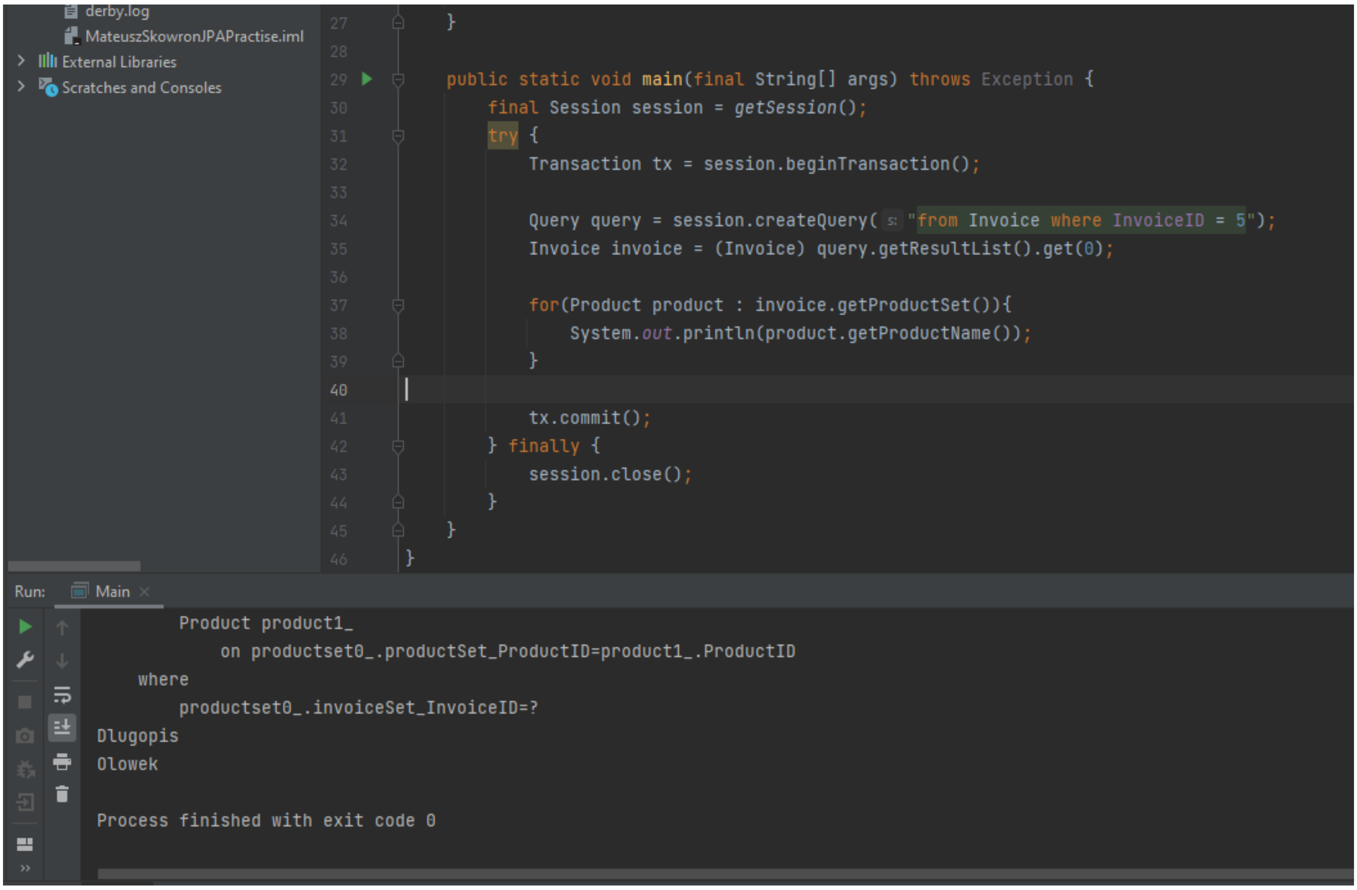
	INVOICEID	INVOICENUMBER	QUANTITY
1	5	123123123	5
2	6	321321321	2

Tabela ***INVOICE_PRODUCT***



	INVOICESET_INVOICEID	PRODUCTSET_PRODUCTID
1	5	1
2	5	2
3	6	3
4	6	4

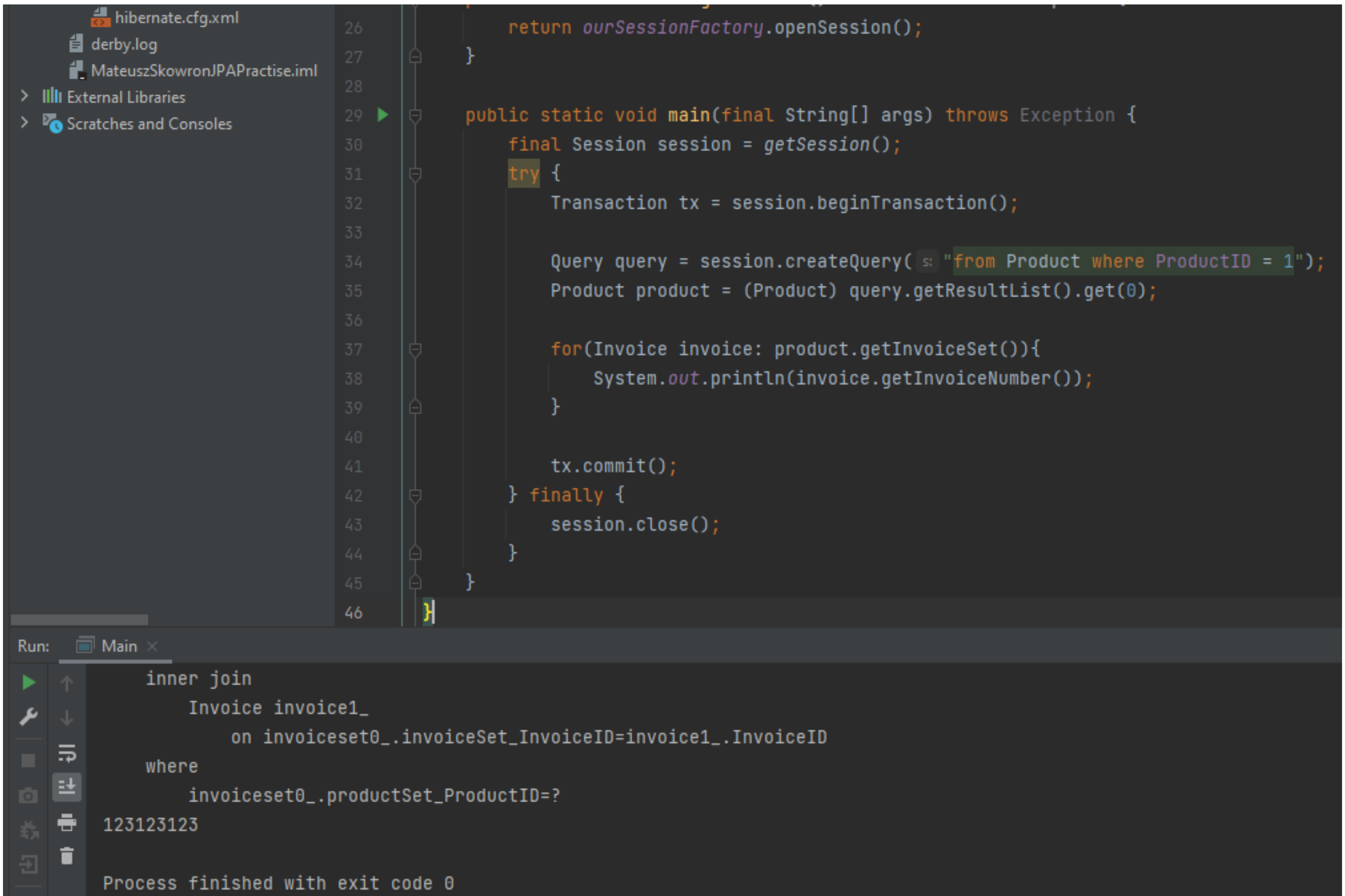
b. Pokaż produkty sprzedane w ramach wybranej faktury/transakcji



```
    }

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

            Query query = session.createQuery( s: "from Invoice where InvoiceID = 5");
            Invoice invoice = (Invoice) query.getResultList().get(0);

            for(Product product : invoice.getProductSet()){
                System.out.println(product.getProductName());
            }

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

```
        Product product1_
            on productset0_.productSet_ProductID=product1_.ProductID
    where
        productset0_.invoiceSet_InvoiceID=?
Dlugopis
Olowek

Process finished with exit code 0
```

c. Pokaż faktury w ramach których był sprzedany wybrany produkt



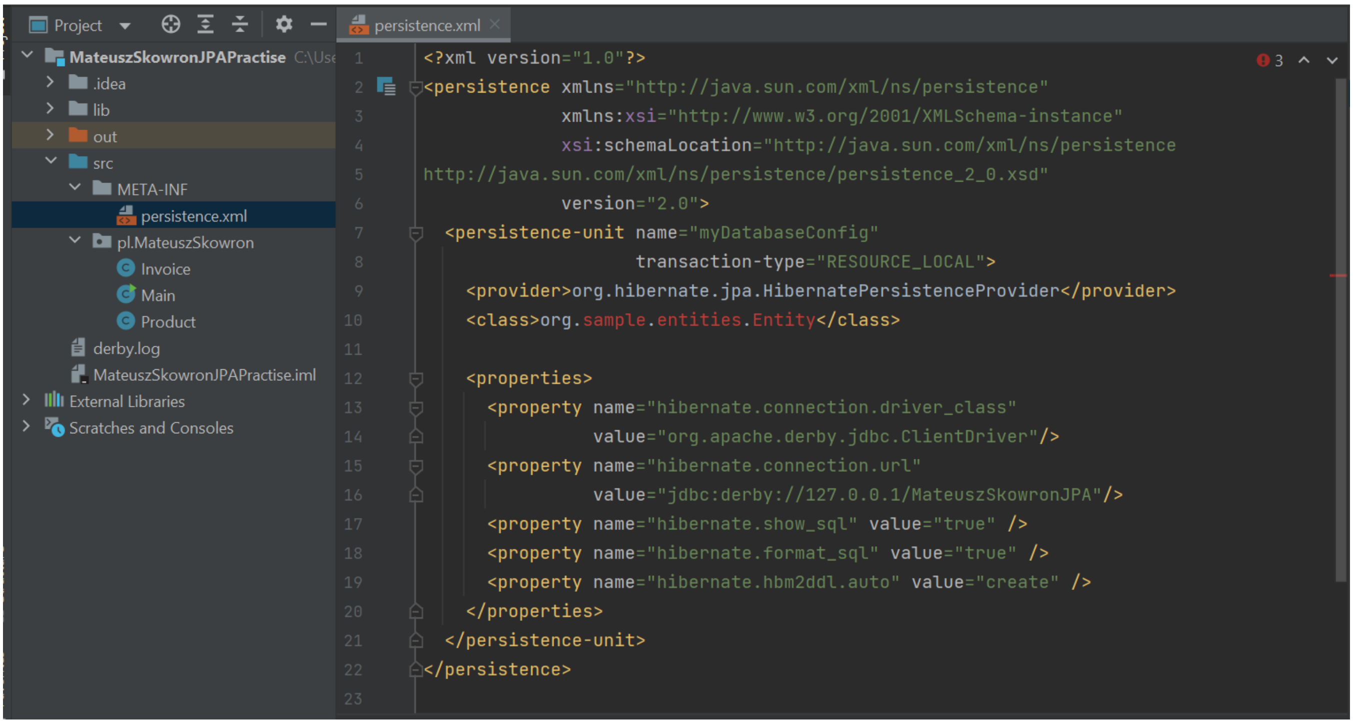
```
        return ourSessionFactory.openSession();
    }

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

            Query query = session.createQuery( s: "from Product where ProductID = 1");
            Product product = (Product) query.getResultList().get(0);

            for(Invoice invoice: product.getInvoiceSet()){
                System.out.println(invoice.getInvoiceNumber());
            }

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

```
    inner join
        Invoice invoice1_
            on invoiceset0_.invoiceSet_InvoiceID=invoice1_.InvoiceID
    where
        invoiceset0_.productSet_ProductID=?
123123123

Process finished with exit code 0
```

6. JPA

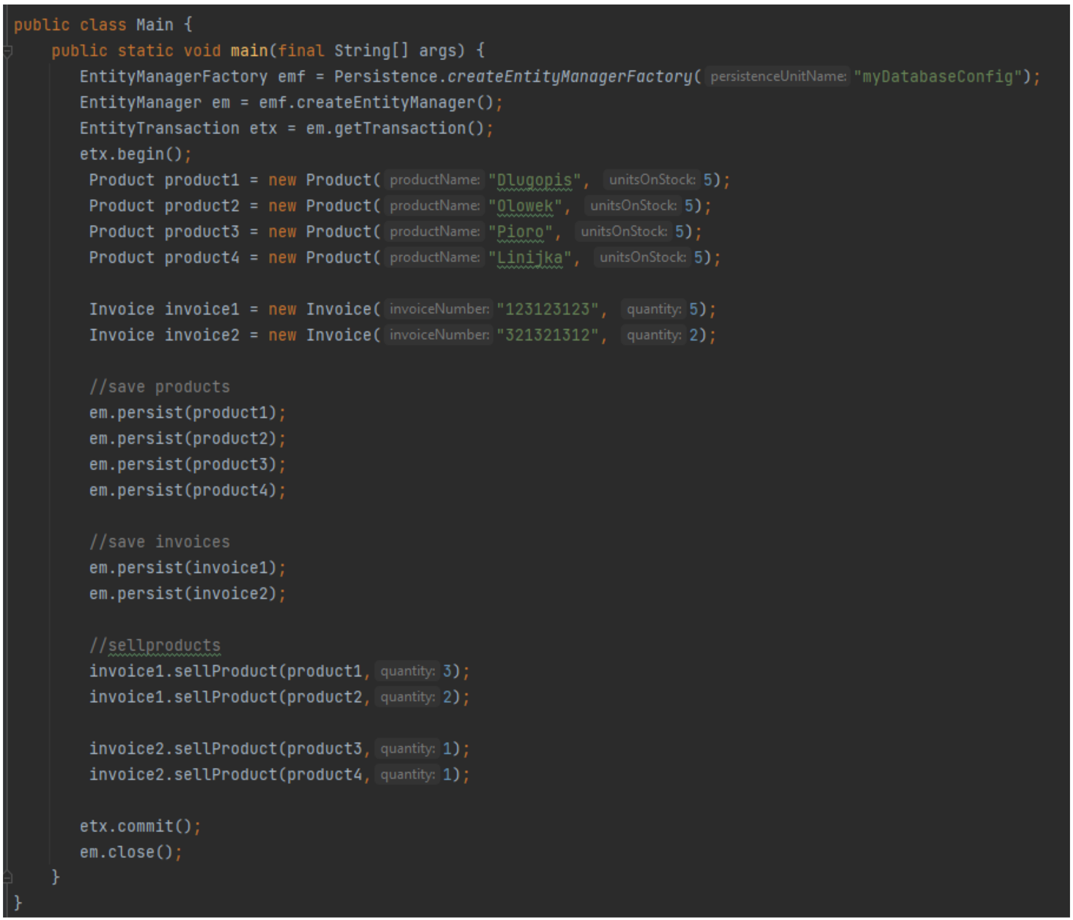
   a. Stwórz nowego main’a w którym zrobisz to samo co w poprzednim ale z wykorzystaniem JPA

persistence.xml



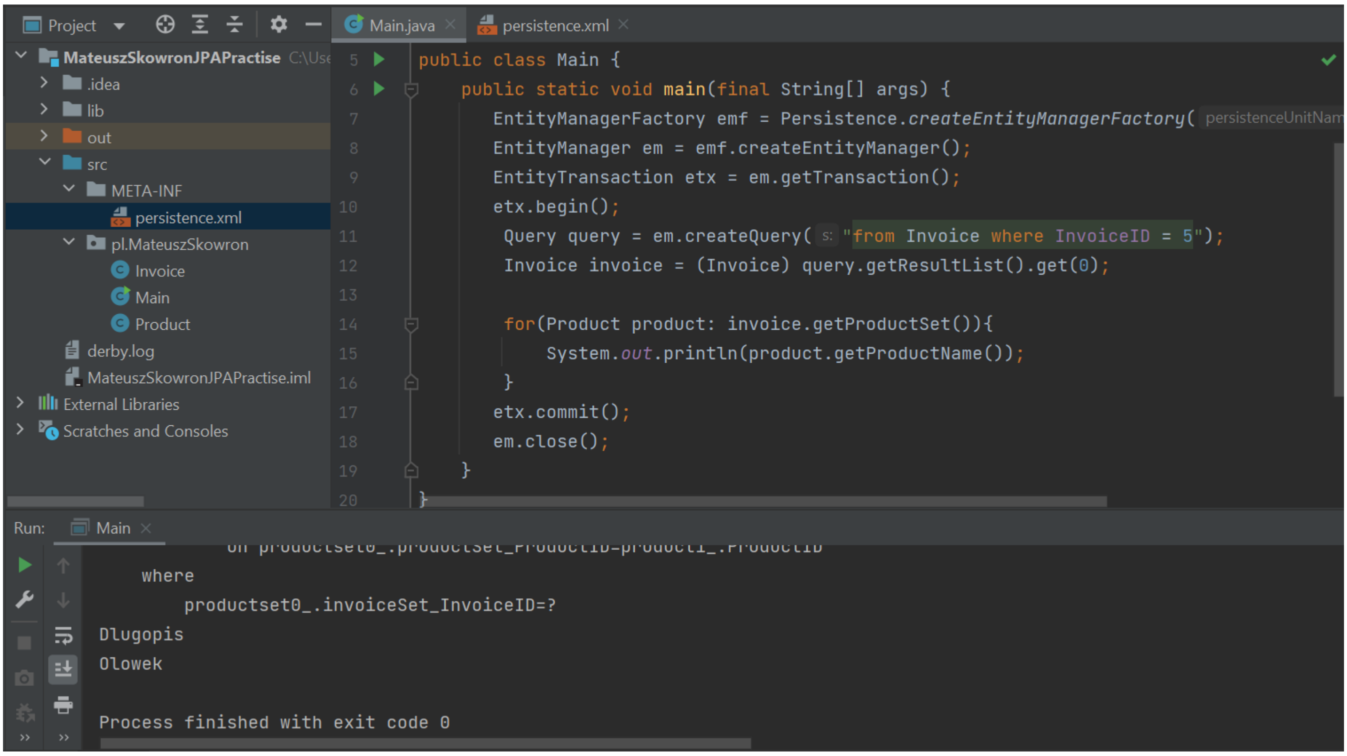
```xml
<?xml version="1.0"?>
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd"
             version="2.0">
  <persistence-unit name="myDatabaseConfig"
                    transaction-type="RESOURCE_LOCAL">
    <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
    <class>org.sample.entities.Entity</class>

    <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/MateuszSkowronJPA"/>
      <property name="hibernate.show_sql" value="true" />
      <property name="hibernate.format_sql" value="true" />
      <property name="hibernate.hbm2ddl.auto" value="create" />
    </properties>
  </persistence-unit>
</persistence>

```

a. Stwórz kilka produktów I “sprzedaj” je na kilku transakcjach



```java
public class Main {
    public static void main(final String[] args) {
       EntityManagerFactory emf = Persistence.createEntityManagerFactory( persistenceUnitName: "myDatabaseConfig");
       EntityManager em = emf.createEntityManager();
       EntityTransaction etx = em.getTransaction();
       etx.begin();
        Product product1 = new Product( productName: "Dlugopis",  unitsOnStock: 5);
        Product product2 = new Product( productName: "Olowek",  unitsOnStock: 5);
        Product product3 = new Product( productName: "Pioro",  unitsOnStock: 5);
        Product product4 = new Product( productName: "Linijka",  unitsOnStock: 5);

        Invoice invoice1 = new Invoice( invoiceNumber: "123123123",  quantity: 5);
        Invoice invoice2 = new Invoice( invoiceNumber: "321321312",  quantity: 2);

        //save products
        em.persist(product1);
        em.persist(product2);
        em.persist(product3);
        em.persist(product4);

        //save invoices
        em.persist(invoice1);
        em.persist(invoice2);

        //sellproducts
        invoice1.sellProduct(product1, quantity: 3);
        invoice1.sellProduct(product2, quantity: 2);

        invoice2.sellProduct(product3, quantity: 1);
        invoice2.sellProduct(product4, quantity: 1);

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

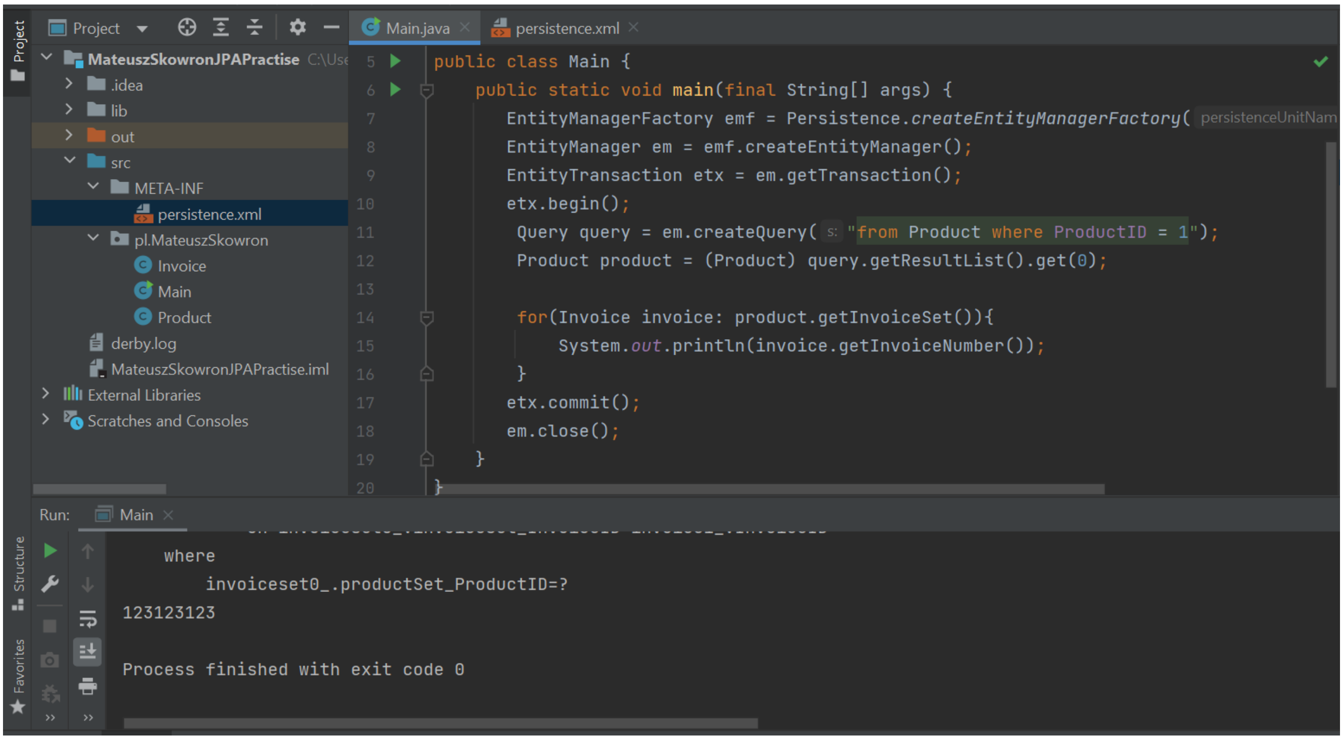
b. Pokaż produkty sprzedane w ramach wybranej faktury/transakcji



```java
public class Main {
    public static void main(final String[] args) {
        EntityManagerFactory emf = Persistence.createEntityManagerFactory( persistenceUnitNam
        EntityManager em = emf.createEntityManager();
        EntityTransaction etx = em.getTransaction();
        etx.begin();
         Query query = em.createQuery( s: "from Invoice where InvoiceID = 5");
         Invoice invoice = (Invoice) query.getResultList().get(0);

         for(Product product: invoice.getProductSet()){
             System.out.println(product.getProductName());
         }
        etx.commit();
        em.close();
    }
}
```

```
        on productset0_.productSet_ProductID=product1_.ProductID
    where
        productset0_.invoiceSet_InvoiceID=?
Dlugopis
Olowek

Process finished with exit code 0
```

c. Pokaż faktury w ramach których był sprzedany wybrany produkt



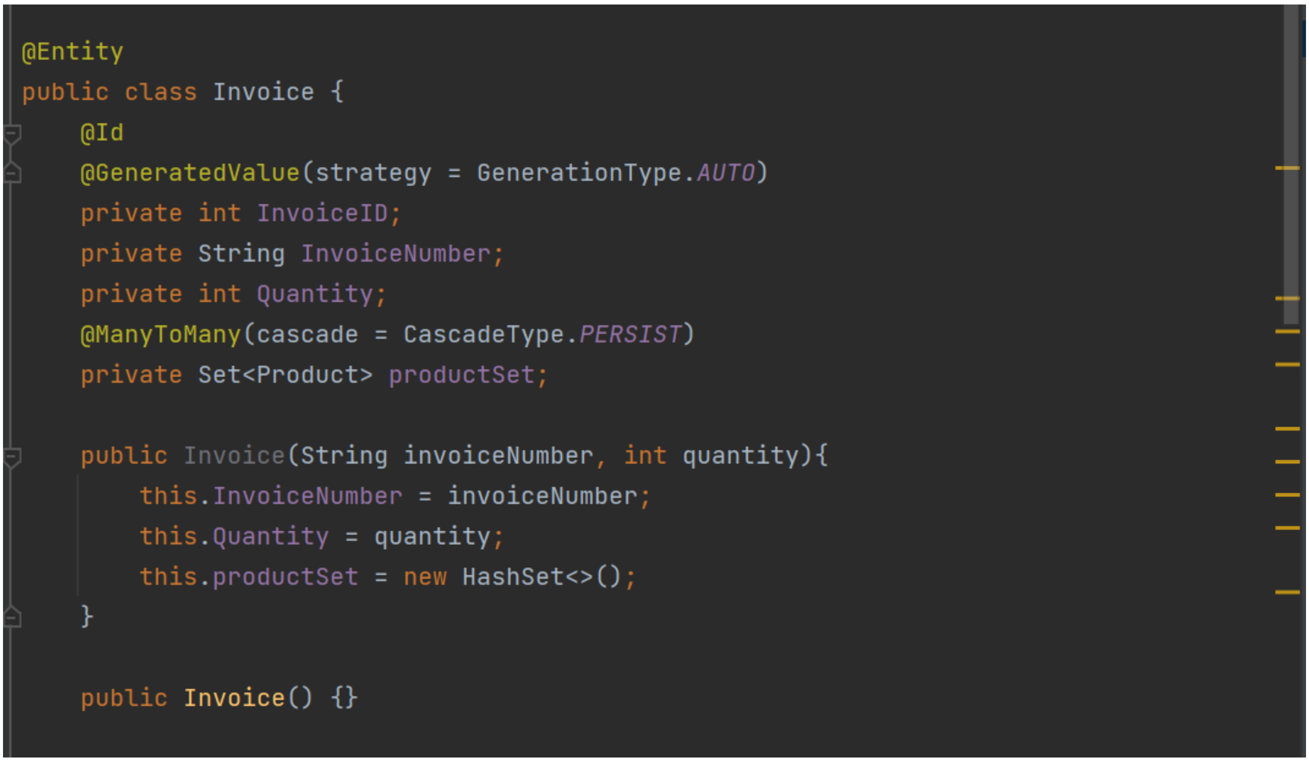
```java
public class Main {
    public static void main(final String[] args) {
        EntityManagerFactory emf = Persistence.createEntityManagerFactory( persistenceUnitNam
        EntityManager em = emf.createEntityManager();
        EntityTransaction etx = em.getTransaction();
        etx.begin();
         Query query = em.createQuery( s: "from Product where ProductID = 1");
         Product product = (Product) query.getResultList().get(0);

         for(Invoice invoice: product.getInvoiceSet()){
             System.out.println(invoice.getInvoiceNumber());
         }
        etx.commit();
        em.close();
    }
}
```

```
        on invoiceset0_.invoiceSet_InvoiceID=invoice1_.InvoiceID
    where
        invoiceset0_.productSet_ProductID=?
123123123

Process finished with exit code 0
```

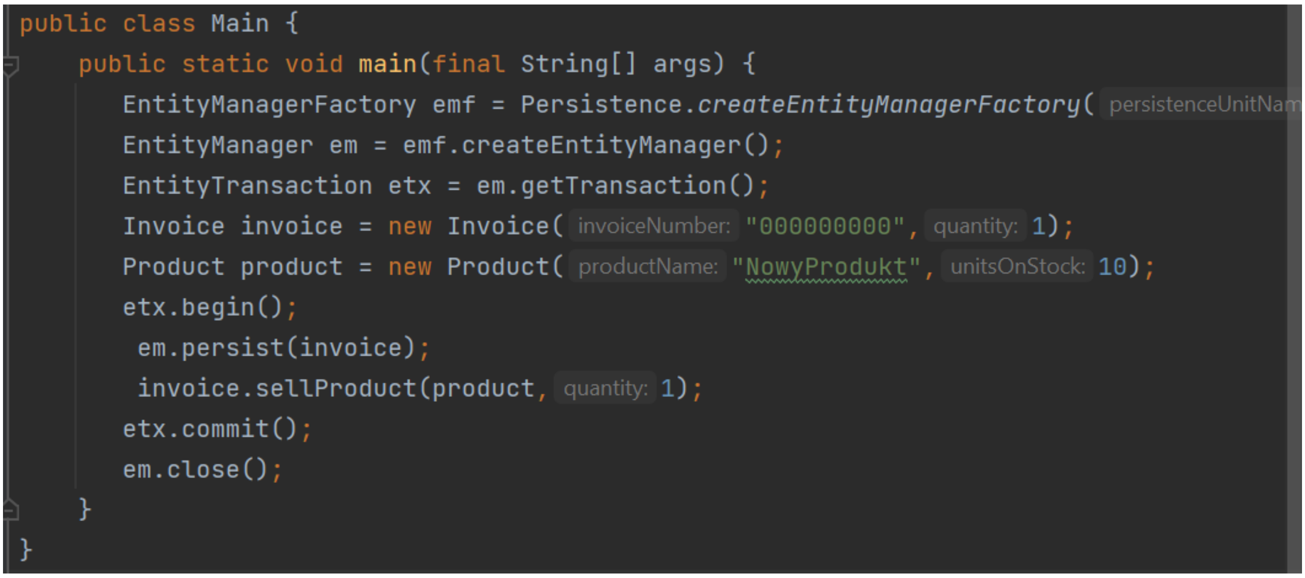
7. Kaskady
    a. Zmodyfikuj model w taki sposób aby było możliwe kaskadowe tworzenie faktur wraz z nowymi produktami, oraz produktów wraz z nową fakturą

Nowy produkt przy nowej fakturze

Klasa ***Invoice***

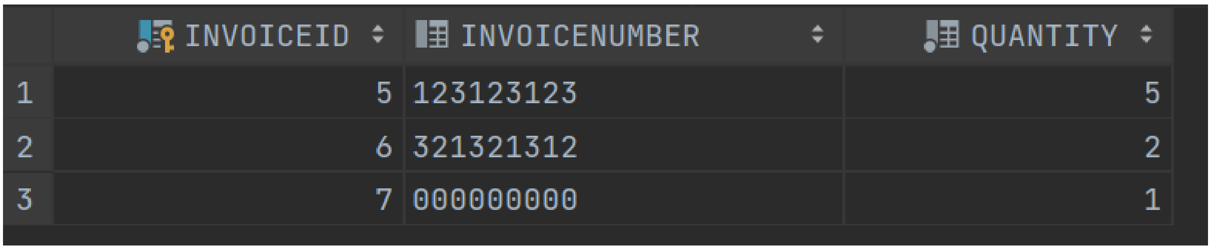


```
@Entity
public class Invoice {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int InvoiceID;
    private String InvoiceNumber;
    private int Quantity;
    @ManyToMany(cascade = CascadeType.PERSIST)
    private Set<Product> productSet;

    public Invoice(String invoiceNumber, int quantity){
        this.InvoiceNumber = invoiceNumber;
        this.Quantity = quantity;
        this.productSet = new HashSet<>();
    }

    public Invoice() {}
```



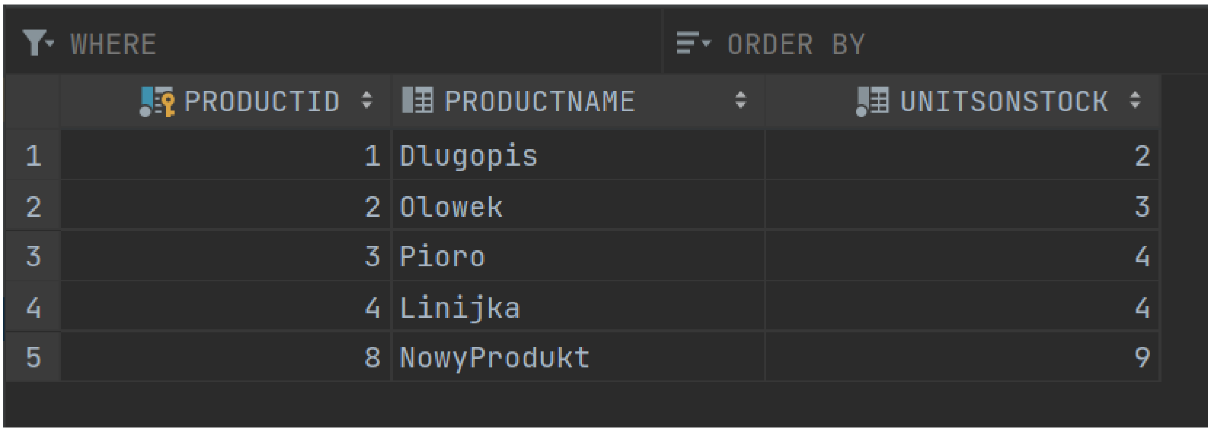
```
public class Main {
    public static void main(final String[] args) {
        EntityManagerFactory emf = Persistence.createEntityManagerFactory( persistenceUnitNam
        EntityManager em = emf.createEntityManager();
        EntityTransaction etx = em.getTransaction();
        Invoice invoice = new Invoice( invoiceNumber: "000000000", quantity: 1);
        Product product = new Product( productName: "NowyProdukt", unitsOnStock: 10);
        etx.begin();
         em.persist(invoice);
         invoice.sellProduct(product, quantity: 1);
        etx.commit();
        em.close();
    }
}
```

Tabela ***INVOICE***



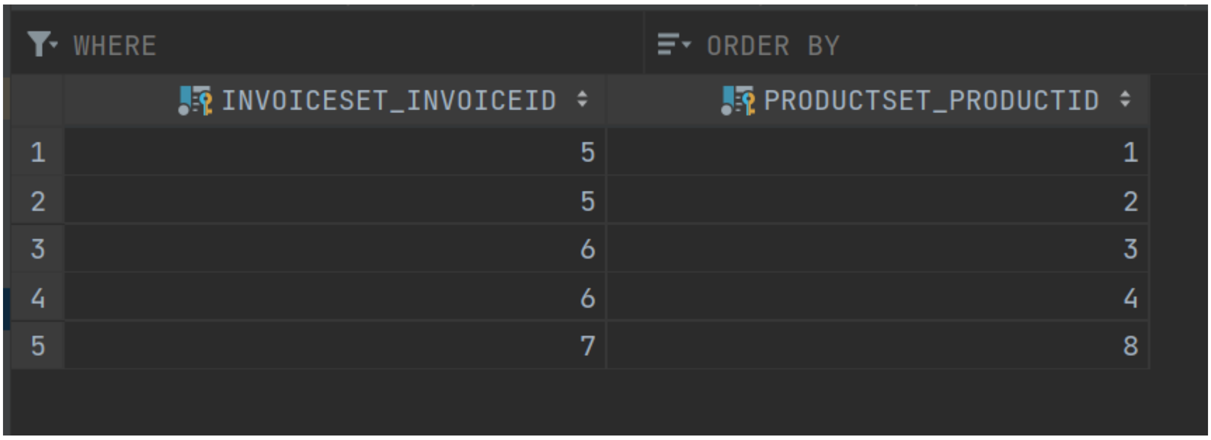
	INVOICEID	INVOICENUMBER	QUANTITY
1	5	123123123	5
2	6	321321312	2
3	7	000000000	1

Tabela ***PRODUCT***



	PRODUCTID	PRODUCTNAME	UNITSONSTOCK
1	1	Dlugopis	2
2	2	Olowek	3
3	3	Pioro	4
4	4	Linijka	4
5	8	NowyProdukt	9

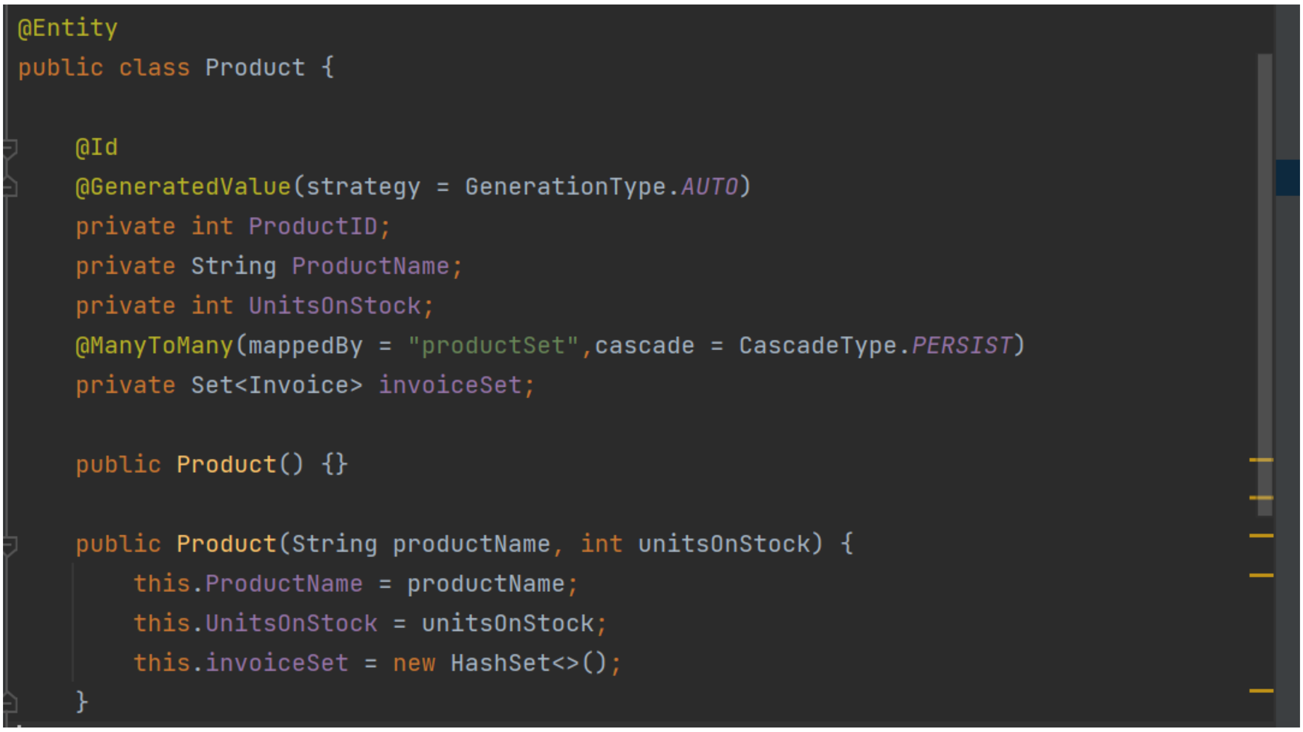
Tabela ***INVOICE_PRODUCT***



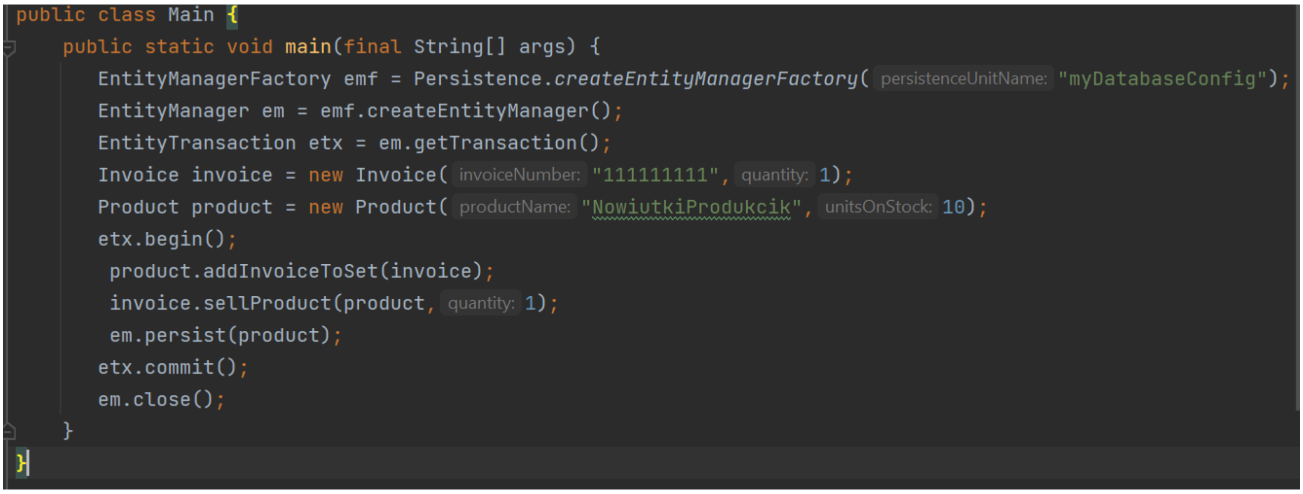
	INVOICESET_INVOICEID	PRODUCTSET_PRODUCTID
1	5	1
2	5	2
3	6	3
4	6	4
5	7	8

Nowa faktura przy nowym produkcie

Klasa ***Product***

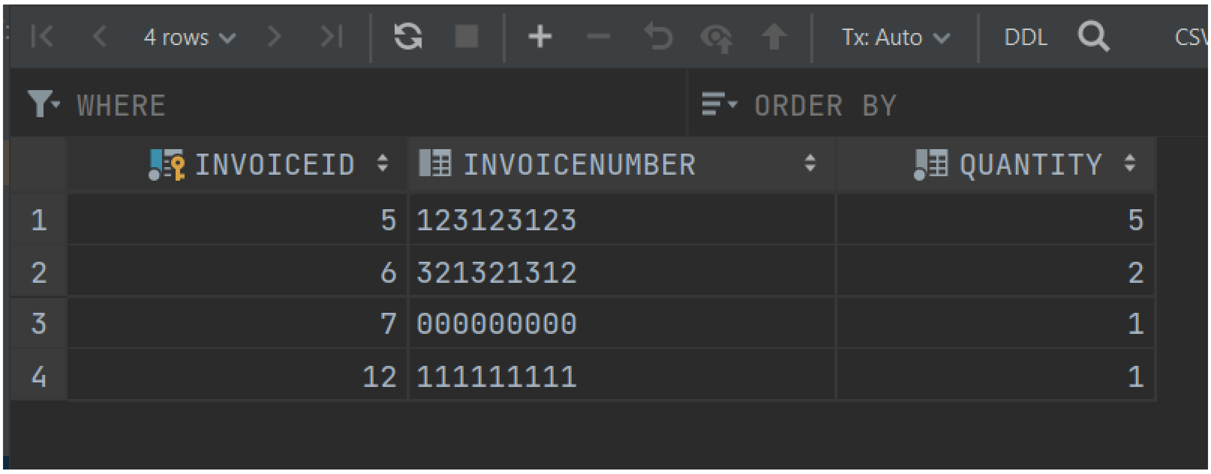


```java
@Entity
public class Product {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int ProductID;
    private String ProductName;
    private int UnitsOnStock;
    @ManyToMany(mappedBy = "productSet",cascade = CascadeType.PERSIST)
    private Set<Invoice> invoiceSet;

    public Product() {}

    public Product(String productName, int unitsOnStock) {
        this.ProductName = productName;
        this.UnitsOnStock = unitsOnStock;
        this.invoiceSet = new HashSet<>();
    }
```



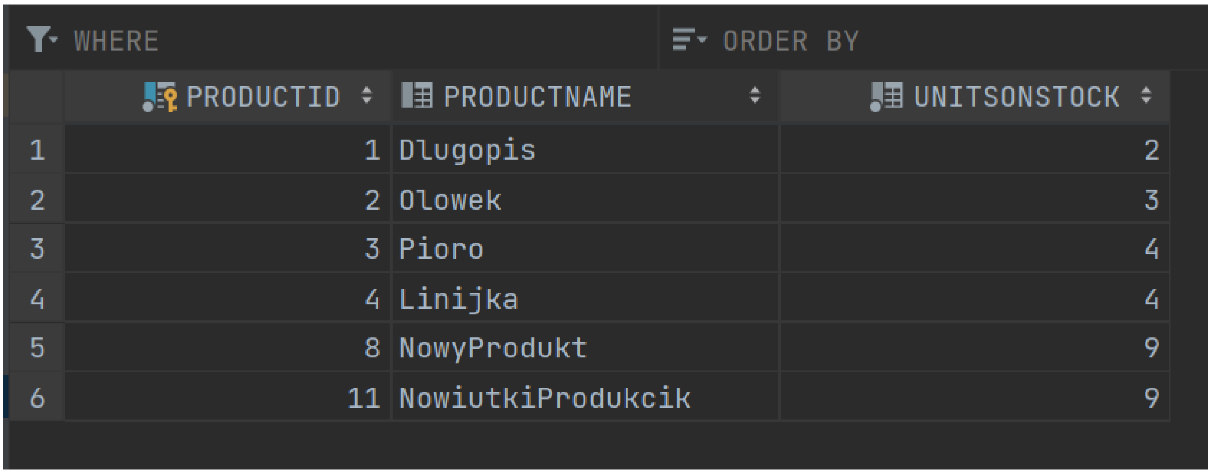
```java
public class Main {
    public static void main(final String[] args) {
        EntityManagerFactory emf = Persistence.createEntityManagerFactory( persistenceUnitName: "myDatabaseConfig");
        EntityManager em = emf.createEntityManager();
        EntityTransaction etx = em.getTransaction();
        Invoice invoice = new Invoice( invoiceNumber: "111111111", quantity: 1);
        Product product = new Product( productName: "NowiutkiProdukcik", unitsOnStock: 10);
        etx.begin();
         product.addInvoiceToSet(invoice);
         invoice.sellProduct(product, quantity: 1);
         em.persist(product);
        etx.commit();
        em.close();
    }
}
```

Tabela ***INVOICE***



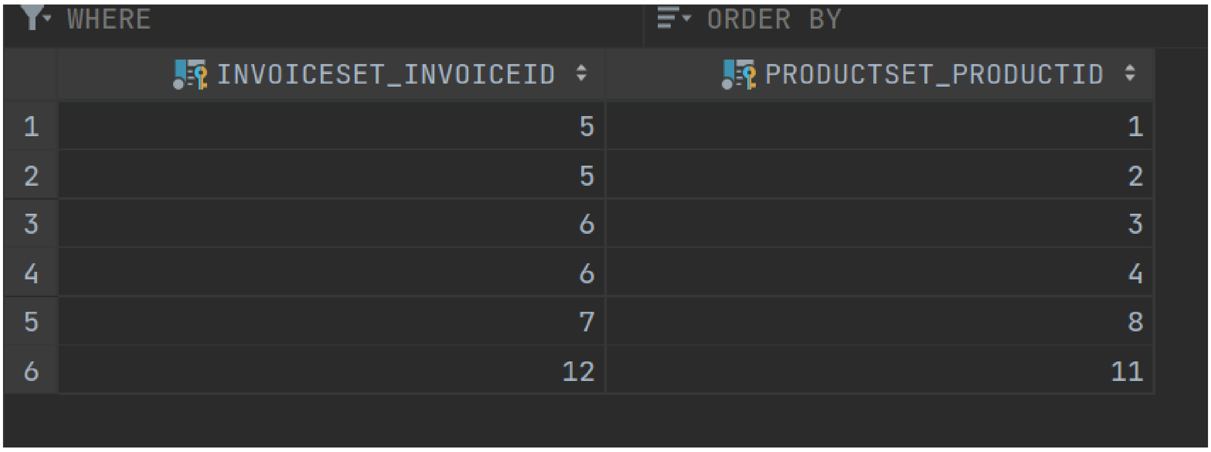
	INVOICEID	INVOICENUMBER	QUANTITY
1	5	123123123	5
2	6	321321312	2
3	7	000000000	1
4	12	111111111	1

Tabela ***PRODUCT***



	PRODUCTID	PRODUCTNAME	UNITSONSTOCK
1	1	Dlugopis	2
2	2	Olowek	3
3	3	Pioro	4
4	4	Linijka	4
5	8	NowyProdukt	9
6	11	NowiutkiProdukcik	9

Tabela ***INVOICE_PRODUCT***

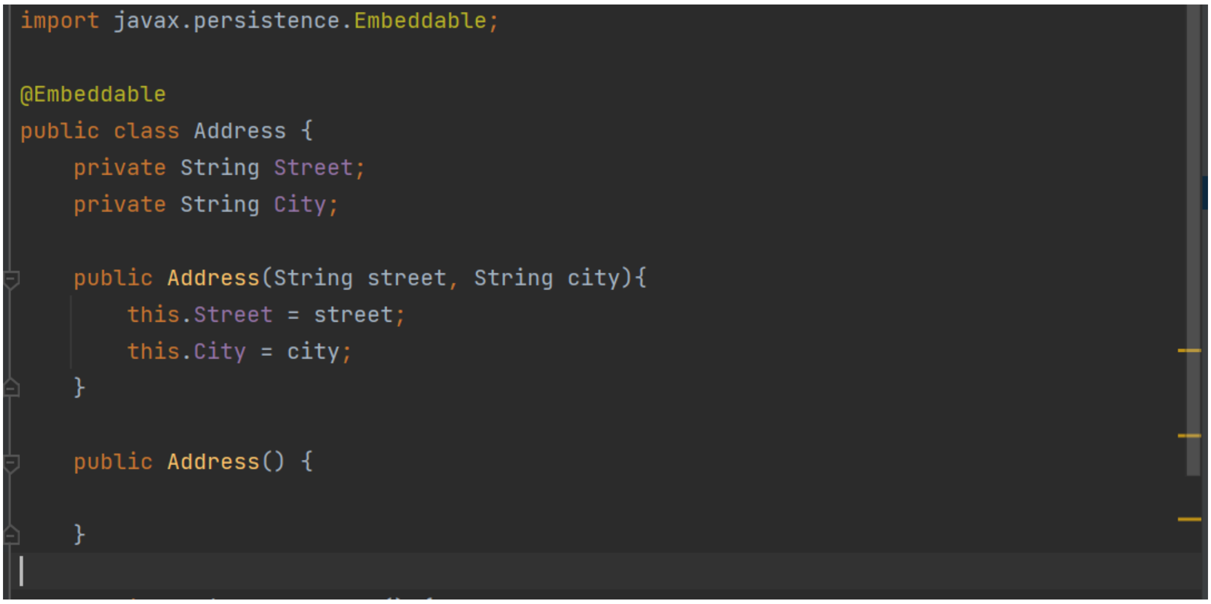


	INVOICESET_INVOICEID	PRODUCTSET_PRODUCTID
1	5	1
2	5	2
3	6	3
4	6	4
5	7	8
6	12	11

8. Embedded class

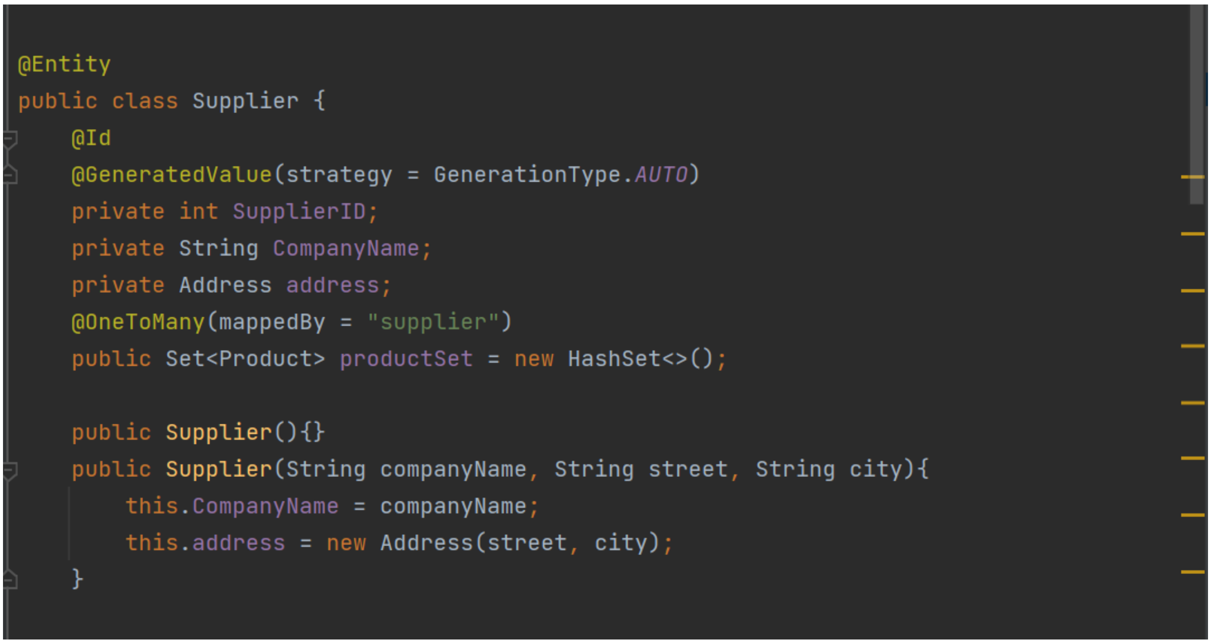
a. Dodaj do modelu klasę adres. „Wbuduj” ją do tabeli Dostawców

Klasa ***Address***



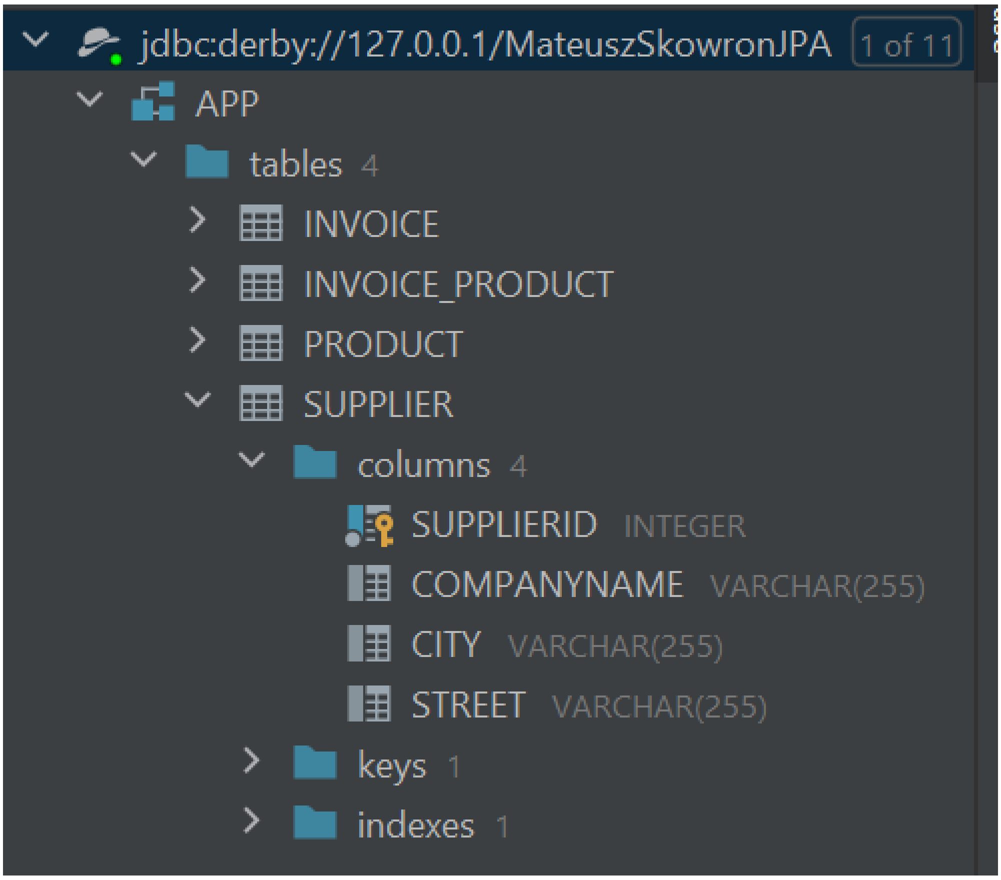
```
import javax.persistence.Embeddable;

@Embeddable
public class Address {
    private String Street;
    private String City;

    public Address(String street, String city){
        this.Street = street;
        this.City = city;
    }

    public Address() {

    }

```

Klasa ***Supplier***



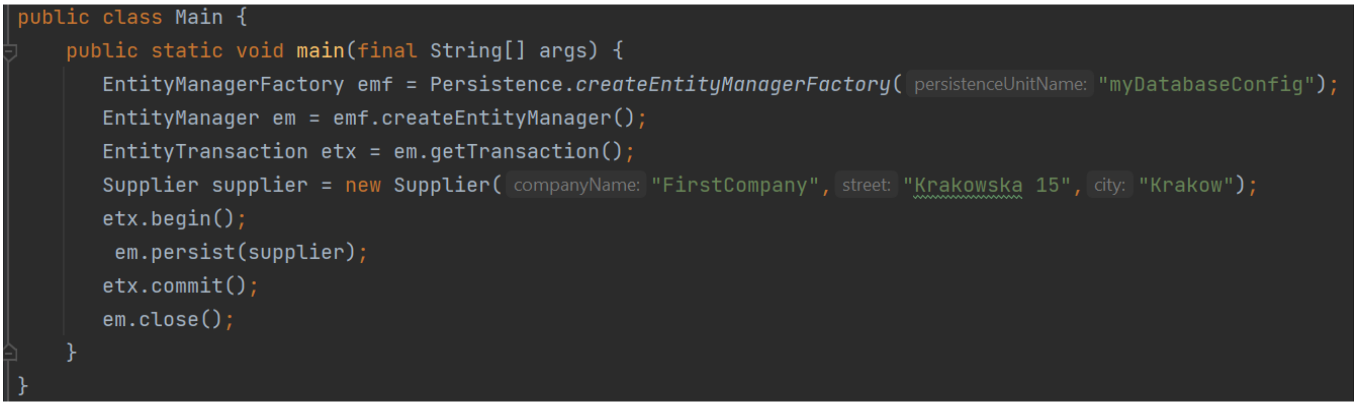
```

@Entity
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int SupplierID;
    private String CompanyName;
    private Address address;
    @OneToMany(mappedBy = "supplier")
    public Set<Product> productSet = new HashSet<>();

    public Supplier(){}
    public Supplier(String companyName, String street, String city){
        this.CompanyName = companyName;
        this.address = new Address(street, city);
    }
```

Struktura bazy danych



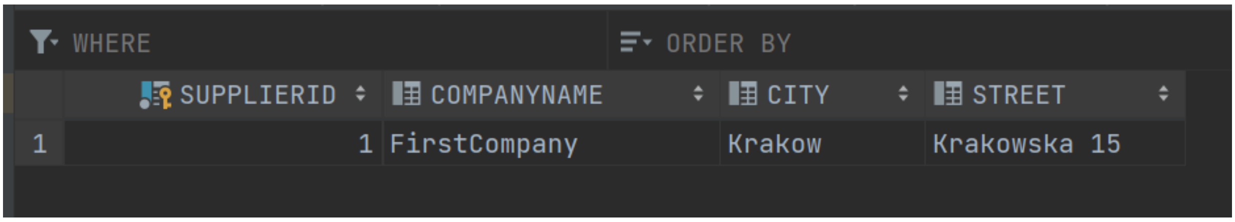
```
jdbc:derby://127.0.0.1/MateuszSkowronJPA  1 of 11
  APP
    tables 4
      INVOICE
      INVOICE_PRODUCT
      PRODUCT
      SUPPLIER
        columns 4
          SUPPLIERID INTEGER
          COMPANYNAME VARCHAR(255)
          CITY VARCHAR(255)
          STREET VARCHAR(255)
        keys 1
        indexes 1
```

Dodanie nowego dostawcy



```java
public class Main {
    public static void main(final String[] args) {
        EntityManagerFactory emf = Persistence.createEntityManagerFactory( persistenceUnitName: "myDatabaseConfig");
        EntityManager em = emf.createEntityManager();
        EntityTransaction etx = em.getTransaction();
        Supplier supplier = new Supplier( companyName: "FirstCompany", street: "Krakowska 15", city: "Krakow");
        etx.begin();
         em.persist(supplier);
        etx.commit();
        em.close();
    }
}
```

Tabela ***SUPPLIER***

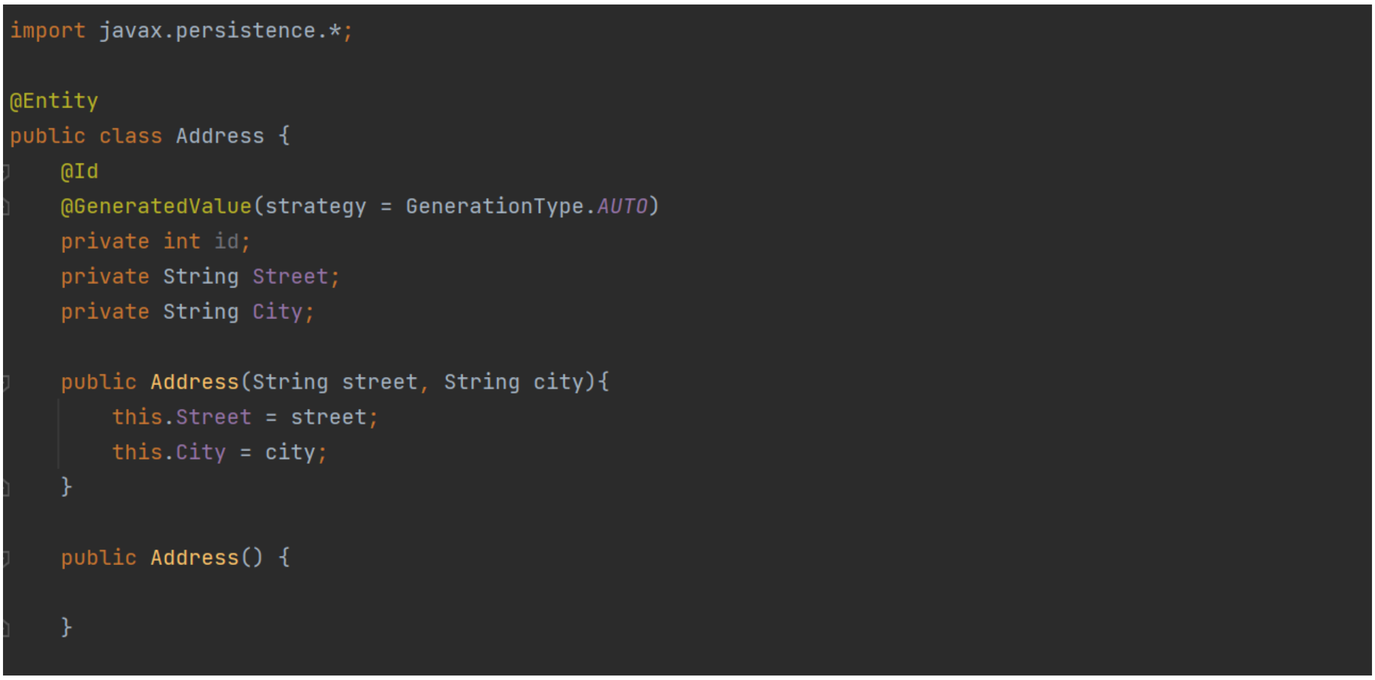


WHERE ORDER BY

	SUPPLIERID	COMPANYNAME	CITY	STREET
1	1	FirstCompany	Krakow	Krakowska 15

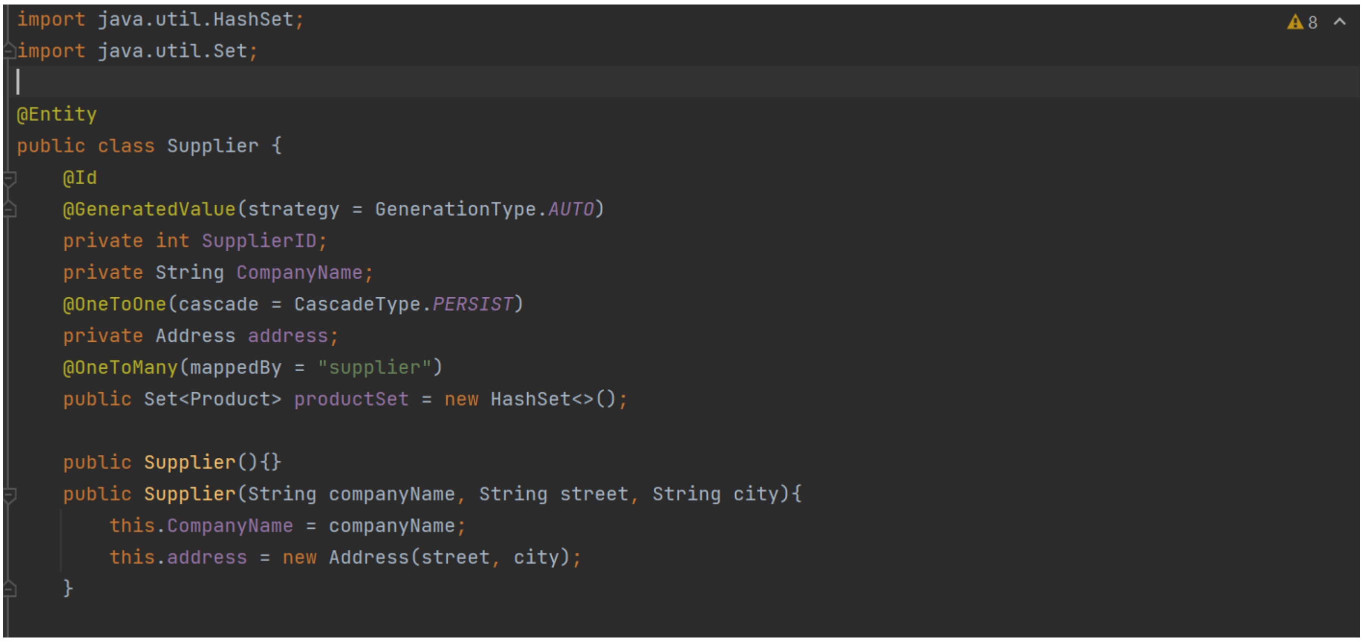
b. Zmodyfikuj model w taki sposób, że dane adresowe znajdują się w klasie dostawców. Zmapuj to do dwóch osobnych tabel

Klasa ***Address***



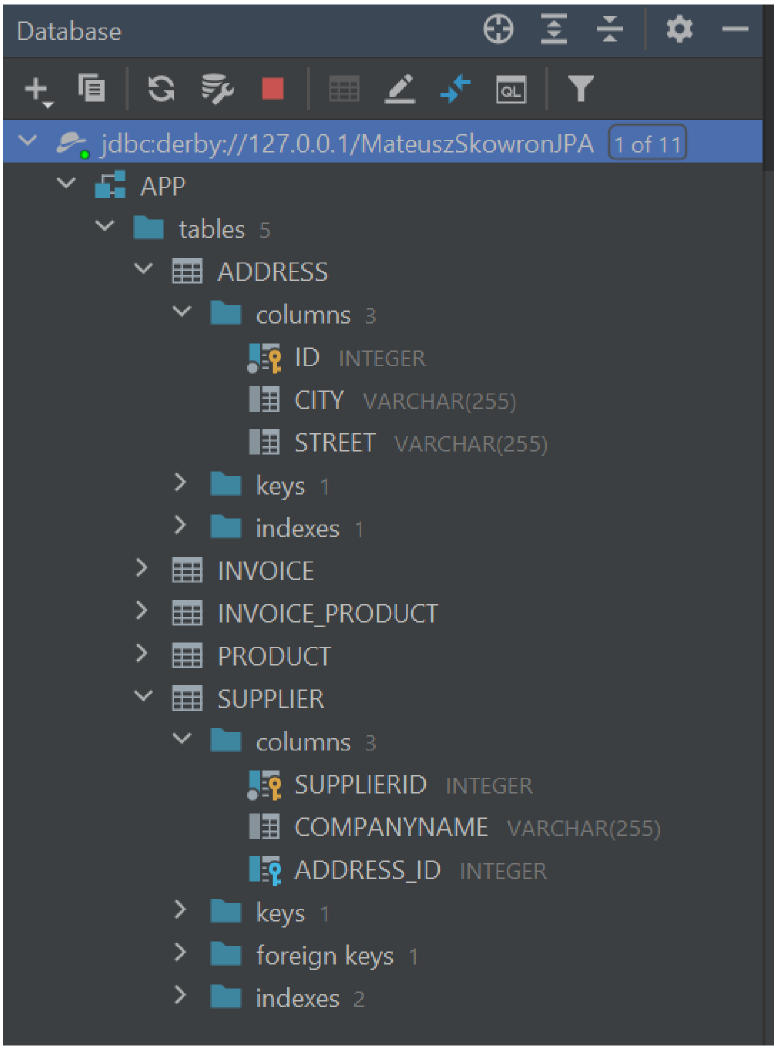
```
import javax.persistence.*;

@Entity
public class Address {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;
    private String Street;
    private String City;

    public Address(String street, String city){
        this.Street = street;
        this.City = city;
    }

    public Address() {

    }
```

Klasa ***Supplier***



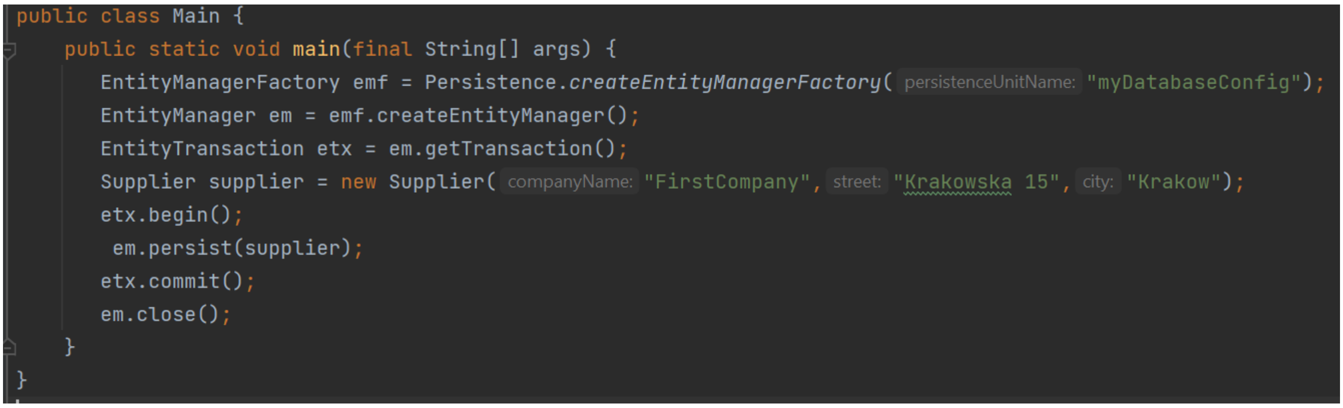
```
import java.util.HashSet;
import java.util.Set;

@Entity
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int SupplierID;
    private String CompanyName;
    @OneToOne(cascade = CascadeType.PERSIST)
    private Address address;
    @OneToMany(mappedBy = "supplier")
    public Set<Product> productSet = new HashSet<>();

    public Supplier(){}
    public Supplier(String companyName, String street, String city){
        this.CompanyName = companyName;
        this.address = new Address(street, city);
    }
```

Struktura bazy danych



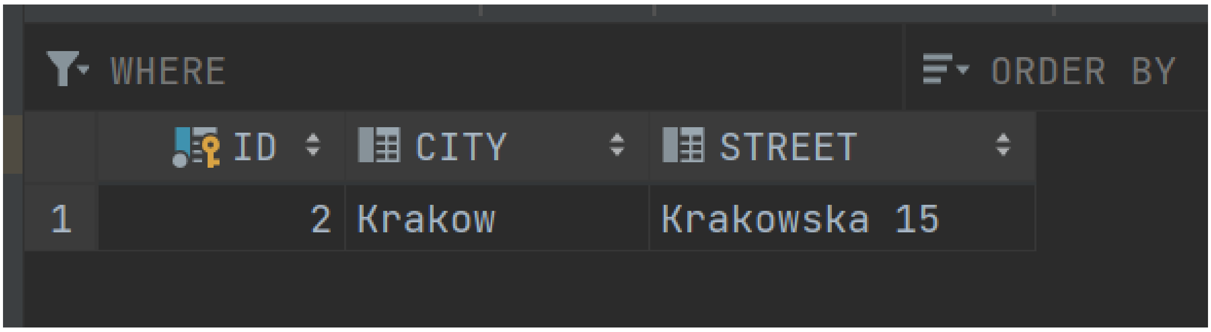
```
Database
jdbc:derby://127.0.0.1/MateuszSkowronJPA  1 of 11
  APP
    tables 5
      ADDRESS
        columns 3
          ID  INTEGER
          CITY  VARCHAR(255)
          STREET  VARCHAR(255)
        keys 1
        indexes 1
      INVOICE
      INVOICE_PRODUCT
      PRODUCT
      SUPPLIER
        columns 3
          SUPPLIERID  INTEGER
          COMPANYNAME  VARCHAR(255)
          ADDRESS_ID  INTEGER
        keys 1
        foreign keys 1
        indexes 2
```

Dodanie dostawcy



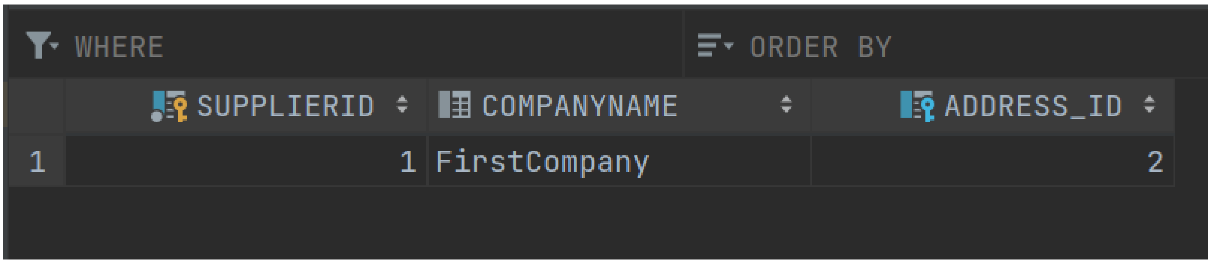
```java
public class Main {
    public static void main(final String[] args) {
        EntityManagerFactory emf = Persistence.createEntityManagerFactory( persistenceUnitName: "myDatabaseConfig");
        EntityManager em = emf.createEntityManager();
        EntityTransaction etx = em.getTransaction();
        Supplier supplier = new Supplier( companyName: "FirstCompany", street: "Krakowska 15", city: "Krakow");
        etx.begin();
         em.persist(supplier);
        etx.commit();
        em.close();
    }
}
```

Tabela ***ADDRESS***



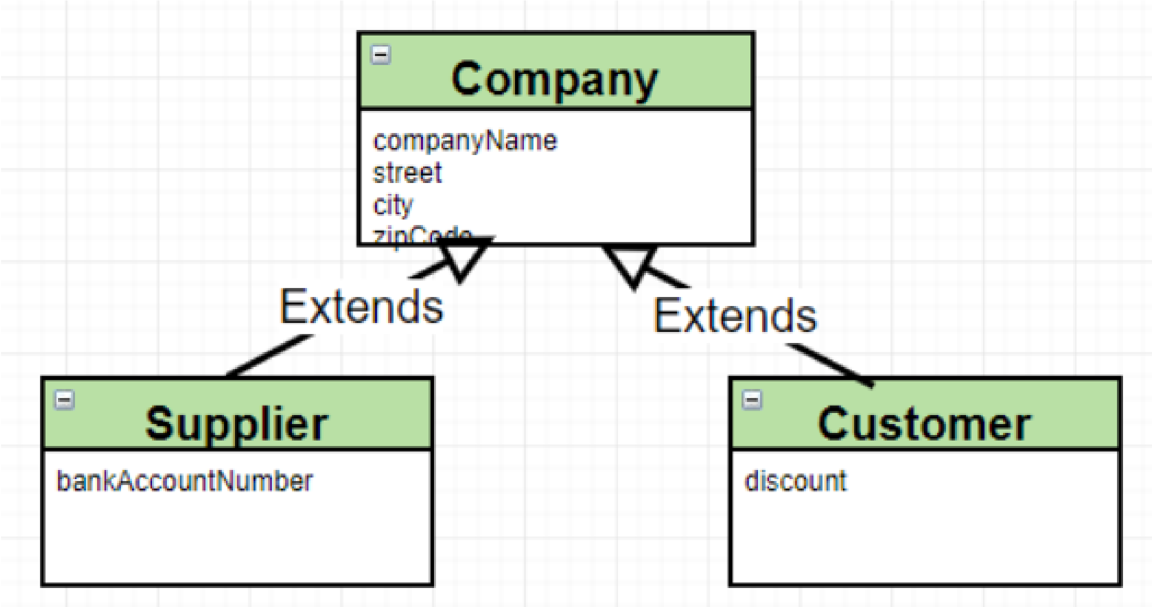
	ID	CITY	STREET
1	2	Krakow	Krakowska 15

Tabela ***SUPPLIER***



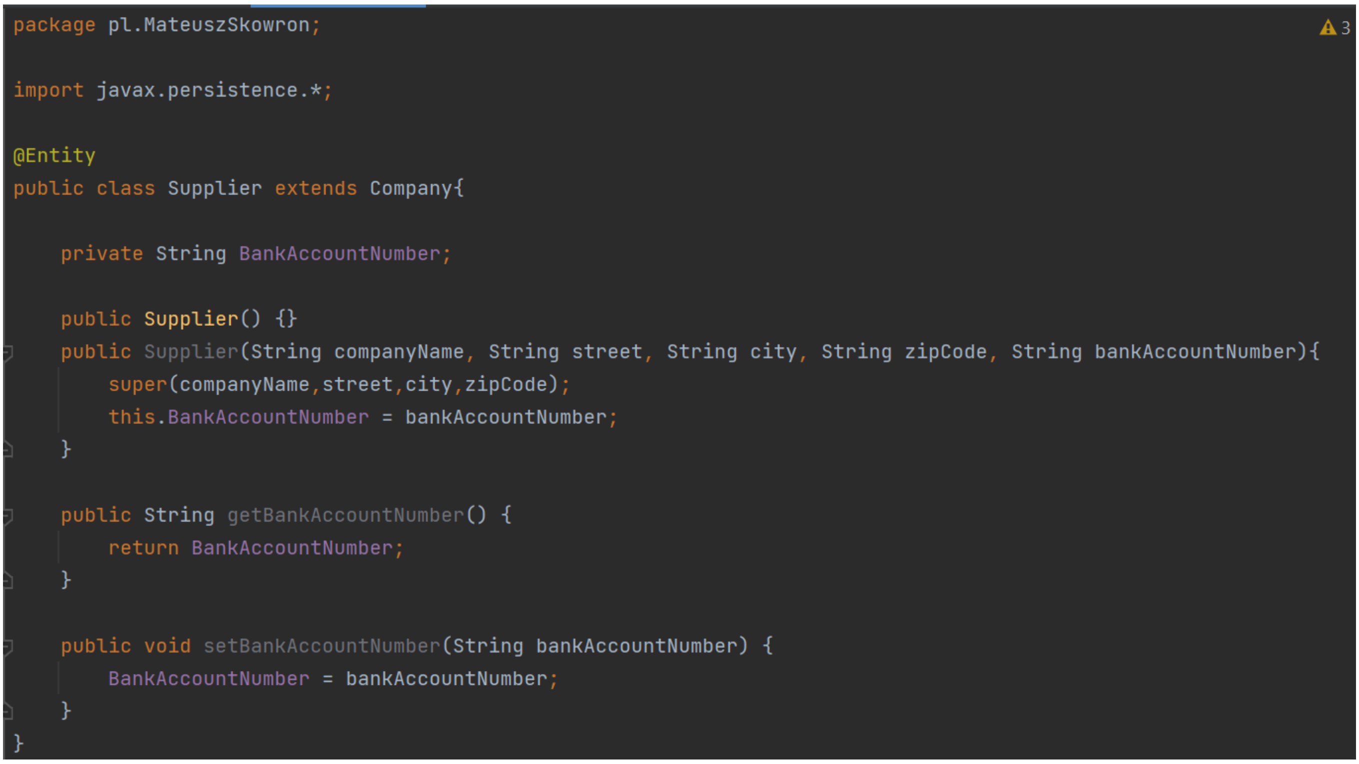
	SUPPLIERID	COMPANYNAME	ADDRESS_ID
1	1	FirstCompany	2

9. Dziedziczenie
    a. Wprowadź do modelu następującą hierarchię:



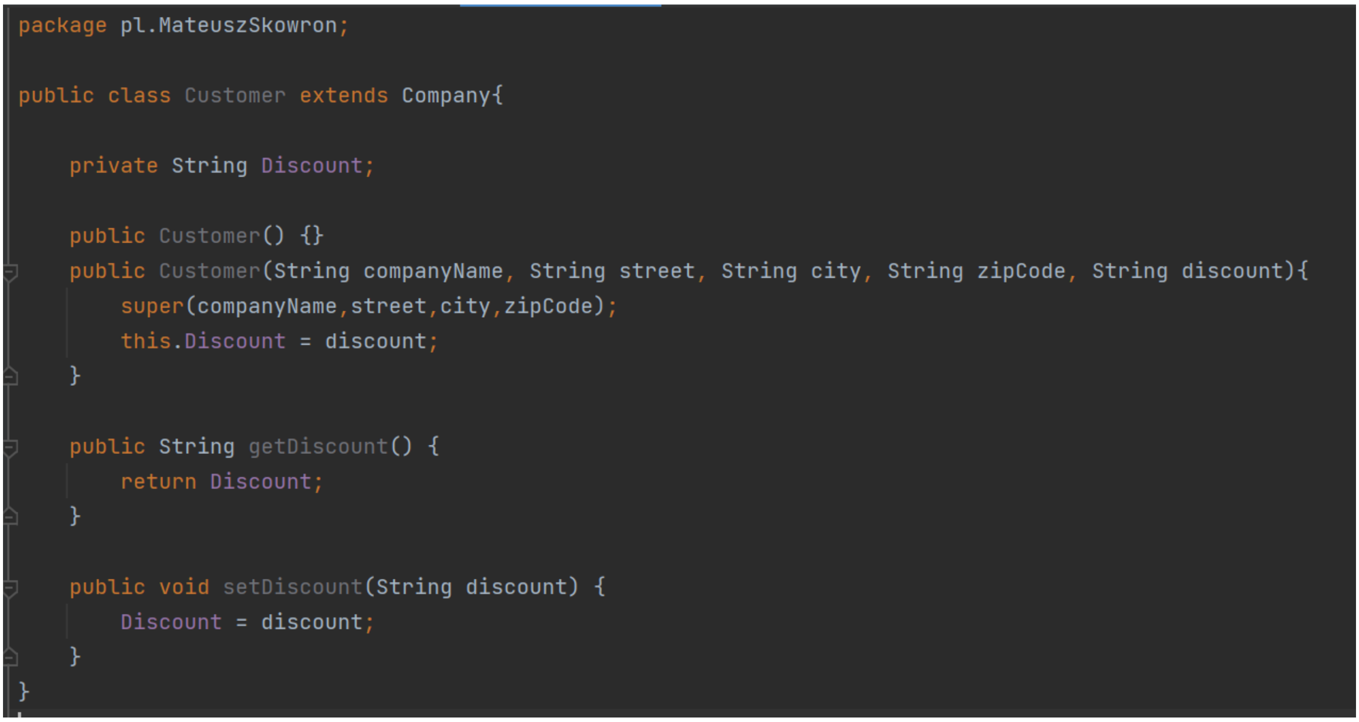
    b. Dodaj i pobierz z bazy kilka firm obu rodzajów stosując po kolei trzy różne strategie mapowania dziedziczenia.

Klasa ***Supplier***



```java
package pl.MateuszSkowron;

import javax.persistence.*;

@Entity
public class Supplier extends Company{

    private String BankAccountNumber;

    public Supplier() {}
    public Supplier(String companyName, String street, String city, String zipCode, String bankAccountNumber){
        super(companyName,street,city,zipCode);
        this.BankAccountNumber = bankAccountNumber;
    }

    public String getBankAccountNumber() {
        return BankAccountNumber;
    }

    public void setBankAccountNumber(String bankAccountNumber) {
        BankAccountNumber = bankAccountNumber;
    }
}
```

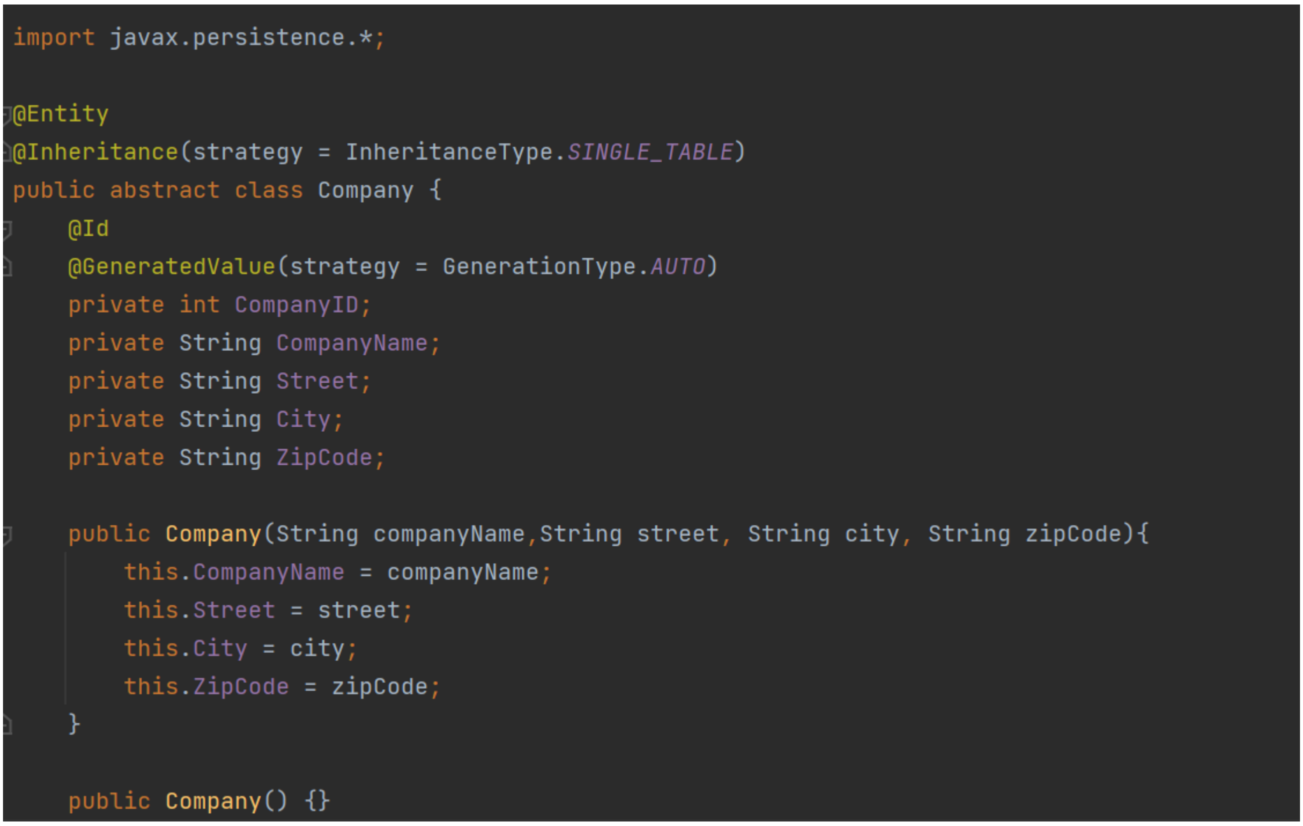
Klasa ***Customer***



```java
package pl.MateuszSkowron;

public class Customer extends Company{

    private String Discount;

    public Customer() {}
    public Customer(String companyName, String street, String city, String zipCode, String discount){
        super(companyName,street,city,zipCode);
        this.Discount = discount;
    }

    public String getDiscount() {
        return Discount;
    }

    public void setDiscount(String discount) {
        Discount = discount;
    }
}
```

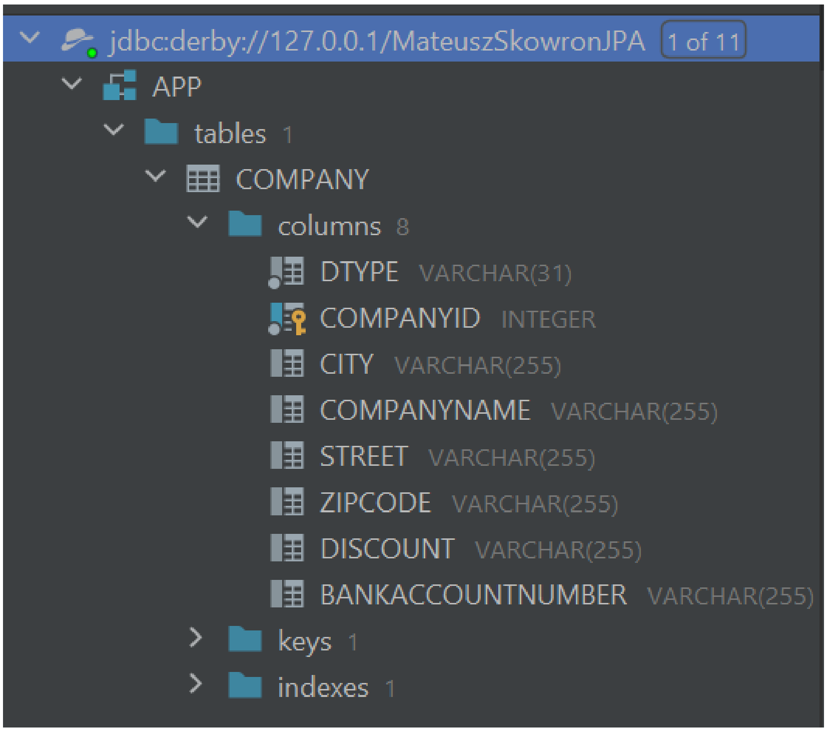
SINGLE_TABLE

Klasa ***Company***



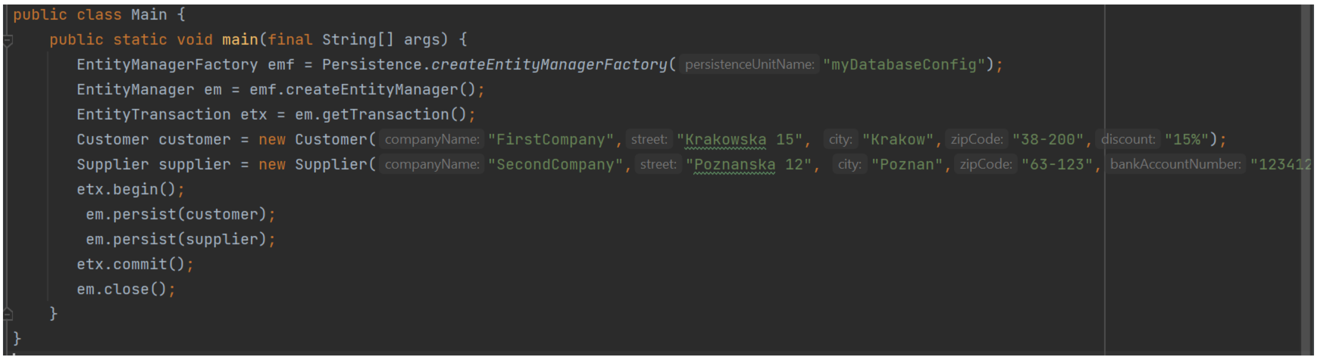
```
import javax.persistence.*;

@Entity
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
public abstract class Company {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int CompanyID;
    private String CompanyName;
    private String Street;
    private String City;
    private String ZipCode;

    public Company(String companyName,String street, String city, String zipCode){
        this.CompanyName = companyName;
        this.Street = street;
        this.City = city;
        this.ZipCode = zipCode;
    }

    public Company() {}
```

Struktura bazy danych



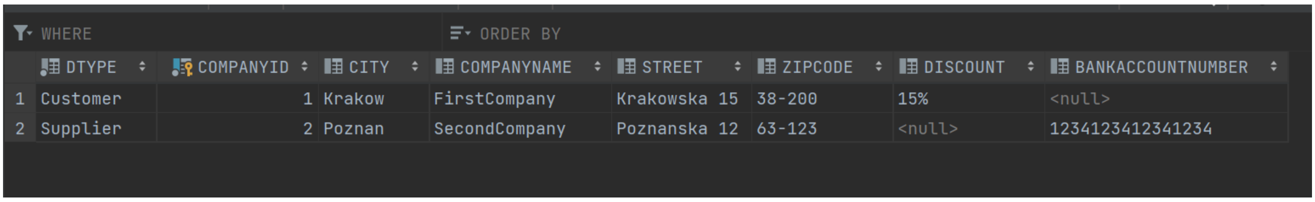
```
jdbc:derby://127.0.0.1/MateuszSkowronJPA  1 of 11
  APP
    tables 1
      COMPANY
        columns 8
          DTYPE VARCHAR(31)
          COMPANYID INTEGER
          CITY VARCHAR(255)
          COMPANYNAME VARCHAR(255)
          STREET VARCHAR(255)
          ZIPCODE VARCHAR(255)
          DISCOUNT VARCHAR(255)
          BANKACCOUNTNUMBER VARCHAR(255)
        keys 1
        indexes 1
```

Dodanie firm



```
public class Main {
    public static void main(final String[] args) {
        EntityManagerFactory emf = Persistence.createEntityManagerFactory( persistenceUnitName: "myDatabaseConfig");
        EntityManager em = emf.createEntityManager();
        EntityTransaction etx = em.getTransaction();
        Customer customer = new Customer( companyName: "FirstCompany", street: "Krakowska 15",  city: "Krakow", zipCode: "38-200", discount: "15%");
        Supplier supplier = new Supplier( companyName: "SecondCompany", street: "Poznanska 12",  city: "Poznan", zipCode: "63-123", bankAccountNumber: "123412
        etx.begin();
         em.persist(customer);
         em.persist(supplier);
        etx.commit();
        em.close();
    }
}
```

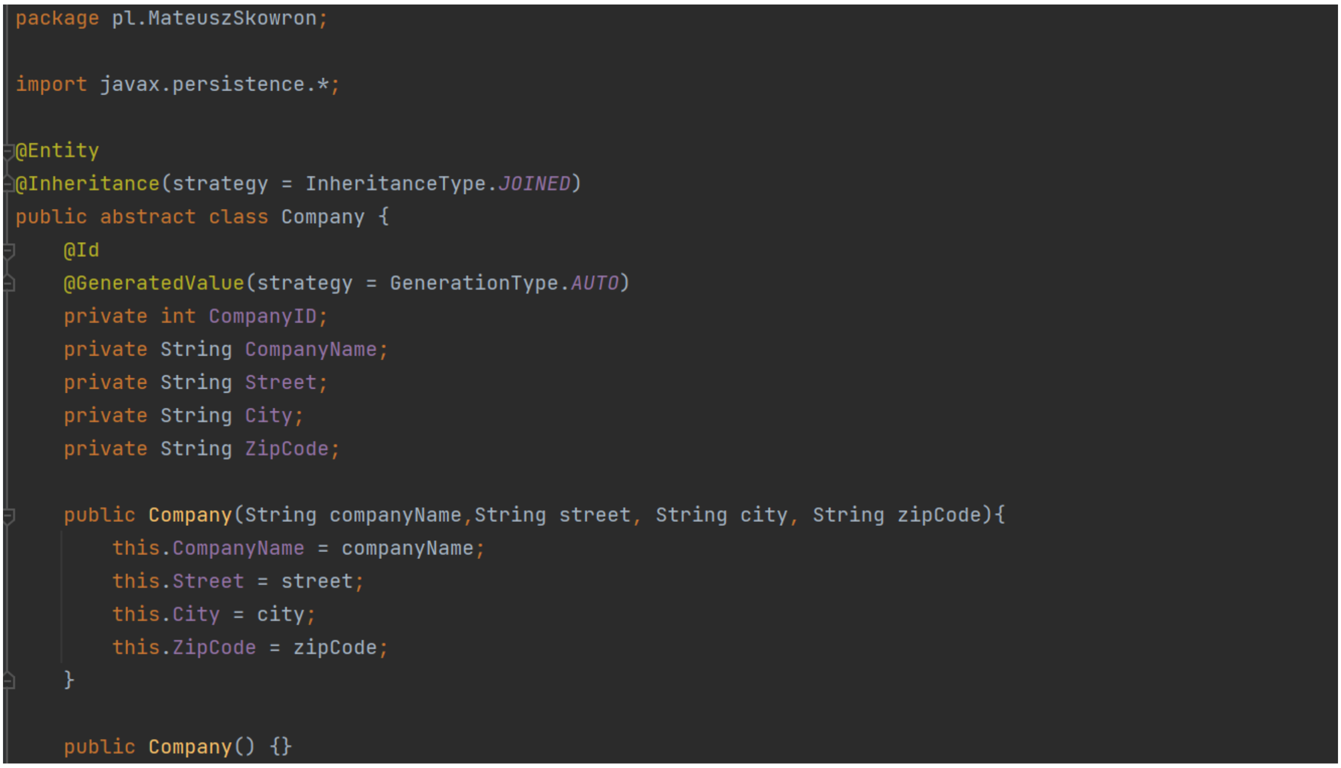
Tabela ***COMPANY***



	DTYPE	COMPANYID	CITY	COMPANYNAME	STREET	ZIPCODE	DISCOUNT	BANKACCOUNTNUMBER
1	Customer	1	Krakow	FirstCompany	Krakowska 15	38-200	15%	<null>
2	Supplier	2	Poznan	SecondCompany	Poznanska 12	63-123	<null>	1234123412341234

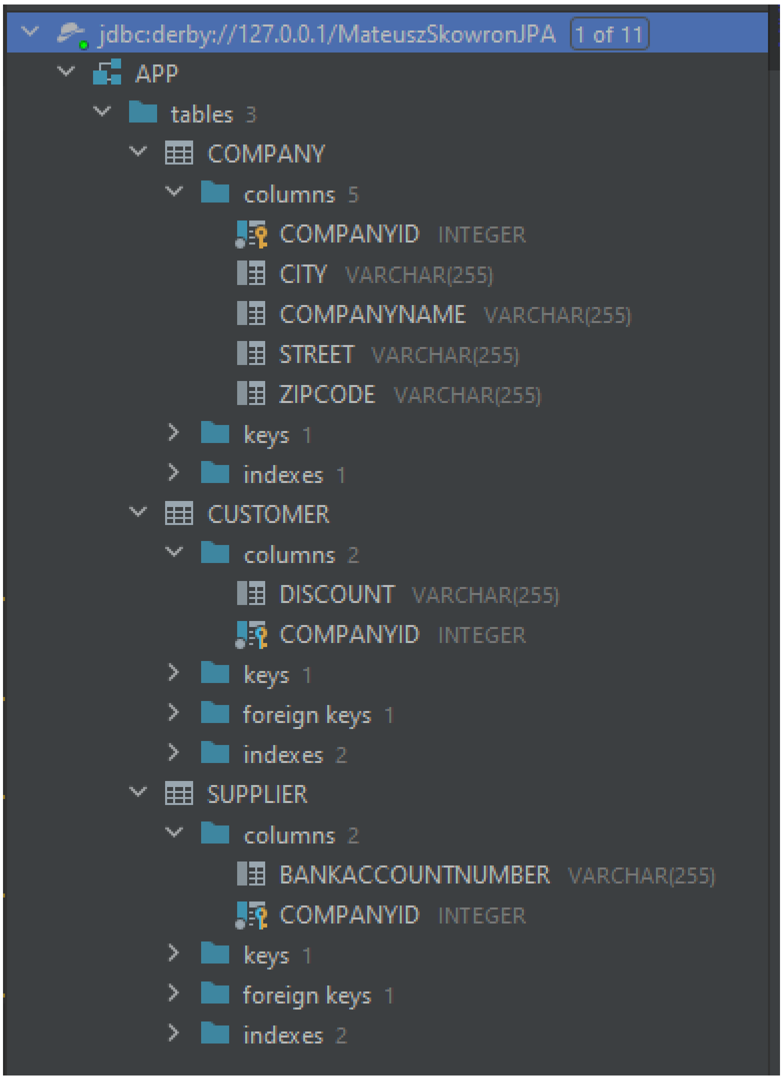
JOINED

Klasa ***Company***



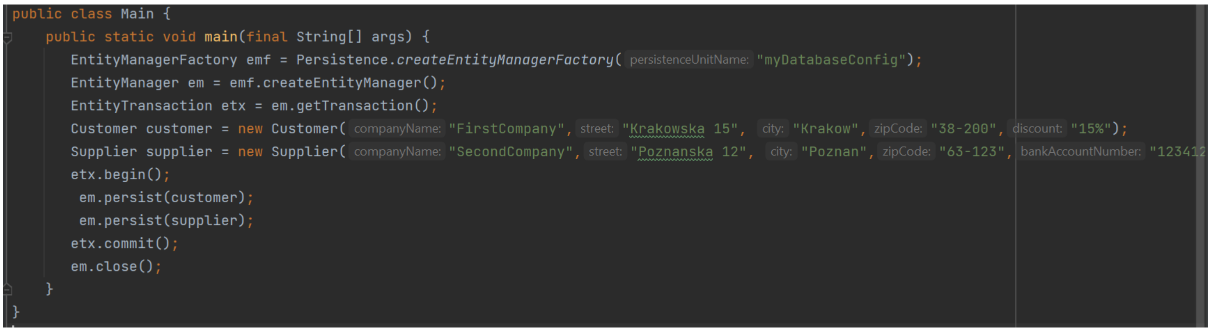
```
package pl.MateuszSkowron;

import javax.persistence.*;

@Entity
@Inheritance(strategy = InheritanceType.JOINED)
public abstract class Company {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int CompanyID;
    private String CompanyName;
    private String Street;
    private String City;
    private String ZipCode;

    public Company(String companyName,String street, String city, String zipCode){
        this.CompanyName = companyName;
        this.Street = street;
        this.City = city;
        this.ZipCode = zipCode;
    }

    public Company() {}
```

Struktura bazy danych



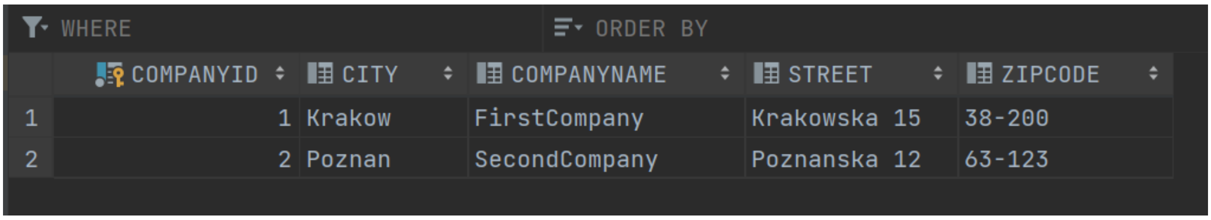
```
jdbc:derby://127.0.0.1/MateuszSkowronJPA  1 of 11
  APP
    tables 3
      COMPANY
        columns 5
          COMPANYID  INTEGER
          CITY  VARCHAR(255)
          COMPANYNAME  VARCHAR(255)
          STREET  VARCHAR(255)
          ZIPCODE  VARCHAR(255)
        keys 1
        indexes 1
      CUSTOMER
        columns 2
          DISCOUNT  VARCHAR(255)
          COMPANYID  INTEGER
        keys 1
        foreign keys 1
        indexes 2
      SUPPLIER
        columns 2
          BANKACCOUNTNUMBER  VARCHAR(255)
          COMPANYID  INTEGER
        keys 1
        foreign keys 1
        indexes 2
```

Dodanie firm



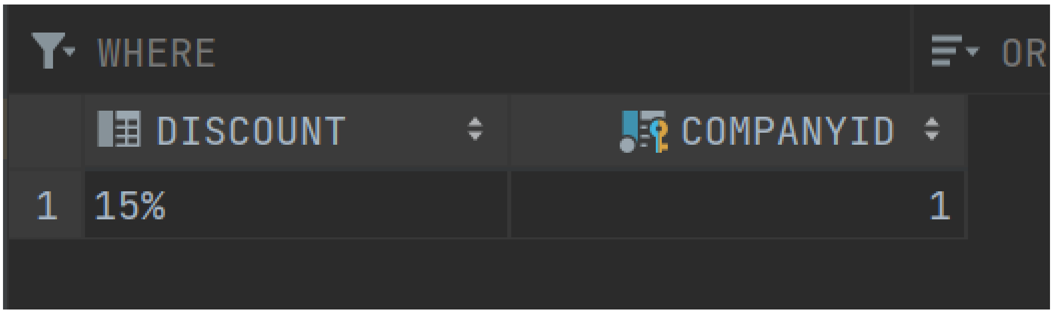
```java
public class Main {
    public static void main(final String[] args) {
        EntityManagerFactory emf = Persistence.createEntityManagerFactory( persistenceUnitName: "myDatabaseConfig");
        EntityManager em = emf.createEntityManager();
        EntityTransaction etx = em.getTransaction();
        Customer customer = new Customer( companyName: "FirstCompany", street: "Krakowska 15",  city: "Krakow", zipCode: "38-200", discount: "15%");
        Supplier supplier = new Supplier( companyName: "SecondCompany", street: "Poznanska 12",  city: "Poznan", zipCode: "63-123", bankAccountNumber: "123412
        etx.begin();
         em.persist(customer);
         em.persist(supplier);
        etx.commit();
        em.close();
    }
}
```

Tabela ***COMPANY***



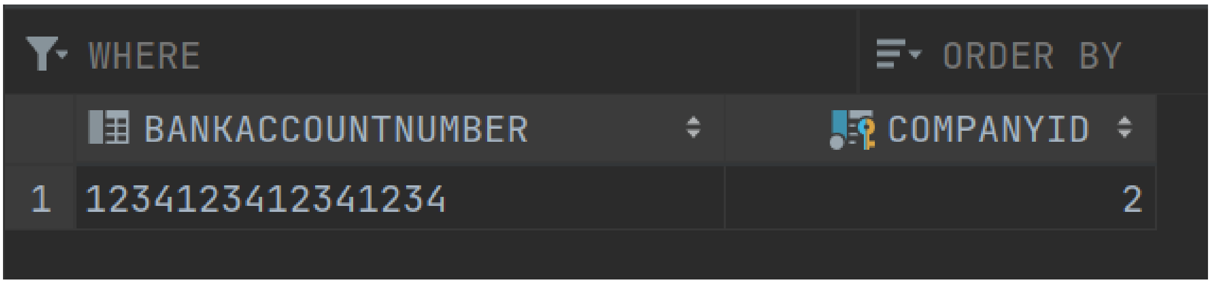
	COMPANYID	CITY	COMPANYNAME	STREET	ZIPCODE
1	1	Krakow	FirstCompany	Krakowska 15	38-200
2	2	Poznan	SecondCompany	Poznanska 12	63-123

Tabela ***CUSTOMER***



	DISCOUNT	COMPANYID
1	15%	1

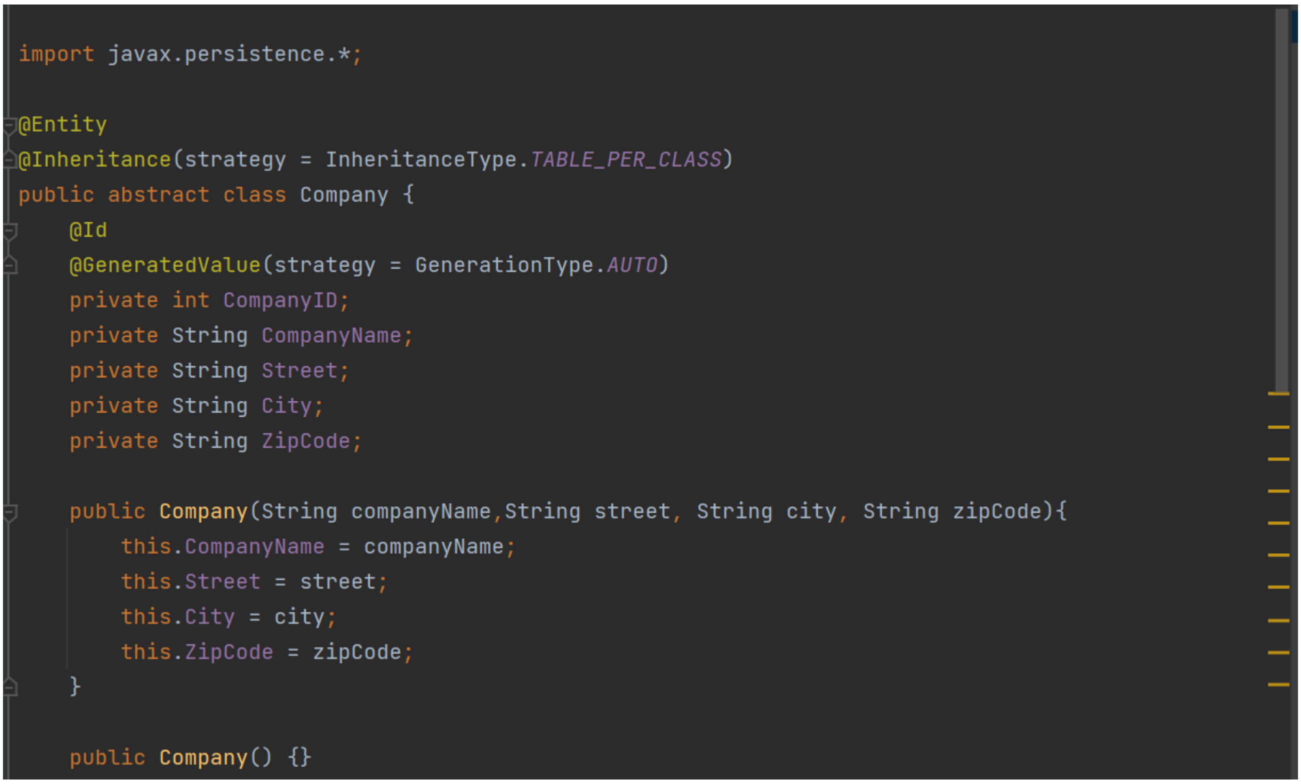
Tabela ***SUPPLIER***



	BANKACCOUNTNUMBER	COMPANYID
1	1234123412341234	2

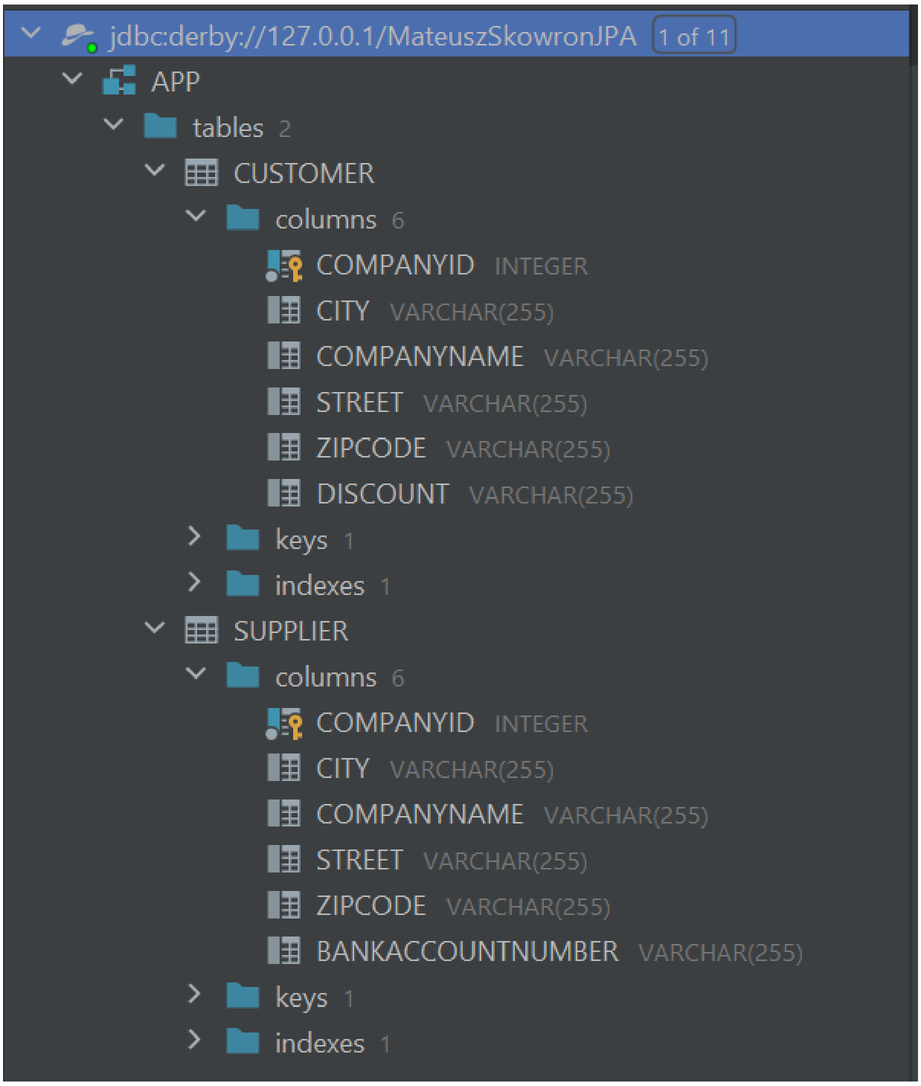
TABLE_PER_CLASS

Klasa ***Company***



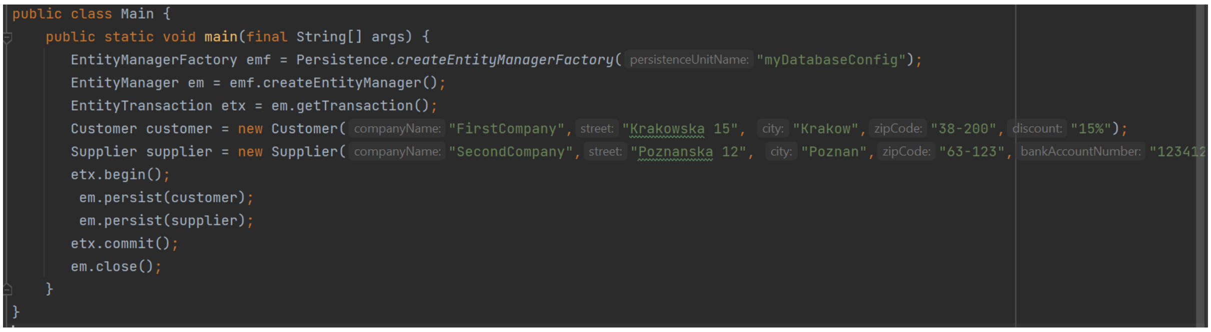
```java
import javax.persistence.*;

@Entity
@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)
public abstract class Company {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int CompanyID;
    private String CompanyName;
    private String Street;
    private String City;
    private String ZipCode;

    public Company(String companyName,String street, String city, String zipCode){
        this.CompanyName = companyName;
        this.Street = street;
        this.City = city;
        this.ZipCode = zipCode;
    }

    public Company() {}
```

Struktura bazy danych



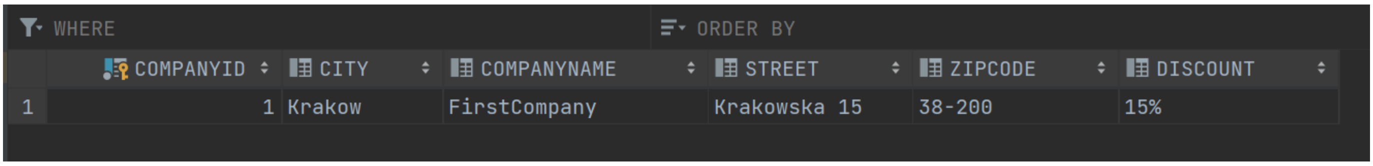
```
jdbc:derby://127.0.0.1/MateuszSkowronJPA  1 of 11
  APP
    tables 2
      CUSTOMER
        columns 6
          COMPANYID INTEGER
          CITY VARCHAR(255)
          COMPANYNAME VARCHAR(255)
          STREET VARCHAR(255)
          ZIPCODE VARCHAR(255)
          DISCOUNT VARCHAR(255)
        keys 1
        indexes 1
      SUPPLIER
        columns 6
          COMPANYID INTEGER
          CITY VARCHAR(255)
          COMPANYNAME VARCHAR(255)
          STREET VARCHAR(255)
          ZIPCODE VARCHAR(255)
          BANKACCOUNTNUMBER VARCHAR(255)
        keys 1
        indexes 1
```

Dodanie firm



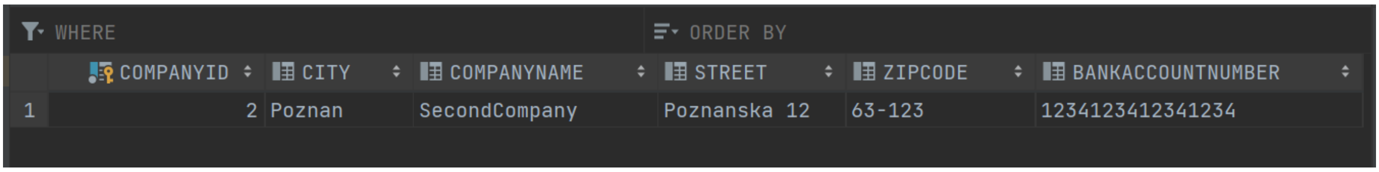
```java
public class Main {
    public static void main(final String[] args) {
        EntityManagerFactory emf = Persistence.createEntityManagerFactory( persistenceUnitName: "myDatabaseConfig");
        EntityManager em = emf.createEntityManager();
        EntityTransaction etx = em.getTransaction();
        Customer customer = new Customer( companyName: "FirstCompany", street: "Krakowska 15",  city: "Krakow", zipCode: "38-200", discount: "15%");
        Supplier supplier = new Supplier( companyName: "SecondCompany", street: "Poznanska 12",  city: "Poznan", zipCode: "63-123", bankAccountNumber: "123412
        etx.begin();
         em.persist(customer);
         em.persist(supplier);
        etx.commit();
        em.close();
    }
}
```

Tabela ***CUSTOMER***



	COMPANYID	CITY	COMPANYNAME	STREET	ZIPCODE	DISCOUNT
1	1	Krakow	FirstCompany	Krakowska 15	38-200	15%

Tabela ***SUPPLIER***



	COMPANYID	CITY	COMPANYNAME	STREET	ZIPCODE	BANKACCOUNTNUMBER
1	2	Poznan	SecondCompany	Poznanska 12	63-123	1234123412341234