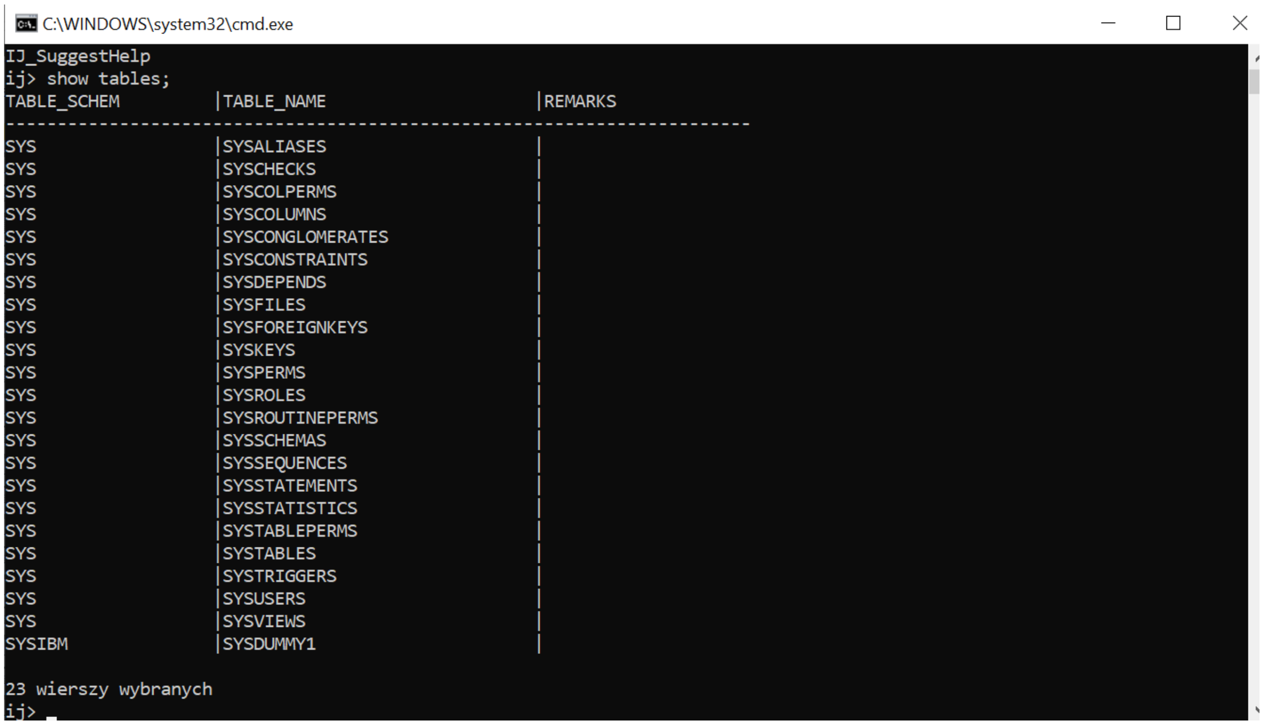
Bazy danych - Hibernate

Mateusz Surjak

305428

1 Konfiguracja

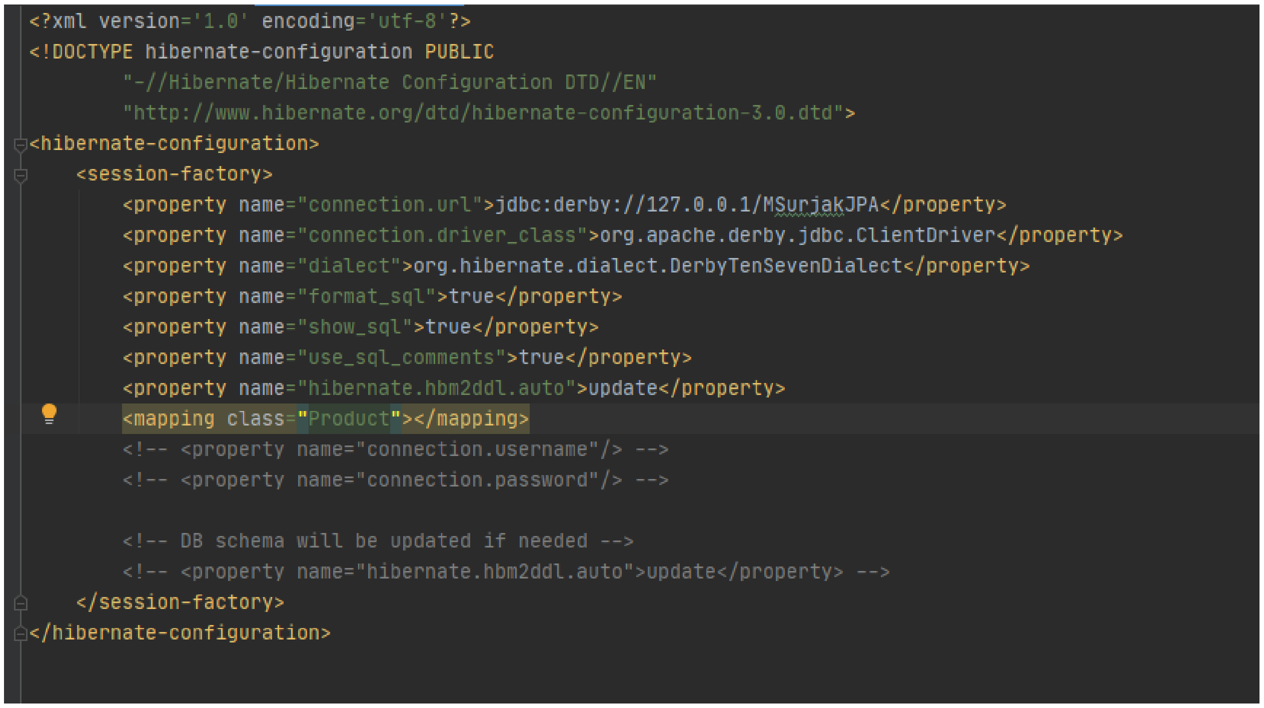
Skonfigurowałem środowisko zgodnie z poleceniem



```
IJ_SuggestHelp
ij> show tables;
TABLE_SCHEM         |TABLE_NAME                    |REMARKS
------------------------------------------------------------------------
SYS                 |SYSALIASES                    |
SYS                 |SYSCHECKS                     |
SYS                 |SYSCOLPERMS                   |
SYS                 |SYSCOLUMNS                    |
SYS                 |SYSCONGLOMERATES              |
SYS                 |SYSCONSTRAINTS                |
SYS                 |SYSDEPENDS                    |
SYS                 |SYSFILES                      |
SYS                 |SYSFOREIGNKEYS                |
SYS                 |SYSKEYS                       |
SYS                 |SYSPERMS                      |
SYS                 |SYSROLES                      |
SYS                 |SYSROUTINEPERMS               |
SYS                 |SYSSCHEMAS                    |
SYS                 |SYSSEQUENCES                  |
SYS                 |SYSSTATEMENTS                 |
SYS                 |SYSSTATISTICS                 |
SYS                 |SYSTABLEPERMS                 |
SYS                 |SYSTABLES                     |
SYS                 |SYSTRIGGERS                   |
SYS                 |SYSUSERS                      |
SYS                 |SYSVIEWS                      |
SYSIBM              |SYSDUMMY1                     |

23 wierszy wybranych
ij>
```

Działające środowisko

Skonfigurowałem również środowisko w którym będę wykonywał kolejne zadania.

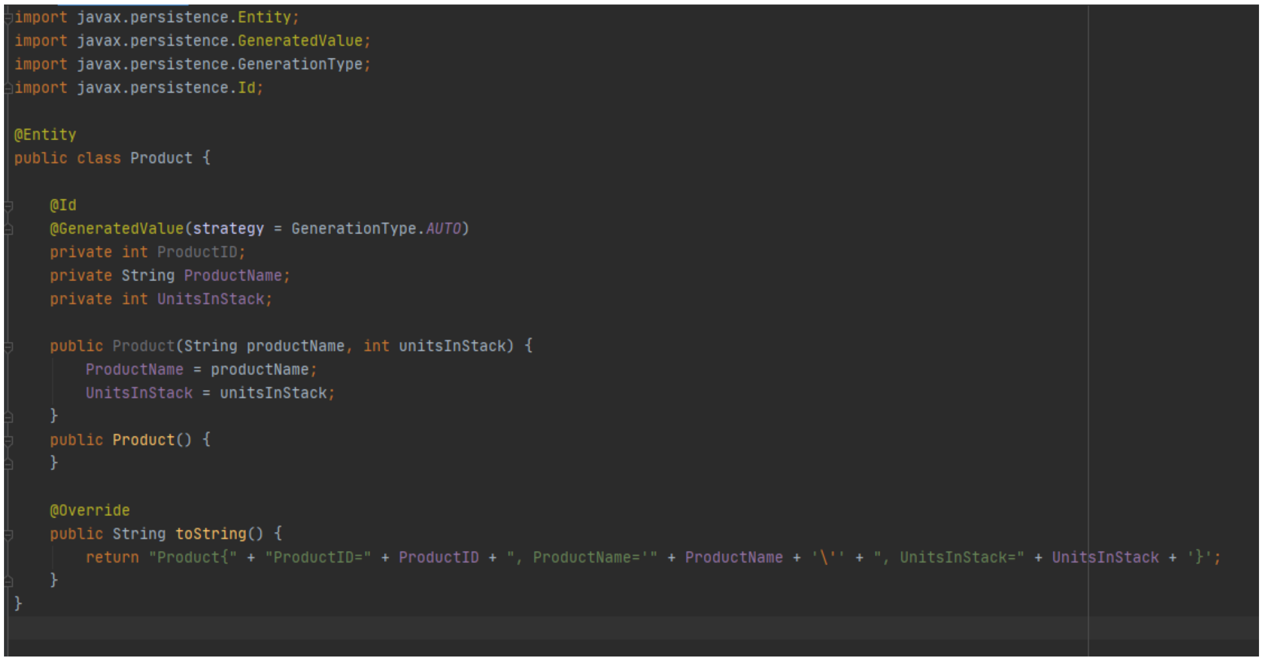


```xml
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="connection.url">jdbc:derby://127.0.0.1/MSurjakJPA</property>
        <property name="connection.driver_class">org.apache.derby.jdbc.ClientDriver</property>
        <property name="dialect">org.hibernate.dialect.DerbyTenSevenDialect</property>
        <property name="format_sql">true</property>
        <property name="show_sql">true</property>
        <property name="use_sql_comments">true</property>
        <property name="hibernate.hbm2ddl.auto">update</property>
        <mapping class="Product"></mapping>
        <!-- <property name="connection.username"/> -->
        <!-- <property name="connection.password"/> -->

        <!-- DB schema will be updated if needed -->
        <!-- <property name="hibernate.hbm2ddl.auto">update</property> -->
    </session-factory>
</hibernate-configuration>
```

Konfiguracja Hibernate'a

2 W maine stwórz przykładowy produkt i utrwal go w BD z wykorzystaniem Hibernate'a

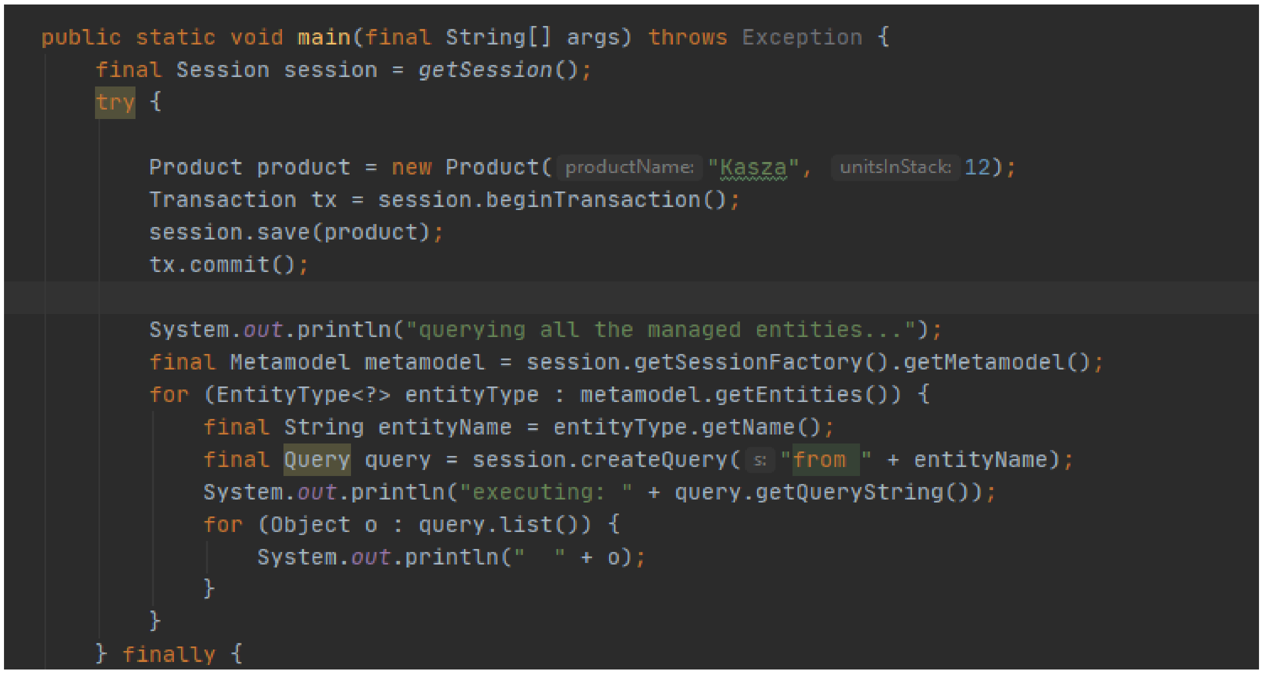
Utworzyłem klasę Product i dodałęm konieczne adnotacje aby klasa mogłą zostać zmapowana do bazy danych przez Hibernate'a. Dodałem również <mapping> w pliku konfiguracyjnym.



```java
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;
    private String ProductName;
    private int UnitsInStack;

    public Product(String productName, int unitsInStack) {
        ProductName = productName;
        UnitsInStack = unitsInStack;
    }
    public Product() {
    }

    @Override
    public String toString() {
        return "Product{" + "ProductID=" + ProductID + ", ProductName='" + ProductName + '\'' + ", UnitsInStack=" + UnitsInStack + '}';
    }
}
```

Klasa Product

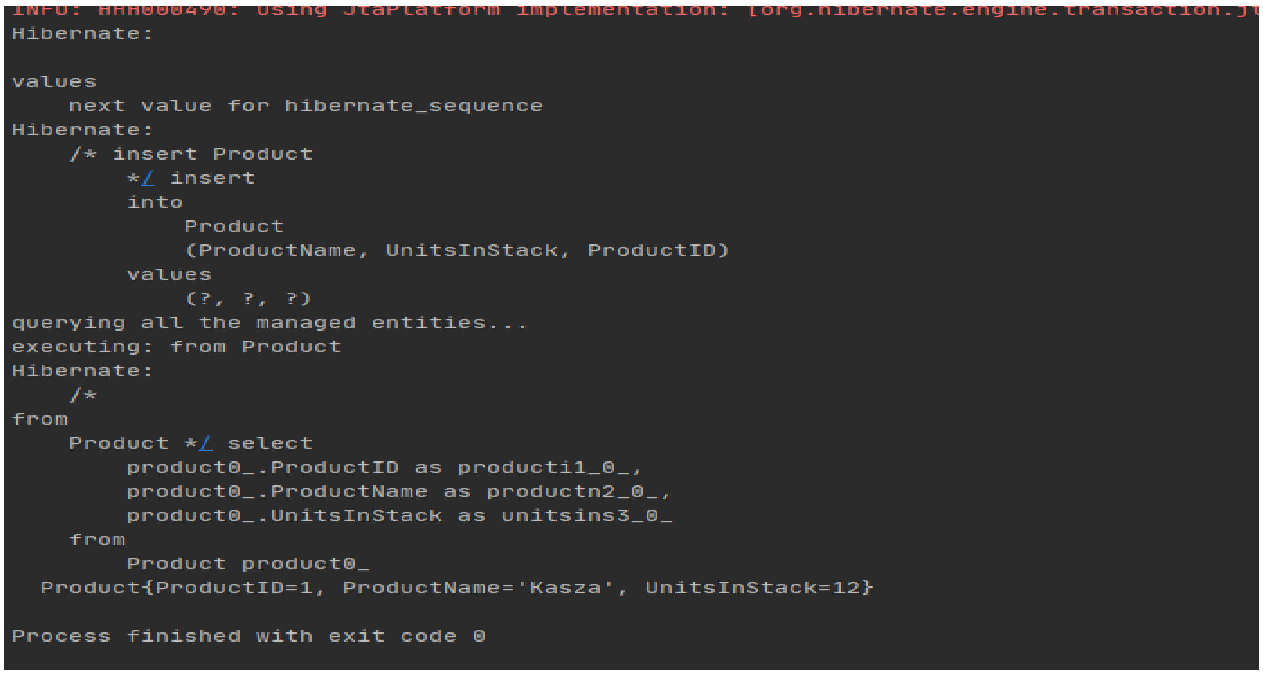
Utworzyłem produkt i zapisałem go w bazie.



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

        Product product = new Product( productName: "Kasza",  unitsInStack: 12);
        Transaction tx = session.beginTransaction();
        session.save(product);
        tx.commit();

        System.out.println("querying all the managed entities...");
        final Metamodel metamodel = session.getSessionFactory().getMetamodel();
        for (EntityType<?> entityType : metamodel.getEntities()) {
            final String entityName = entityType.getName();
            final Query query = session.createQuery( s: "from " + entityName);
            System.out.println("executing: " + query.getQueryString());
            for (Object o : query.list()) {
                System.out.println("  " + o);
            }
        }
    } finally {
```

Funkcja main

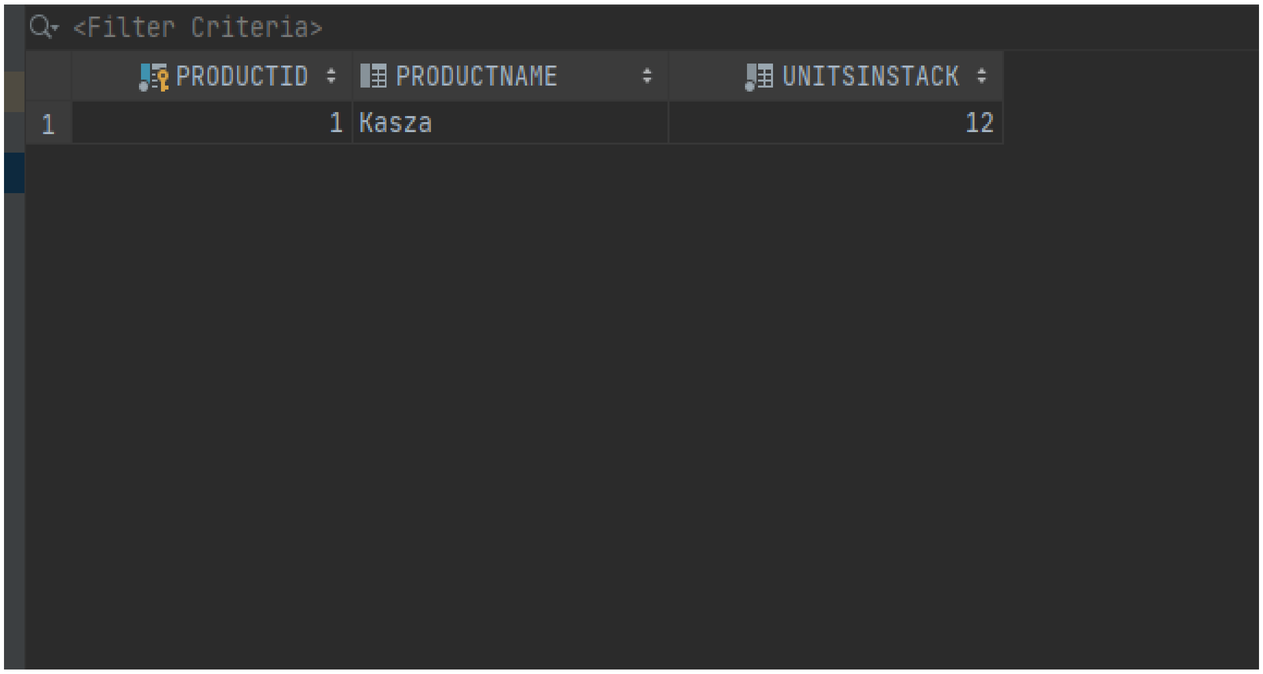
Poniżej prezentuję logi wywołań Hibernate'a



```
INFO: HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jt
Hibernate:

values
    next value for hibernate_sequence
Hibernate:
    /* insert Product
        */ insert
        into
            Product
            (ProductName, UnitsInStack, ProductID)
        values
            (?, ?, ?)
querying all the managed entities...
executing: from Product
Hibernate:
    /*
from
    Product */ select
        product0_.ProductID as producti1_0_,
        product0_.ProductName as productn2_0_,
        product0_.UnitsInStack as unitsins3_0_
    from
        Product product0_
  Product{ProductID=1, ProductName='Kasza', UnitsInStack=12}

Process finished with exit code 0
```

Wynik wywołania

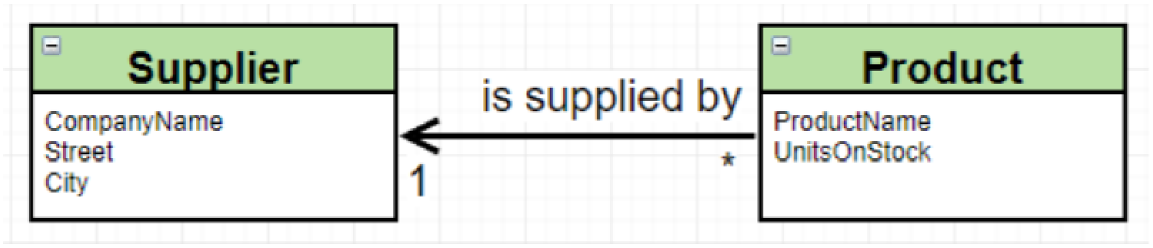
Piniżej prezentuję tabelę Product w bazie danych.

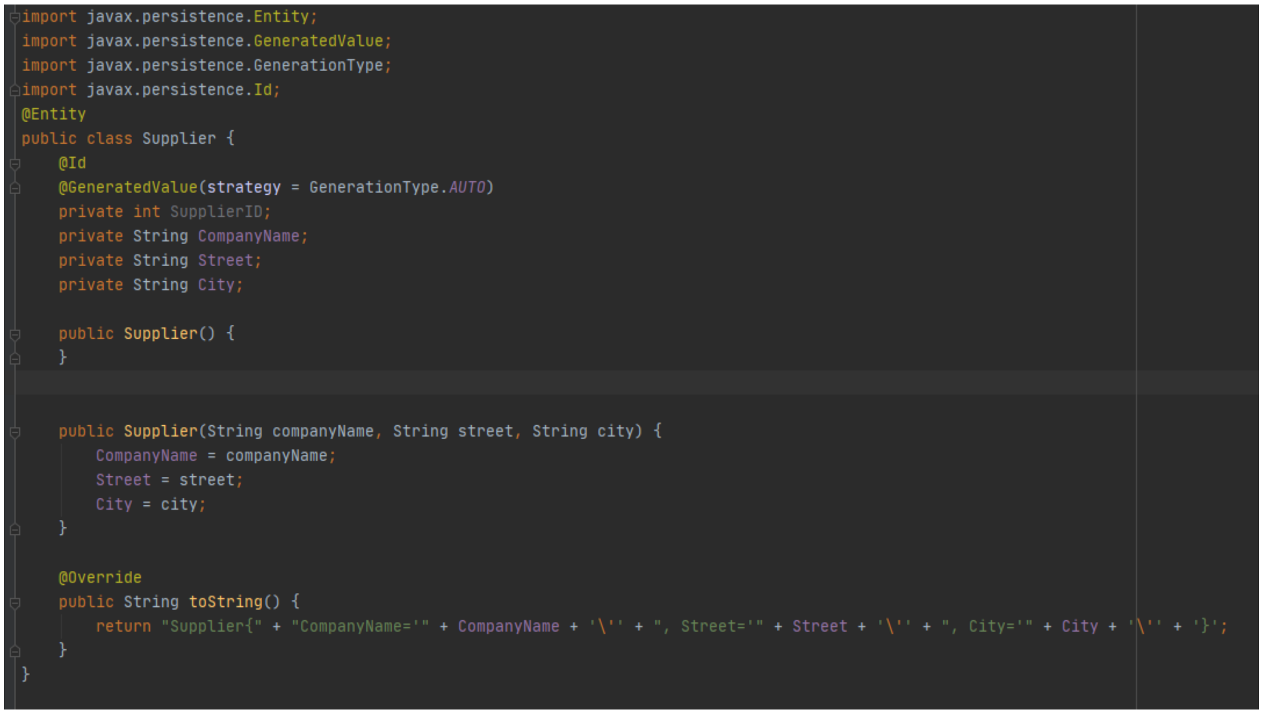


	PRODUCTID	PRODUCTNAME	UNITSINSTACK
1	1	Kasza	12

Tabela products w bazie danych

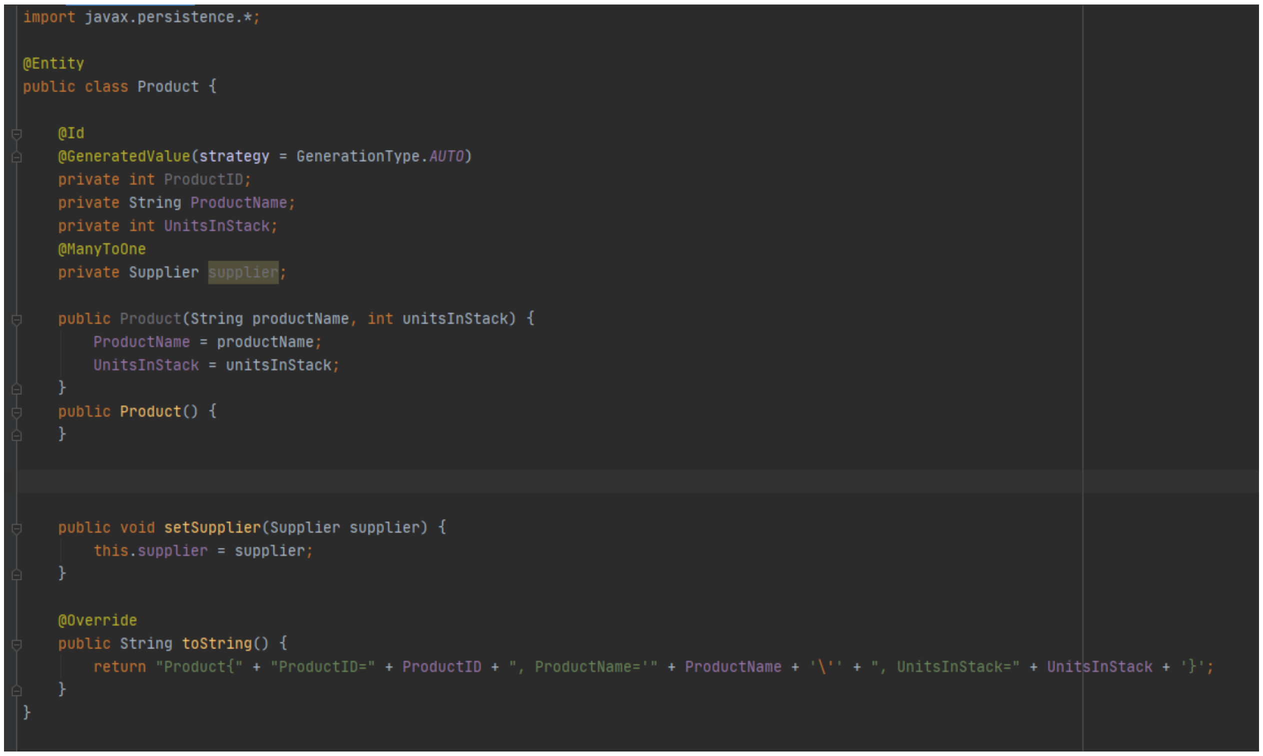
3 Zmodyfikuj model wprowadzając pojęcie Dostawcy jak poniżej





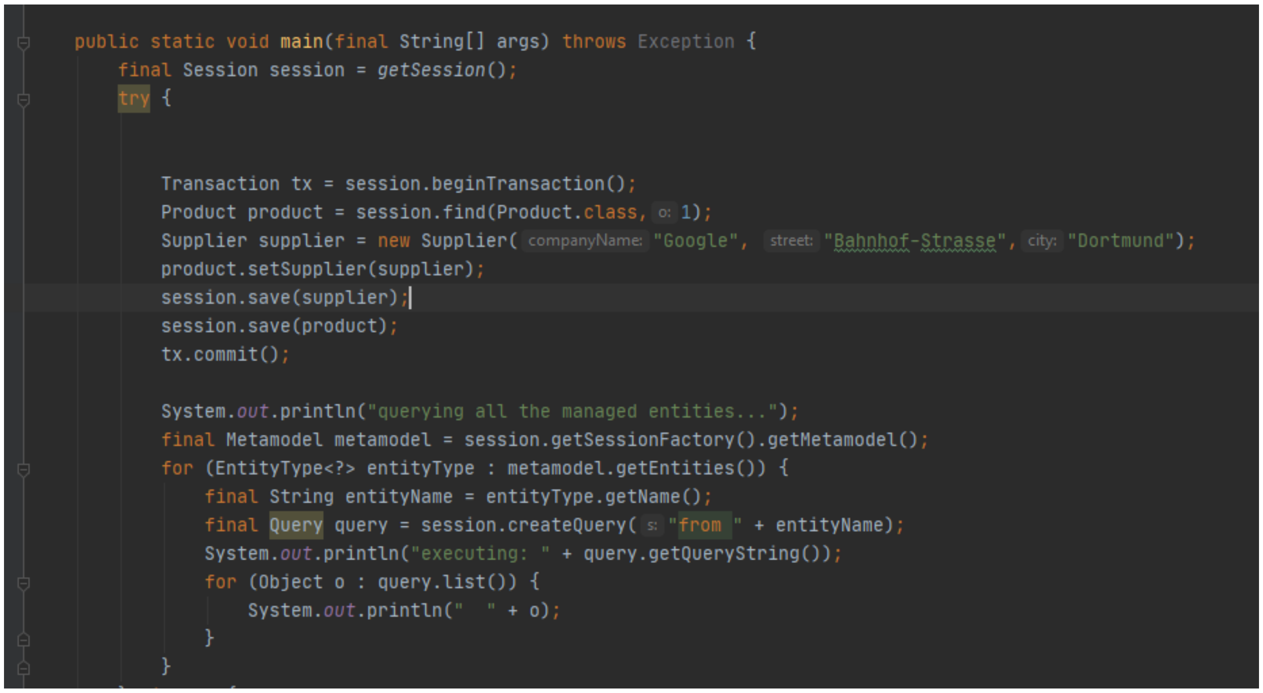
```java
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
@Entity
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int SupplierID;
    private String CompanyName;
    private String Street;
    private String City;

    public Supplier() {
    }


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

    @Override
    public String toString() {
        return "Supplier{" + "CompanyName='" + CompanyName + '\'' + ", Street='" + Street + '\'' + ", City='" + City + '\'' + '}';
    }
}
```

Klasa Supplier



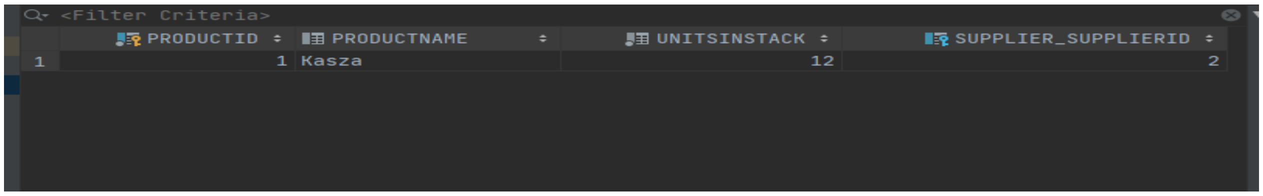
```java
import javax.persistence.*;

@Entity
public class Product {

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

    public Product(String productName, int unitsInStack) {
        ProductName = productName;
        UnitsInStack = unitsInStack;
    }
    public Product() {
    }


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

    @Override
    public String toString() {
        return "Product{" + "ProductID=" + ProductID + ", ProductName='" + ProductName + '\'' + ", UnitsInStack=" + UnitsInStack + '}';
    }
}
```

Klasa Product



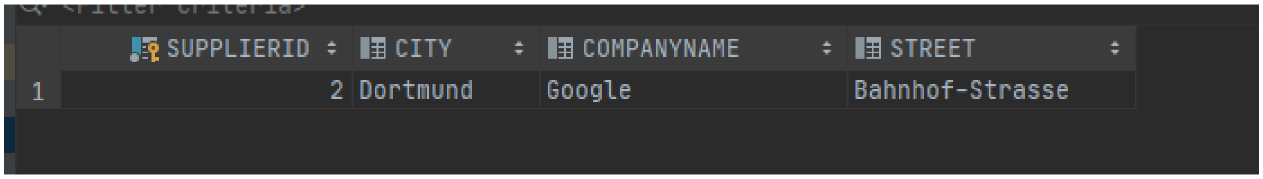
```java
    public static void main(final String[] args) throws Exception {
        final Session session = getSession();
        try {


            Transaction tx = session.beginTransaction();
            Product product = session.find(Product.class, o: 1);
            Supplier supplier = new Supplier( companyName: "Google",  street: "Bahnhof-Strasse", city: "Dortmund");
            product.setSupplier(supplier);
            session.save(supplier);
            session.save(product);
            tx.commit();

            System.out.println("querying all the managed entities...");
            final Metamodel metamodel = session.getSessionFactory().getMetamodel();
            for (EntityType<?> entityType : metamodel.getEntities()) {
                final String entityName = entityType.getName();
                final Query query = session.createQuery( s: "from " + entityName);
                System.out.println("executing: " + query.getQueryString());
                for (Object o : query.list()) {
                    System.out.println("  " + o);
                }
            }
```

Main



	PRODUCTID	PRODUCTNAME	UNITSINSTACK	SUPPLIER_SUPPLIERID
1	1	Kasza	12	2

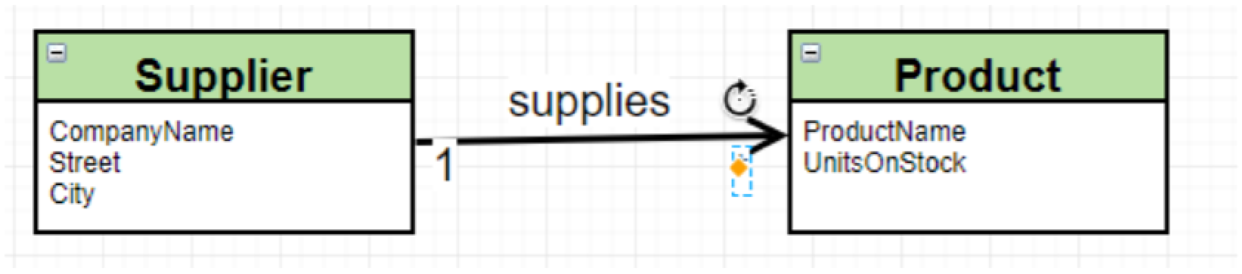
Tabela products



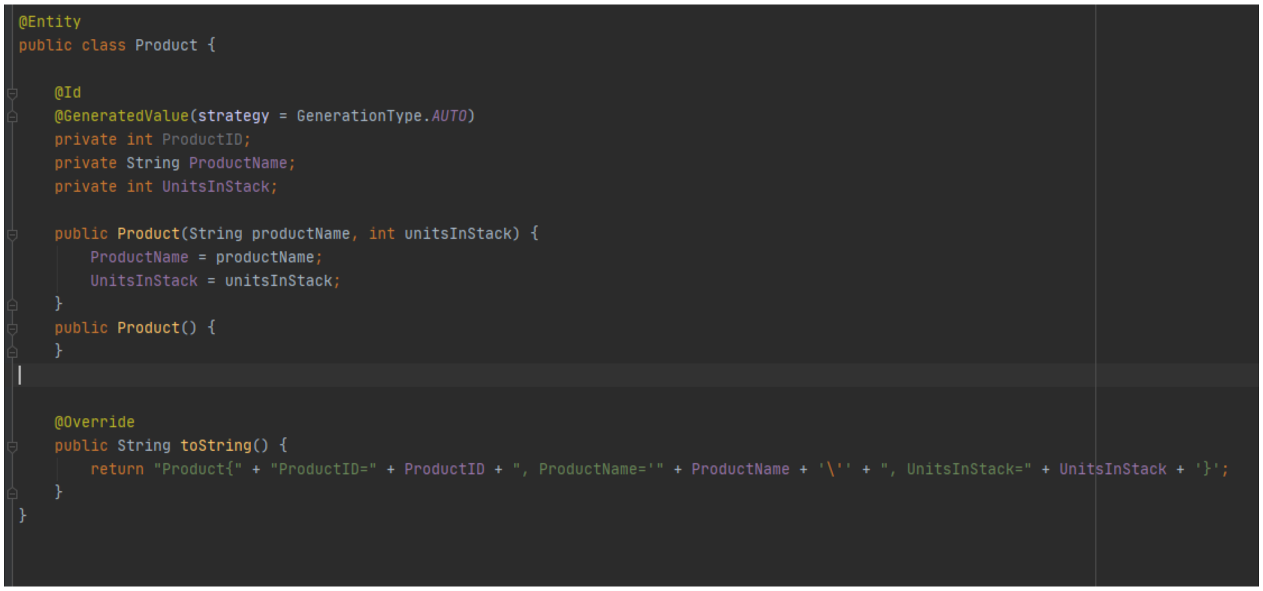
	SUPPLIERID	CITY	COMPANYNAME	STREET
1	2	Dortmund	Google	Bahnhof-Strasse

Tabela suppliers

4 Odwróć relacje zgodnie z poniższym schematem

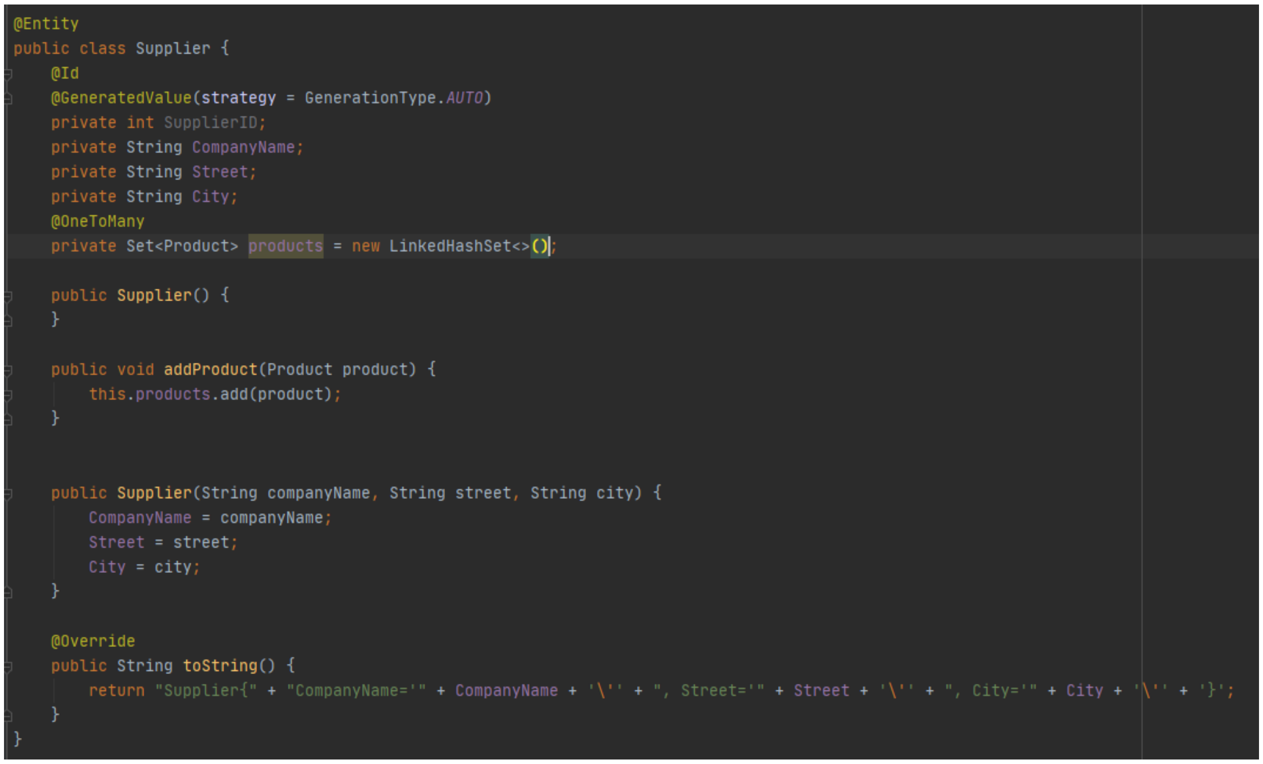


4.1 Z tabelą łącznikową



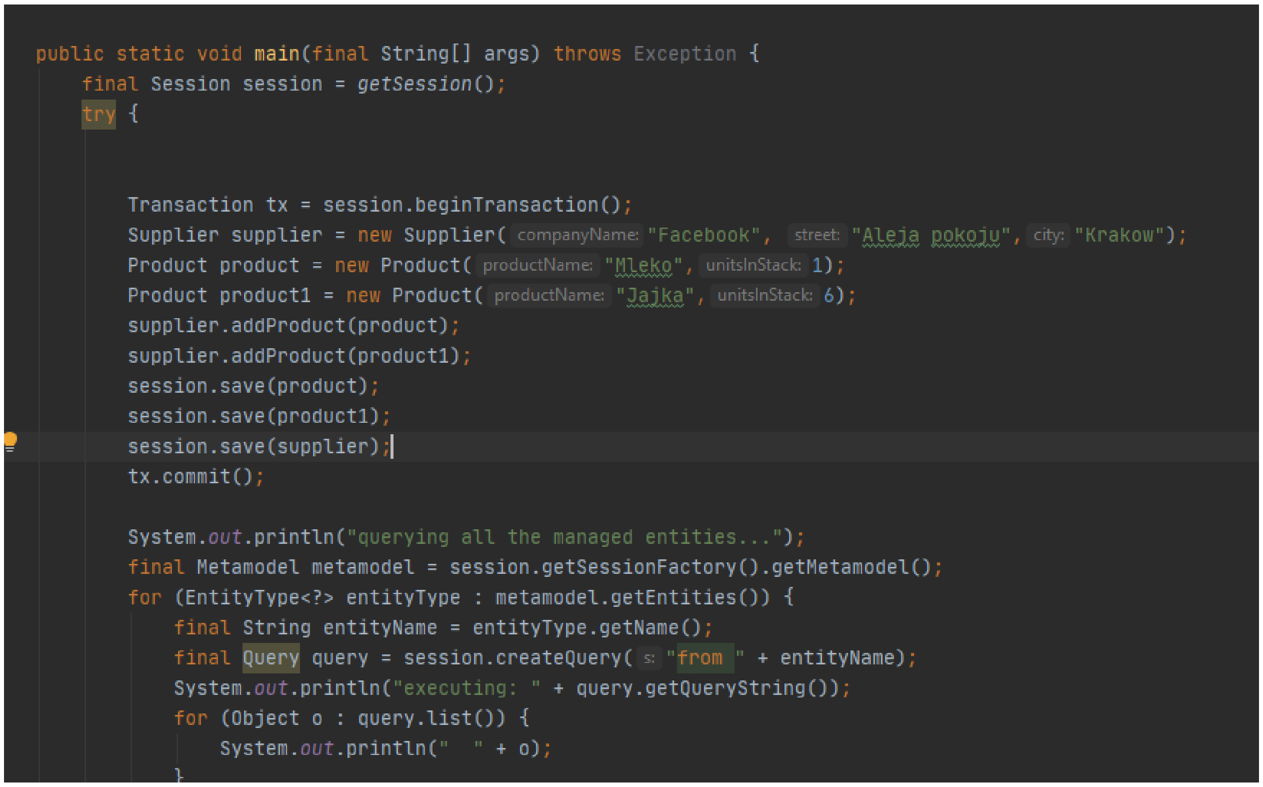
```
@Entity
public class Product {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int ProductID;
    private String ProductName;
    private int UnitsInStack;

    public Product(String productName, int unitsInStack) {
        ProductName = productName;
        UnitsInStack = unitsInStack;
    }
    public Product() {
    }


    @Override
    public String toString() {
        return "Product{" + "ProductID=" + ProductID + ", ProductName='" + ProductName + '\'' + ", UnitsInStack=" + UnitsInStack + '}';
    }
}
```

Klasa Product



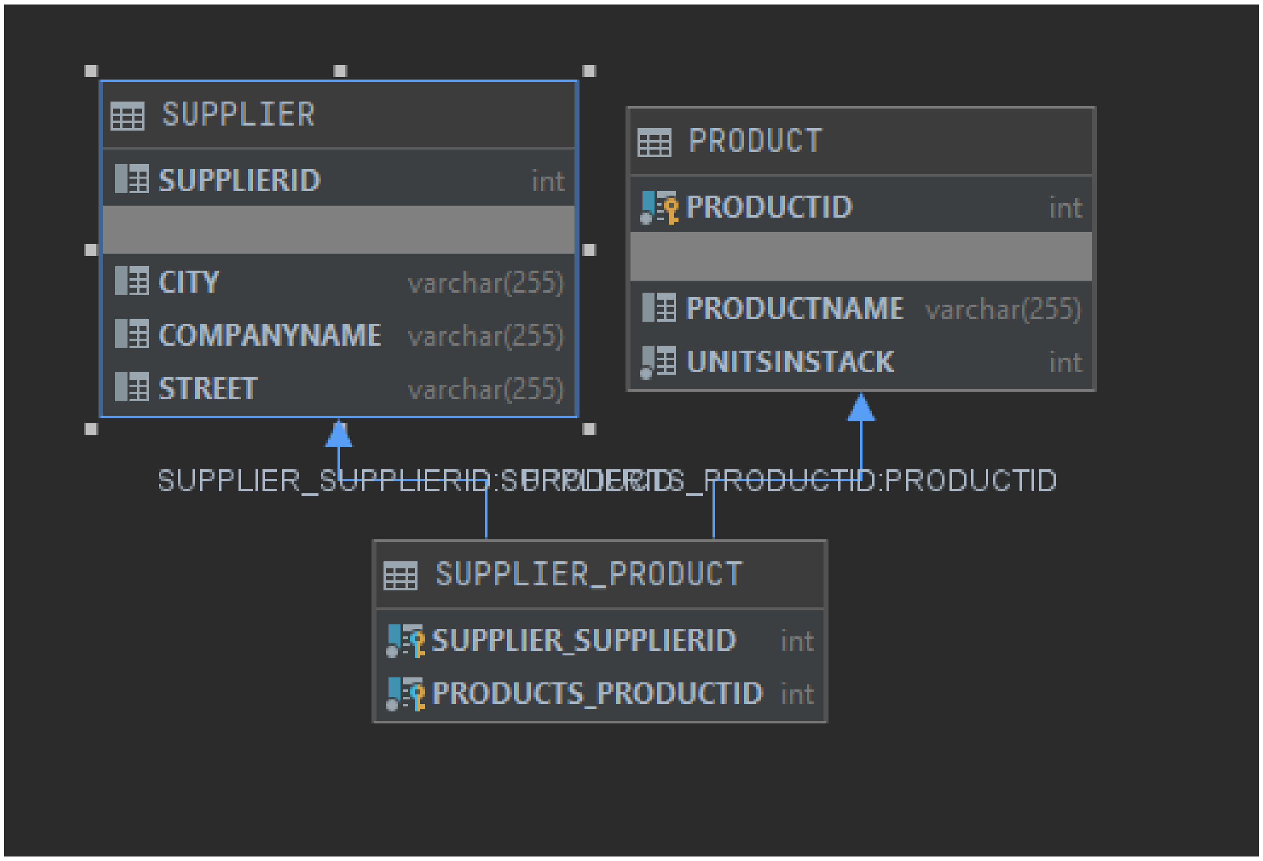
```java
@Entity
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int SupplierID;
    private String CompanyName;
    private String Street;
    private String City;
    @OneToMany
    private Set<Product> products = new LinkedHashSet<>();

    public Supplier() {
    }

    public void addProduct(Product product) {
        this.products.add(product);
    }


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

    @Override
    public String toString() {
        return "Supplier{" + "CompanyName='" + CompanyName + '\'' + ", Street='" + Street + '\'' + ", City='" + City + '\'' + '}';
    }
}
```

Klasa Supplier



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


        Transaction tx = session.beginTransaction();
        Supplier supplier = new Supplier( companyName: "Facebook",  street: "Aleja pokoju", city: "Krakow");
        Product product = new Product( productName: "Mleko", unitsInStack: 1);
        Product product1 = new Product( productName: "Jajka", unitsInStack: 6);
        supplier.addProduct(product);
        supplier.addProduct(product1);
        session.save(product);
        session.save(product1);
        session.save(supplier);
        tx.commit();

        System.out.println("querying all the managed entities...");
        final Metamodel metamodel = session.getSessionFactory().getMetamodel();
        for (EntityType<?> entityType : metamodel.getEntities()) {
            final String entityName = entityType.getName();
            final Query query = session.createQuery( s: "from " + entityName);
            System.out.println("executing: " + query.getQueryString());
            for (Object o : query.list()) {
                System.out.println("  " + o);
            }
```

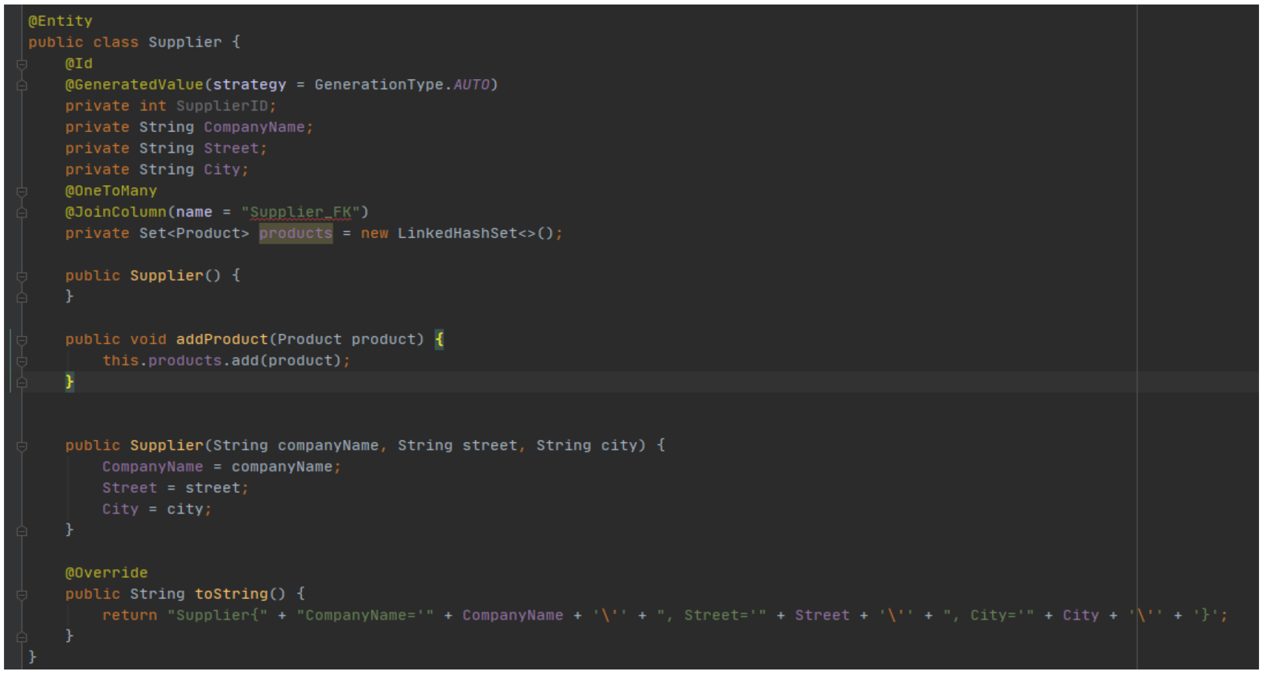
Main



Schemat bazy

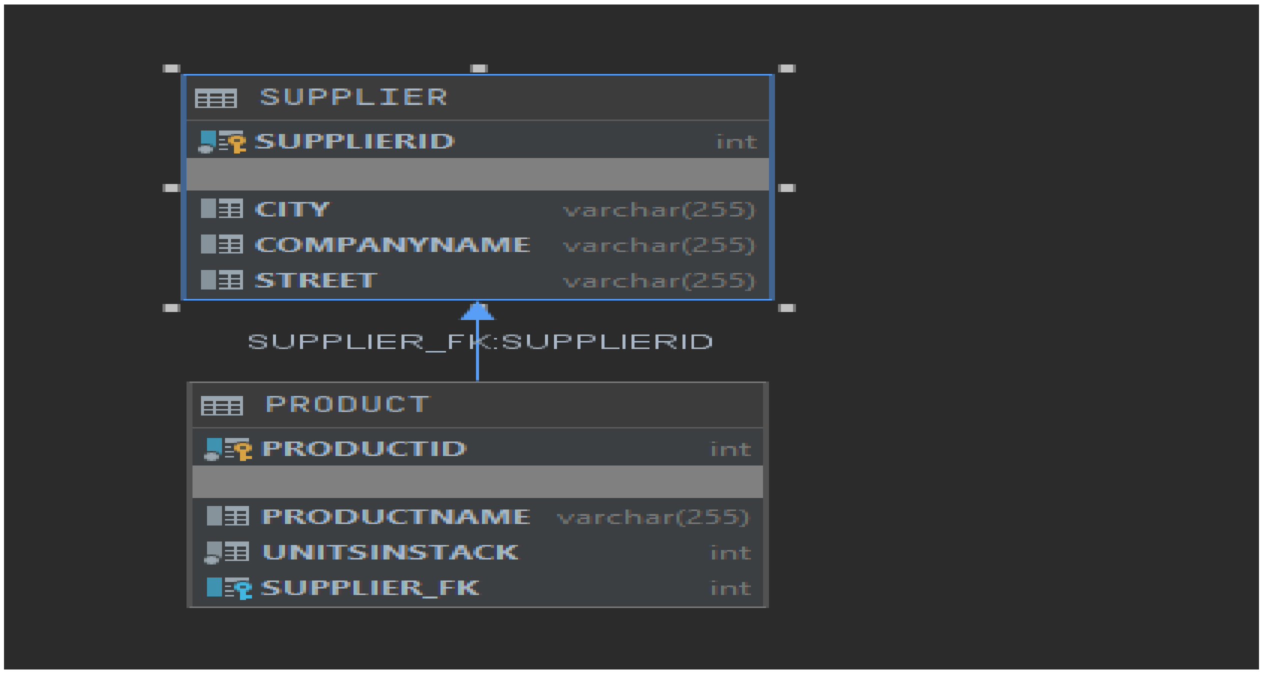
4.2 Bez tabeli łącznikowej

Informuję iż od tego momentu w kodzie przy @JoinColumn pojawia się czerwone podkreślenie, nie mogłem się go pozbyć ale wszystko z nim działa jak powinno, także należy je zignorować.



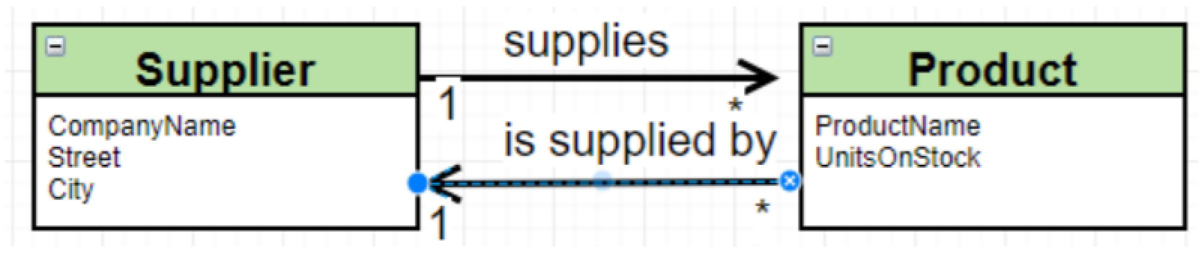
```java
@Entity
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int SupplierID;
    private String CompanyName;
    private String Street;
    private String City;
    @OneToMany
    @JoinColumn(name = "Supplier_FK")
    private Set<Product> products =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

    public Supplier() {
    }

    public void addProduct(Product product) {
        this.products.add(product);
    }


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

    @Override
    public String toString() {
        return "Supplier{" + "CompanyName='" + CompanyName + '\'' + ", Street='" + Street + '\'' + ", City='" + City + '\'' + '}';
    }
}
```

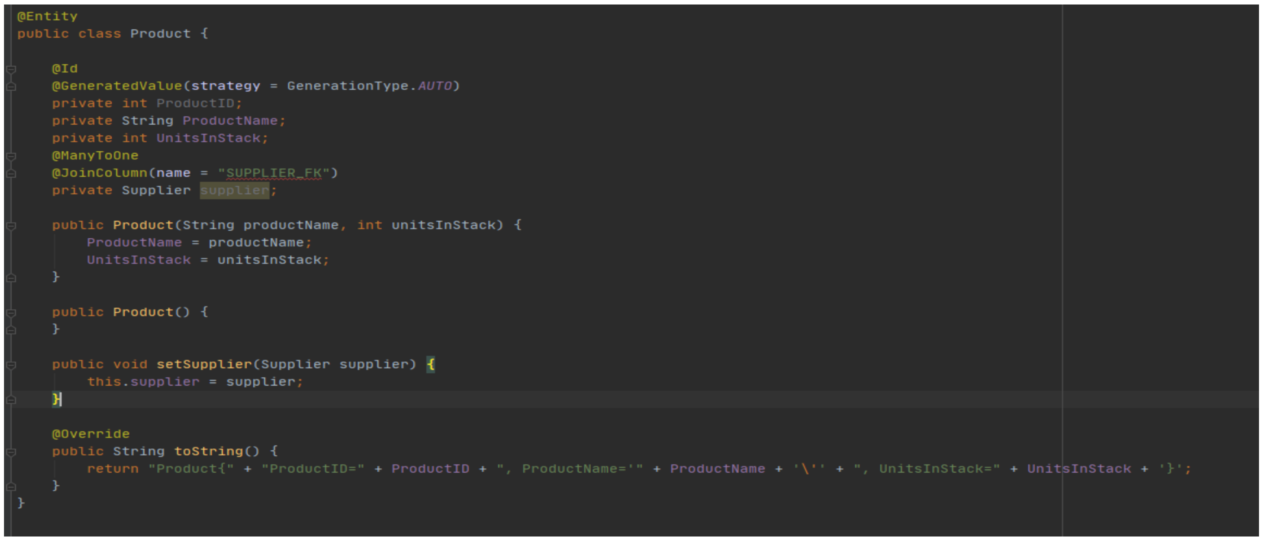
Zmiana w klasie Supplier



Wyglad bazy

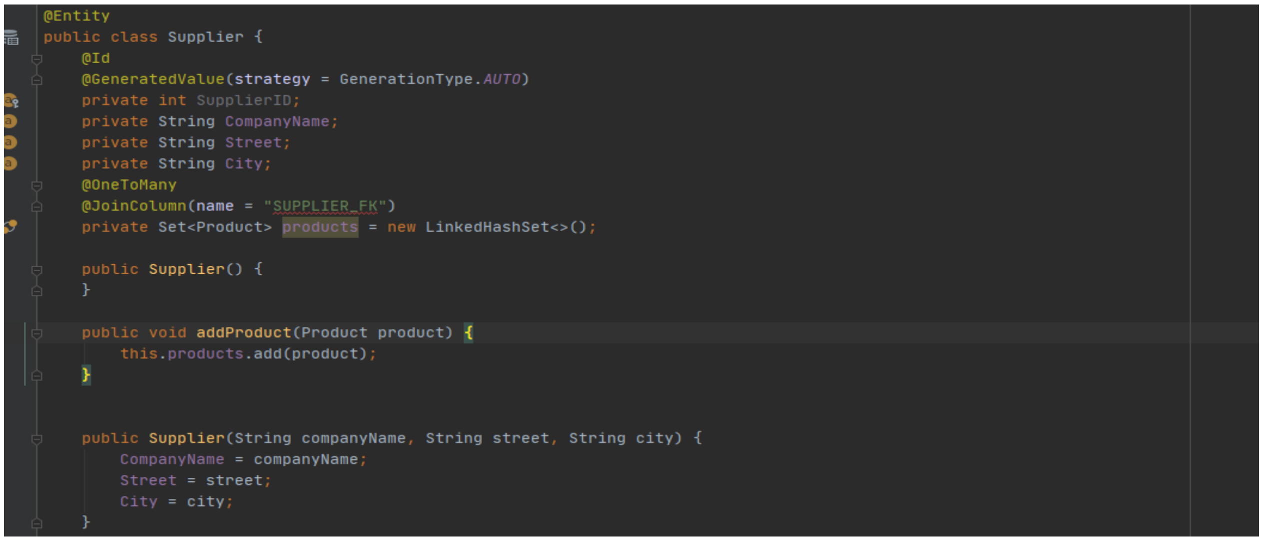
5 Zamodeluj relacje dwustronną jak poniżej:





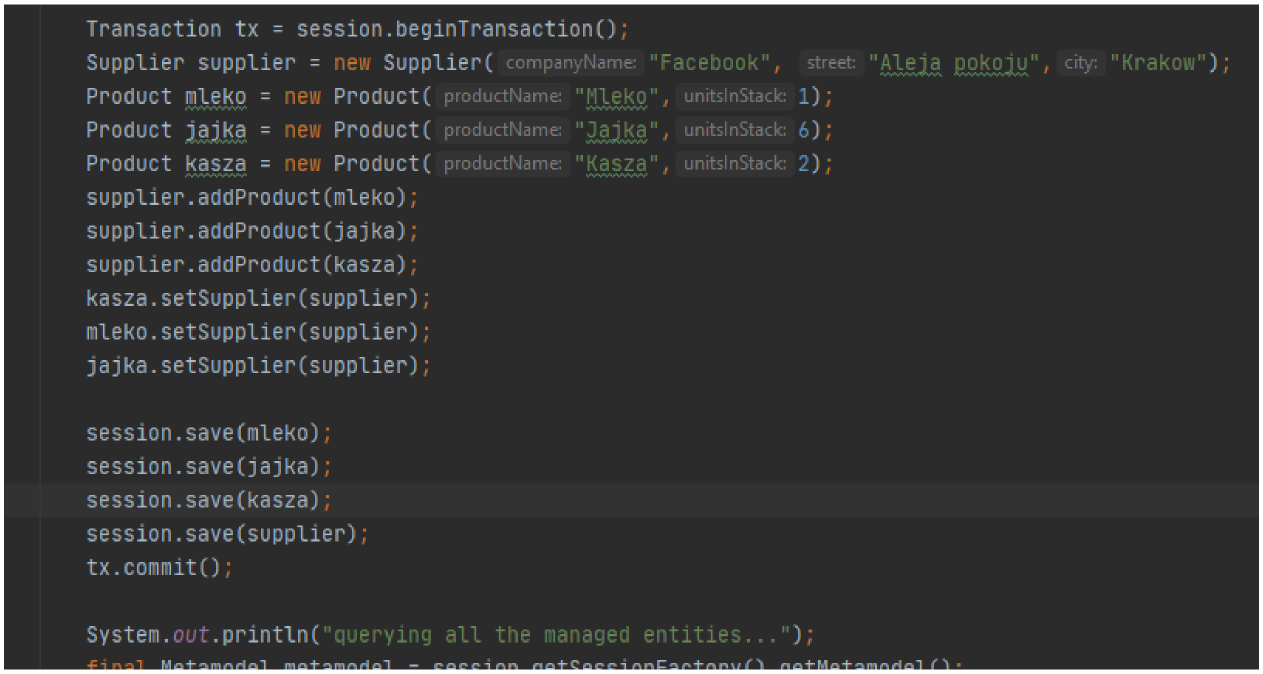
```java
@Entity
public class Product {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int ProductID;
    private String ProductName;
    private int UnitsInStack;
    @ManyToOne
    @JoinColumn(name = "SUPPLIER_FK")
    private Supplier supplier;

    public Product(String productName, int unitsInStack) {
        ProductName = productName;
        UnitsInStack = unitsInStack;
    }

    public Product() {
    }

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

    @Override
    public String toString() {
        return "Product{" + "ProductID=" + ProductID + ", ProductName='" + ProductName + '\'' + ", UnitsInStack=" + UnitsInStack + '}';
    }
}
```

Klasa Product



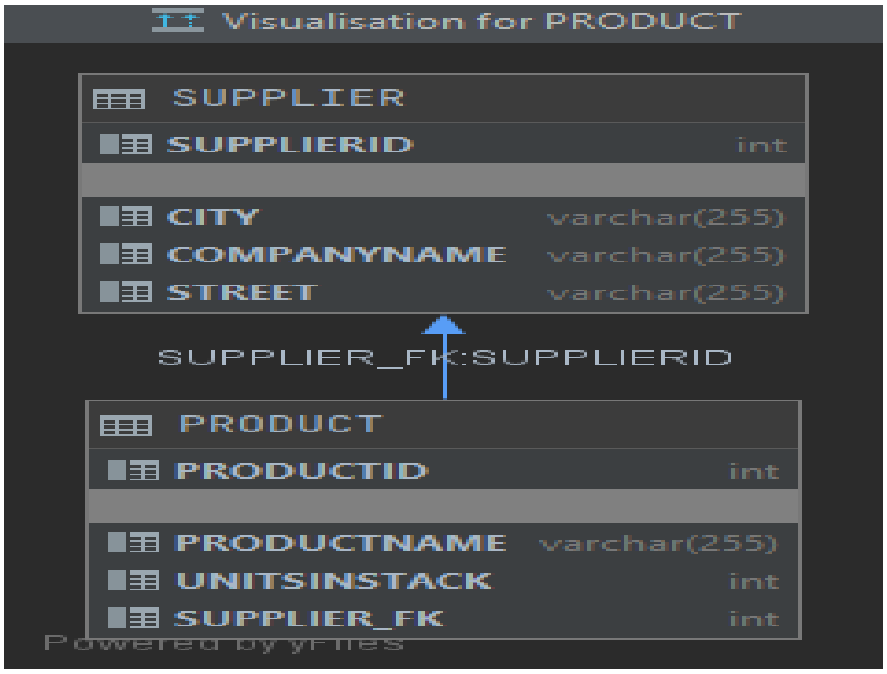
```java
@Entity
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int SupplierID;
    private String CompanyName;
    private String Street;
    private String City;
    @OneToMany
    @JoinColumn(name = "SUPPLIER_FK")
    private Set<Product> products =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

    public Supplier() {
    }

    public void addProduct(Product product) {
        this.products.add(product);
    }


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

Klasa Supplier



```
Transaction tx = session.beginTransaction();
Supplier supplier = new Supplier( companyName: "Facebook",  street: "Aleja pokoju", city: "Krakow");
Product mleko = new Product( productName: "Mleko", unitsInStack: 1);
Product jajka = new Product( productName: "Jajka", unitsInStack: 6);
Product kasza = new Product( productName: "Kasza", unitsInStack: 2);
supplier.addProduct(mleko);
supplier.addProduct(jajka);
supplier.addProduct(kasza);
kasza.setSupplier(supplier);
mleko.setSupplier(supplier);
jajka.setSupplier(supplier);

session.save(mleko);
session.save(jajka);
session.save(kasza);
session.save(supplier);
tx.commit();

System.out.println("querying all the managed entities...");
```

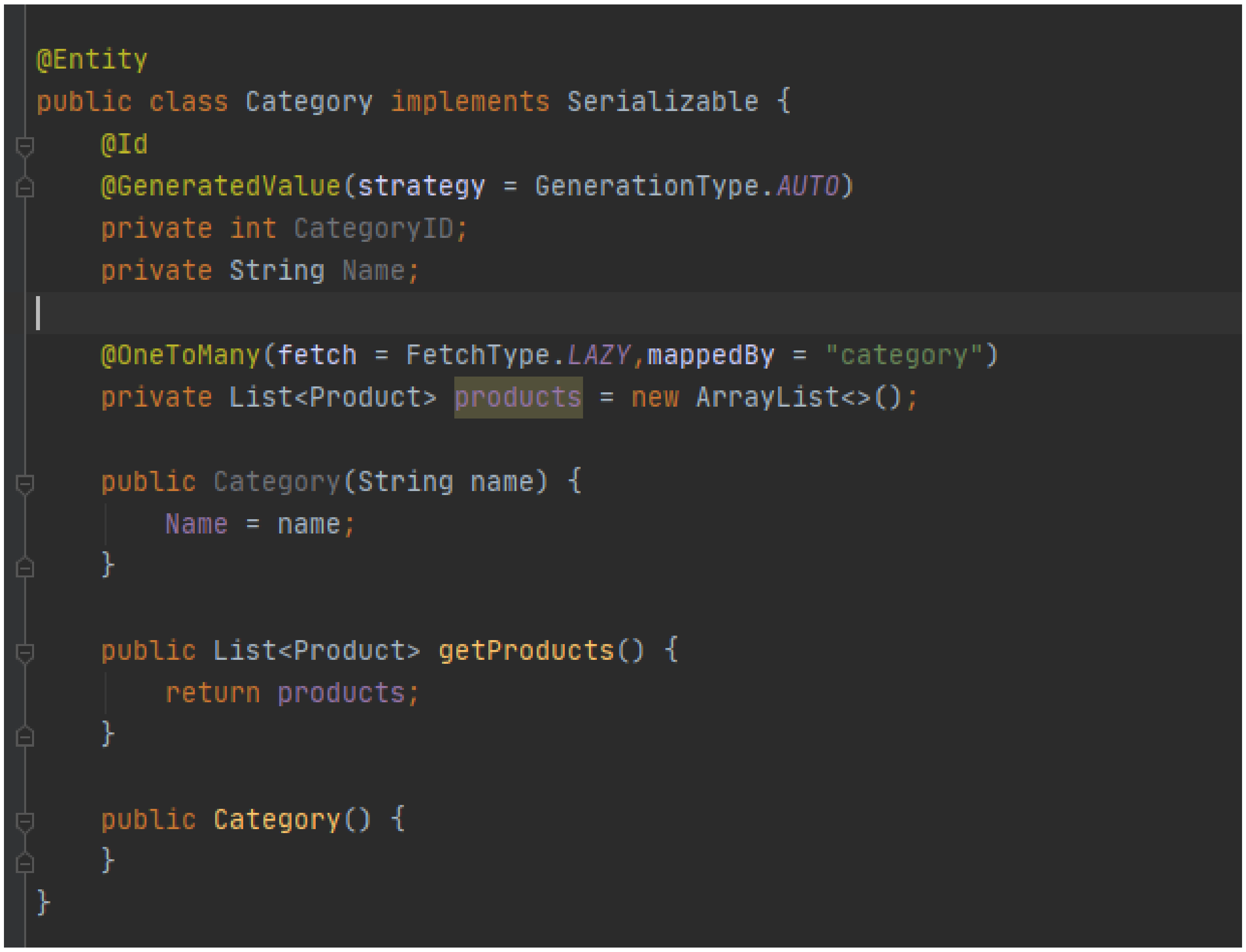
Main



Schemat bazy danych

6 Dodaj klase Category z property int CategoryID, String Name oraz listą produktow List<Product> Products

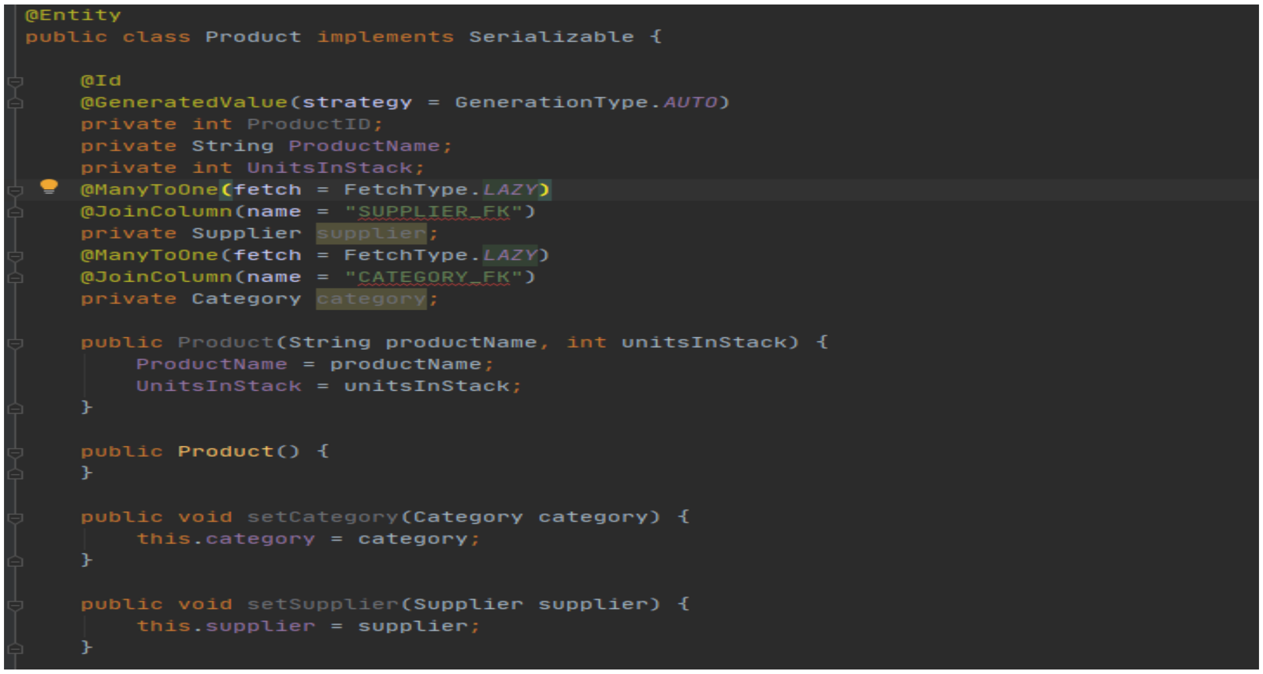
Zauważyłem, że bardziej wydajne będzie ustawienie Lazy loadingu aby uniknąć wyciągania z bazy danych których nie potrzebujemy.



```
@Entity
public class Category implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int CategoryID;
    private String Name;

    @OneToMany(fetch = FetchType.LAZY,mappedBy = "category")
    private List<Product> products = new ArrayList<>();

    public Category(String name) {
        Name = name;
    }

    public List<Product> getProducts() {
        return products;
    }

    public Category() {
    }
}
```

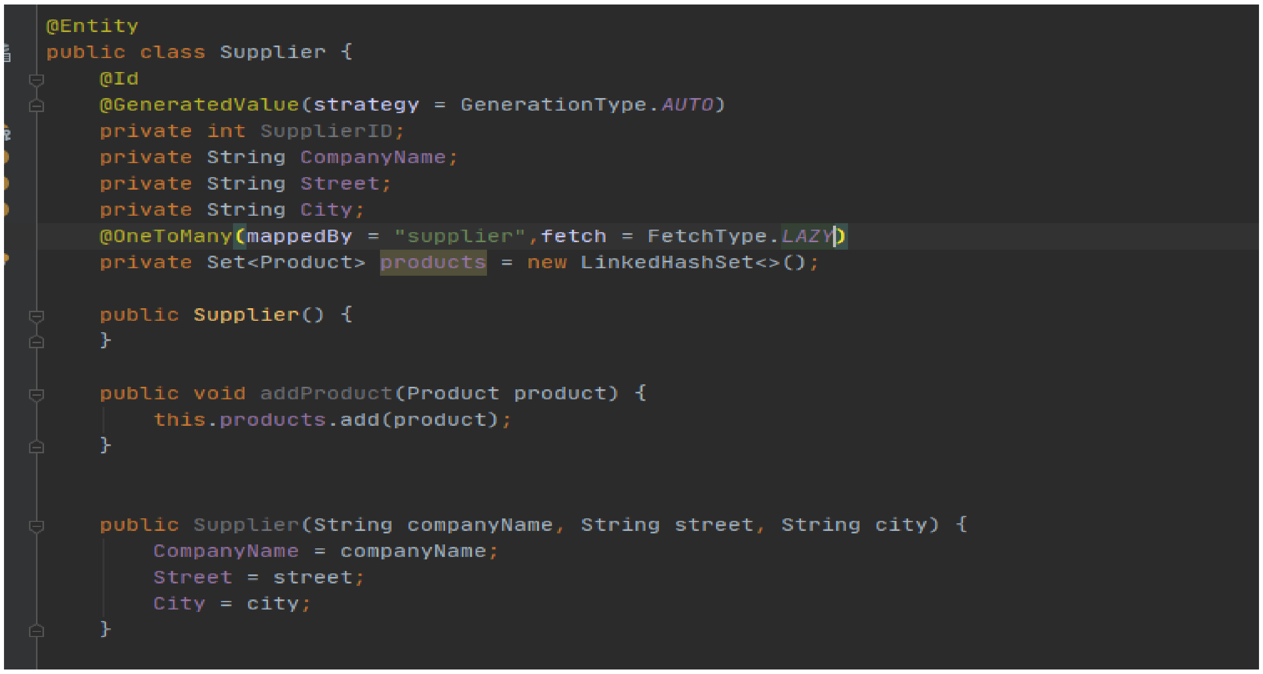
Klasa Category

6.1 Zmodyfikuj produkty dodając wskazanie na kategorie do której należy.



```java
@Entity
public class Product implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int ProductID;
    private String ProductName;
    private int UnitsInStack;
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "SUPPLIER_FK")
    private Supplier supplier;
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "CATEGORY_FK")
    private Category category;

    public Product(String productName, int unitsInStack) {
        ProductName = productName;
        UnitsInStack = unitsInStack;
    }

    public Product() {
    }

    public void setCategory(Category category) {
        this.category = category;
    }

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

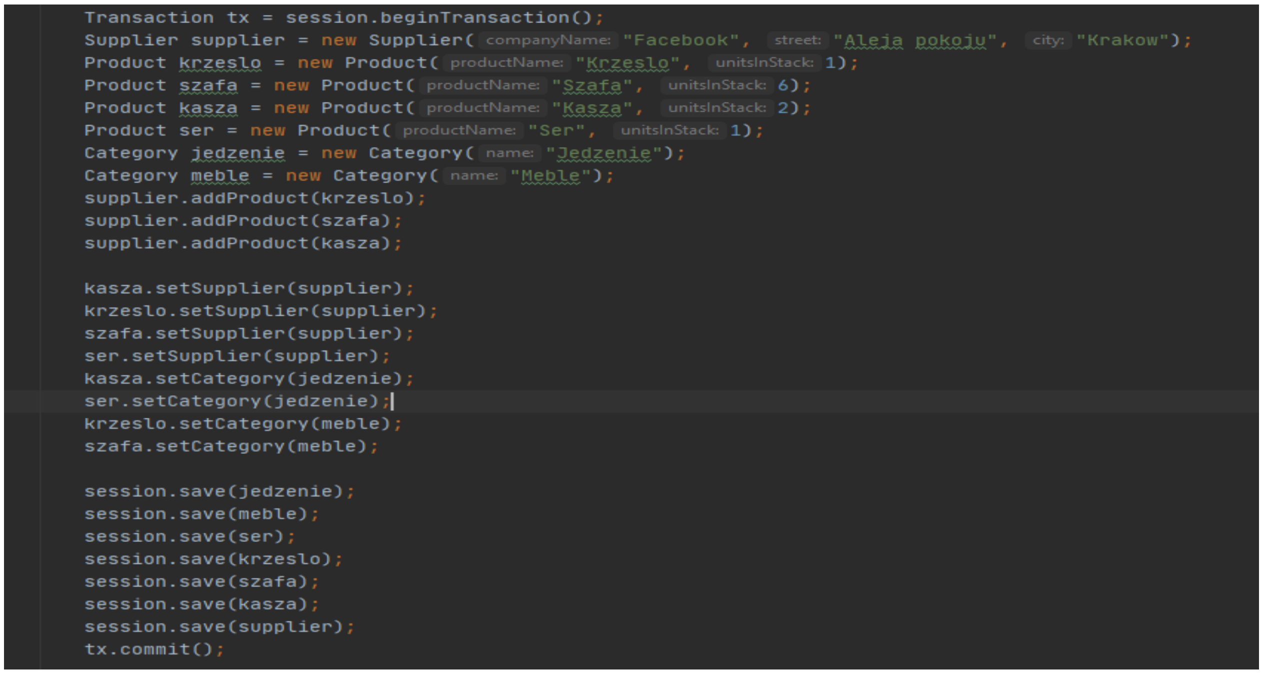
Klasa Product



```java
@Entity
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int SupplierID;
    private String CompanyName;
    private String Street;
    private String City;
    @OneToMany(mappedBy = "supplier",fetch = FetchType.LAZY)
    private Set<Product> products = new LinkedHashSet<>();

    public Supplier() {
    }

    public void addProduct(Product product) {
        this.products.add(product);
    }


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

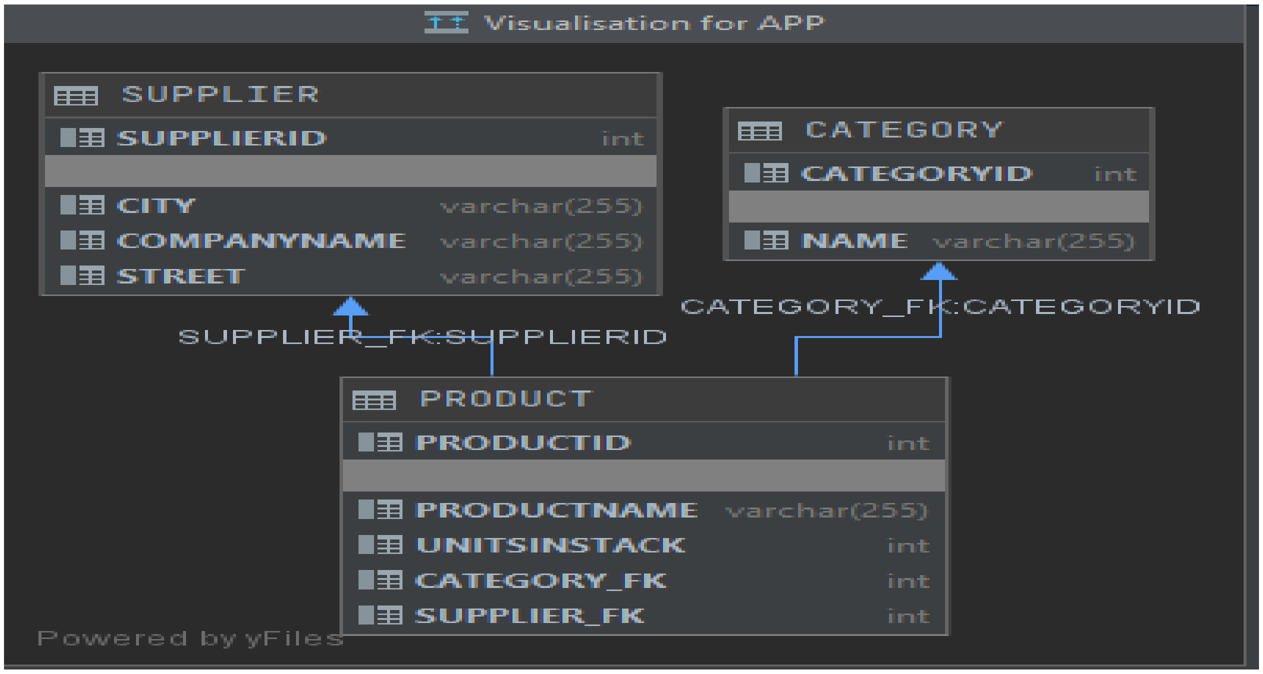
Klasa Supplier

6.2 Stworz kilka produktow i kilka kategorii



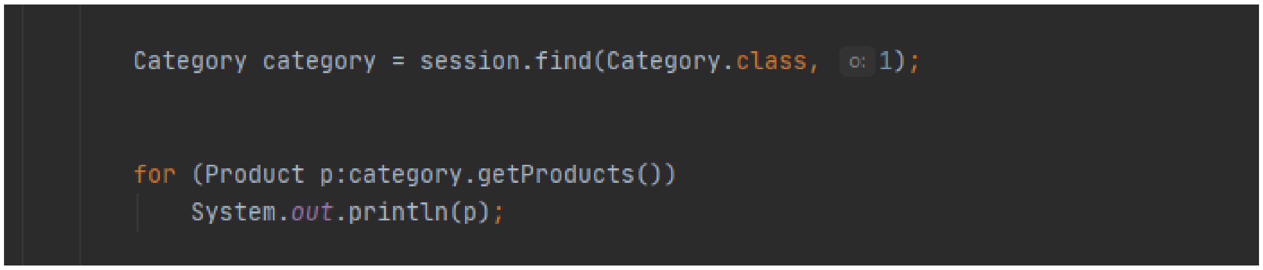
```
Transaction tx = session.beginTransaction();
Supplier supplier = new Supplier( companyName: "Facebook",  street: "Aleja pokoju",  city: "Krakow");
Product krzeslo = new Product( productName: "Krzeslo",  unitsInStack: 1);
Product szafa = new Product( productName: "Szafa",  unitsInStack: 6);
Product kasza = new Product( productName: "Kasza",  unitsInStack: 2);
Product ser = new Product( productName: "Ser",  unitsInStack: 1);
Category jedzenie = new Category( name: "Jedzenie");
Category meble = new Category( name: "Meble");
supplier.addProduct(krzeslo);
supplier.addProduct(szafa);
supplier.addProduct(kasza);

kasza.setSupplier(supplier);
krzeslo.setSupplier(supplier);
szafa.setSupplier(supplier);
ser.setSupplier(supplier);
kasza.setCategory(jedzenie);
ser.setCategory(jedzenie);
krzeslo.setCategory(meble);
szafa.setCategory(meble);

session.save(jedzenie);
session.save(meble);
session.save(ser);
session.save(krzeslo);
session.save(szafa);
session.save(kasza);
session.save(supplier);
tx.commit();
```

Main



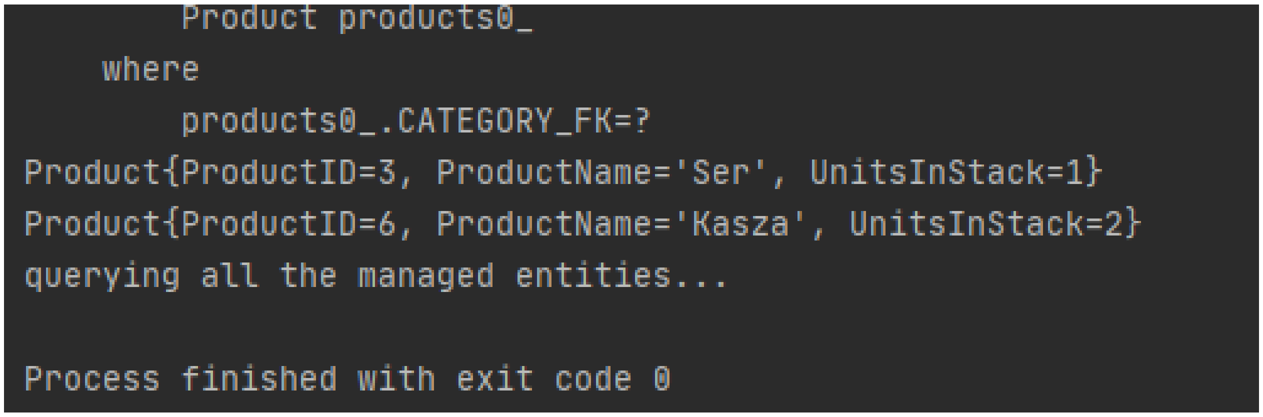
Schemat bazy danych

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



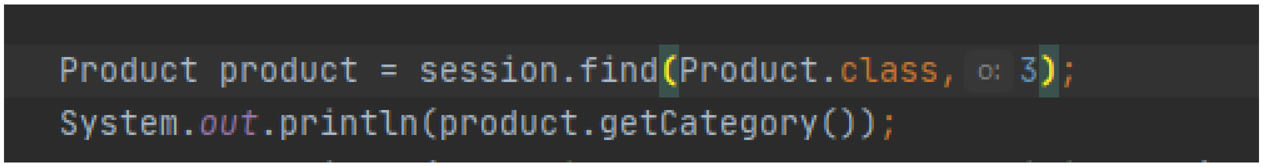
```
Category category = session.find(Category.class,  o: 1);


for (Product p:category.getProducts())
    System.out.println(p);
```

Wydobywanie produktów z danej katogorii



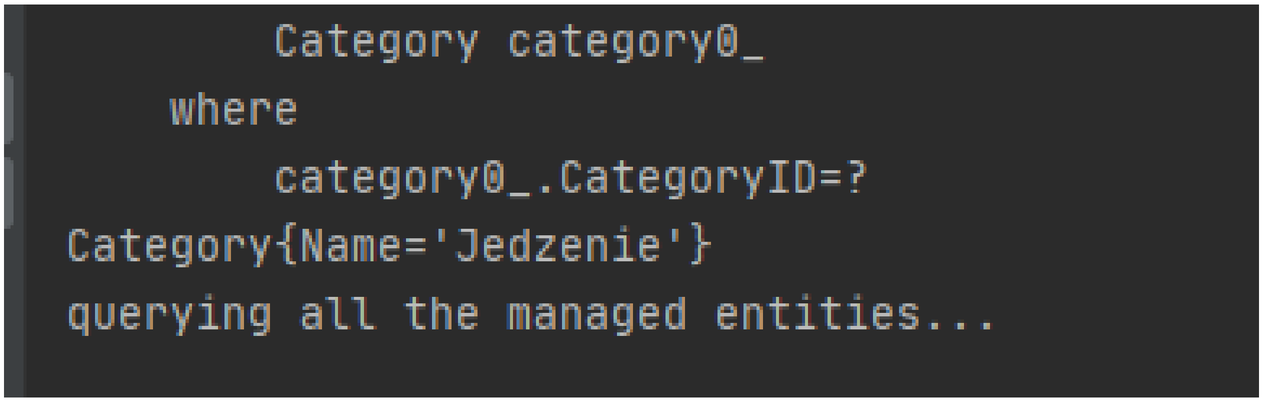
```
        Product products0_
    where
        products0_.CATEGORY_FK=?
Product{ProductID=3, ProductName='Ser', UnitsInStack=1}
Product{ProductID=6, ProductName='Kasza', UnitsInStack=2}
querying all the managed entities...

Process finished with exit code 0
```

Produkty z danej katogorii



```
Product product = session.find(Product.class,  o: 3);
System.out.println(product.getCategory());
```

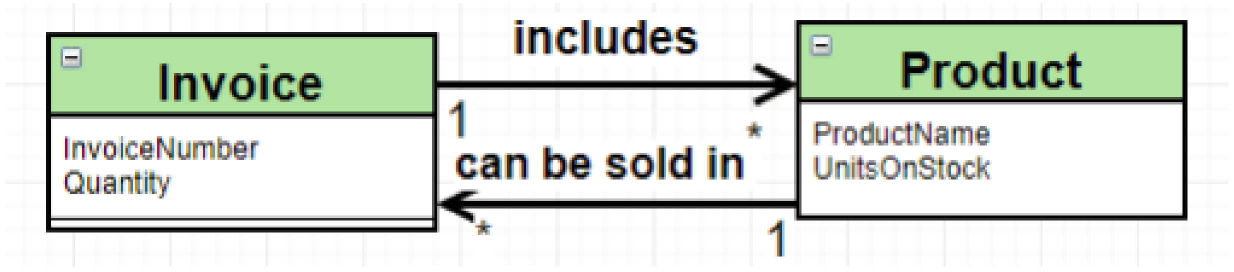
Wydobywanie katogorii produktu

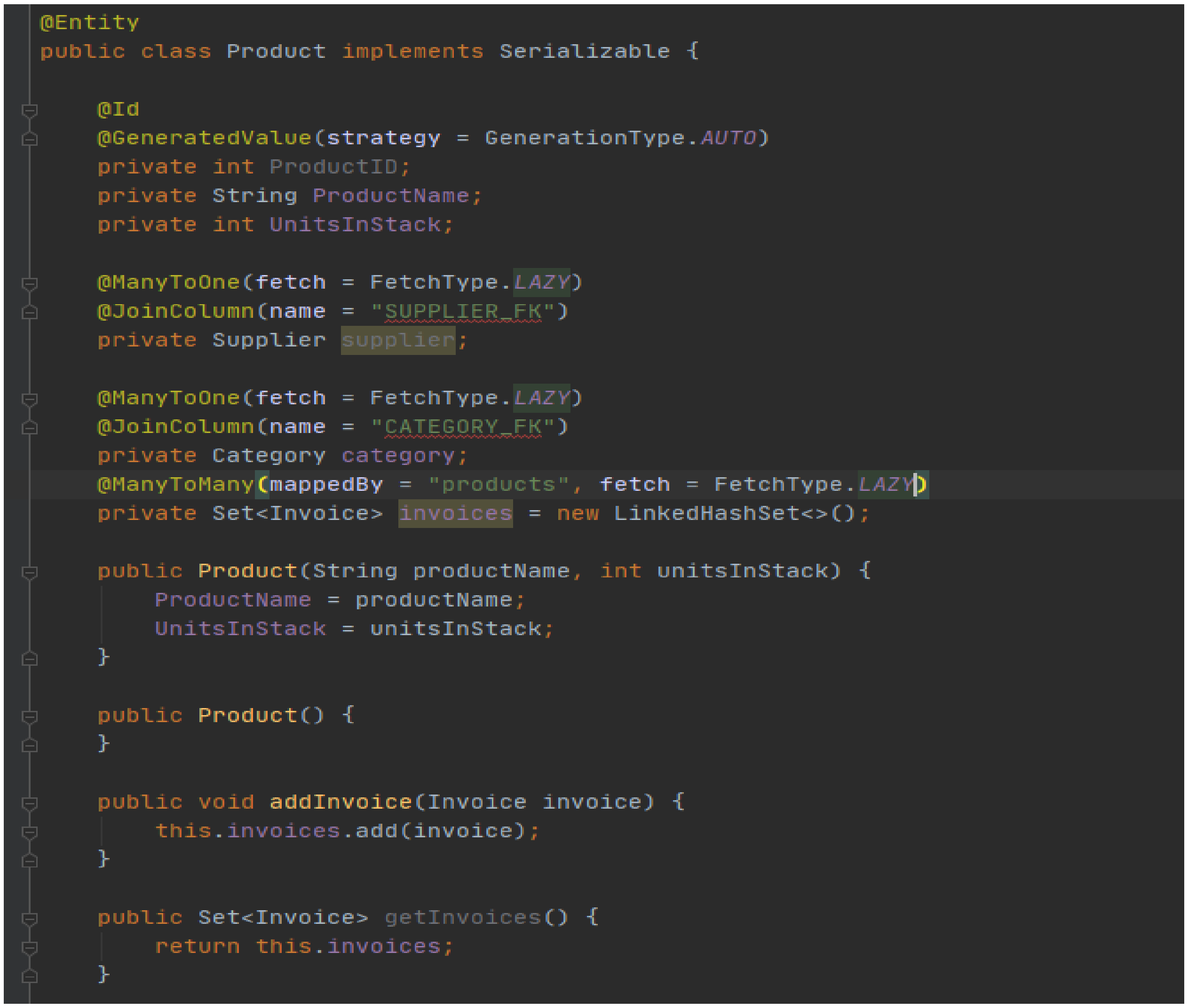


```
        Category category0_
    where
        category0_.CategoryID=?
Category{Name='Jedzenie'}
querying all the managed entities...
```

Kategoria produktu

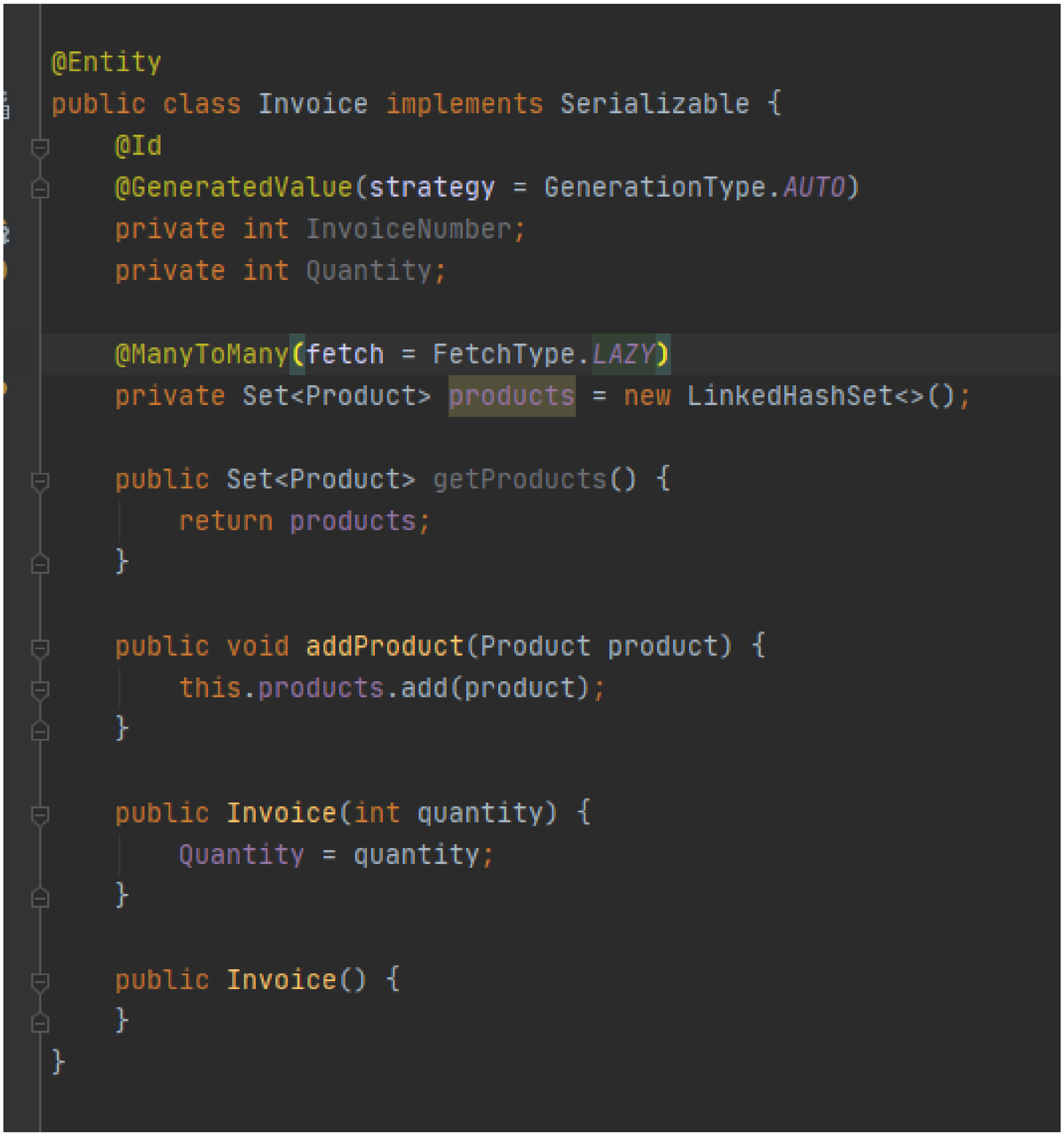
7 Zamodeluj relacje wiele-do-wielu, jak poniżej:





```
@Entity
public class Product implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int ProductID;
    private String ProductName;
    private int UnitsInStack;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "SUPPLIER_FK")
    private Supplier supplier;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "CATEGORY_FK")
    private Category category;
    @ManyToMany(mappedBy = "products", fetch = FetchType.LAZY)
    private Set<Invoice> invoices = new LinkedHashSet<>();

    public Product(String productName, int unitsInStack) {
        ProductName = productName;
        UnitsInStack = unitsInStack;
    }

    public Product() {
    }

    public void addInvoice(Invoice invoice) {
        this.invoices.add(invoice);
    }

    public Set<Invoice> getInvoices() {
        return this.invoices;
    }
```

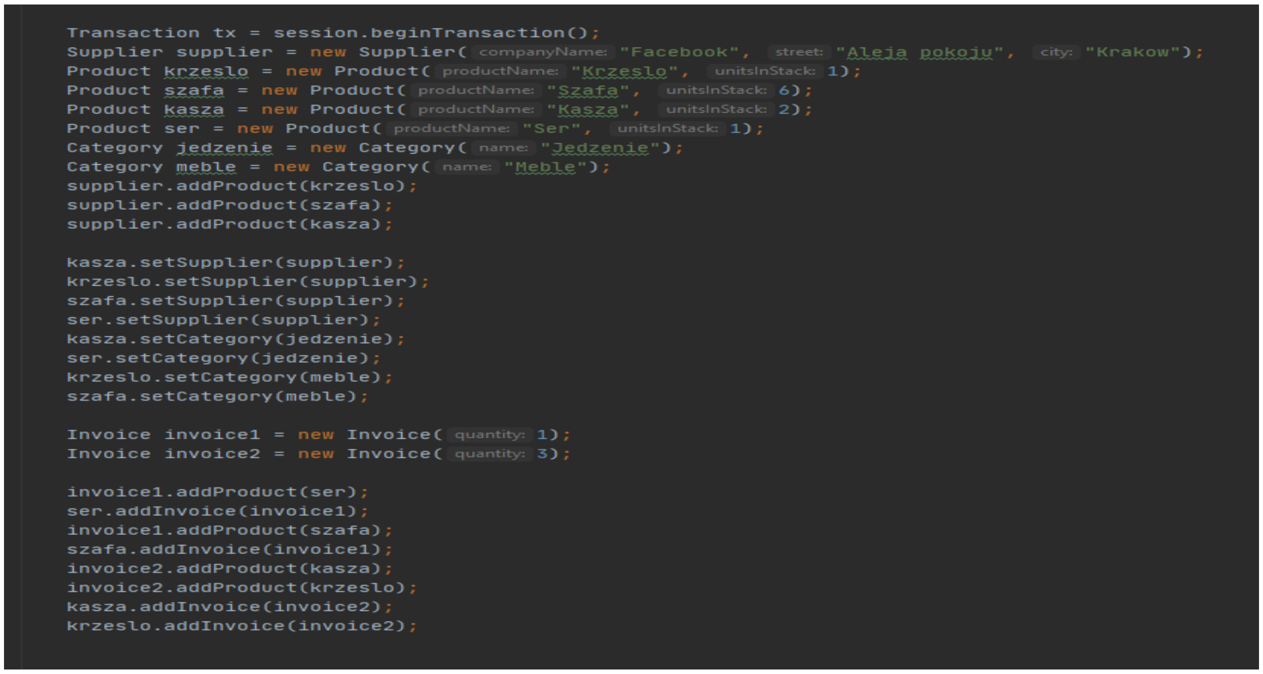
Klasa Product



```
@Entity
public class Invoice implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int InvoiceNumber;
    private int Quantity;

    @ManyToMany(fetch = FetchType.LAZY)
    private Set<Product> products = new LinkedHashSet<>();

    public Set<Product> getProducts() {
        return products;
    }

    public void addProduct(Product product) {
        this.products.add(product);
    }

    public Invoice(int quantity) {
        Quantity = quantity;
    }

    public Invoice() {
    }
}
```

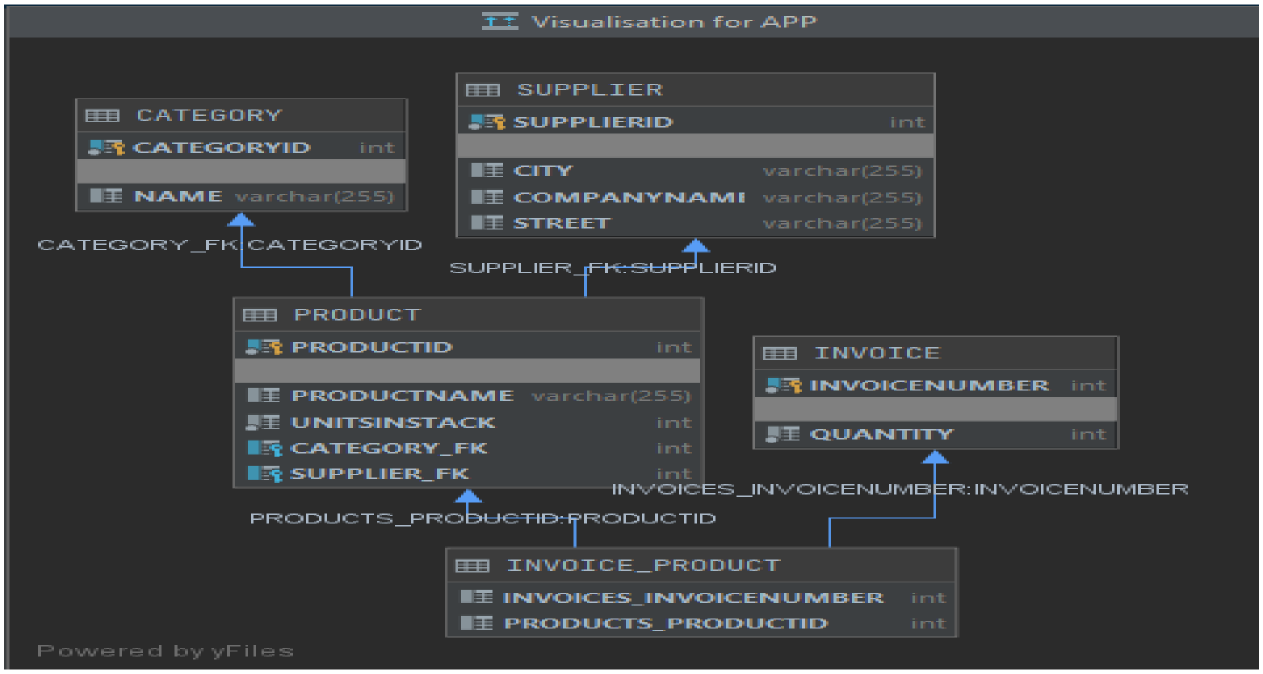
Klasa Invoice

7.1 Stórz kilka produktów I “sprzedaj” je na kilku transakcjach



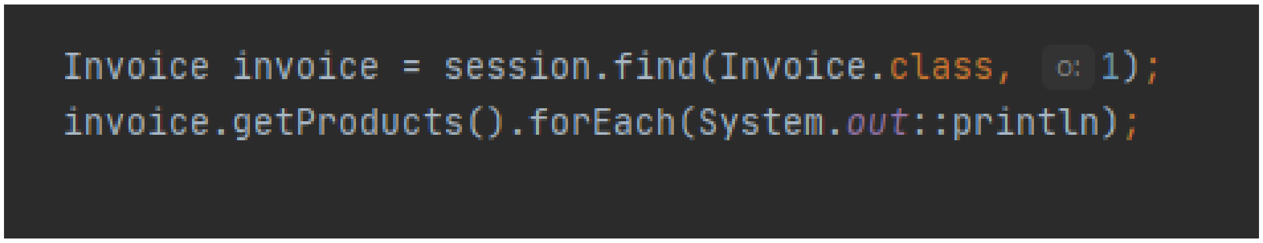
```
Transaction tx = session.beginTransaction();
Supplier supplier = new Supplier( companyName: "Facebook",  street: "Aleja pokoju",  city: "Krakow");
Product krzeslo = new Product( productName: "Krzeslo",  unitsInStack: 1);
Product szafa = new Product( productName: "Szafa",  unitsInStack: 6);
Product kasza = new Product( productName: "Kasza",  unitsInStack: 2);
Product ser = new Product( productName: "Ser",  unitsInStack: 1);
Category jedzenie = new Category( name: "Jedzenie");
Category meble = new Category( name: "Meble");
supplier.addProduct(krzeslo);
supplier.addProduct(szafa);
supplier.addProduct(kasza);

kasza.setSupplier(supplier);
krzeslo.setSupplier(supplier);
szafa.setSupplier(supplier);
ser.setSupplier(supplier);
kasza.setCategory(jedzenie);
ser.setCategory(jedzenie);
krzeslo.setCategory(meble);
szafa.setCategory(meble);

Invoice invoice1 = new Invoice( quantity: 1);
Invoice invoice2 = new Invoice( quantity: 3);

invoice1.addProduct(ser);
ser.addInvoice(invoice1);
invoice1.addProduct(szafa);
szafa.addInvoice(invoice1);
invoice2.addProduct(kasza);
invoice2.addProduct(krzeslo);
kasza.addInvoice(invoice2);
krzeslo.addInvoice(invoice2);
```

Main



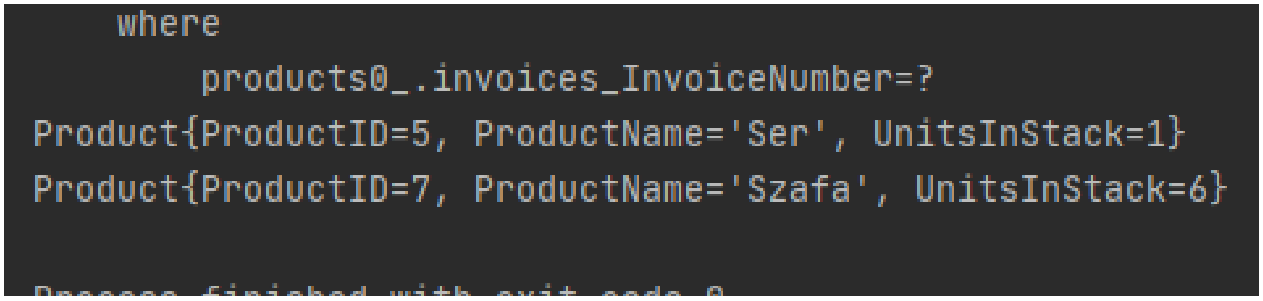
Schemat Bazy danych

7.2 Pokaż produkty sprzedane w ramach wybranej faktury/transakcji



```
Invoice invoice = session.find(Invoice.class,  o: 1);
invoice.getProducts().forEach(System.out::println);
```

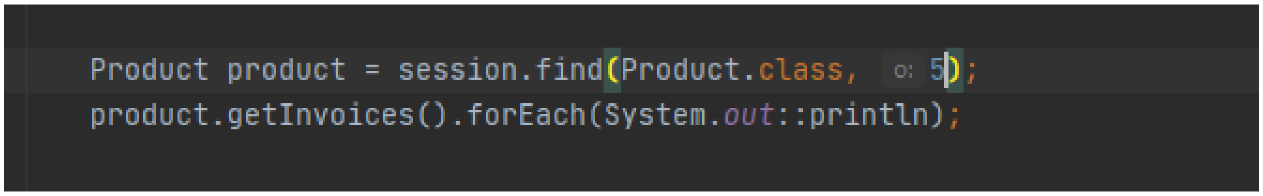
Zapytanie



```
    where
        products0_.invoices_InvoiceNumber=?
Product{ProductID=5, ProductName='Ser', UnitsInStack=1}
Product{ProductID=7, ProductName='Szafa', UnitsInStack=6}
```

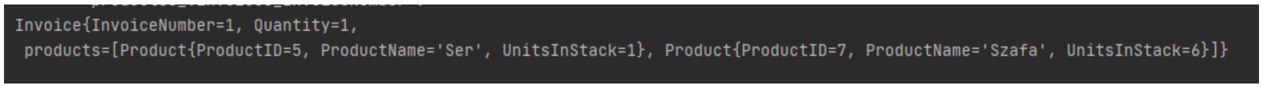
Rezultat

7.3 Pokaż faktury w ramach których był sprzedany wybrany produkt



```
Product product = session.find(Product.class,  o: 5);
product.getInvoices().forEach(System.out::println);
```

Zapytanie



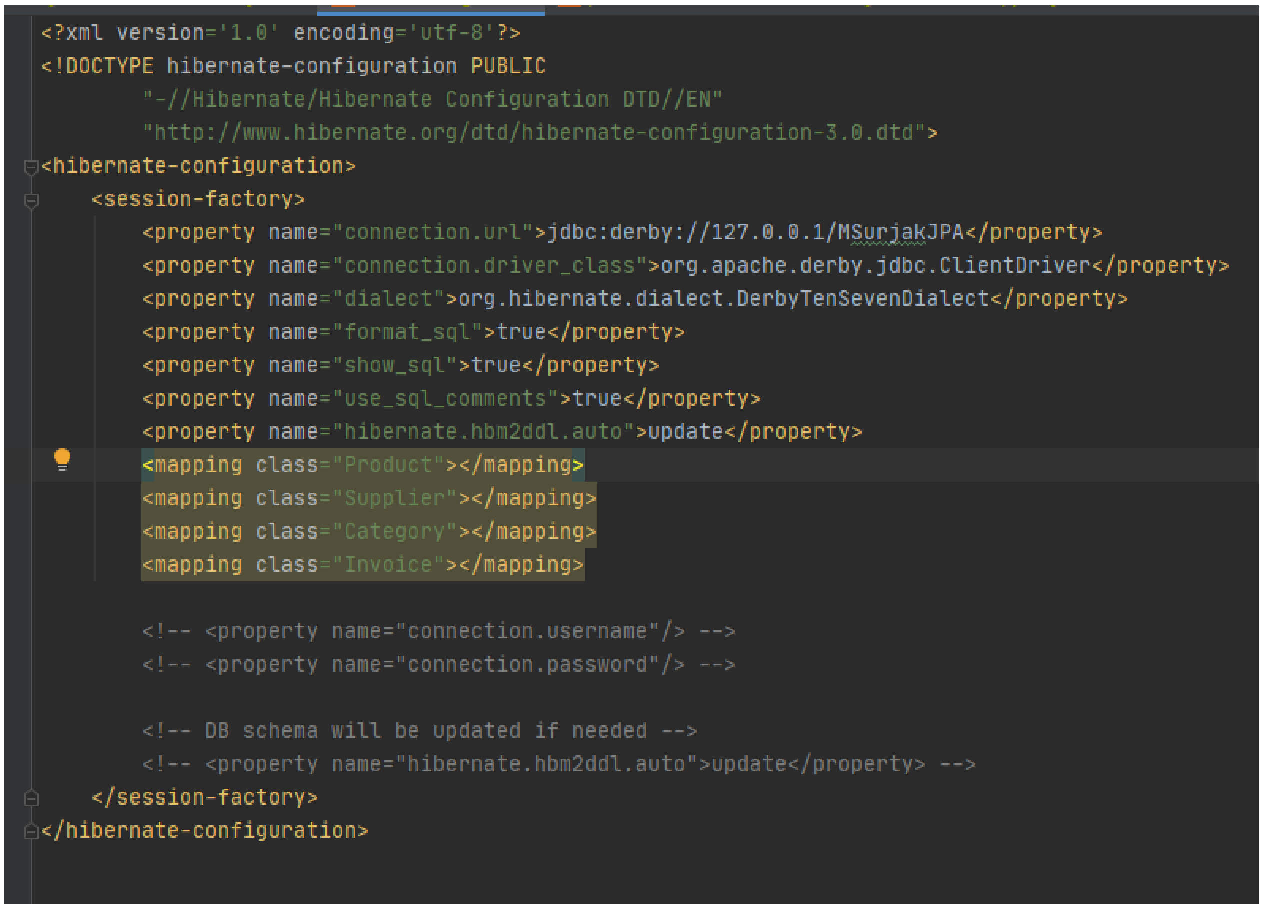
```
Invoice{InvoiceNumber=1, Quantity=1,
 products=[Product{ProductID=5, ProductName='Ser', UnitsInStack=1}, Product{ProductID=7, ProductName='Szafa', UnitsInStack=6}]}
```

Rezultat

8 JPA

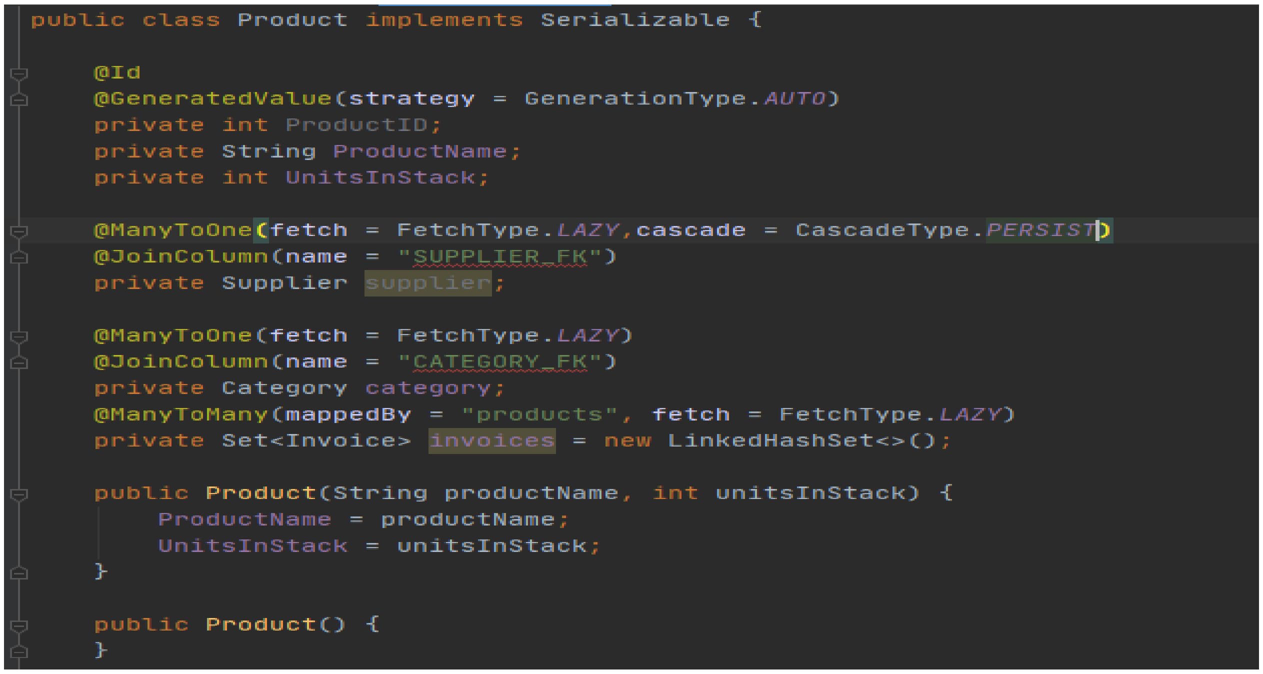
8.1 Stwórz nowego maina w którym zrobisz to samo co w punkcie VI ale z wykorzystaniem JPA

Utworzyłem plik persistence.yml w folderze META-INF w celu skonfigurowania JPA



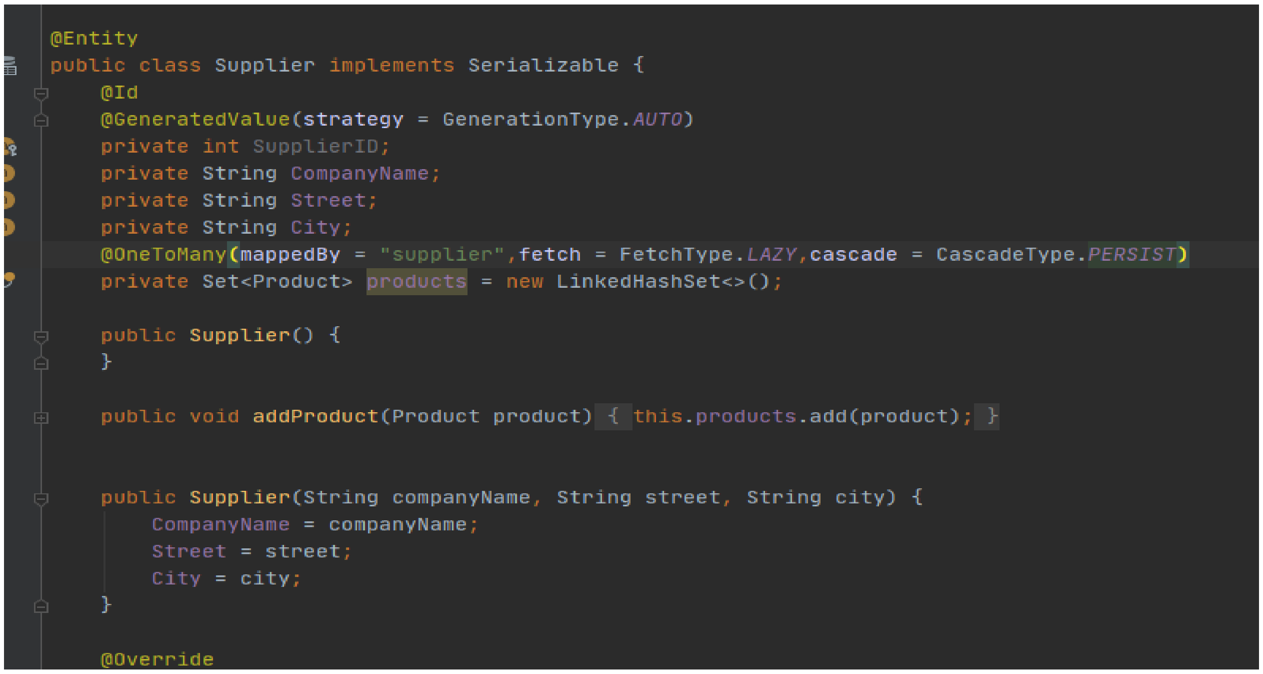
```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="connection.url">jdbc:derby://127.0.0.1/MSurjakJPA</property>
        <property name="connection.driver_class">org.apache.derby.jdbc.ClientDriver</property>
        <property name="dialect">org.hibernate.dialect.DerbyTenSevenDialect</property>
        <property name="format_sql">true</property>
        <property name="show_sql">true</property>
        <property name="use_sql_comments">true</property>
        <property name="hibernate.hbm2ddl.auto">update</property>
        <mapping class="Product"></mapping>
        <mapping class="Supplier"></mapping>
        <mapping class="Category"></mapping>
        <mapping class="Invoice"></mapping>

        <!-- <property name="connection.username"/> -->
        <!-- <property name="connection.password"/> -->

        <!-- DB schema will be updated if needed -->
        <!-- <property name="hibernate.hbm2ddl.auto">update</property> -->
    </session-factory>
</hibernate-configuration>
```

persistence.yml



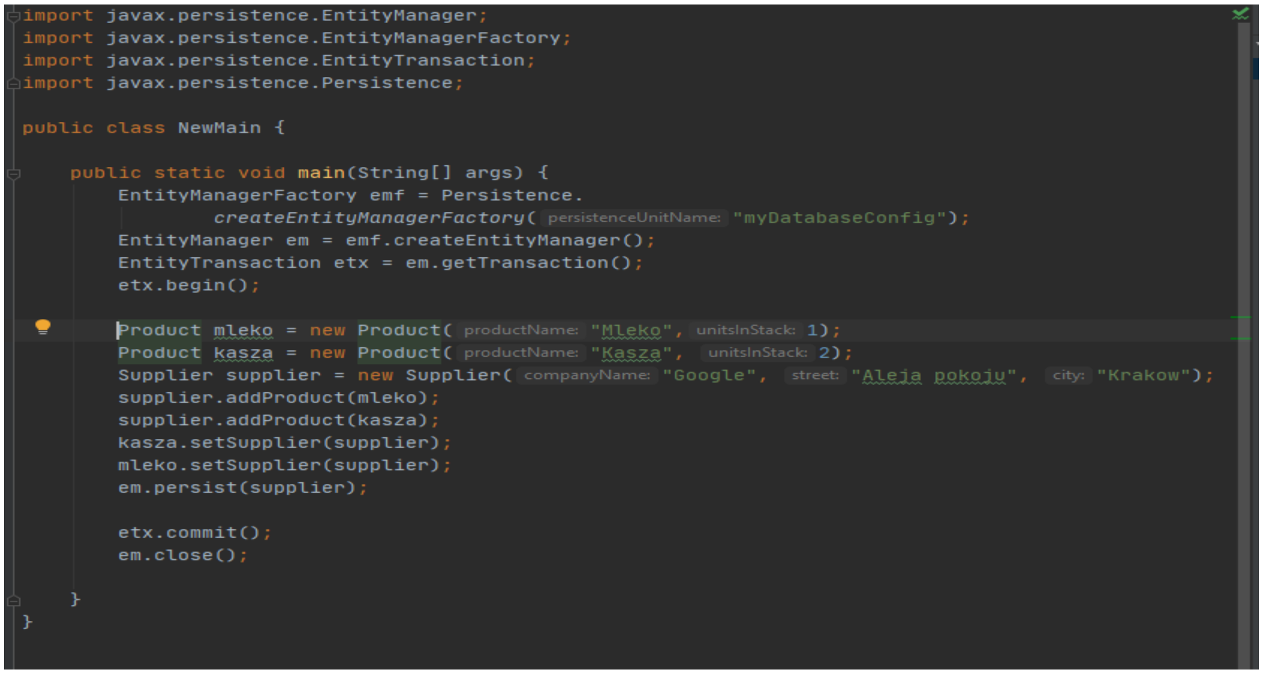
```
public class Product implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int ProductID;
    private String ProductName;
    private int UnitsInStack;

    @ManyToOne(fetch = FetchType.LAZY,cascade = CascadeType.PERSIST)
    @JoinColumn(name = "SUPPLIER_FK")
    private Supplier supplier;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "CATEGORY_FK")
    private Category category;
    @ManyToMany(mappedBy = "products", fetch = FetchType.LAZY)
    private Set<Invoice> invoices = new LinkedHashSet<>();

    public Product(String productName, int unitsInStack) {
        ProductName = productName;
        UnitsInStack = unitsInStack;
    }

    public Product() {
    }
```

Klasa Product



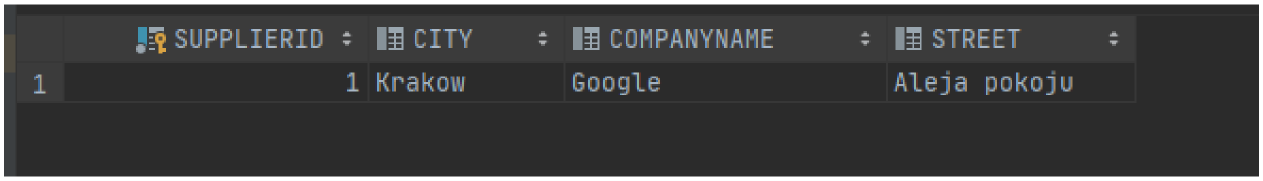
```
@Entity
public class Supplier implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int SupplierID;
    private String CompanyName;
    private String Street;
    private String City;
    @OneToMany(mappedBy = "supplier",fetch = FetchType.LAZY,cascade = CascadeType.PERSIST)
    private Set<Product> products = new LinkedHashSet<>();

    public Supplier() {
    }

    public void addProduct(Product product) { this.products.add(product); }


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

    @Override
```

Klasa Supplier



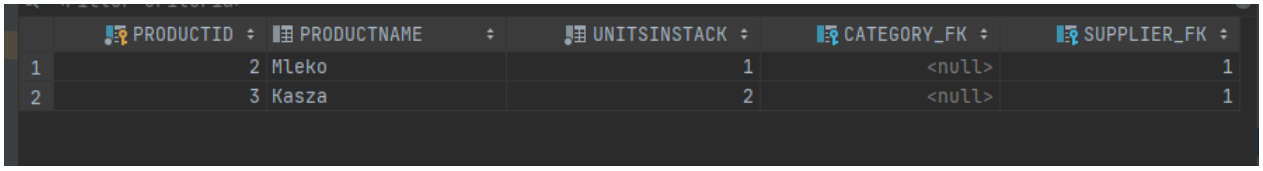
```
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;

public class NewMain {

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

        Product mleko = new Product( productName: "Mleko", unitsInStack: 1);
        Product kasza = new Product( productName: "Kasza", unitsInStack: 2);
        Supplier supplier = new Supplier( companyName: "Google", street: "Aleja pokoju", city: "Krakow");
        supplier.addProduct(mleko);
        supplier.addProduct(kasza);
        kasza.setSupplier(supplier);
        mleko.setSupplier(supplier);
        em.persist(supplier);

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

    }
}
```

Main



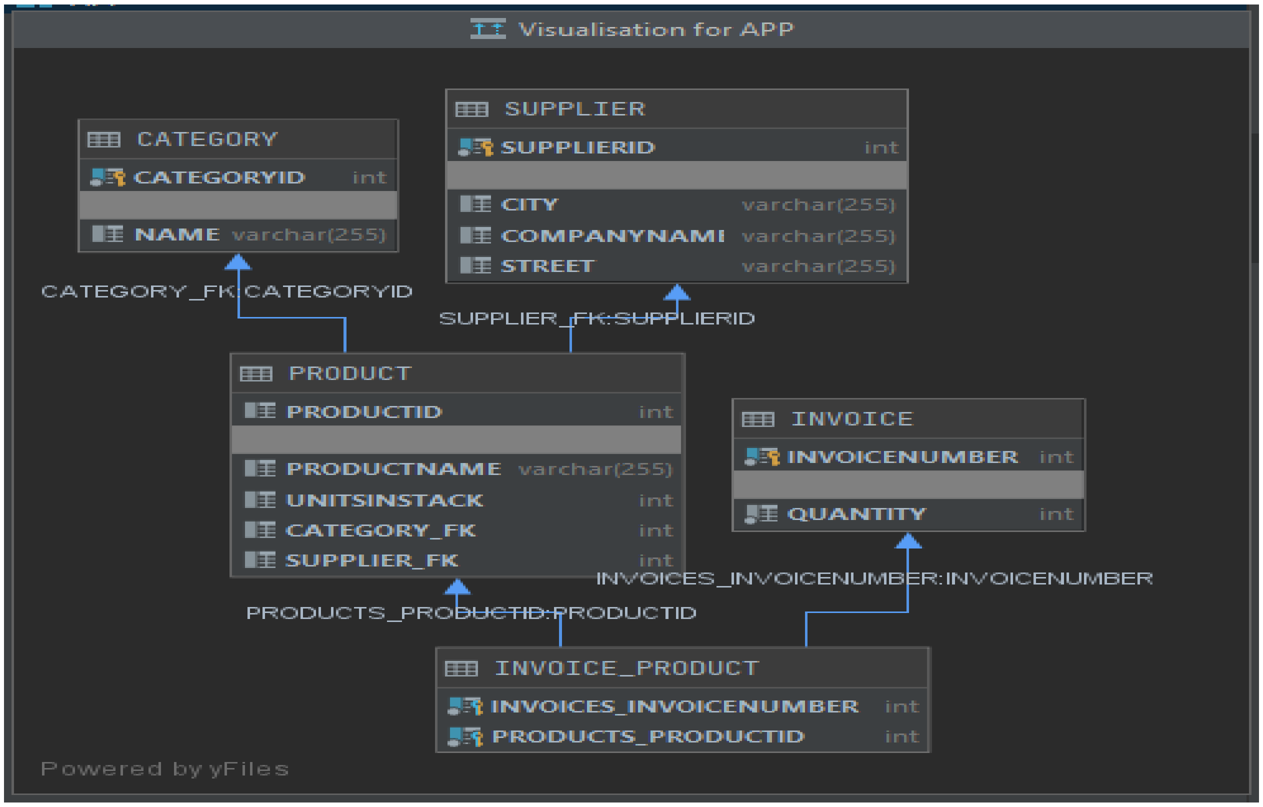
	SUPPLIERID	CITY	COMPANYNAME	STREET
1	1	Krakow	Google	Aleja pokoju

Tabela Supplier



	PRODUCTID	PRODUCTNAME	UNITSINSTACK	CATEGORY_FK	SUPPLIER_FK
1	2	Mleko	1	<null>	1
2	3	Kasza	2	<null>	1

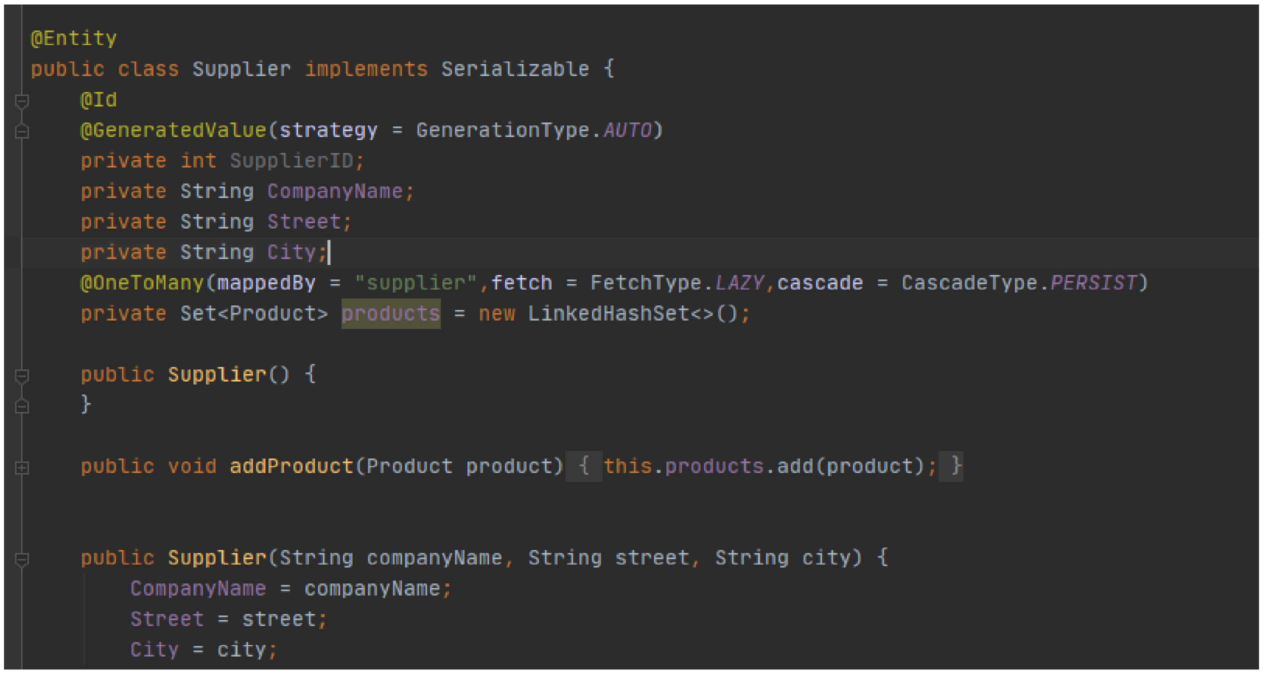
Tabela Product



Schemat Bazy danych

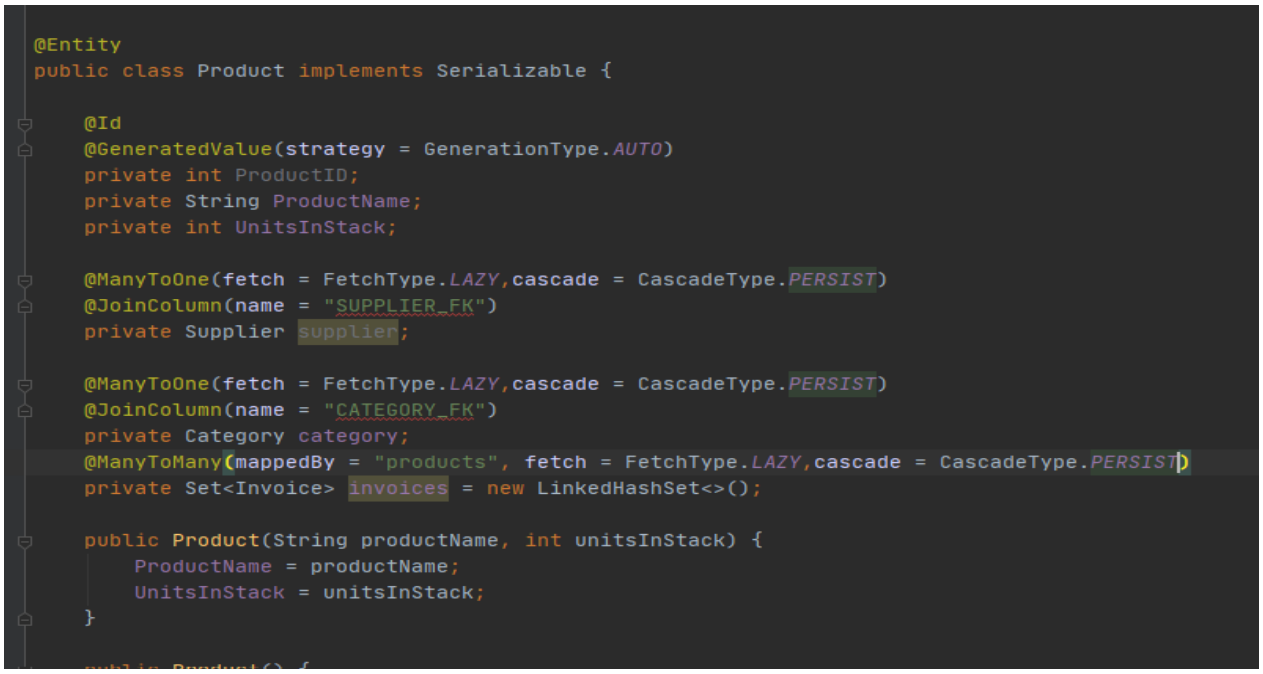
9 Kaskady

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



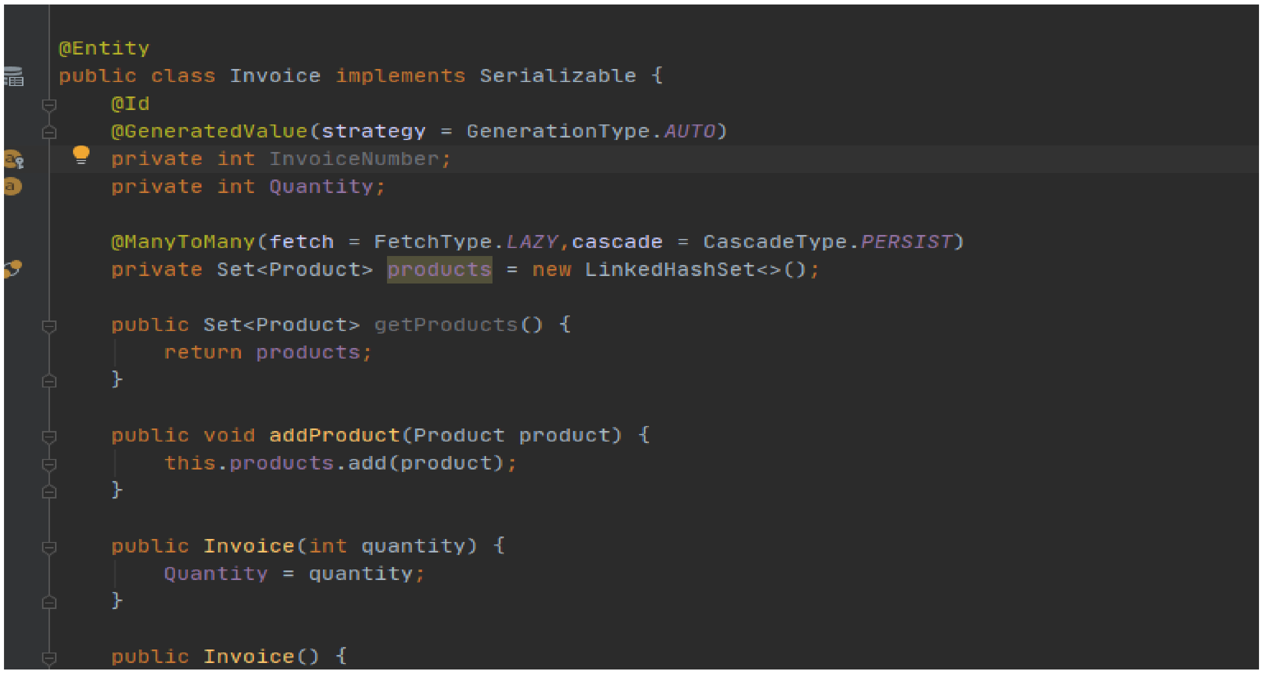
```java
@Entity
public class Supplier implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int SupplierID;
    private String CompanyName;
    private String Street;
    private String City;
    @OneToMany(mappedBy = "supplier",fetch = FetchType.LAZY,cascade = CascadeType.PERSIST)
    private Set<Product> products = new LinkedHashSet<>();

    public Supplier() {
    }

    public void addProduct(Product product) { this.products.add(product); }


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

Klasa Supplier



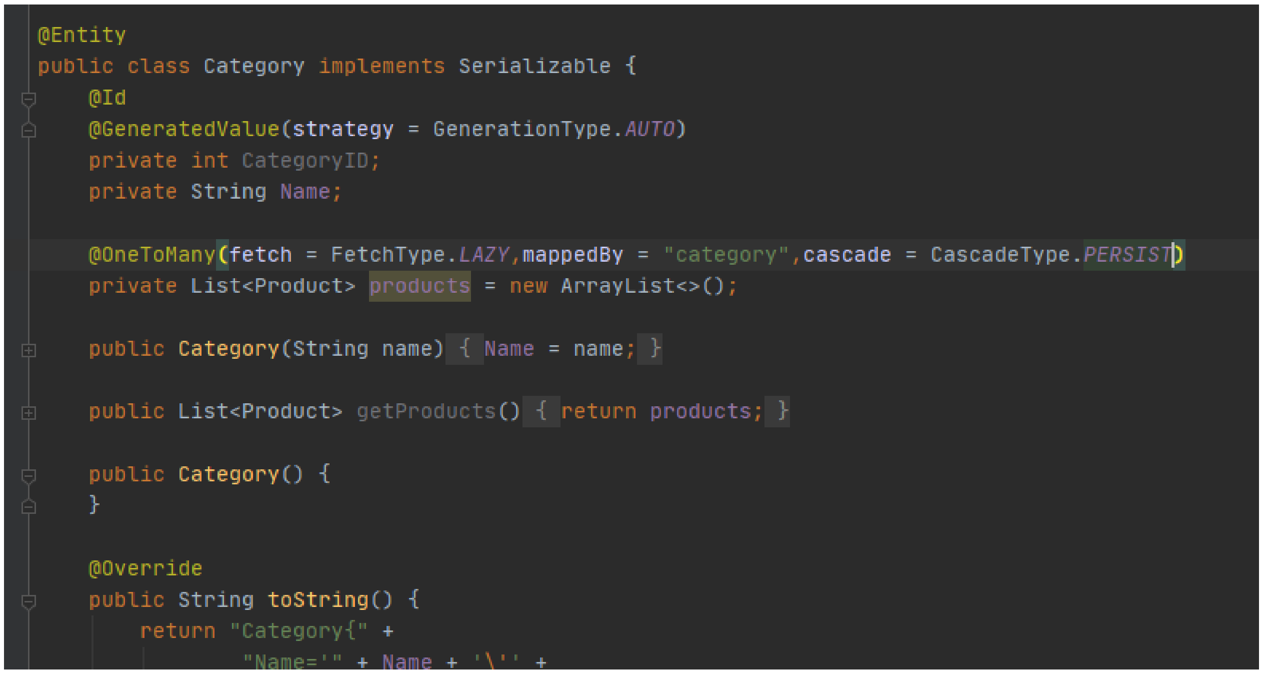
```java
@Entity
public class Product implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int ProductID;
    private String ProductName;
    private int UnitsInStack;

    @ManyToOne(fetch = FetchType.LAZY,cascade = CascadeType.PERSIST)
    @JoinColumn(name = "SUPPLIER_FK")
    private Supplier supplier;

    @ManyToOne(fetch = FetchType.LAZY,cascade = CascadeType.PERSIST)
    @JoinColumn(name = "CATEGORY_FK")
    private Category category;
    @ManyToMany(mappedBy = "products", fetch = FetchType.LAZY,cascade = CascadeType.PERSIST)
    private Set<Invoice> invoices = new LinkedHashSet<>();

    public Product(String productName, int unitsInStack) {
        ProductName = productName;
        UnitsInStack = unitsInStack;
    }

    public Product() {
```

Klasa Product



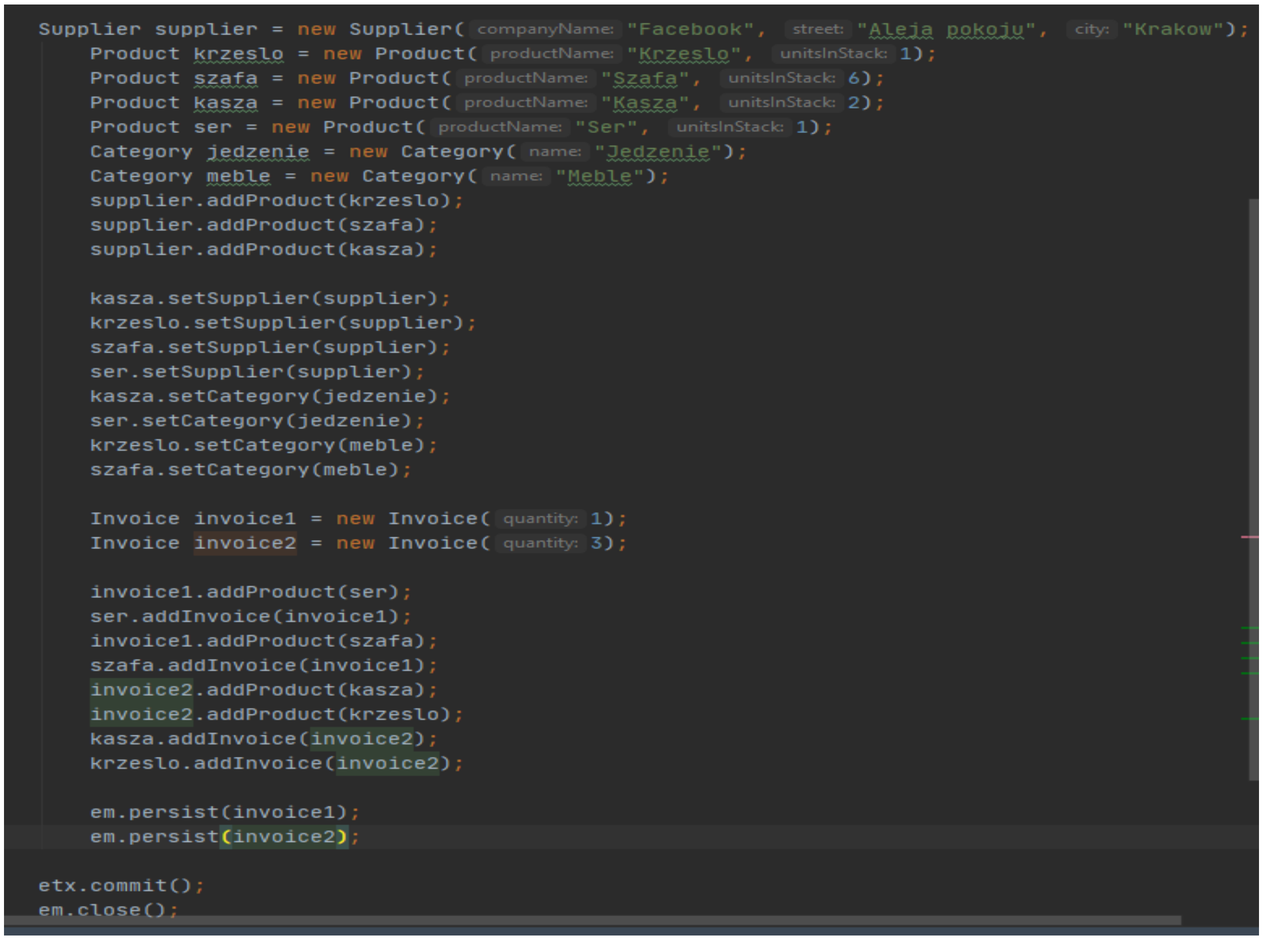
```
@Entity
public class Invoice implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int InvoiceNumber;
    private int Quantity;

    @ManyToMany(fetch = FetchType.LAZY,cascade = CascadeType.PERSIST)
    private Set<Product> products = new LinkedHashSet<>();

    public Set<Product> getProducts() {
        return products;
    }

    public void addProduct(Product product) {
        this.products.add(product);
    }

    public Invoice(int quantity) {
        Quantity = quantity;
    }

    public Invoice() {
```

Klasa Invoice



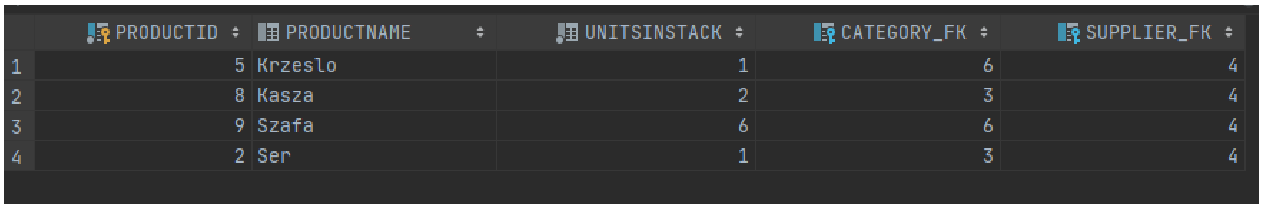
```
@Entity
public class Category implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int CategoryID;
    private String Name;

    @OneToMany(fetch = FetchType.LAZY,mappedBy = "category",cascade = CascadeType.PERSIST)
    private List<Product> products = new ArrayList<>();

    public Category(String name) { Name = name; }

    public List<Product> getProducts() { return products; }

    public Category() {
    }

    @Override
    public String toString() {
        return "Category{" +
                "Name='" + Name + '\'' +
```

Klasa Category



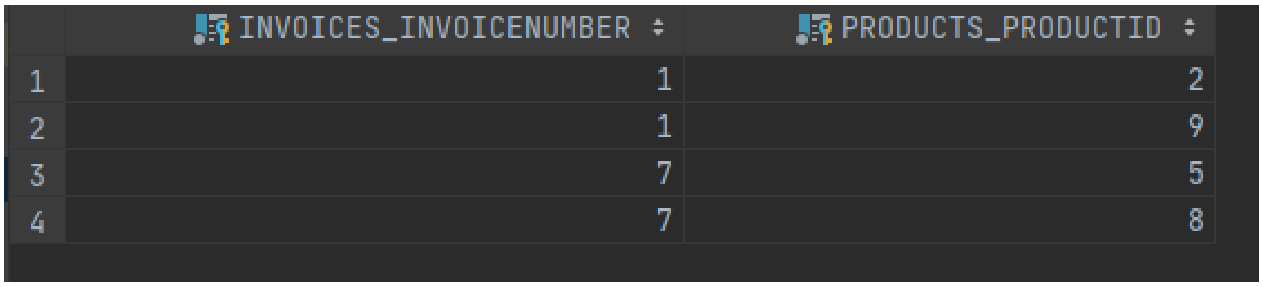
```
Supplier supplier = new Supplier( companyName: "Facebook",  street: "Aleja pokoju",  city: "Krakow");
    Product krzeslo = new Product( productName: "Krzeslo",  unitsInStack: 1);
    Product szafa = new Product( productName: "Szafa",  unitsInStack: 6);
    Product kasza = new Product( productName: "Kasza",  unitsInStack: 2);
    Product ser = new Product( productName: "Ser",  unitsInStack: 1);
    Category jedzenie = new Category( name: "Jedzenie");
    Category meble = new Category( name: "Meble");
    supplier.addProduct(krzeslo);
    supplier.addProduct(szafa);
    supplier.addProduct(kasza);

    kasza.setSupplier(supplier);
    krzeslo.setSupplier(supplier);
    szafa.setSupplier(supplier);
    ser.setSupplier(supplier);
    kasza.setCategory(jedzenie);
    ser.setCategory(jedzenie);
    krzeslo.setCategory(meble);
    szafa.setCategory(meble);

    Invoice invoice1 = new Invoice( quantity: 1);
    Invoice invoice2 = new Invoice( quantity: 3);

    invoice1.addProduct(ser);
    ser.addInvoice(invoice1);
    invoice1.addProduct(szafa);
    szafa.addInvoice(invoice1);
    invoice2.addProduct(kasza);
    invoice2.addProduct(krzeslo);
    kasza.addInvoice(invoice2);
    krzeslo.addInvoice(invoice2);

    em.persist(invoice1);
    em.persist(invoice2);

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

Main



	PRODUCTID	PRODUCTNAME	UNITSINSTACK	CATEGORY_FK	SUPPLIER_FK
1	5	Krzeslo	1	6	4
2	8	Kasza	2	3	4
3	9	Szafa	6	6	4
4	2	Ser	1	3	4

Tabela Products

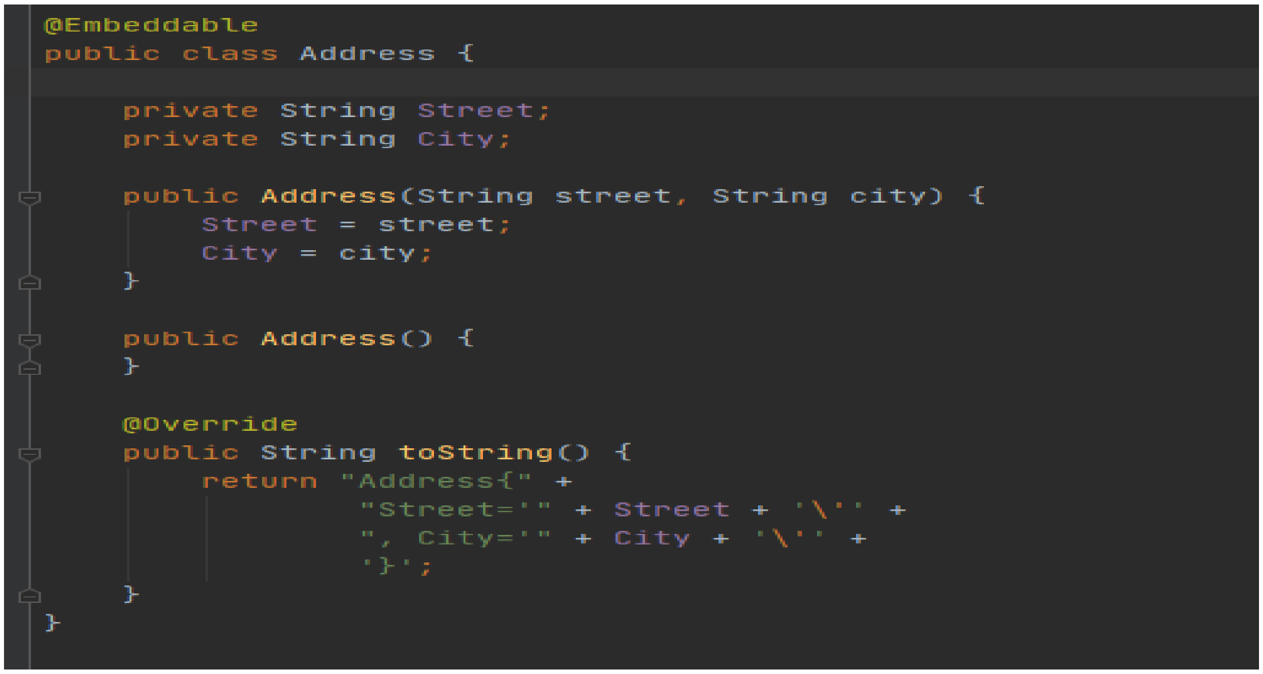


	INVOICES_INVOICENUMBER	PRODUCTS_PRODUCTID
1	1	2
2	1	9
3	7	5
4	7	8

Tabela Invoice-Products

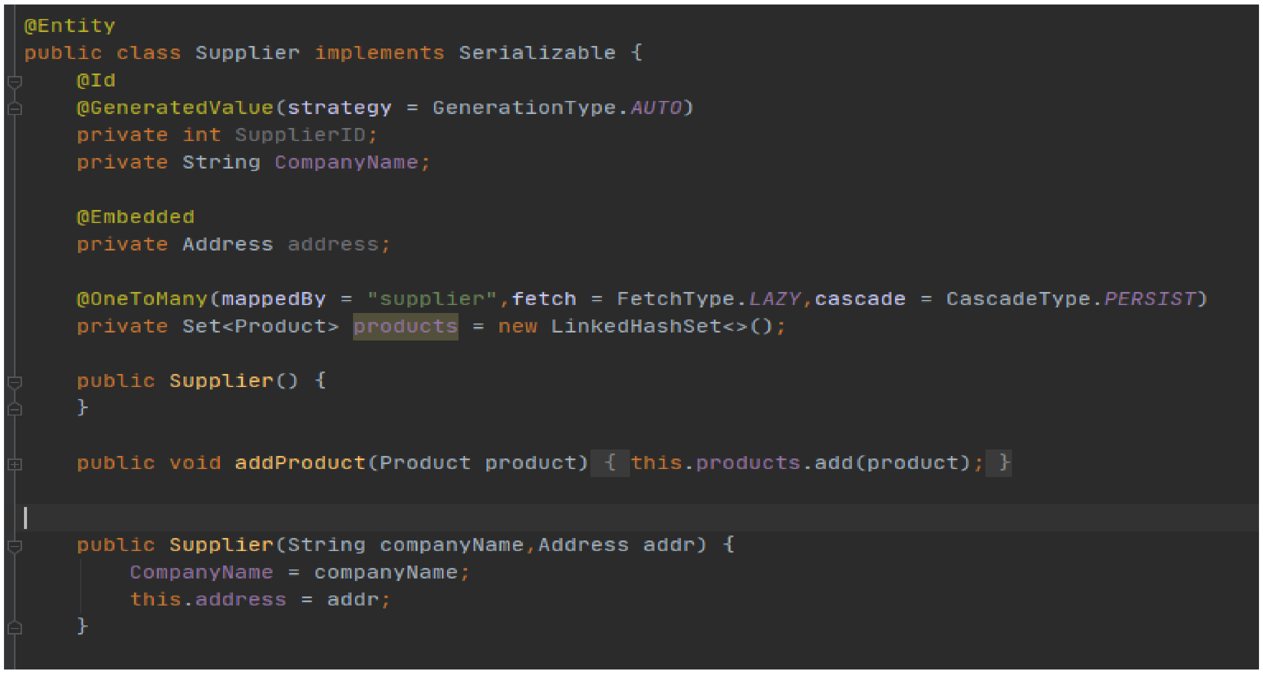
10 Embedded class

10.1 Dodaj do modelu klase adres. „Wbuduj” ją do tabeli Dostawców.



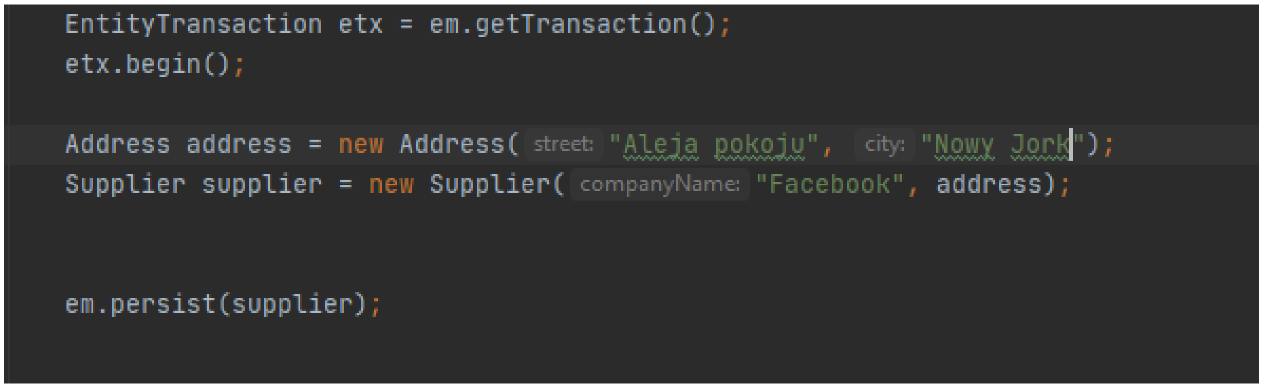
```java
@Embeddable
public class Address {

    private String Street;
    private String City;

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

    public Address() {
    }

    @Override
    public String toString() {
        return "Address{" +
                "Street='" + Street + '\'' +
                ", City='" + City + '\'' +
                '}';
    }
}
```

Klasa Address



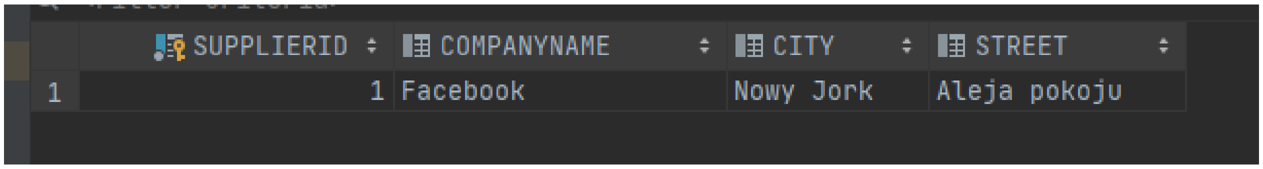
```java
@Entity
public class Supplier implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int SupplierID;
    private String CompanyName;

    @Embedded
    private Address address;

    @OneToMany(mappedBy = "supplier",fetch = FetchType.LAZY,cascade = CascadeType.PERSIST)
    private Set<Product> products = new LinkedHashSet<>();

    public Supplier() {
    }

    public void addProduct(Product product) { this.products.add(product); }

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

Klasa Supplier



```
EntityTransaction etx = em.getTransaction();
etx.begin();

Address address = new Address( street: "Aleja pokoju",  city: "Nowy Jork");
Supplier supplier = new Supplier( companyName: "Facebook", address);


em.persist(supplier);
```

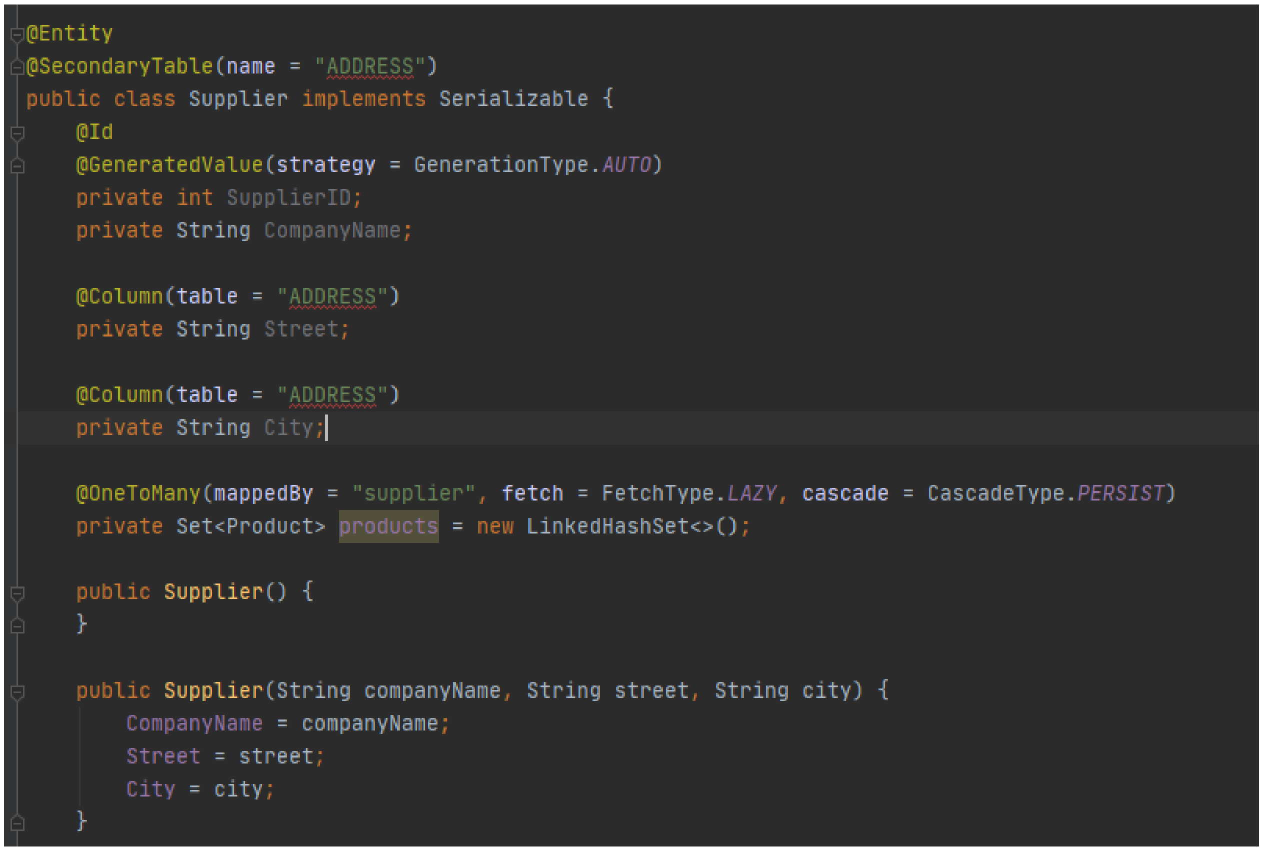
Main



	SUPPLIERID	COMPANYNAME	CITY	STREET
1	1	Facebook	Nowy Jork	Aleja pokoju

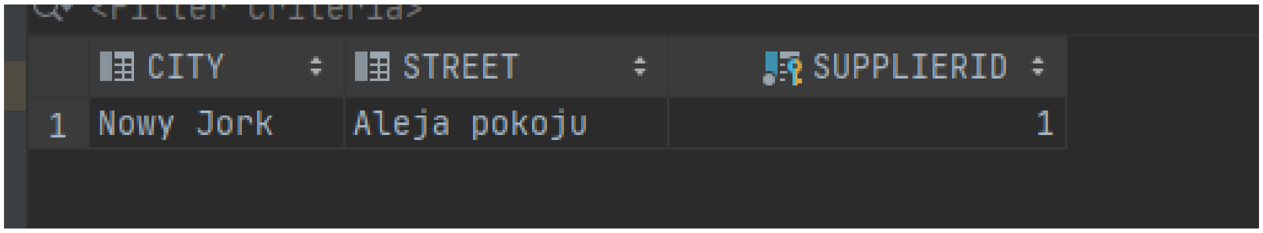
Tabela Supplier

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



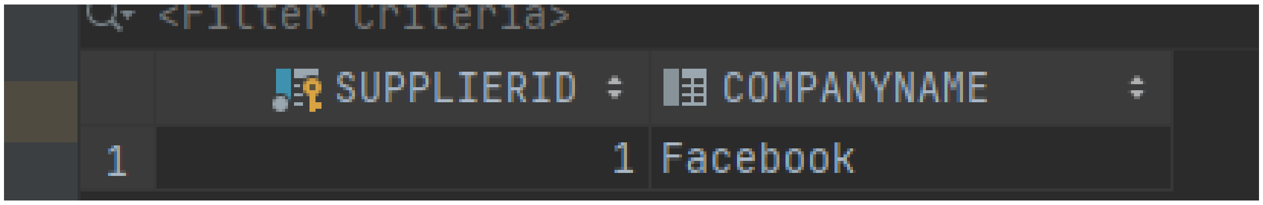
```
@Entity
@SecondaryTable(name = "ADDRESS")
public class Supplier implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int SupplierID;
    private String CompanyName;

    @Column(table = "ADDRESS")
    private String Street;

    @Column(table = "ADDRESS")
    private String City;

    @OneToMany(mappedBy = "supplier", fetch = FetchType.LAZY, cascade = CascadeType.PERSIST)
    private Set<Product> products = new LinkedHashSet<>();

    public Supplier() {
    }

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

Klasa Supplier



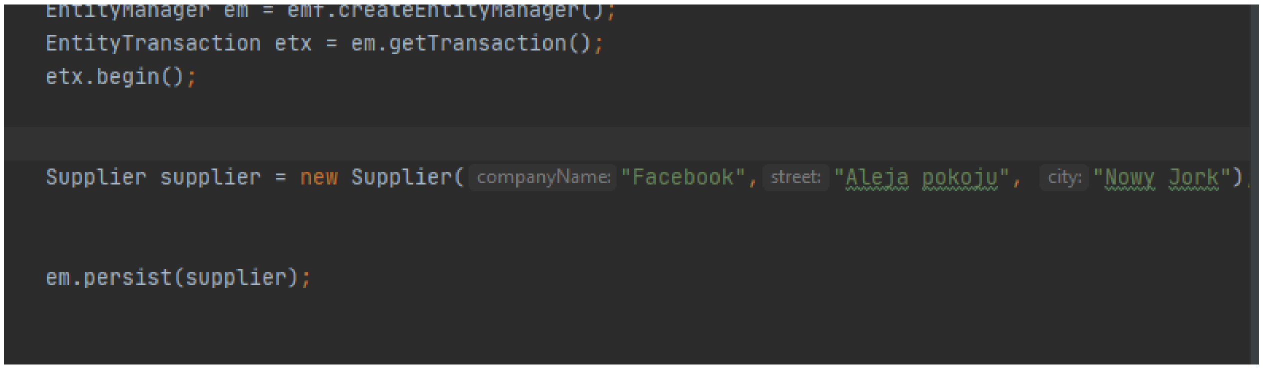
	CITY	STREET	SUPPLIERID
1	Nowy Jork	Aleja pokoju	1

Tabela Address



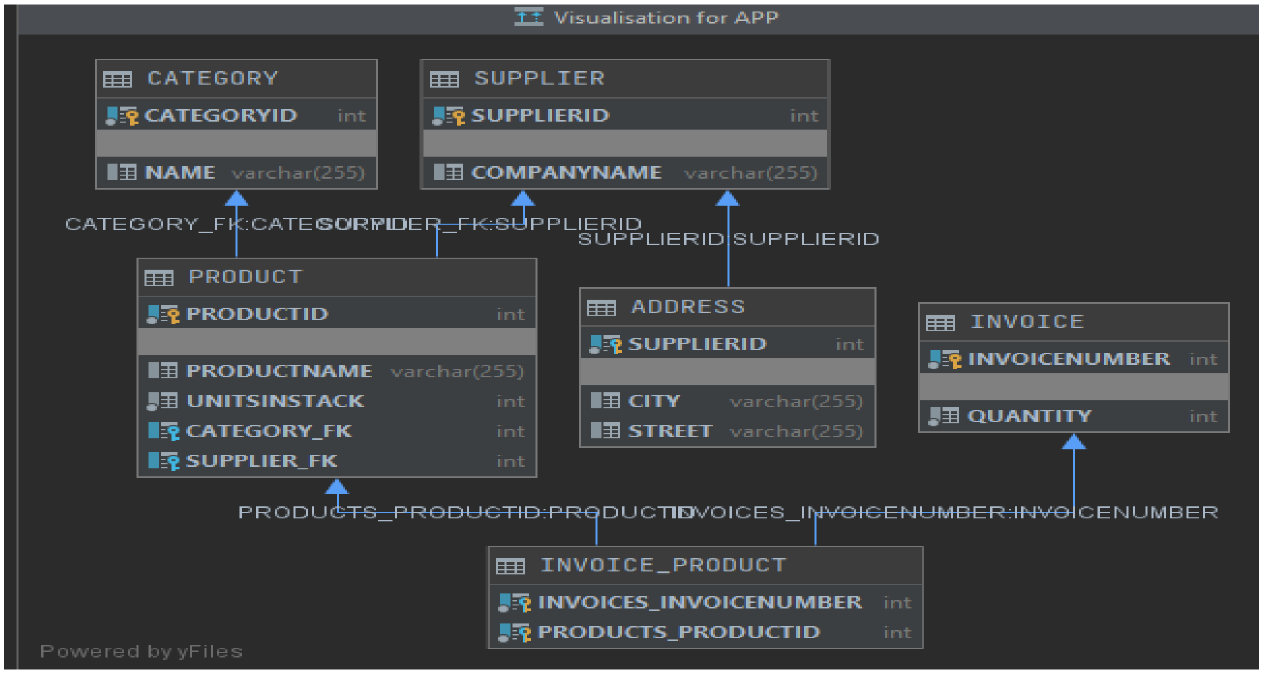
	SUPPLIERID	COMPANYNAME
1	1	Facebook

Tabela Supplier



```
EntityManager em = emf.createEntityManager();
EntityTransaction etx = em.getTransaction();
etx.begin();


Supplier supplier = new Supplier( companyName: "Facebook", street: "Aleja pokoju",  city: "Nowy Jork").


em.persist(supplier);
```

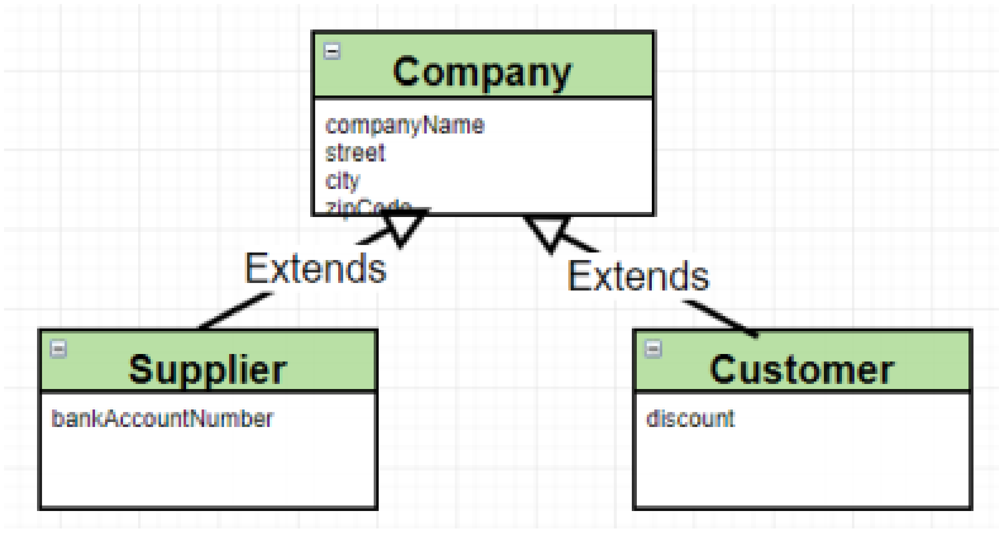
Main



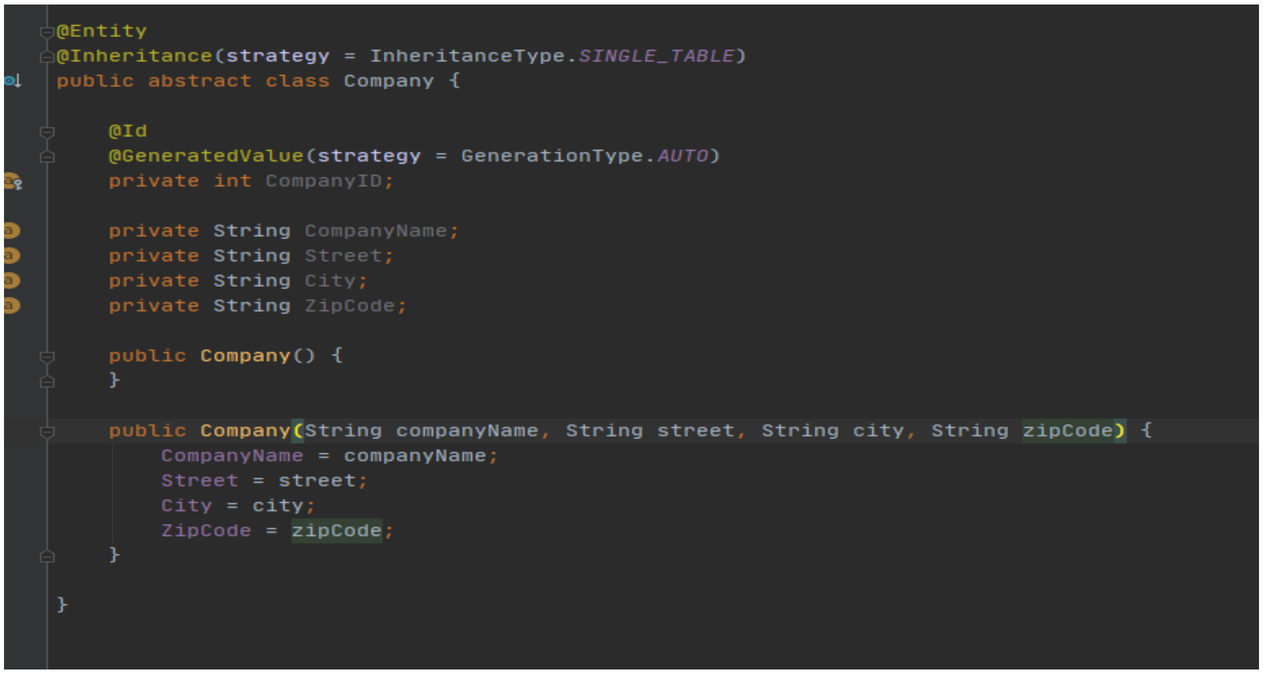
Schemat Bazy danych

11 Dziedziczenie

11.1 Wprowadź do modelu następującą hierarchie:

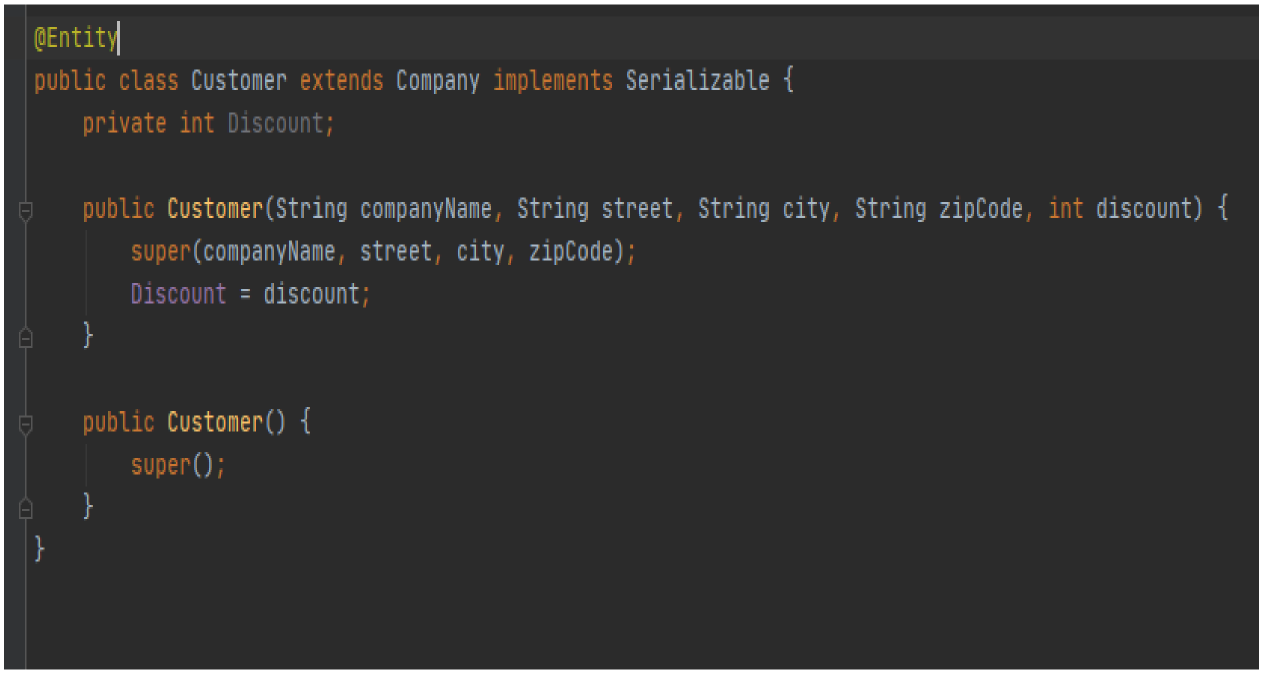


11.2 SINGLE TABLE



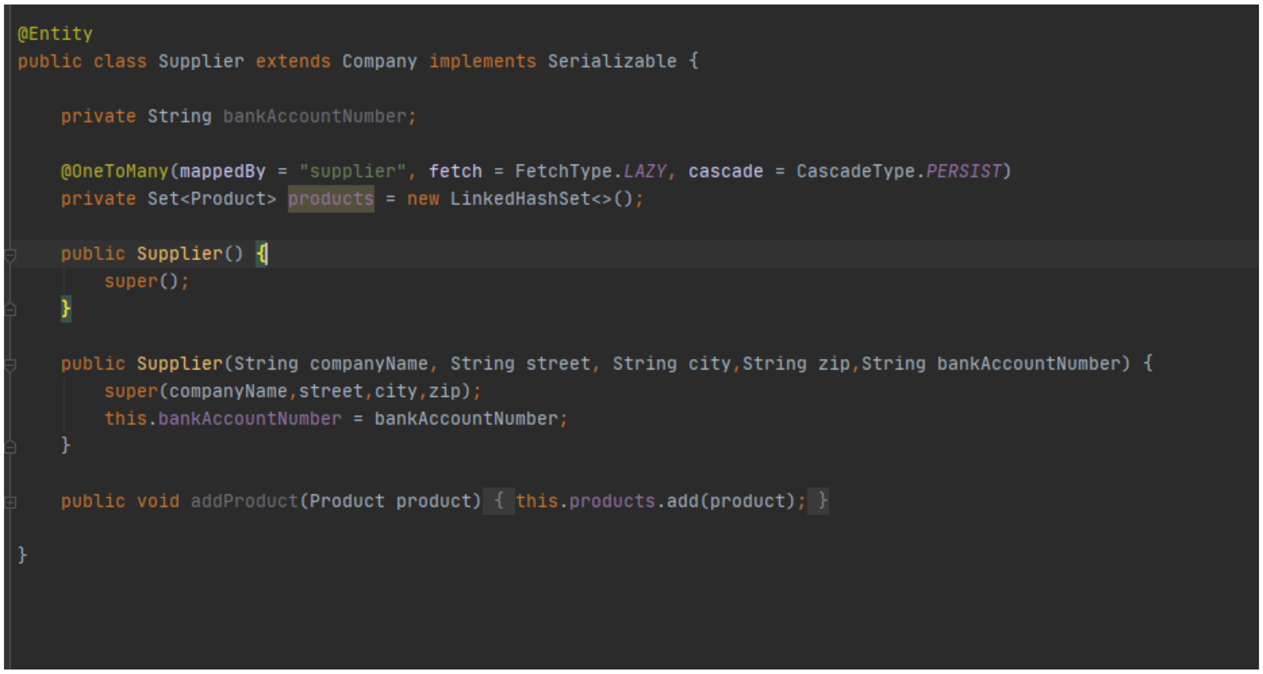
```
@Entity
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
public abstract class Company {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int CompanyID;

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

    public Company() {
    }

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

}
```

Klasa Company



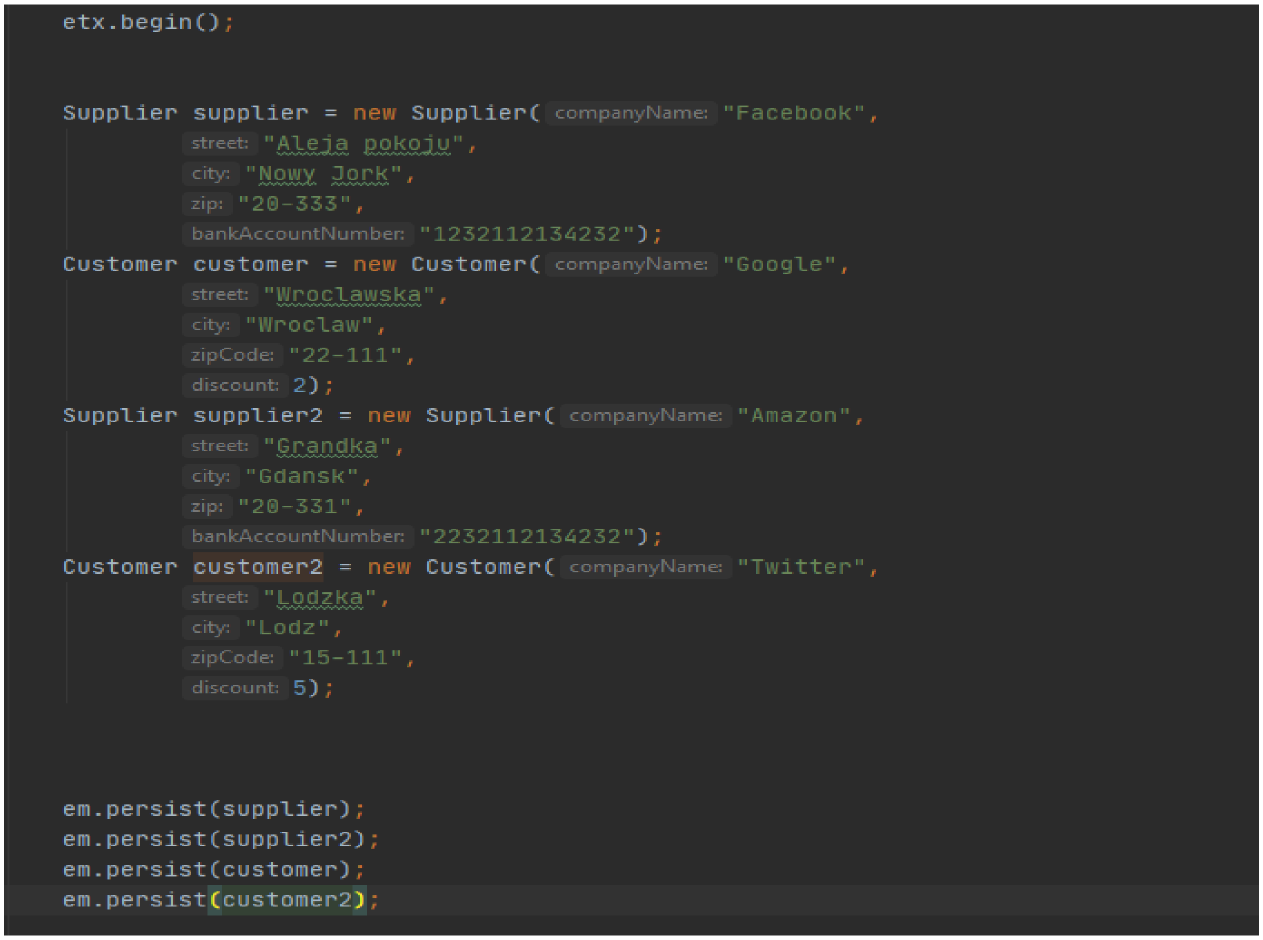
```java
@Entity
public class Customer extends Company implements Serializable {
    private int Discount;

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

    public Customer() {
        super();
    }
}
```

Klasa Customer



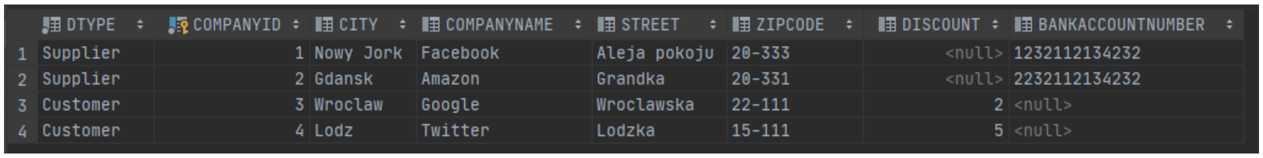
```java
@Entity
public class Supplier extends Company implements Serializable {

    private String bankAccountNumber;

    @OneToMany(mappedBy = "supplier", fetch = FetchType.LAZY, cascade = CascadeType.PERSIST)
    private Set<Product> products = new LinkedHashSet<>();

    public Supplier() {
        super();
    }

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

    public void addProduct(Product product) { this.products.add(product); }

}
```

Klasa Supplier



```
etx.begin();


Supplier supplier = new Supplier( companyName: "Facebook",
        street: "Aleja pokoju",
        city: "Nowy Jork",
        zip: "20-333",
        bankAccountNumber: "1232112134232");
Customer customer = new Customer( companyName: "Google",
        street: "Wroclawska",
        city: "Wroclaw",
        zipCode: "22-111",
        discount: 2);
Supplier supplier2 = new Supplier( companyName: "Amazon",
        street: "Grandka",
        city: "Gdansk",
        zip: "20-331",
        bankAccountNumber: "2232112134232");
Customer customer2 = new Customer( companyName: "Twitter",
        street: "Lodzka",
        city: "Lodz",
        zipCode: "15-111",
        discount: 5);



em.persist(supplier);
em.persist(supplier2);
em.persist(customer);
em.persist(customer2);
```

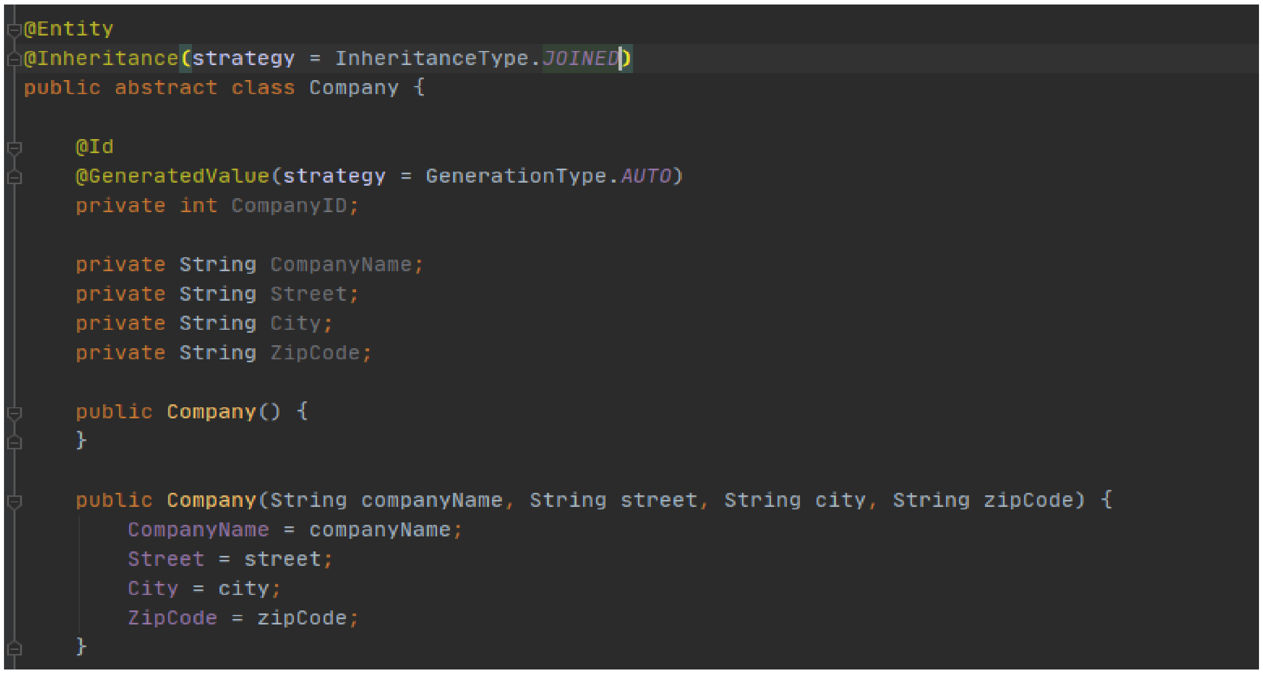
Main



	DTYPE	COMPANYID	CITY	COMPANYNAME	STREET	ZIPCODE	DISCOUNT	BANKACCOUNTNUMBER
1	Supplier	1	Nowy Jork	Facebook	Aleja pokoju	20-333	<null>	1232112134232
2	Supplier	2	Gdansk	Amazon	Grandka	20-331	<null>	2232112134232
3	Customer	3	Wroclaw	Google	Wroclawska	22-111	2	<null>
4	Customer	4	Lodz	Twitter	Lodzka	15-111	5	<null>

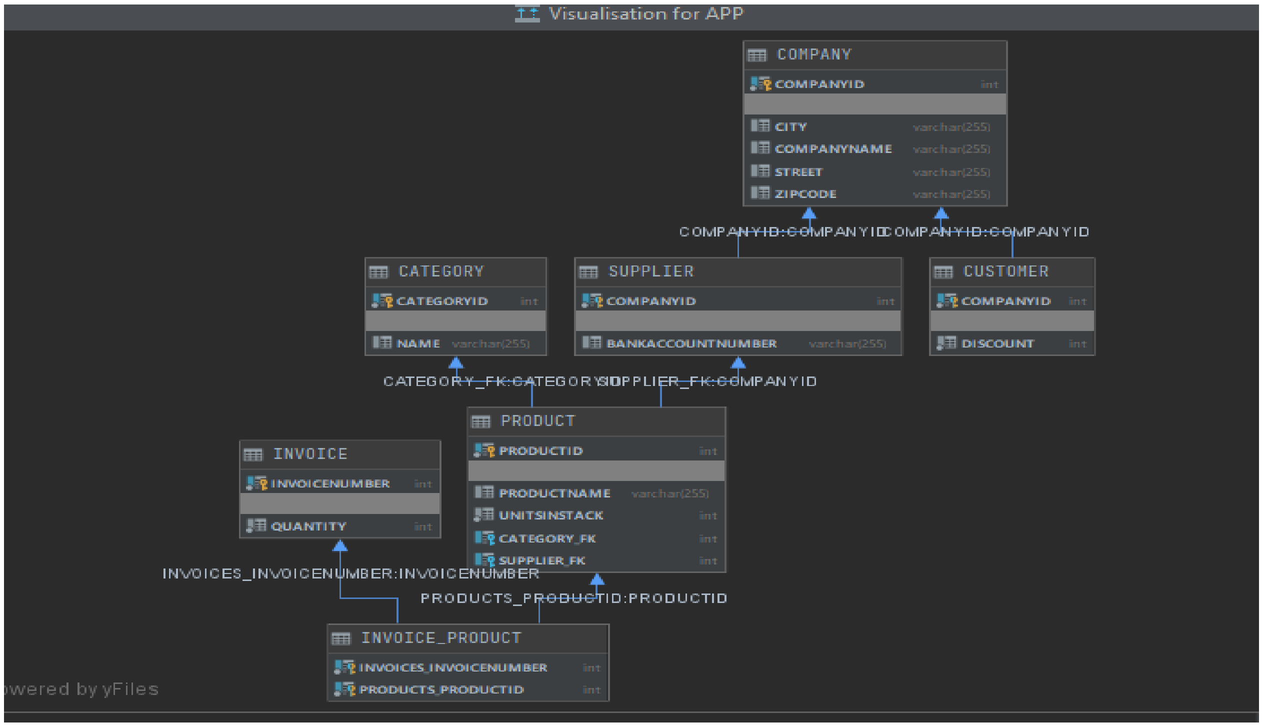
Tabela Company

11.3 JOINED



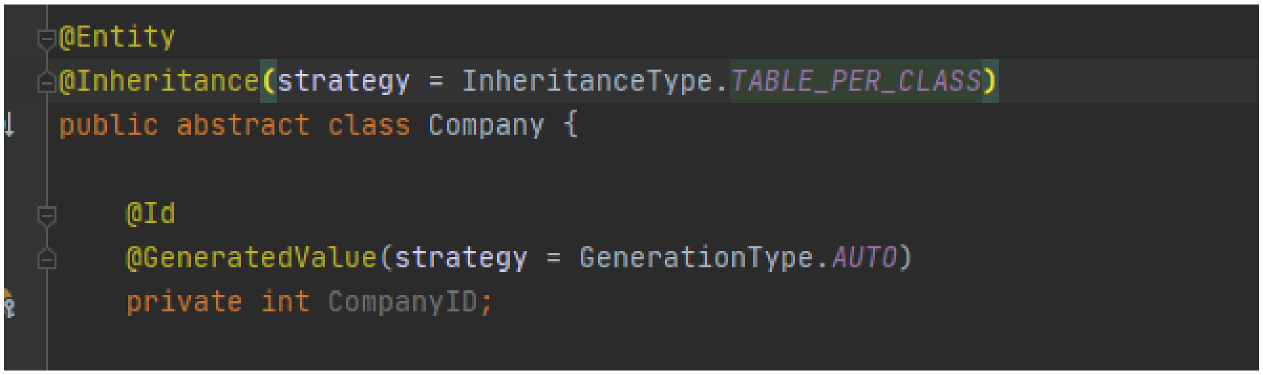
```java
@Entity
@Inheritance(strategy = InheritanceType.JOINED)
public abstract class Company {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int CompanyID;

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

    public Company() {
    }

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

Klasa Company



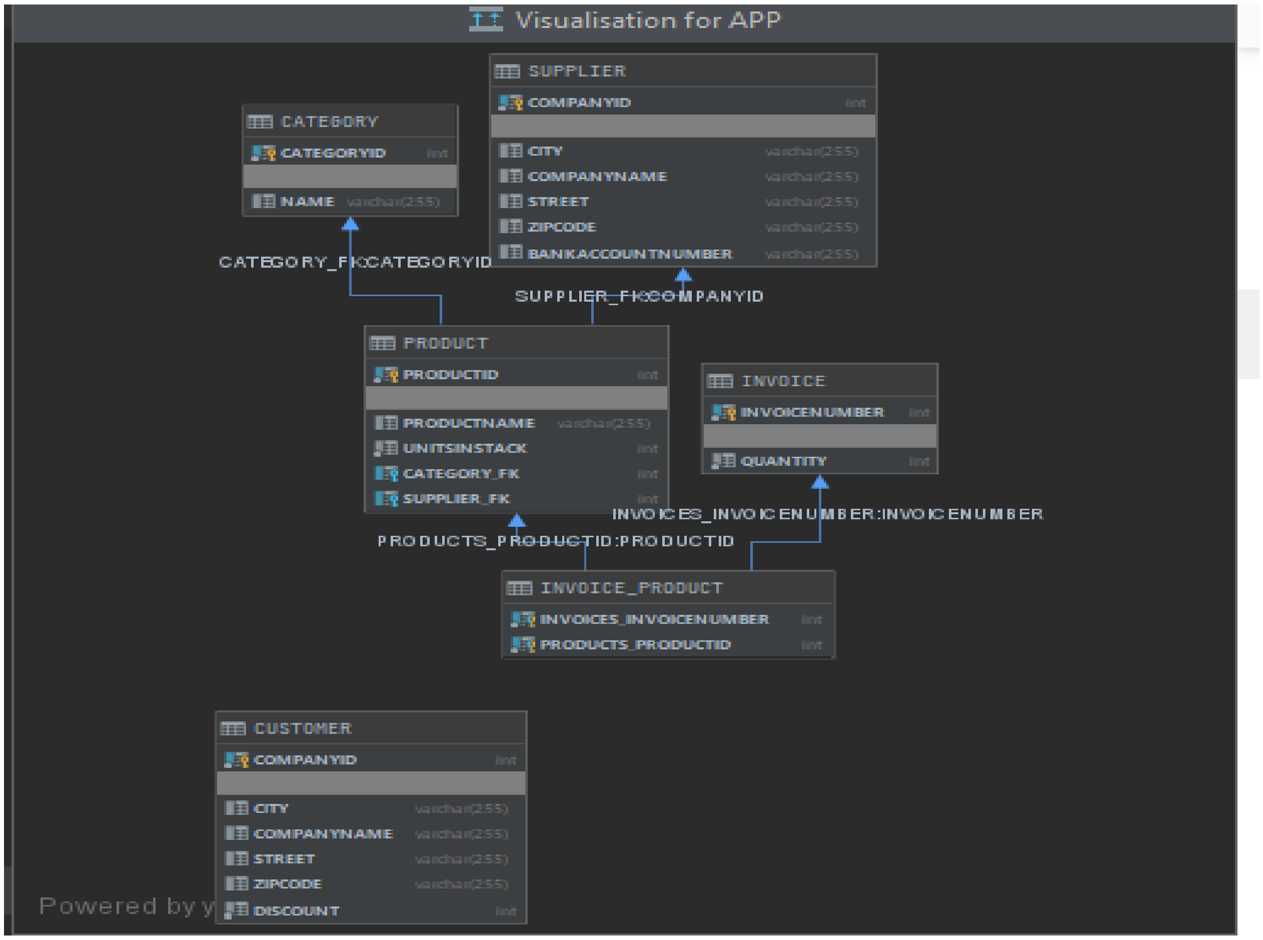
Schemat Bazy danych

11.4 TABLE PER CLASS



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

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int CompanyID;
```

Klasa Company



Schemat Bazy danych