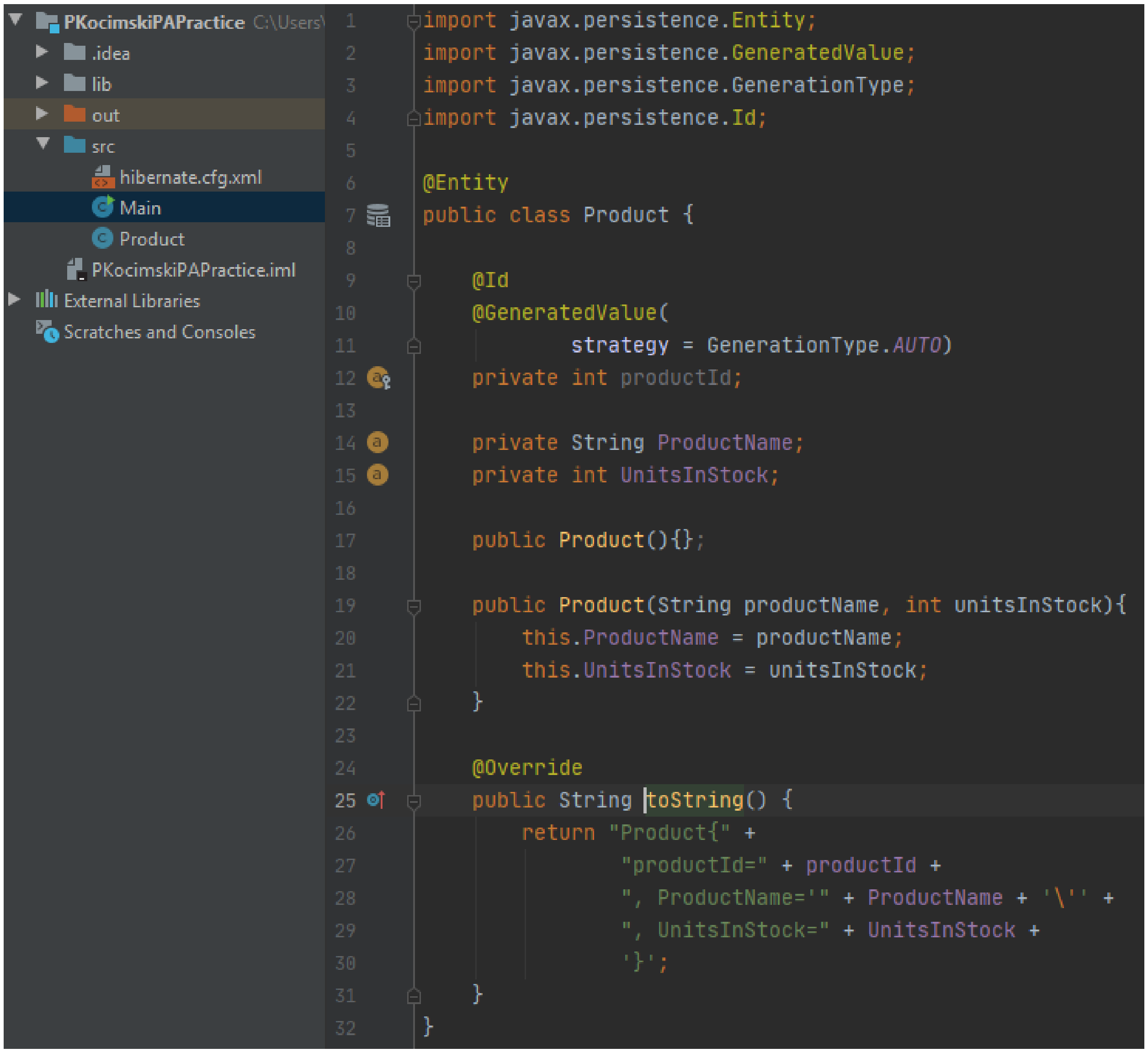
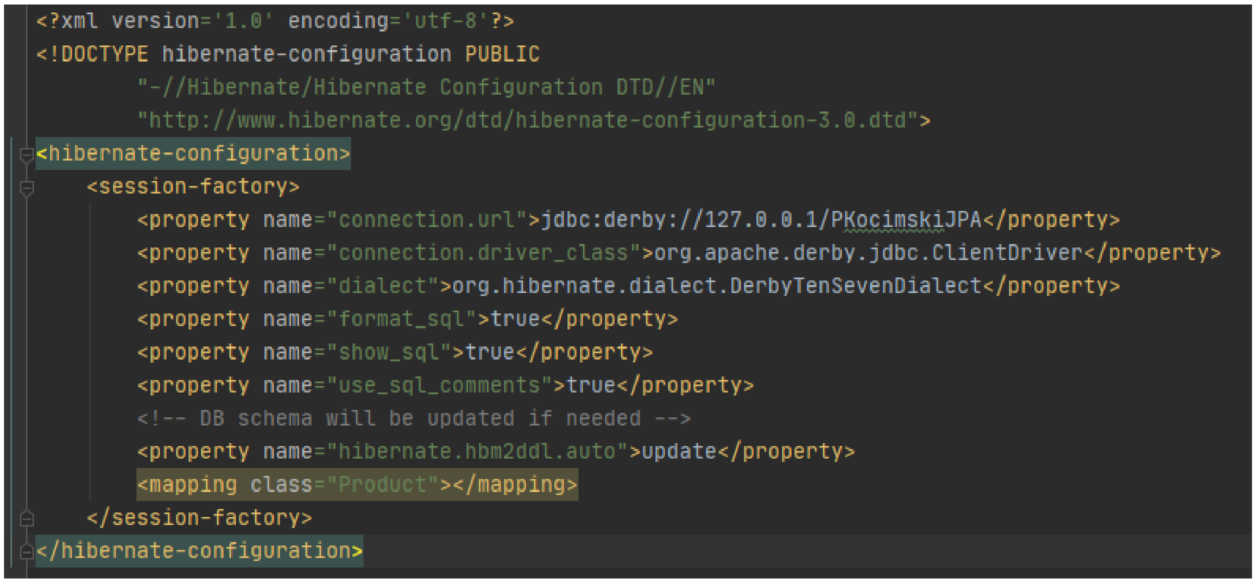
Stworzona klasa Product z możliwością mapowania do bazy



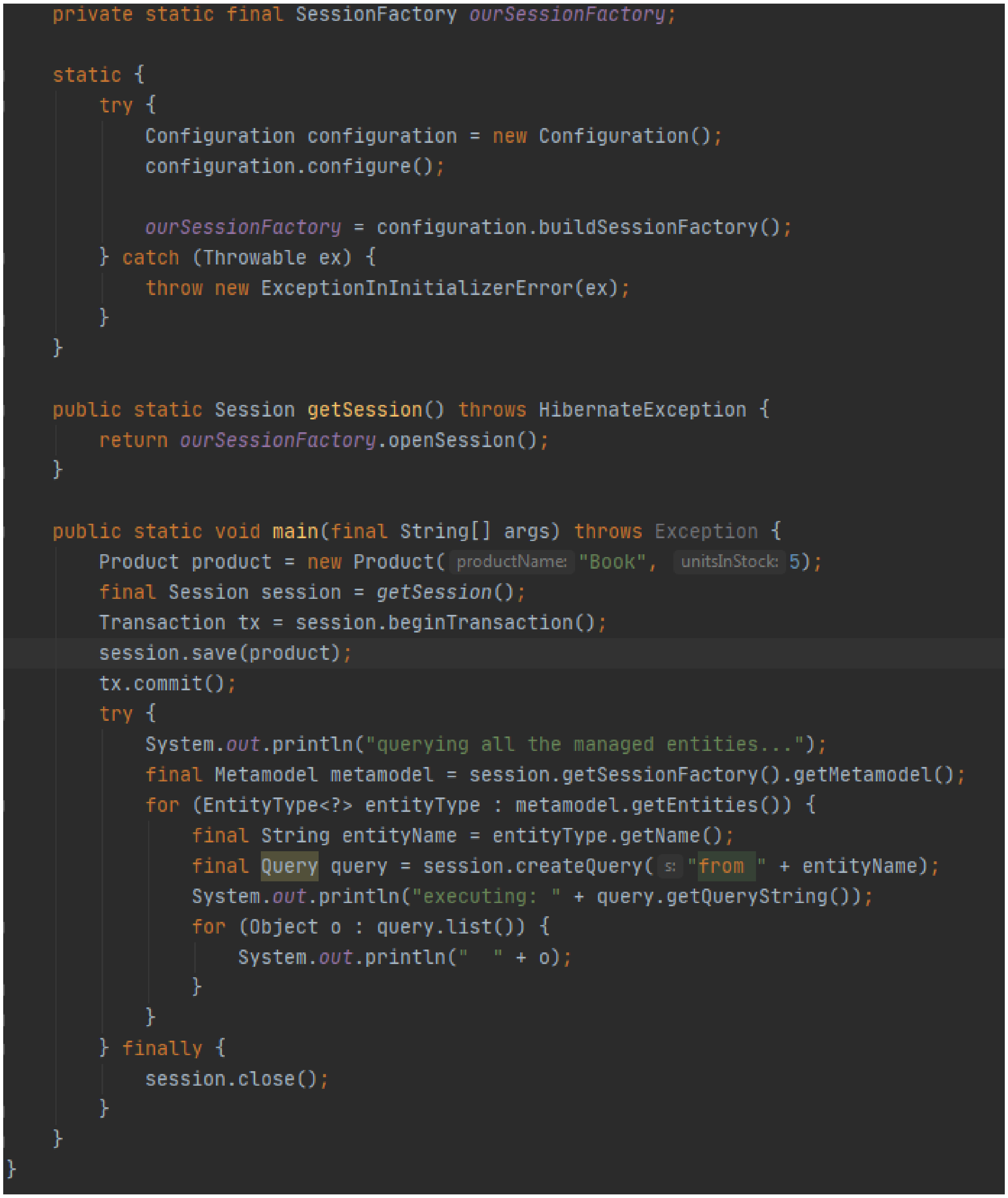
```java
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Product {

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

    private String ProductName;
    private int UnitsInStock;

    public Product(){};

    public Product(String productName, int unitsInStock){
        this.ProductName = productName;
        this.UnitsInStock = unitsInStock;
    }

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

Konfiguracja Hibernate:

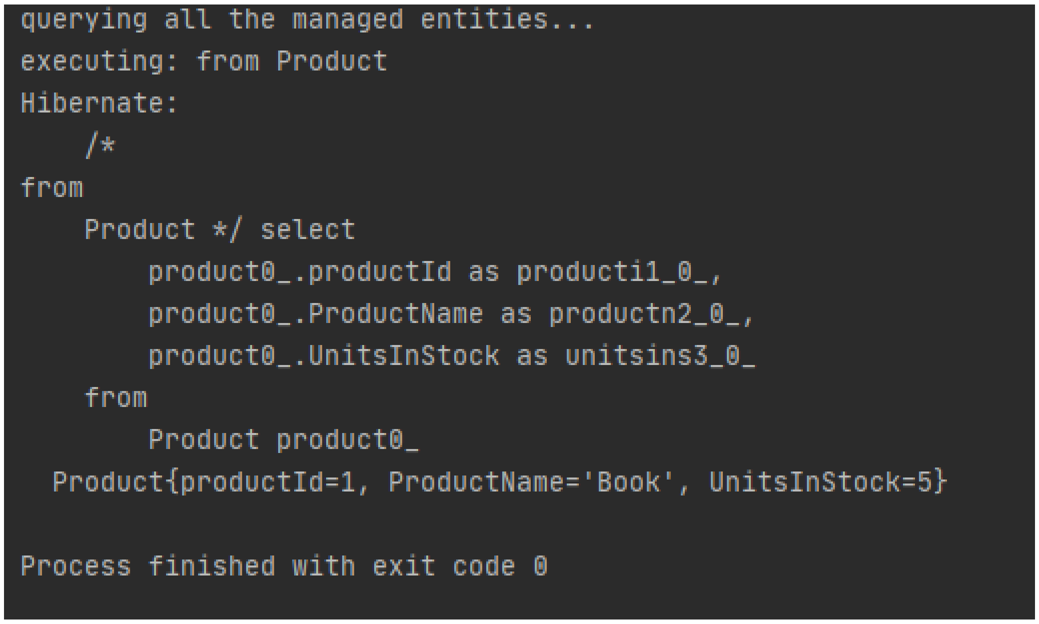


```xml
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="connection.url">jdbc:derby://127.0.0.1/PKocimskiJPA</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>
        <!-- DB schema will be updated if needed -->
        <property name="hibernate.hbm2ddl.auto">update</property>
        <mapping class="Product"></mapping>
    </session-factory>
</hibernate-configuration>
```

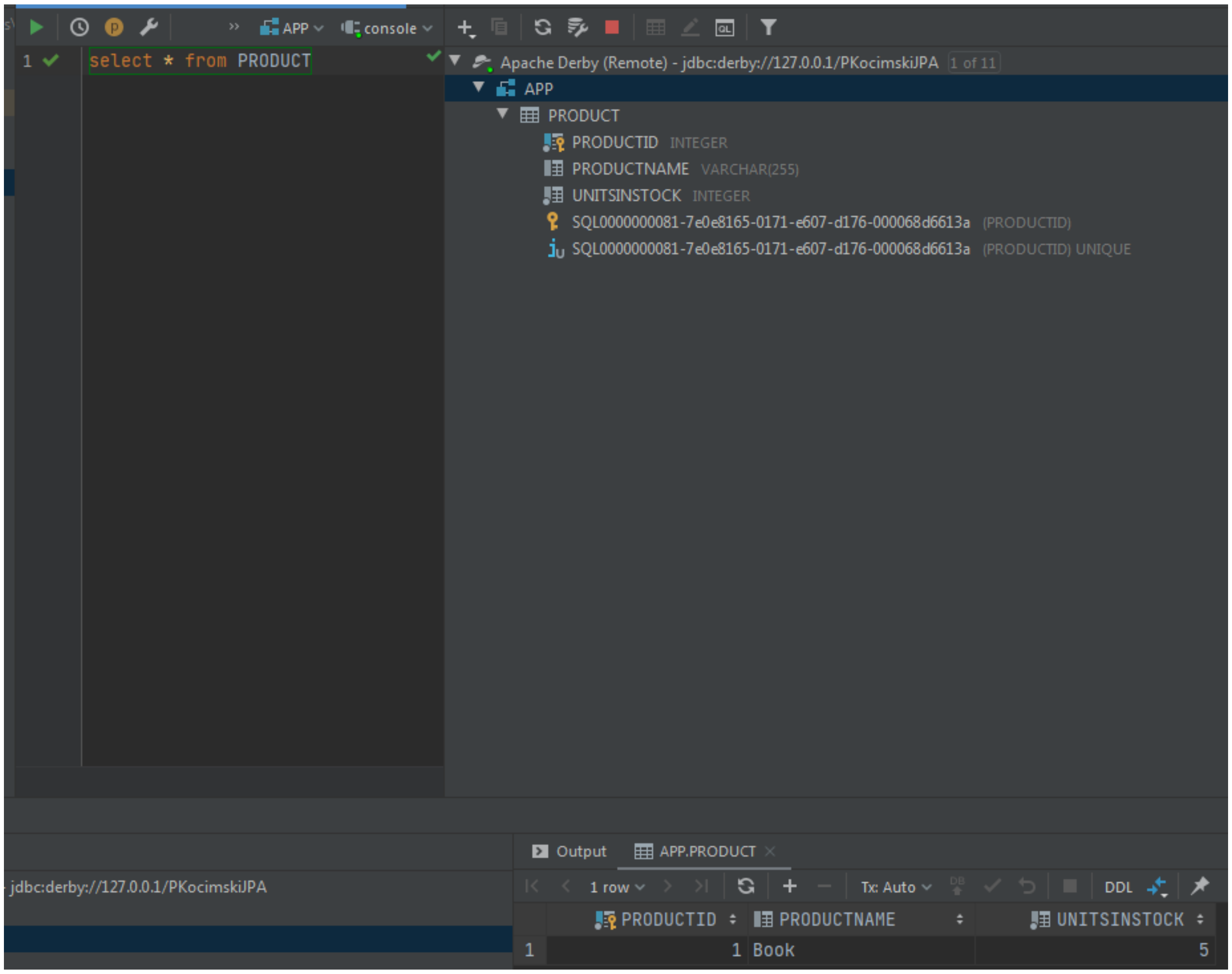
Klasa main



```java
    private static final SessionFactory ourSessionFactory;

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

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

    public static Session getSession() throws HibernateException {
        return ourSessionFactory.openSession();
    }

    public static void main(final String[] args) throws Exception {
        Product product = new Product( productName: "Book",  unitsInStock: 5);
        final Session session = getSession();
        Transaction tx = session.beginTransaction();
        session.save(product);
        tx.commit();
        try {
            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 {
            session.close();
        }
    }
}
```



```
querying all the managed entities...
executing: from Product
Hibernate:
    /*
from
    Product */ select
        product0_.productId as producti1_0_,
        product0_.ProductName as productn2_0_,
        product0_.UnitsInStock as unitsins3_0_
    from
        Product product0_
  Product{productId=1, ProductName='Book', UnitsInStock=5}

Process finished with exit code 0
```



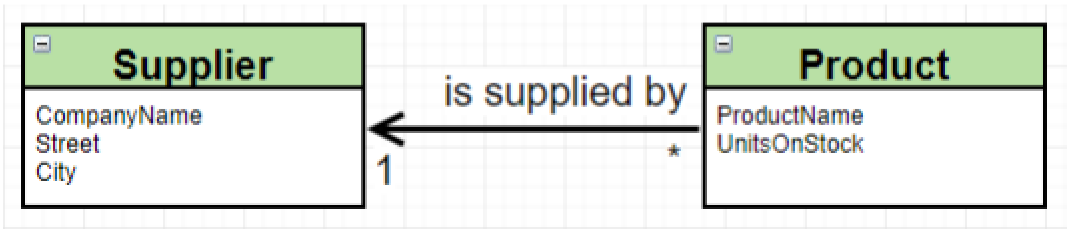
```
select * from PRODUCT
```

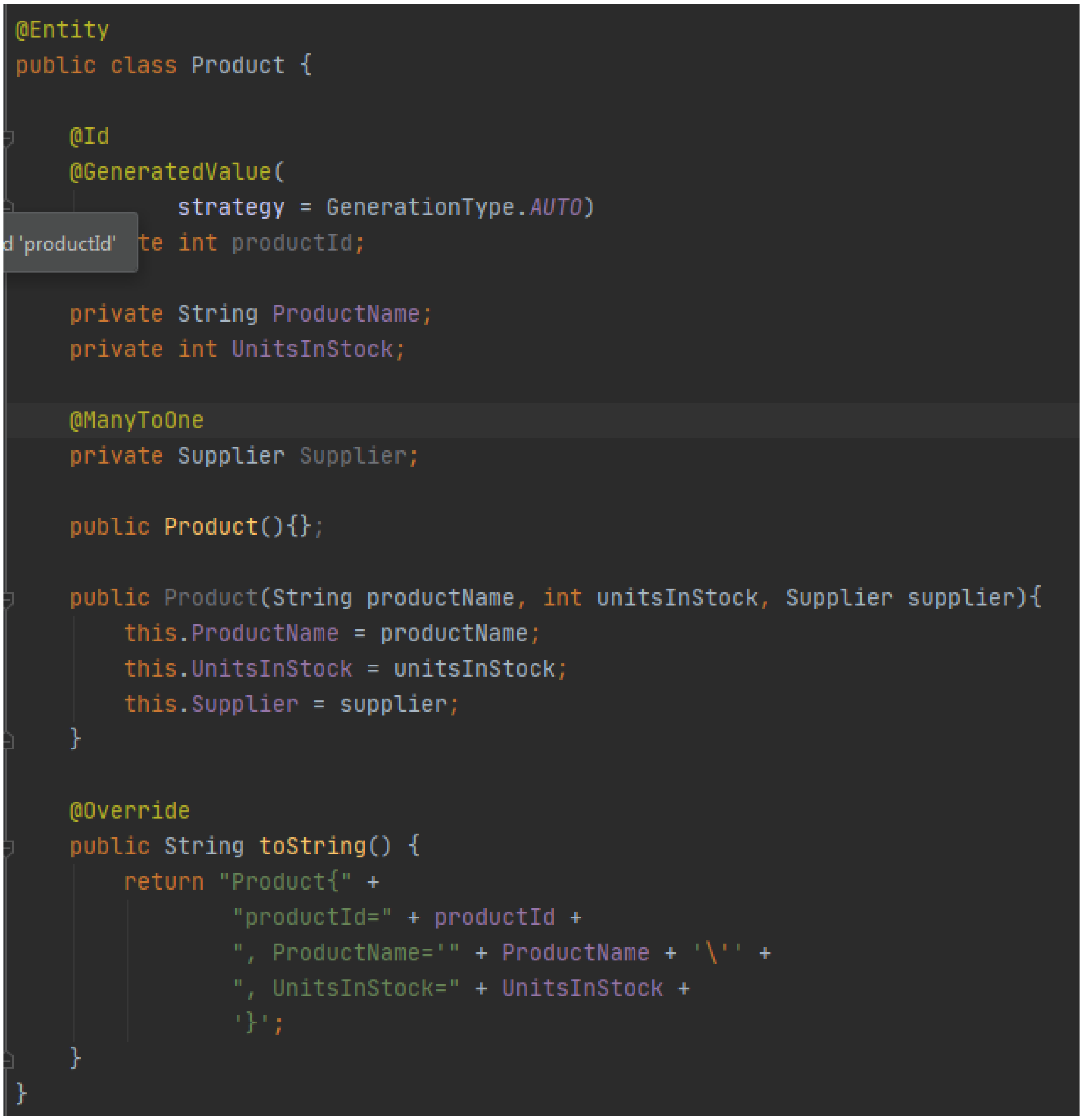
```
Apache Derby (Remote) - jdbc:derby://127.0.0.1/PKocimskiJPA  1 of 11
  APP
    PRODUCT
      PRODUCTID  INTEGER
      PRODUCTNAME  VARCHAR(255)
      UNITSINSTOCK  INTEGER
      SQL0000000081-7e0e8165-0171-e607-d176-000068d6613a  (PRODUCTID)
      SQL0000000081-7e0e8165-0171-e607-d176-000068d6613a  (PRODUCTID) UNIQUE
```

jdbc:derby://127.0.0.1/PKocimskiJPA

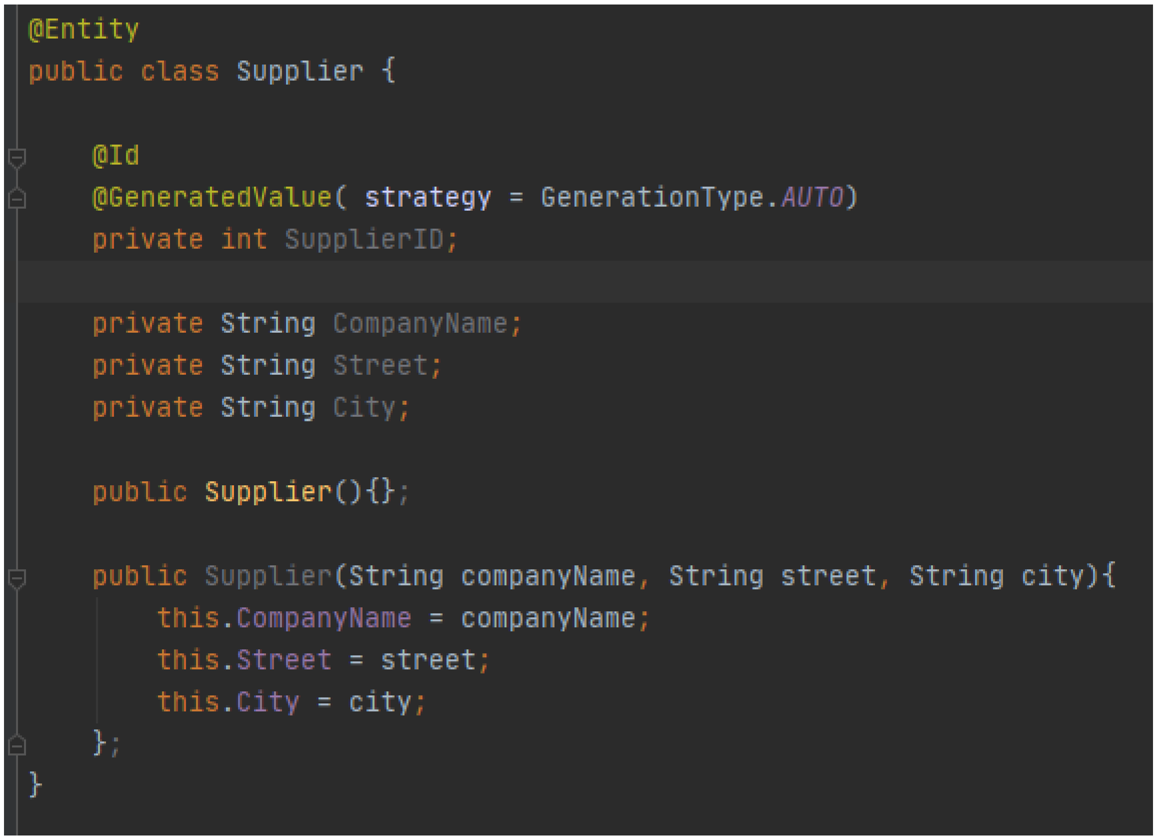
	PRODUCTID	PRODUCTNAME	UNITSINSTOCK
1	1	Book	5

IV. Zmodyfikuj model wprowadzając pojęcie Dostawcy jak poniżej



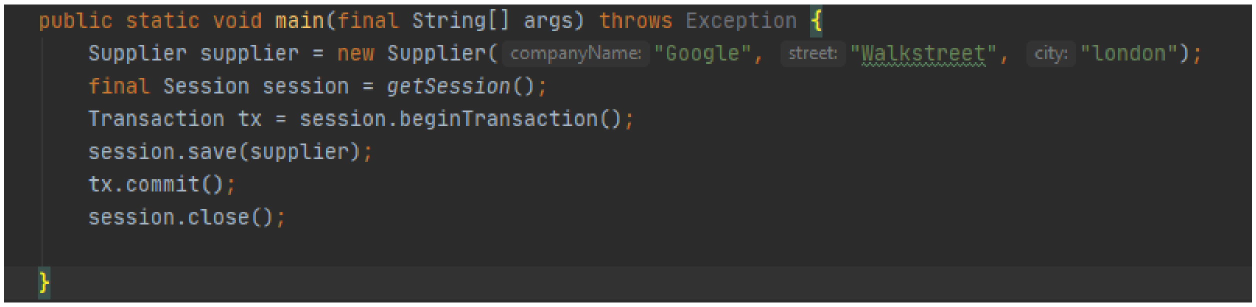


```
@Entity
public class Product {

    @Id
    @GeneratedValue(
            strategy = GenerationType.AUTO)
d 'productId' te int productId;

    private String ProductName;
    private int UnitsInStock;

    @ManyToOne
    private Supplier Supplier;

    public Product(){};

    public Product(String productName, int unitsInStock, Supplier supplier){
        this.ProductName = productName;
        this.UnitsInStock = unitsInStock;
        this.Supplier = supplier;
    }

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

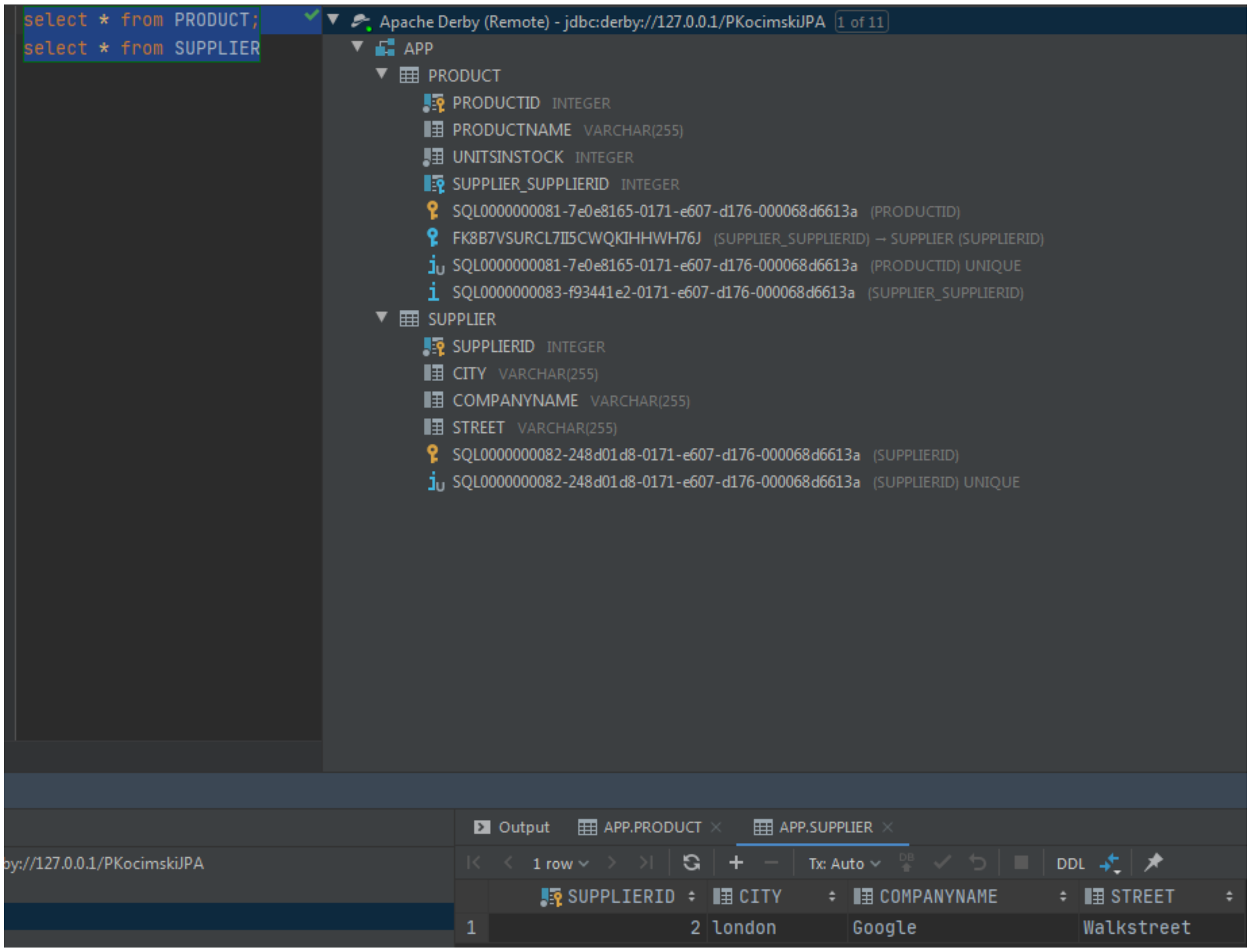


```java
@Entity
public class Supplier {

    @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;
    };
}
```

a. Stwórz nowego dostawce



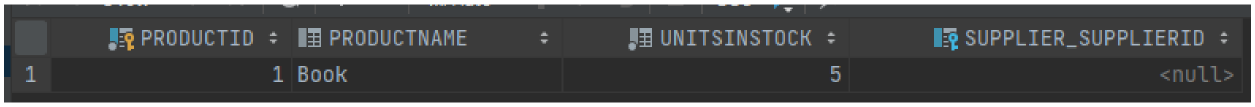
```java
public static void main(final String[] args) throws Exception {
    Supplier supplier = new Supplier( companyName: "Google", street: "Walkstreet", city: "london");
    final Session session = getSession();
    Transaction tx = session.beginTransaction();
    session.save(supplier);
    tx.commit();
    session.close();

}
```



```
select * from PRODUCT;
select * from SUPPLIER
```

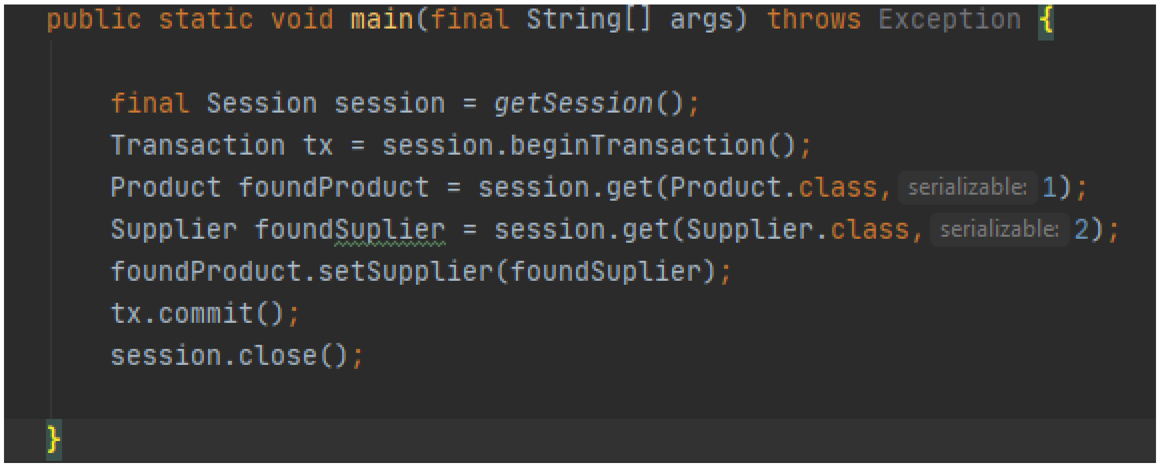
```
Apache Derby (Remote) - jdbc:derby://127.0.0.1/PKocimskiJPA  1 of 11
  APP
    PRODUCT
      PRODUCTID  INTEGER
      PRODUCTNAME  VARCHAR(255)
      UNITSINSTOCK  INTEGER
      SUPPLIER_SUPPLIERID  INTEGER
      SQL0000000081-7e0e8165-0171-e607-d176-000068d6613a  (PRODUCTID)
      FK8B7VSURCL7II5CWQKIHHWH76J  (SUPPLIER_SUPPLIERID) → SUPPLIER (SUPPLIERID)
      SQL0000000081-7e0e8165-0171-e607-d176-000068d6613a  (PRODUCTID) UNIQUE
      SQL0000000083-f93441e2-0171-e607-d176-000068d6613a  (SUPPLIER_SUPPLIERID)
    SUPPLIER
      SUPPLIERID  INTEGER
      CITY  VARCHAR(255)
      COMPANYNAME  VARCHAR(255)
      STREET  VARCHAR(255)
      SQL0000000082-248d01d8-0171-e607-d176-000068d6613a  (SUPPLIERID)
      SQL0000000082-248d01d8-0171-e607-d176-000068d6613a  (SUPPLIERID) UNIQUE
```

	SUPPLIERID	CITY	COMPANYNAME	STREET
1	2	london	Google	Walkstreet

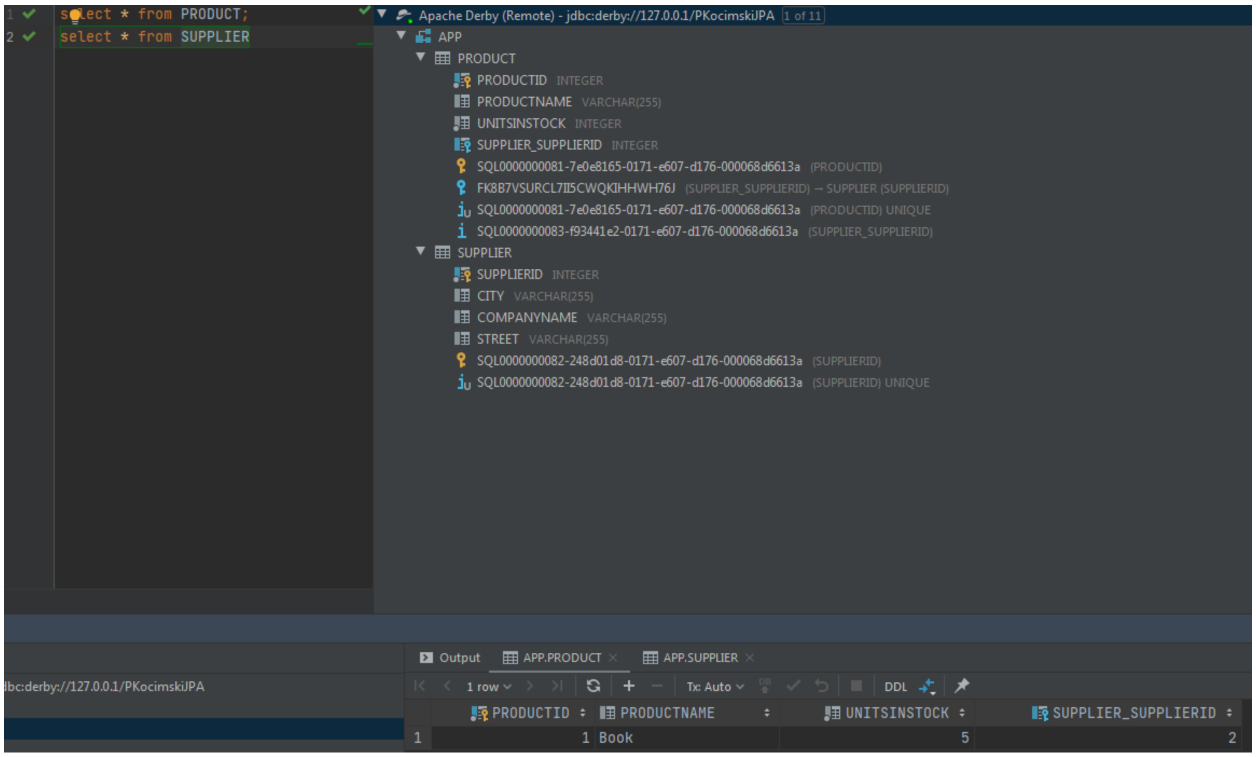


	PRODUCTID	PRODUCTNAME	UNITSINSTOCK	SUPPLIER_SUPPLIERID
1	1	Book	5	<null>

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



```java
public static void main(final String[] args) throws Exception {

    final Session session = getSession();
    Transaction tx = session.beginTransaction();
    Product foundProduct = session.get(Product.class, serializable: 1);
    Supplier foundSuplier = session.get(Supplier.class, serializable: 2);
    foundProduct.setSupplier(foundSuplier);
    tx.commit();
    session.close();

}
```

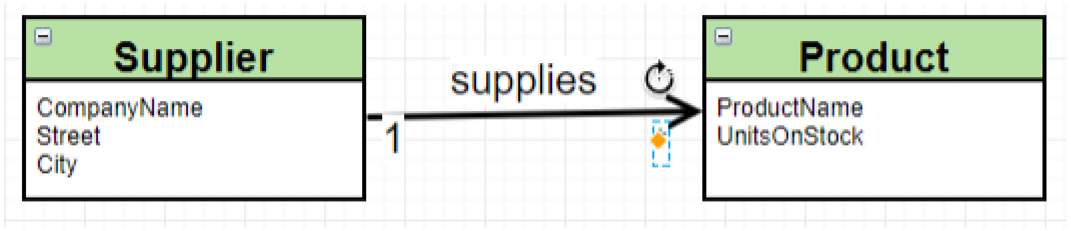


```
select * from PRODUCT;
select * from SUPPLIER
```

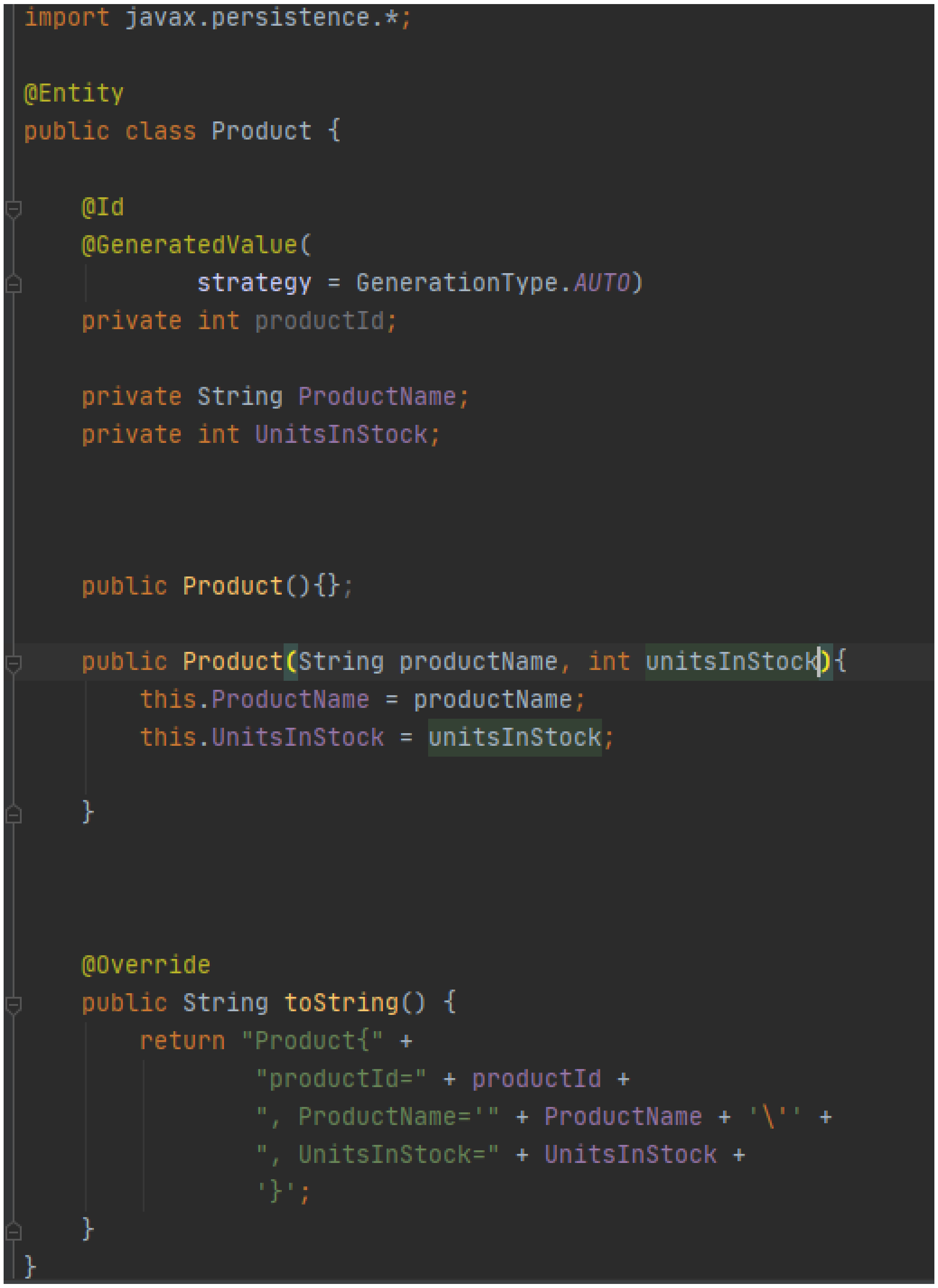
```
Apache Derby (Remote) - jdbc:derby://127.0.0.1/PKocimskiJPA  1 of 11
  APP
    PRODUCT
      PRODUCTID  INTEGER
      PRODUCTNAME  VARCHAR(255)
      UNITSINSTOCK  INTEGER
      SUPPLIER_SUPPLIERID  INTEGER
      SQL0000000081-7e0e8165-0171-e607-d176-000068d6613a  (PRODUCTID)
      FK8B7VSURCL7II5CWQKIHHWH76J  (SUPPLIER_SUPPLIERID) → SUPPLIER (SUPPLIERID)
      SQL0000000081-7e0e8165-0171-e607-d176-000068d6613a  (PRODUCTID) UNIQUE
      SQL0000000083-f93441e2-0171-e607-d176-000068d6613a  (SUPPLIER_SUPPLIERID)
    SUPPLIER
      SUPPLIERID  INTEGER
      CITY  VARCHAR(255)
      COMPANYNAME  VARCHAR(255)
      STREET  VARCHAR(255)
      SQL0000000082-248d01d8-0171-e607-d176-000068d6613a  (SUPPLIERID)
      SQL0000000082-248d01d8-0171-e607-d176-000068d6613a  (SUPPLIERID) UNIQUE
```

jdbc:derby://127.0.0.1/PKocimskiJPA

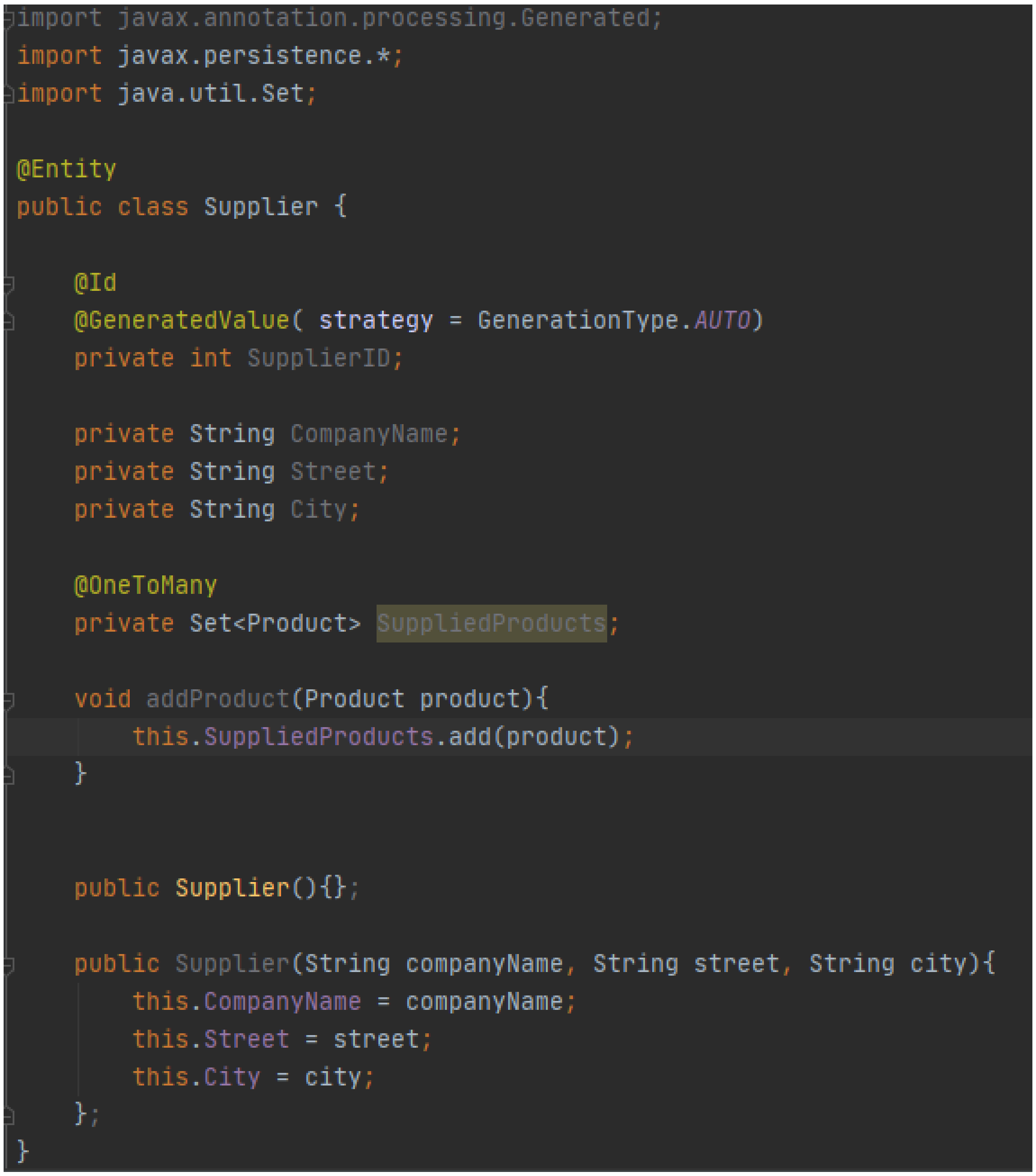
	PRODUCTID	PRODUCTNAME	UNITSINSTOCK	SUPPLIER_SUPPLIERID
1	1	Book	5	2



a. Zamodeluj powyższe „z” tabelą łącznikową

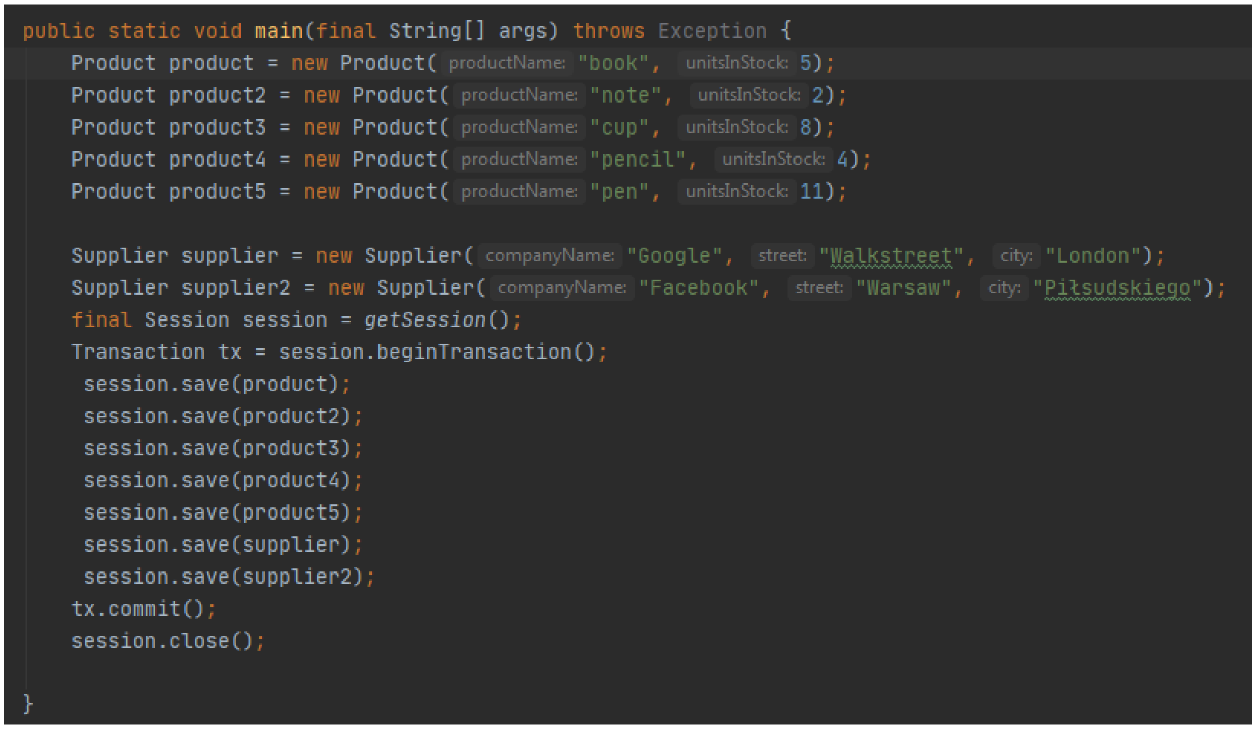


```java
import javax.persistence.*;

@Entity
public class Product {

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

    private String ProductName;
    private int UnitsInStock;



    public Product(){};

    public Product(String productName, int unitsInStock){
        this.ProductName = productName;
        this.UnitsInStock = unitsInStock;

    }



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



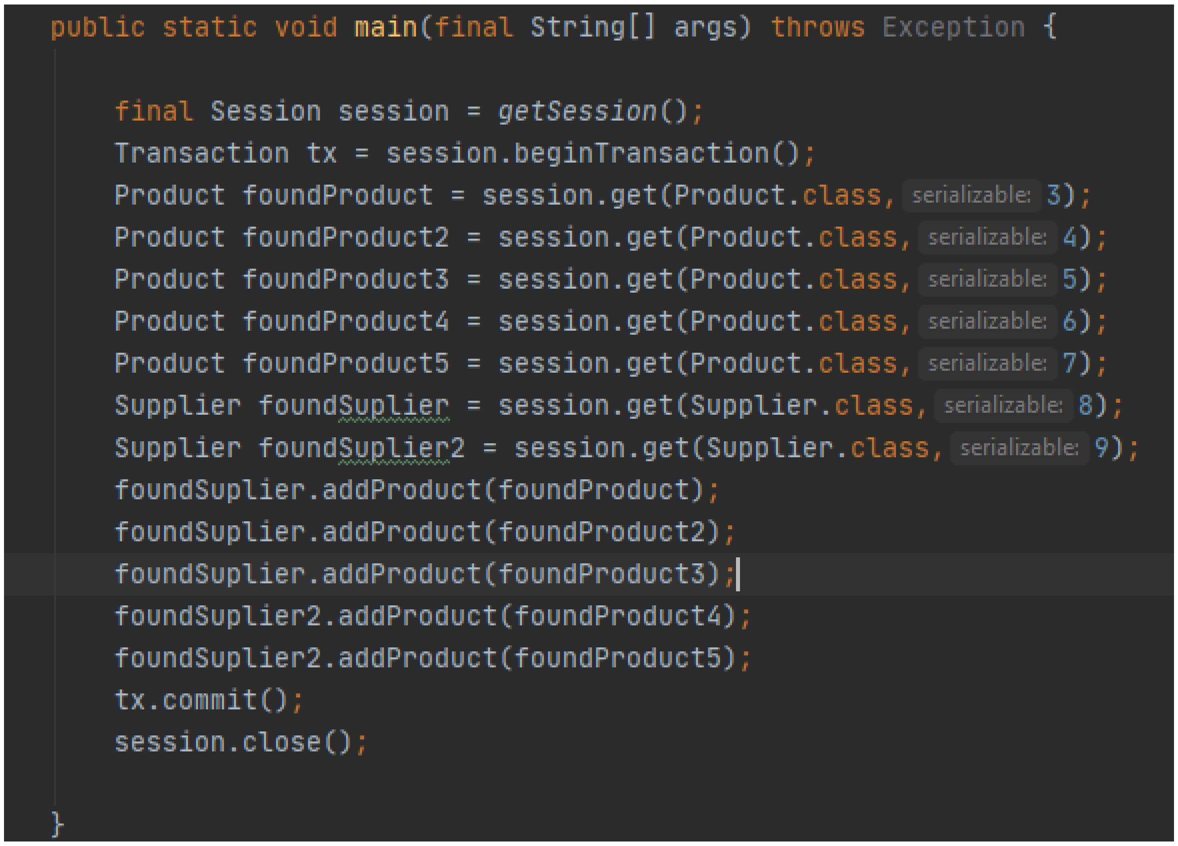
```java
import javax.annotation.processing.Generated;
import javax.persistence.*;
import java.util.Set;

@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> SuppliedProducts;

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


    public Supplier(){};

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

b. Stwórz kilka produktów

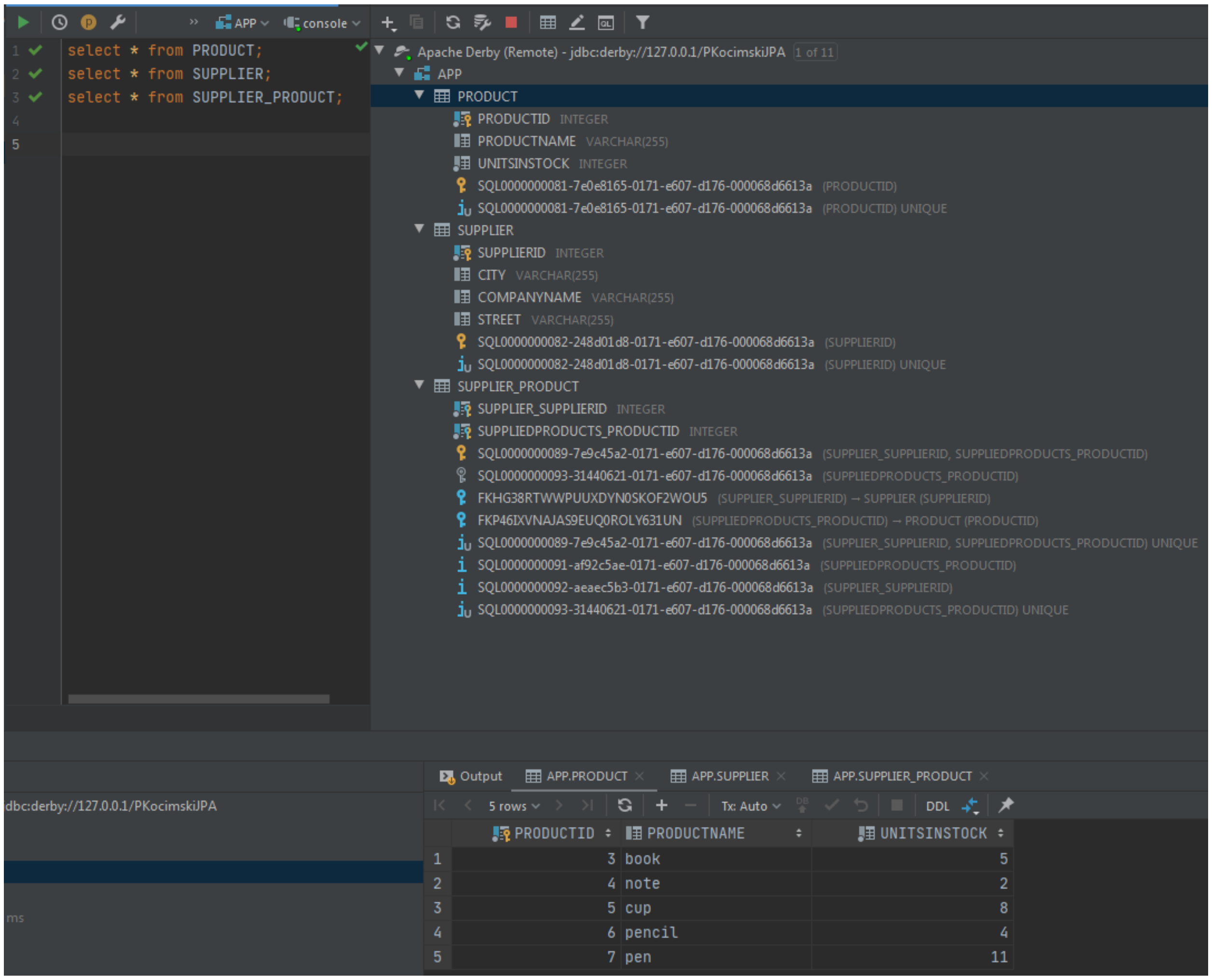


```java
public static void main(final String[] args) throws Exception {
    Product product = new Product( productName: "book",  unitsInStock: 5);
    Product product2 = new Product( productName: "note",  unitsInStock: 2);
    Product product3 = new Product( productName: "cup",  unitsInStock: 8);
    Product product4 = new Product( productName: "pencil",  unitsInStock: 4);
    Product product5 = new Product( productName: "pen",  unitsInStock: 11);

    Supplier supplier = new Supplier( companyName: "Google",  street: "Walkstreet",  city: "London");
    Supplier supplier2 = new Supplier( companyName: "Facebook",  street: "Warsaw",  city: "Piłsudskiego");
    final Session session = getSession();
    Transaction tx = session.beginTransaction();
     session.save(product);
     session.save(product2);
     session.save(product3);
     session.save(product4);
     session.save(product5);
     session.save(supplier);
     session.save(supplier2);
    tx.commit();
    session.close();

}
```

c)Dodaj je do produktów dostarczanych przez nowo stworzonego dostawcę



```java
public static void main(final String[] args) throws Exception {

    final Session session = getSession();
    Transaction tx = session.beginTransaction();
    Product foundProduct = session.get(Product.class, serializable: 3);
    Product foundProduct2 = session.get(Product.class, serializable: 4);
    Product foundProduct3 = session.get(Product.class, serializable: 5);
    Product foundProduct4 = session.get(Product.class, serializable: 6);
    Product foundProduct5 = session.get(Product.class, serializable: 7);
    Supplier foundSuplier = session.get(Supplier.class, serializable: 8);
    Supplier foundSuplier2 = session.get(Supplier.class, serializable: 9);
    foundSuplier.addProduct(foundProduct);
    foundSuplier.addProduct(foundProduct2);
    foundSuplier.addProduct(foundProduct3);
    foundSuplier2.addProduct(foundProduct4);
    foundSuplier2.addProduct(foundProduct5);
    tx.commit();
    session.close();

}
```



```sql
select * from PRODUCT;
select * from SUPPLIER;
select * from SUPPLIER_PRODUCT;
```

```
Apache Derby (Remote) - jdbc:derby://127.0.0.1/PKocimskiJPA  1 of 11
  APP
    PRODUCT
      PRODUCTID  INTEGER
      PRODUCTNAME  VARCHAR(255)
      UNITSINSTOCK  INTEGER
      SQL0000000081-7e0e8165-0171-e607-d176-000068d6613a  (PRODUCTID)
      SQL0000000081-7e0e8165-0171-e607-d176-000068d6613a  (PRODUCTID) UNIQUE
    SUPPLIER
      SUPPLIERID  INTEGER
      CITY  VARCHAR(255)
      COMPANYNAME  VARCHAR(255)
      STREET  VARCHAR(255)
      SQL0000000082-248d01d8-0171-e607-d176-000068d6613a  (SUPPLIERID)
      SQL0000000082-248d01d8-0171-e607-d176-000068d6613a  (SUPPLIERID) UNIQUE
    SUPPLIER_PRODUCT
      SUPPLIER_SUPPLIERID  INTEGER
      SUPPLIEDPRODUCTS_PRODUCTID  INTEGER
      SQL0000000089-7e9c45a2-0171-e607-d176-000068d6613a  (SUPPLIER_SUPPLIERID, SUPPLIEDPRODUCTS_PRODUCTID)
      SQL0000000093-31440621-0171-e607-d176-000068d6613a  (SUPPLIEDPRODUCTS_PRODUCTID)
      FKHG38RTWWPUUXDYN0SKOF2WOU5  (SUPPLIER_SUPPLIERID) → SUPPLIER (SUPPLIERID)
      FKP46IXVNAJAS9EUQ0ROLY631UN  (SUPPLIEDPRODUCTS_PRODUCTID) → PRODUCT (PRODUCTID)
      SQL0000000089-7e9c45a2-0171-e607-d176-000068d6613a  (SUPPLIER_SUPPLIERID, SUPPLIEDPRODUCTS_PRODUCTID) UNIQUE
      SQL0000000091-af92c5ae-0171-e607-d176-000068d6613a  (SUPPLIEDPRODUCTS_PRODUCTID)
      SQL0000000092-aeaec5b3-0171-e607-d176-000068d6613a  (SUPPLIER_SUPPLIERID)
      SQL0000000093-31440621-0171-e607-d176-000068d6613a  (SUPPLIEDPRODUCTS_PRODUCTID) UNIQUE
```

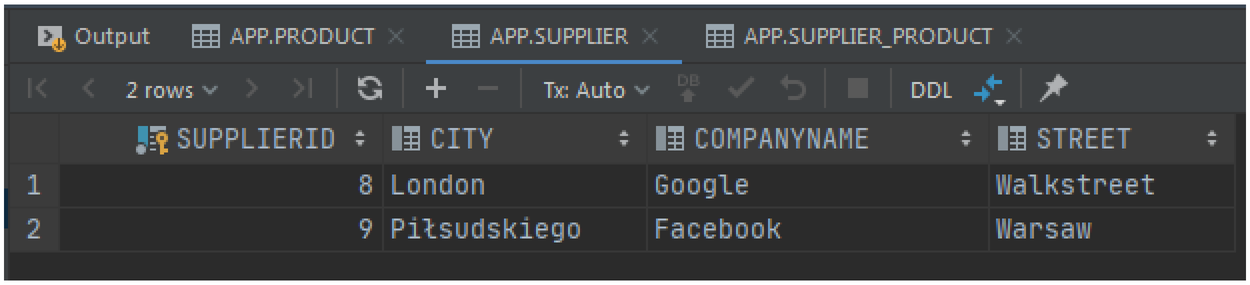
dbc:derby://127.0.0.1/PKocimskiJPA

ms

Output | APP.PRODUCT | APP.SUPPLIER | APP.SUPPLIER_PRODUCT

5 rows   Tx: Auto   DDL

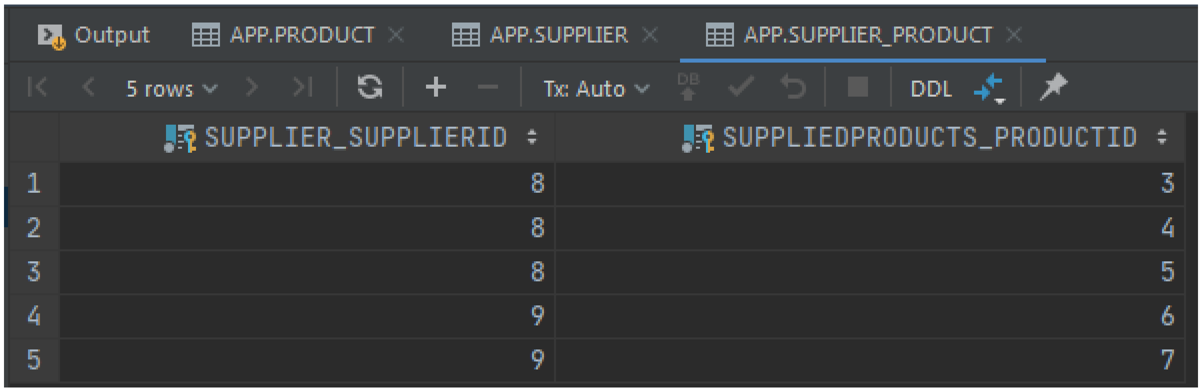
	PRODUCTID	PRODUCTNAME	UNITSINSTOCK
1	3	book	5
2	4	note	2
3	5	cup	8
4	6	pencil	4
5	7	pen	11



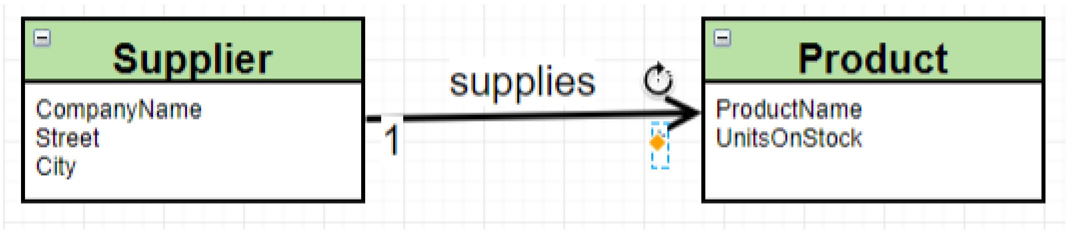
Output | APP.PRODUCT | APP.SUPPLIER | APP.SUPPLIER_PRODUCT

2 rows   Tx: Auto   DDL

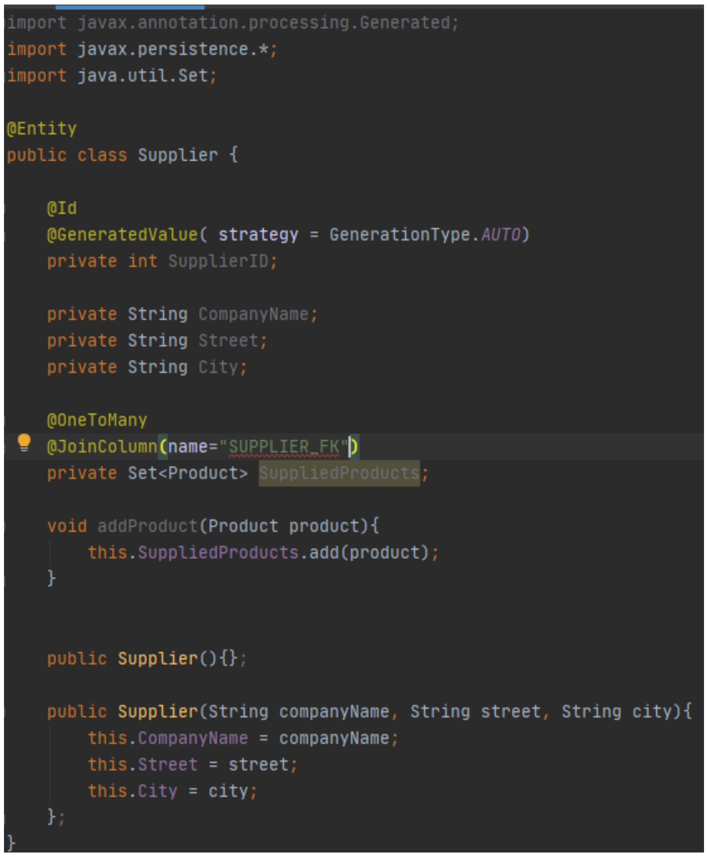
	SUPPLIERID	CITY	COMPANYNAME	STREET
1	8	London	Google	Walkstreet
2	9	Piłsudskiego	Facebook	Warsaw



	SUPPLIER_SUPPLIERID	SUPPLIEDPRODUCTS_PRODUCTID
1	8	3
2	8	4
3	8	5
4	9	6
5	9	7

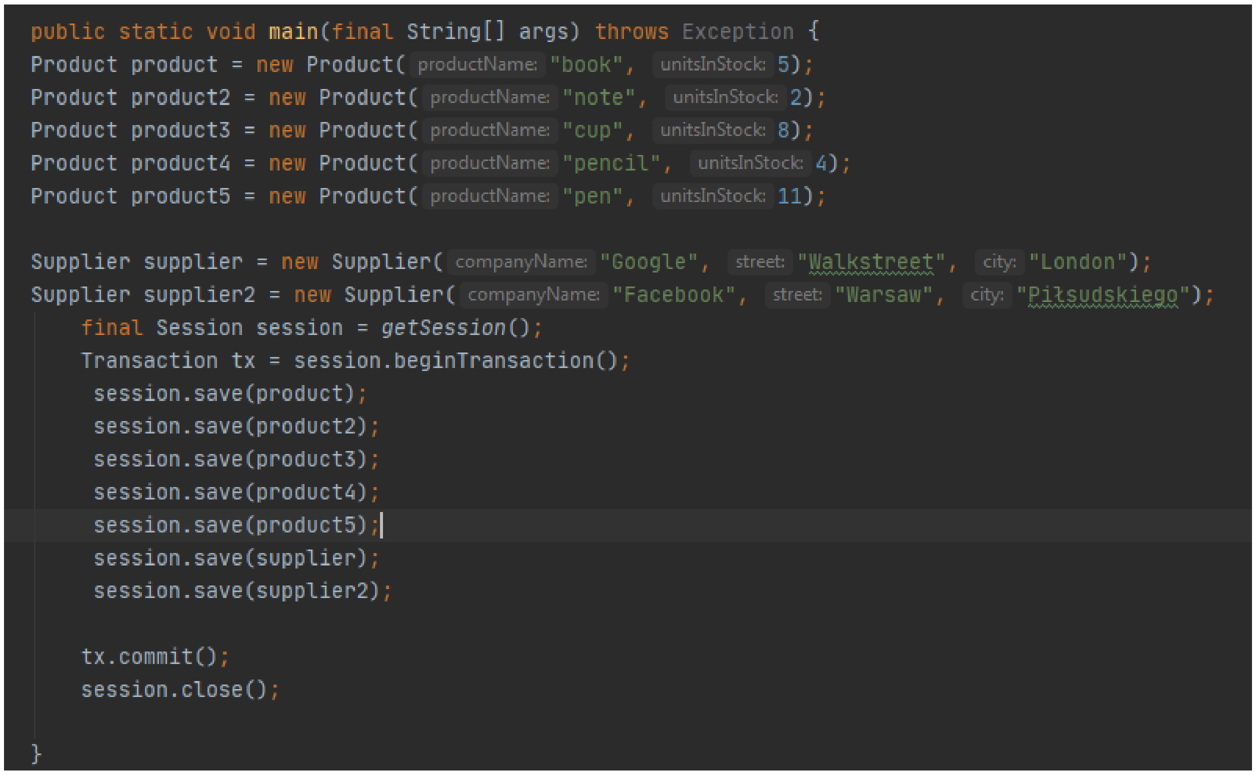


a. Zamodeluj powyższe „bez” tabeli łącznikowej



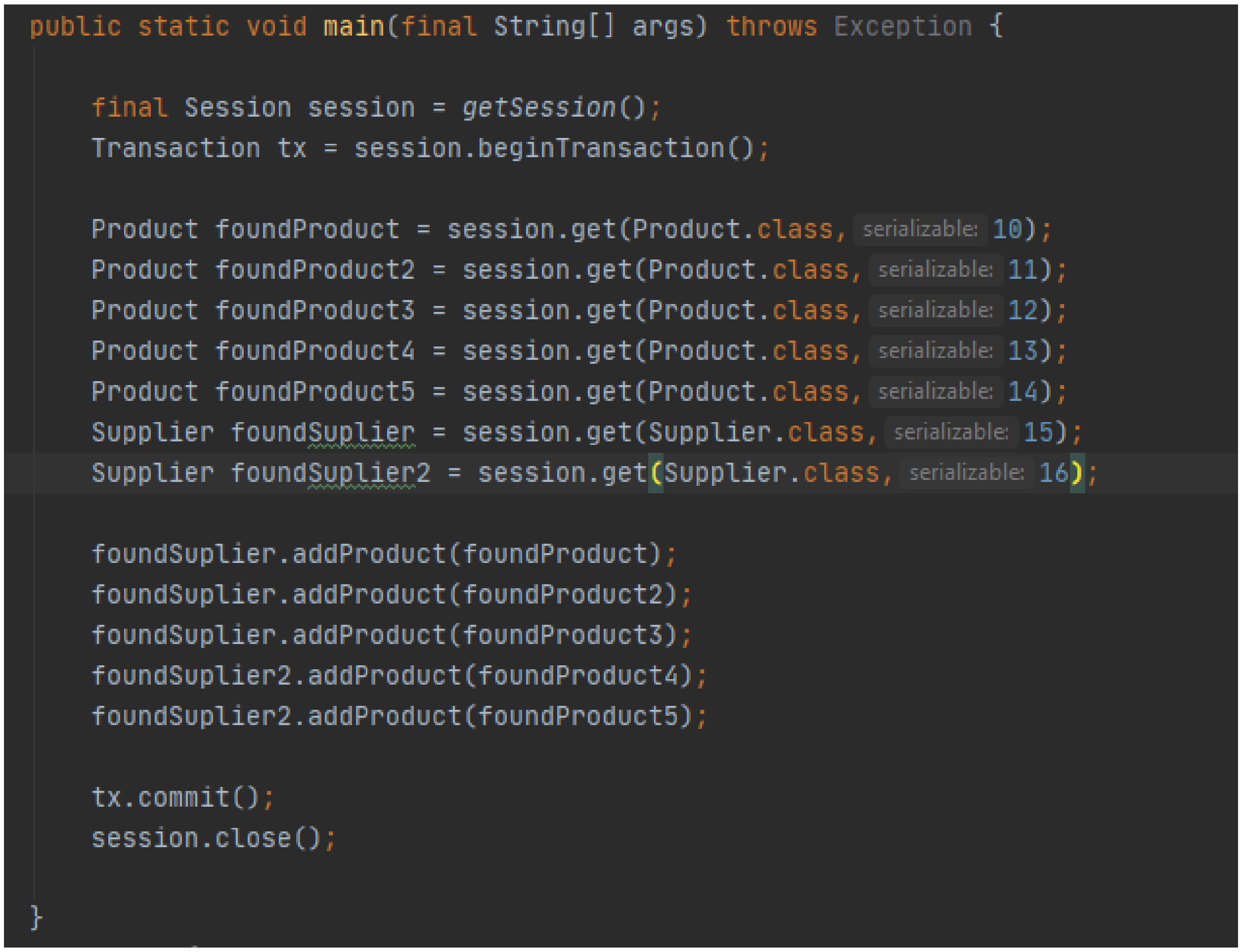
```
import javax.annotation.processing.Generated;
import javax.persistence.*;
import java.util.Set;

@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> SuppliedProducts;

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


    public Supplier(){};

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

b. Stwórz kilka produktów

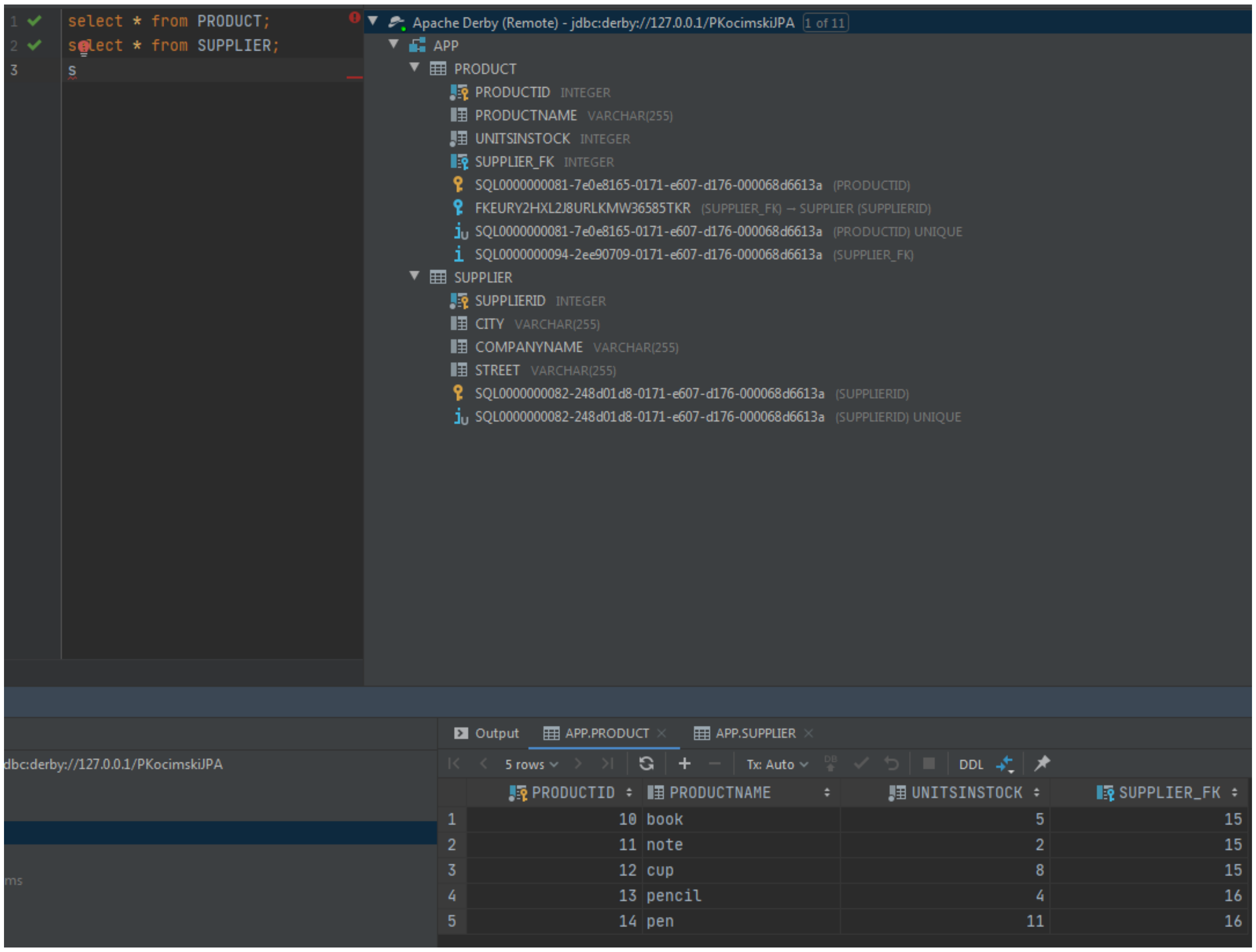


```java
public static void main(final String[] args) throws Exception {
Product product = new Product( productName: "book",  unitsInStock: 5);
Product product2 = new Product( productName: "note",  unitsInStock: 2);
Product product3 = new Product( productName: "cup",  unitsInStock: 8);
Product product4 = new Product( productName: "pencil",  unitsInStock: 4);
Product product5 = new Product( productName: "pen",  unitsInStock: 11);

Supplier supplier = new Supplier( companyName: "Google",  street: "Walkstreet",  city: "London");
Supplier supplier2 = new Supplier( companyName: "Facebook",  street: "Warsaw",  city: "Piłsudskiego");
    final Session session = getSession();
    Transaction tx = session.beginTransaction();
     session.save(product);
     session.save(product2);
     session.save(product3);
     session.save(product4);
     session.save(product5);
     session.save(supplier);
     session.save(supplier2);

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

}
```

c)Dodaj je do produktów dostarczanych przez nowo stworzonego dostawcę



```java
public static void main(final String[] args) throws Exception {

    final Session session = getSession();
    Transaction tx = session.beginTransaction();

    Product foundProduct = session.get(Product.class, serializable: 10);
    Product foundProduct2 = session.get(Product.class, serializable: 11);
    Product foundProduct3 = session.get(Product.class, serializable: 12);
    Product foundProduct4 = session.get(Product.class, serializable: 13);
    Product foundProduct5 = session.get(Product.class, serializable: 14);
    Supplier foundSuplier = session.get(Supplier.class, serializable: 15);
    Supplier foundSuplier2 = session.get(Supplier.class, serializable: 16);

    foundSuplier.addProduct(foundProduct);
    foundSuplier.addProduct(foundProduct2);
    foundSuplier.addProduct(foundProduct3);
    foundSuplier2.addProduct(foundProduct4);
    foundSuplier2.addProduct(foundProduct5);

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

}
```

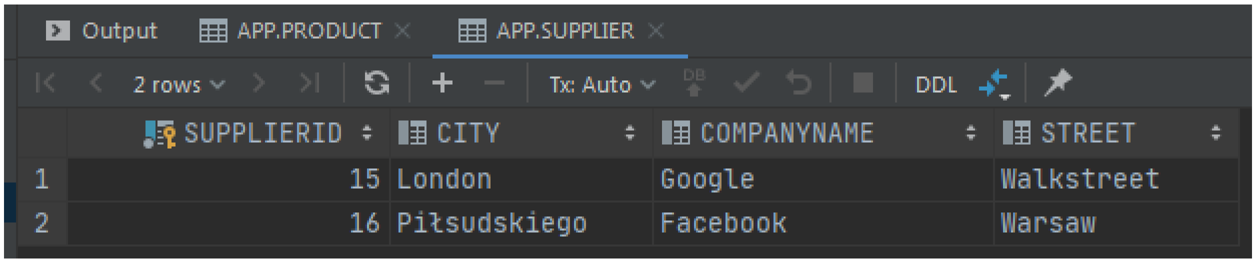


```
select * from PRODUCT;
select * from SUPPLIER;
s
```

```
Apache Derby (Remote) - jdbc:derby://127.0.0.1/PKocimskiJPA  1 of 11
  APP
    PRODUCT
      PRODUCTID  INTEGER
      PRODUCTNAME  VARCHAR(255)
      UNITSINSTOCK  INTEGER
      SUPPLIER_FK  INTEGER
      SQL0000000081-7e0e8165-0171-e607-d176-000068d6613a  (PRODUCTID)
      FKEURY2HXL2J8URLKMW36585TKR  (SUPPLIER_FK) → SUPPLIER (SUPPLIERID)
      SQL0000000081-7e0e8165-0171-e607-d176-000068d6613a  (PRODUCTID) UNIQUE
      SQL0000000094-2ee90709-0171-e607-d176-000068d6613a  (SUPPLIER_FK)
    SUPPLIER
      SUPPLIERID  INTEGER
      CITY  VARCHAR(255)
      COMPANYNAME  VARCHAR(255)
      STREET  VARCHAR(255)
      SQL0000000082-248d01d8-0171-e607-d176-000068d6613a  (SUPPLIERID)
      SQL0000000082-248d01d8-0171-e607-d176-000068d6613a  (SUPPLIERID) UNIQUE
```

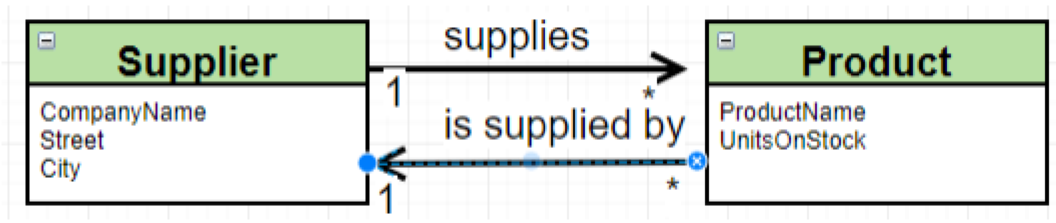
dbc:derby://127.0.0.1/PKocimskiJPA

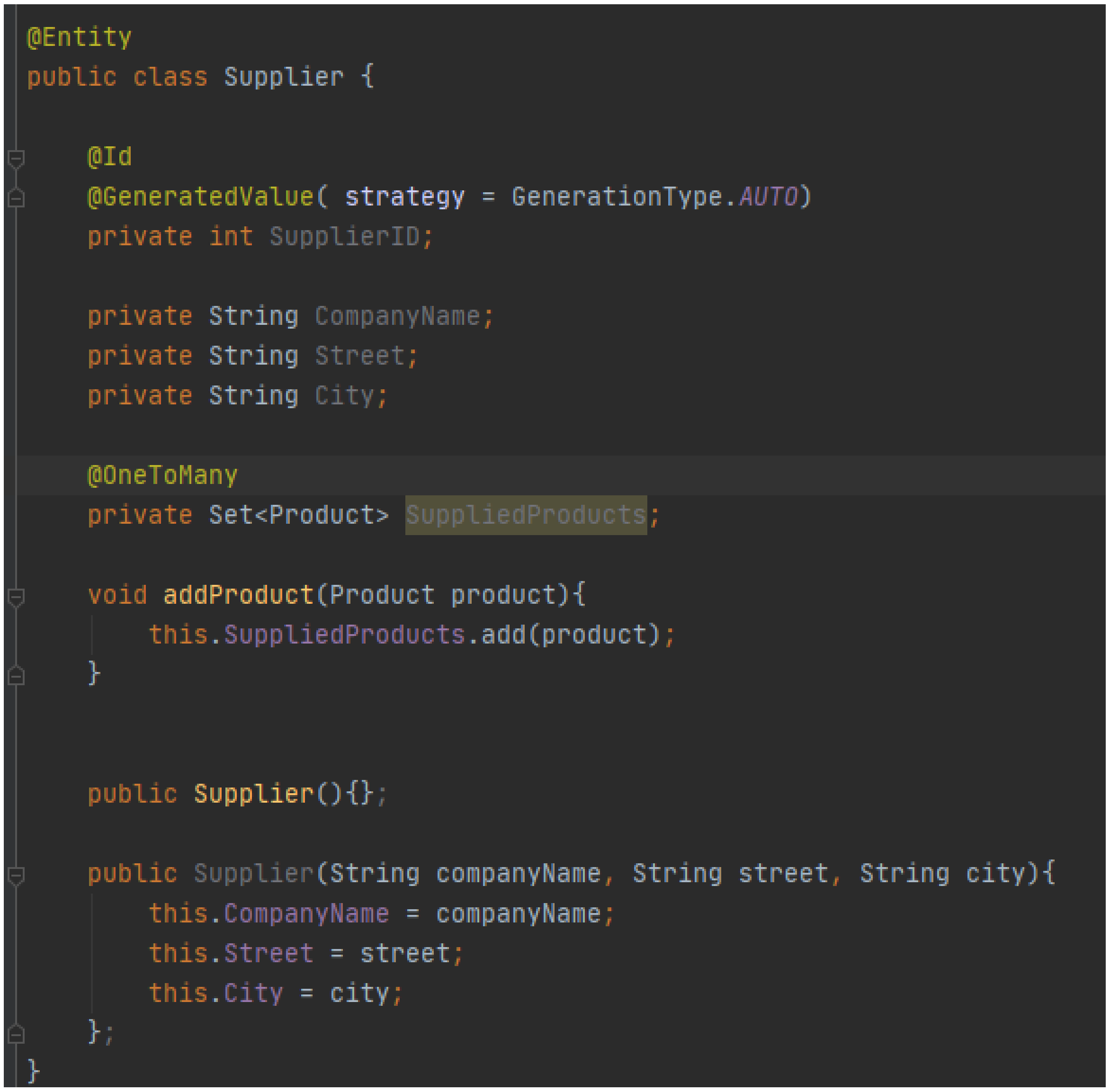
	PRODUCTID	PRODUCTNAME	UNITSINSTOCK	SUPPLIER_FK
1	10	book	5	15
2	11	note	2	15
3	12	cup	8	15
4	13	pencil	4	16
5	14	pen	11	16



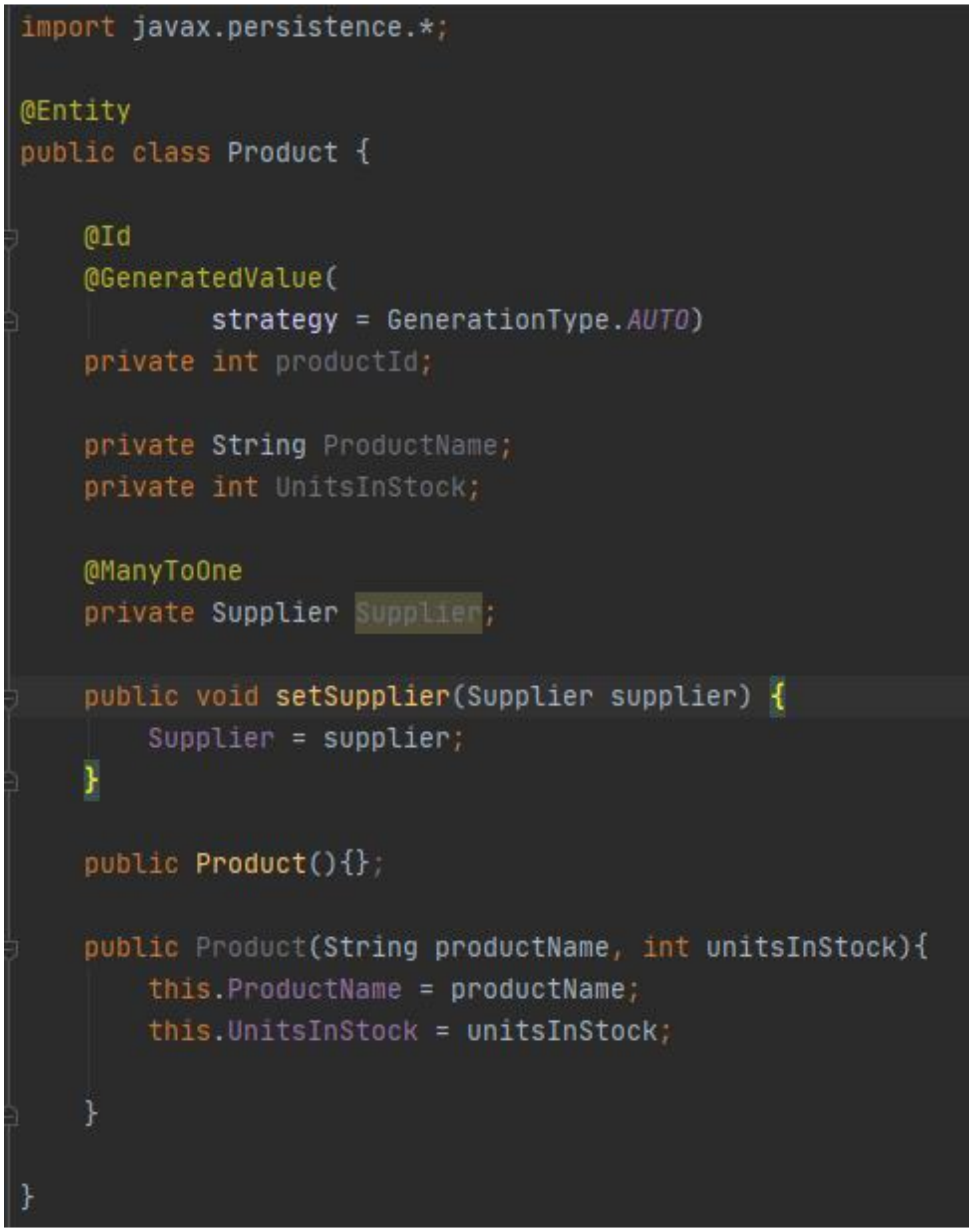
	SUPPLIERID	CITY	COMPANYNAME	STREET
1	15	London	Google	Walkstreet
2	16	Piłsudskiego	Facebook	Warsaw

VI. Zamodeluj relacje dwustronną jak poniżej:



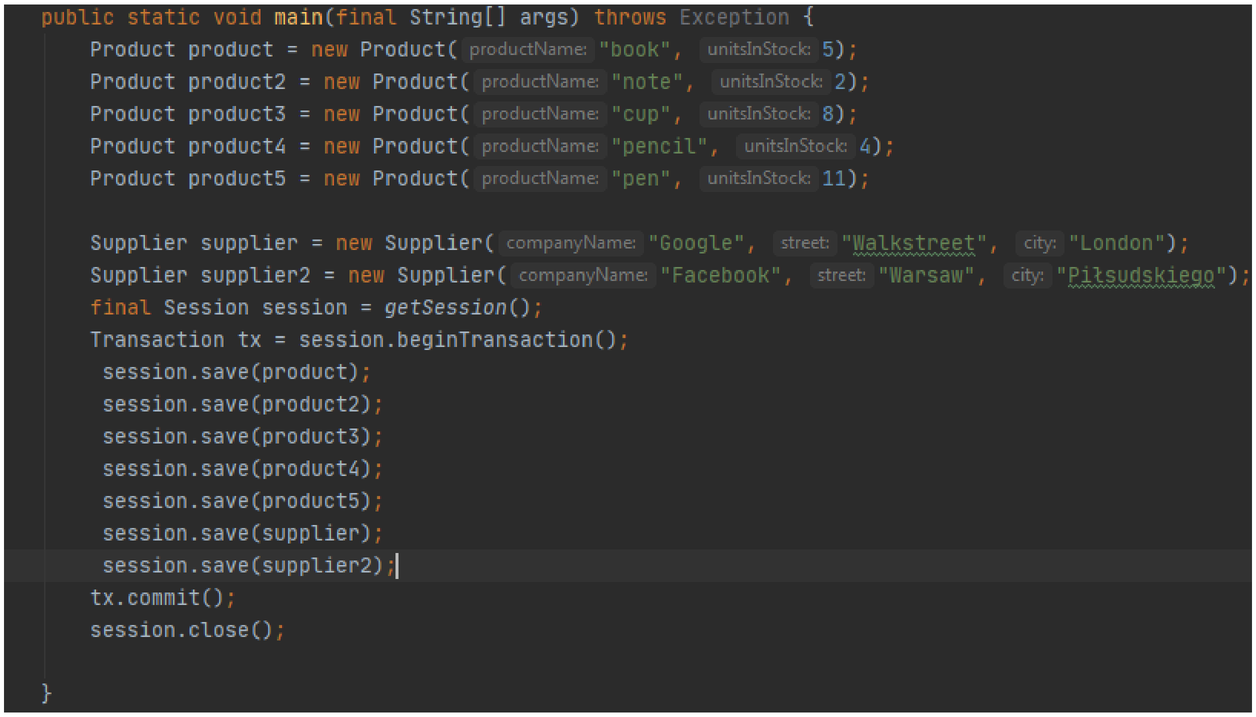


```java
@Entity
public class Supplier {

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

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

    @OneToMany
    private Set<Product> SuppliedProducts;

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


    public Supplier(){};

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



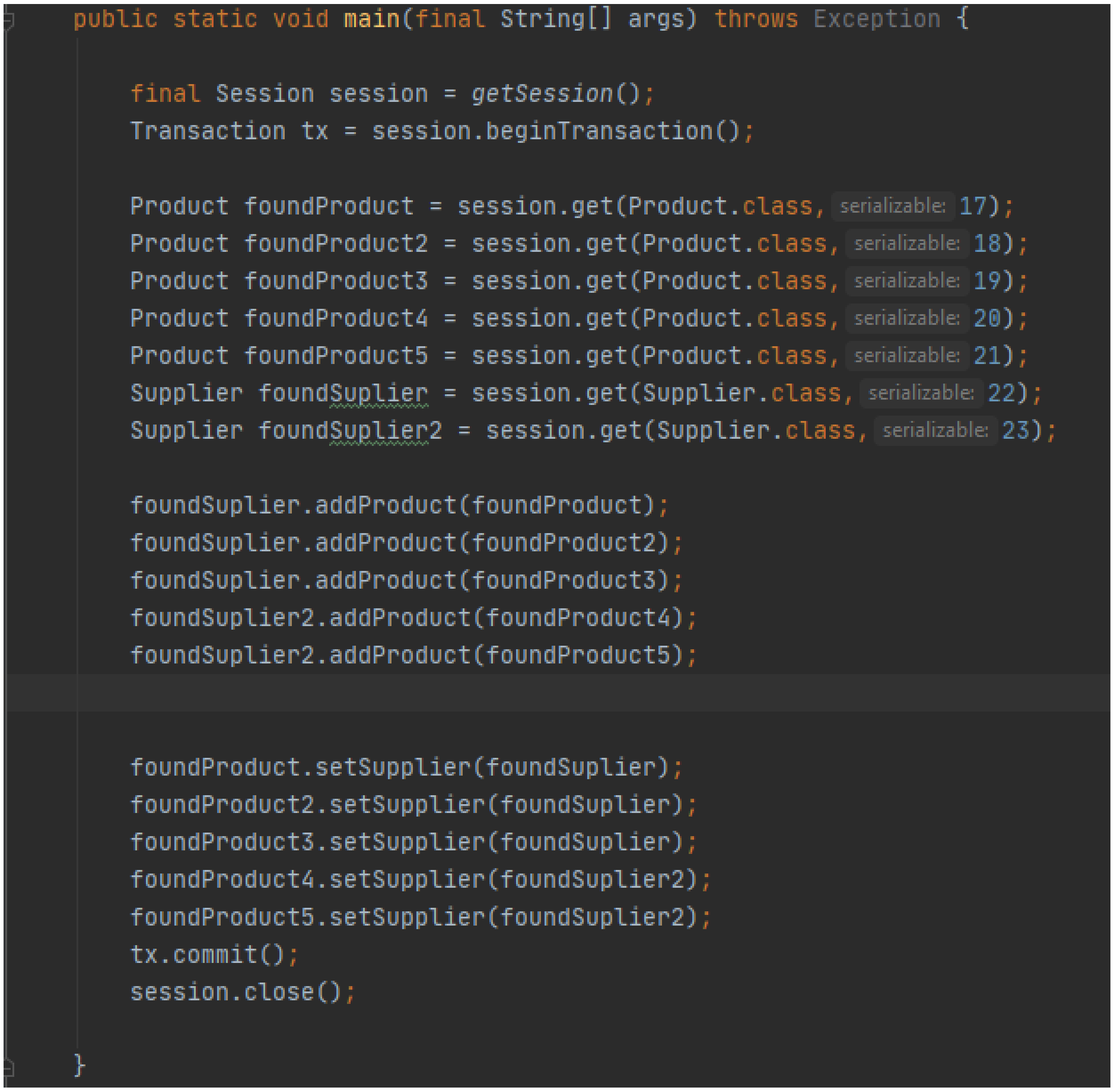
```java
import javax.persistence.*;

@Entity
public class Product {

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

    private String ProductName;
    private int UnitsInStock;

    @ManyToOne
    private Supplier Supplier;

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

    public Product(){};

    public Product(String productName, int unitsInStock){
        this.ProductName = productName;
        this.UnitsInStock = unitsInStock;

    }

}
```

a. Tradycyjnie: Stwórz kilka produktów

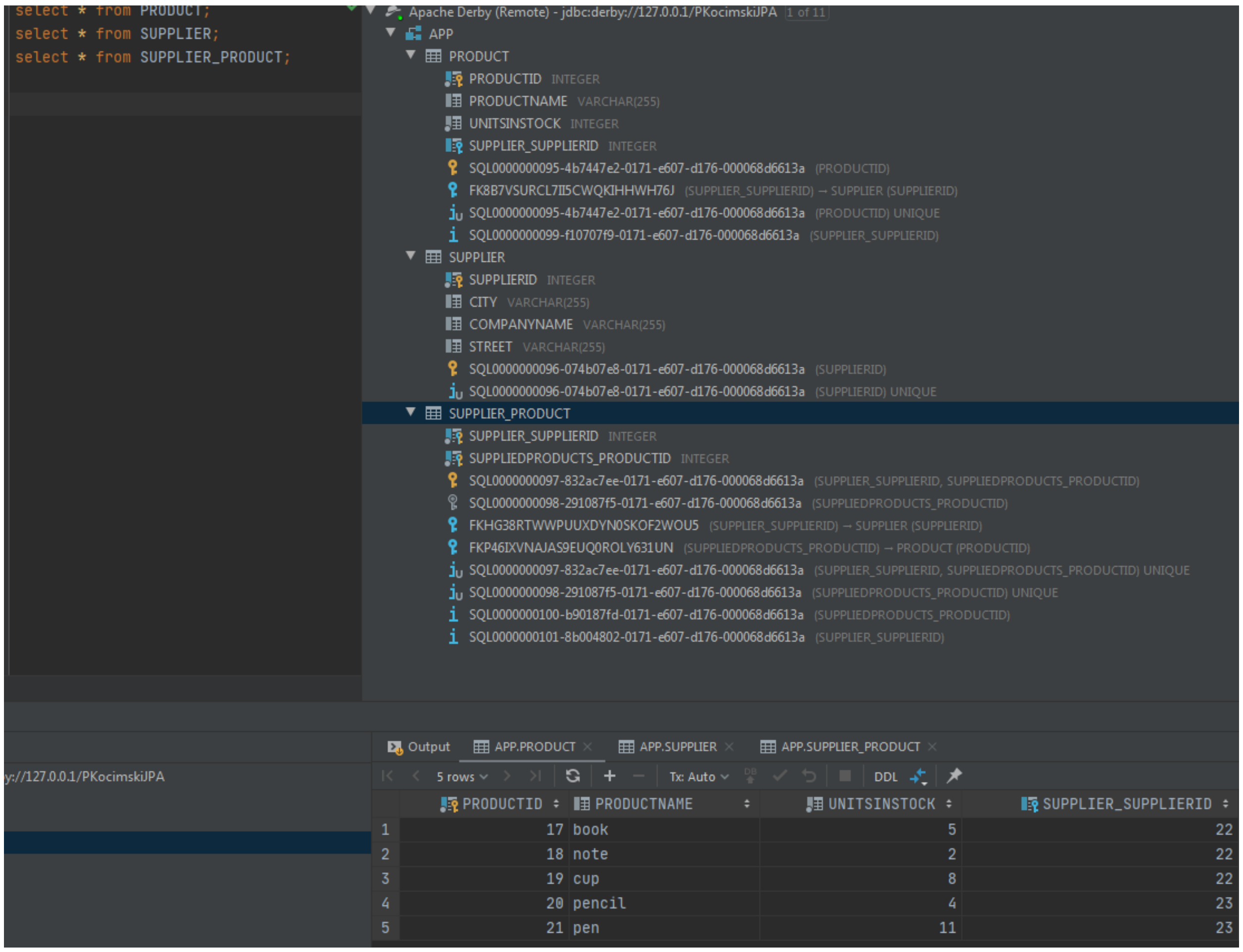


```java
public static void main(final String[] args) throws Exception {
    Product product = new Product( productName: "book",  unitsInStock: 5);
    Product product2 = new Product( productName: "note",  unitsInStock: 2);
    Product product3 = new Product( productName: "cup",  unitsInStock: 8);
    Product product4 = new Product( productName: "pencil",  unitsInStock: 4);
    Product product5 = new Product( productName: "pen",  unitsInStock: 11);

    Supplier supplier = new Supplier( companyName: "Google",  street: "Walkstreet",  city: "London");
    Supplier supplier2 = new Supplier( companyName: "Facebook",  street: "Warsaw",  city: "Piłsudskiego");
    final Session session = getSession();
    Transaction tx = session.beginTransaction();
     session.save(product);
     session.save(product2);
     session.save(product3);
     session.save(product4);
     session.save(product5);
     session.save(supplier);
     session.save(supplier2);
    tx.commit();
    session.close();

}
```

b. Dodaj je do produktów dostarczanych przez nowo stworzonego dostawcę (pamiętaj o poprawnej obsłudze dwustronności relacji)



```
public static void main(final String[] args) throws Exception {

    final Session session = getSession();
    Transaction tx = session.beginTransaction();

    Product foundProduct = session.get(Product.class, serializable: 17);
    Product foundProduct2 = session.get(Product.class, serializable: 18);
    Product foundProduct3 = session.get(Product.class, serializable: 19);
    Product foundProduct4 = session.get(Product.class, serializable: 20);
    Product foundProduct5 = session.get(Product.class, serializable: 21);
    Supplier foundSuplier = session.get(Supplier.class, serializable: 22);
    Supplier foundSuplier2 = session.get(Supplier.class, serializable: 23);

    foundSuplier.addProduct(foundProduct);
    foundSuplier.addProduct(foundProduct2);
    foundSuplier.addProduct(foundProduct3);
    foundSuplier2.addProduct(foundProduct4);
    foundSuplier2.addProduct(foundProduct5);


    foundProduct.setSupplier(foundSuplier);
    foundProduct2.setSupplier(foundSuplier);
    foundProduct3.setSupplier(foundSuplier);
    foundProduct4.setSupplier(foundSuplier2);
    foundProduct5.setSupplier(foundSuplier2);
    tx.commit();
    session.close();

}
```

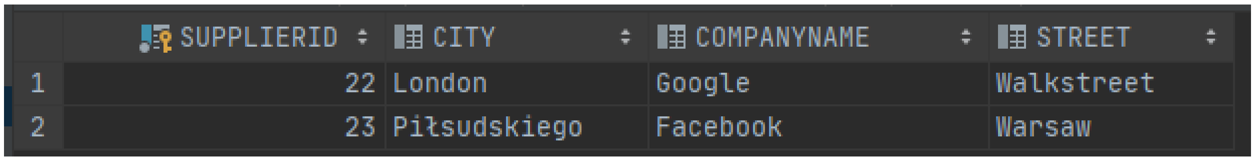


```
select * from PRODUCT;
select * from SUPPLIER;
select * from SUPPLIER_PRODUCT;
```

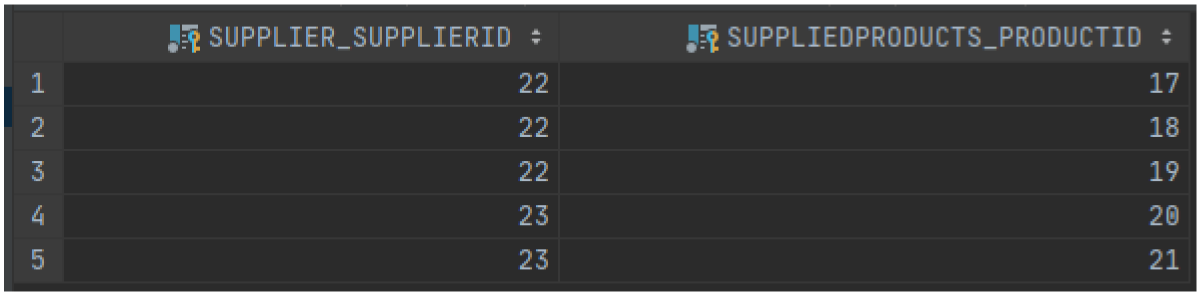
```
Apache Derby (Remote) - jdbc:derby://127.0.0.1/PKocimskiJPA  1 of 11
▼ APP
  ▼ PRODUCT
      PRODUCTID  INTEGER
      PRODUCTNAME  VARCHAR(255)
      UNITSINSTOCK  INTEGER
      SUPPLIER_SUPPLIERID  INTEGER
      SQL0000000095-4b7447e2-0171-e607-d176-000068d6613a  (PRODUCTID)
      FK8B7VSURCL7II5CWQKIHHWH76J  (SUPPLIER_SUPPLIERID) → SUPPLIER (SUPPLIERID)
      SQL0000000095-4b7447e2-0171-e607-d176-000068d6613a  (PRODUCTID) UNIQUE
      SQL0000000099-f10707f9-0171-e607-d176-000068d6613a  (SUPPLIER_SUPPLIERID)
  ▼ SUPPLIER
      SUPPLIERID  INTEGER
      CITY  VARCHAR(255)
      COMPANYNAME  VARCHAR(255)
      STREET  VARCHAR(255)
      SQL0000000096-074b07e8-0171-e607-d176-000068d6613a  (SUPPLIERID)
      SQL0000000096-074b07e8-0171-e607-d176-000068d6613a  (SUPPLIERID) UNIQUE
  ▼ SUPPLIER_PRODUCT
      SUPPLIER_SUPPLIERID  INTEGER
      SUPPLIEDPRODUCTS_PRODUCTID  INTEGER
      SQL0000000097-832ac7ee-0171-e607-d176-000068d6613a  (SUPPLIER_SUPPLIERID, SUPPLIEDPRODUCTS_PRODUCTID)
      SQL0000000098-291087f5-0171-e607-d176-000068d6613a  (SUPPLIEDPRODUCTS_PRODUCTID)
      FKHG38RTWWPUUXDYN0SKOF2WOU5  (SUPPLIER_SUPPLIERID) → SUPPLIER (SUPPLIERID)
      FKP46IXVNAJAS9EUQ0ROLY631UN  (SUPPLIEDPRODUCTS_PRODUCTID) → PRODUCT (PRODUCTID)
      SQL0000000097-832ac7ee-0171-e607-d176-000068d6613a  (SUPPLIER_SUPPLIERID, SUPPLIEDPRODUCTS_PRODUCTID) UNIQUE
      SQL0000000098-291087f5-0171-e607-d176-000068d6613a  (SUPPLIEDPRODUCTS_PRODUCTID) UNIQUE
      SQL0000000100-b90187fd-0171-e607-d176-000068d6613a  (SUPPLIEDPRODUCTS_PRODUCTID)
      SQL0000000101-8b004802-0171-e607-d176-000068d6613a  (SUPPLIER_SUPPLIERID)
```

y://127.0.0.1/PKocimskiJPA

	PRODUCTID	PRODUCTNAME	UNITSINSTOCK	SUPPLIER_SUPPLIERID
1	17	book	5	22
2	18	note	2	22
3	19	cup	8	22
4	20	pencil	4	23
5	21	pen	11	23

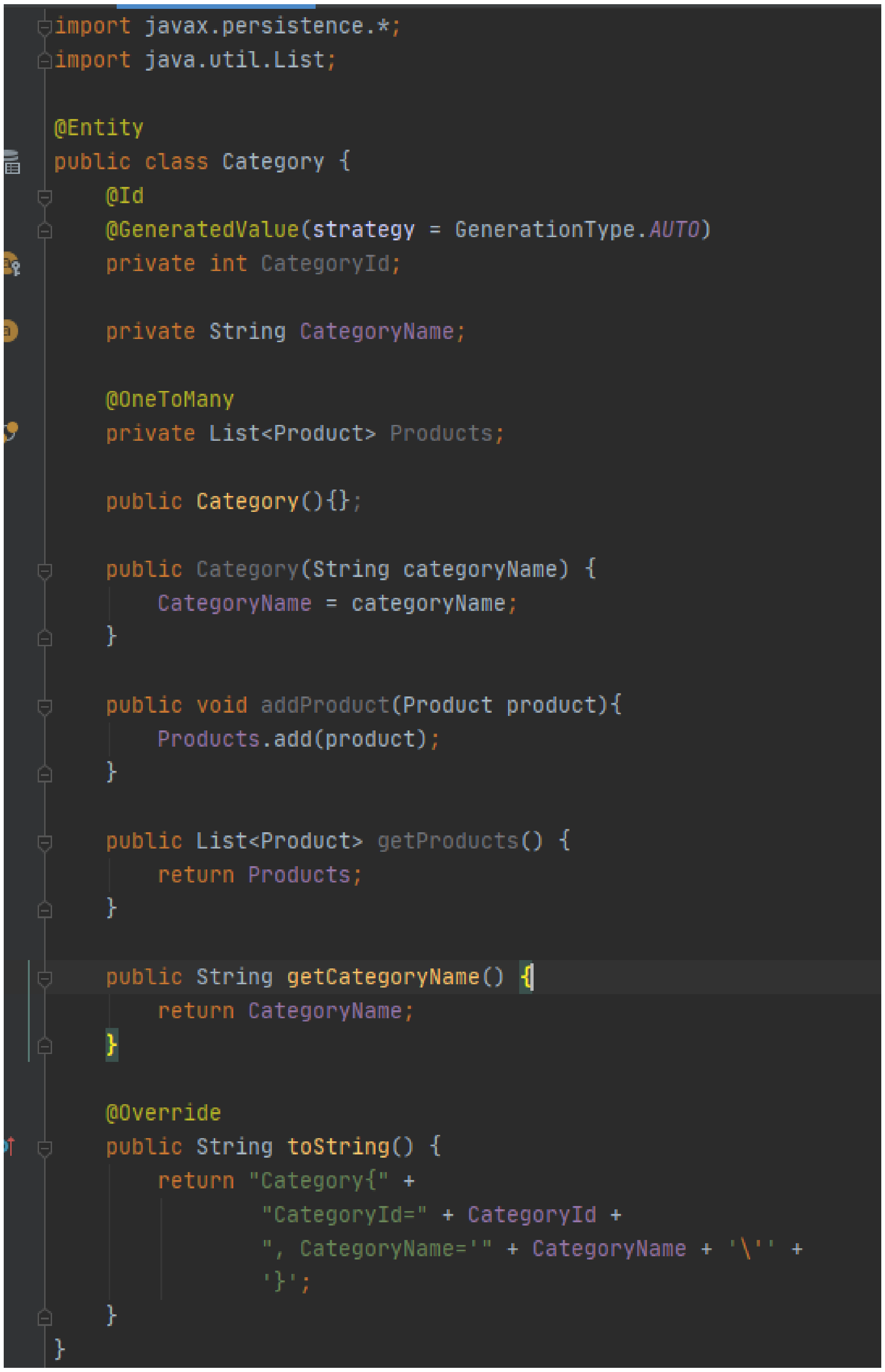


	SUPPLIERID	CITY	COMPANYNAME	STREET
1	22	London	Google	Walkstreet
2	23	Piłsudskiego	Facebook	Warsaw



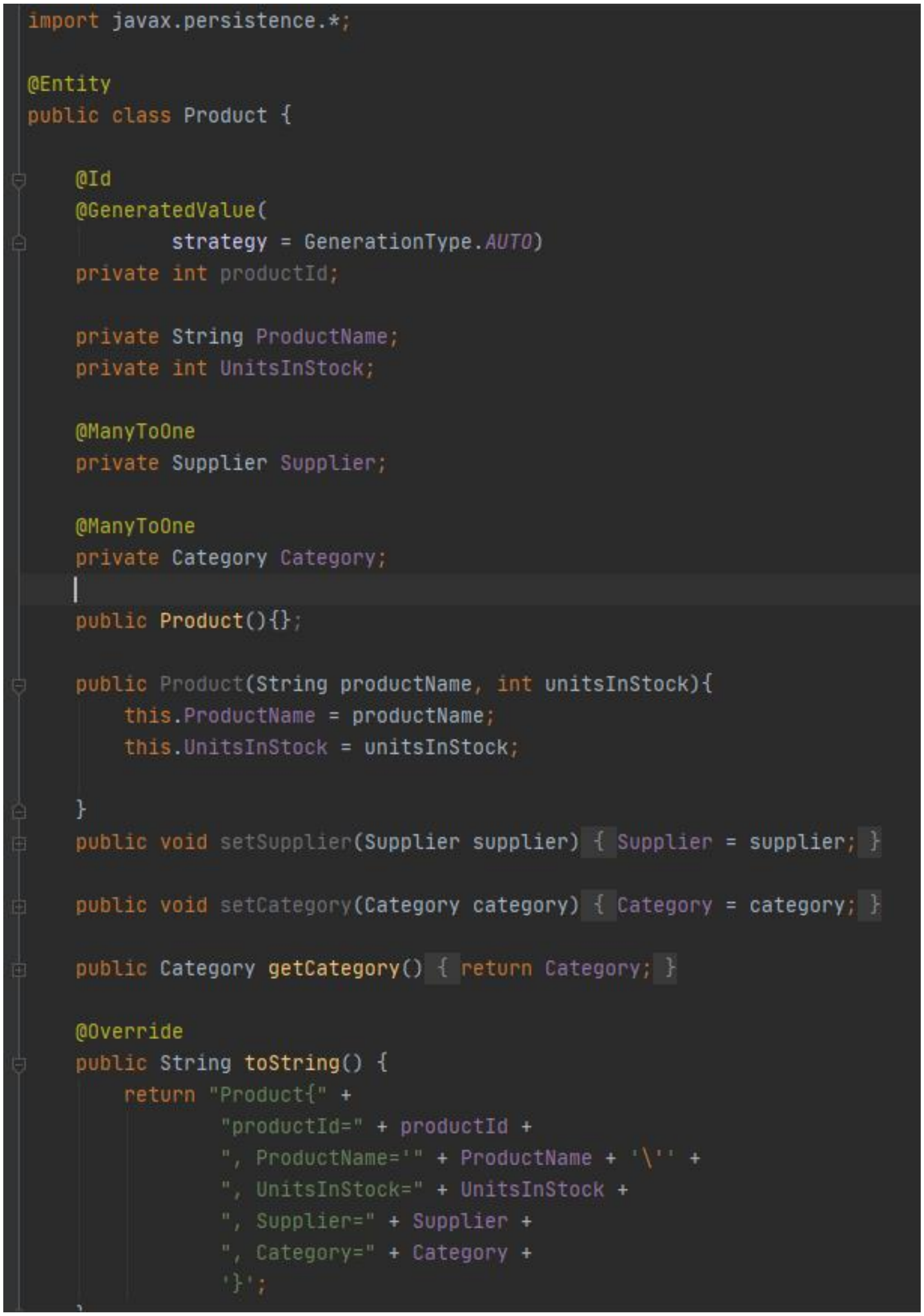
	SUPPLIER_SUPPLIERID	SUPPLIEDPRODUCTS_PRODUCTID
1	22	17
2	22	18
3	22	19
4	23	20
5	23	21

VII. Dodaj klase Category z property int CategoryID, String Name oraz listą produktow List Products



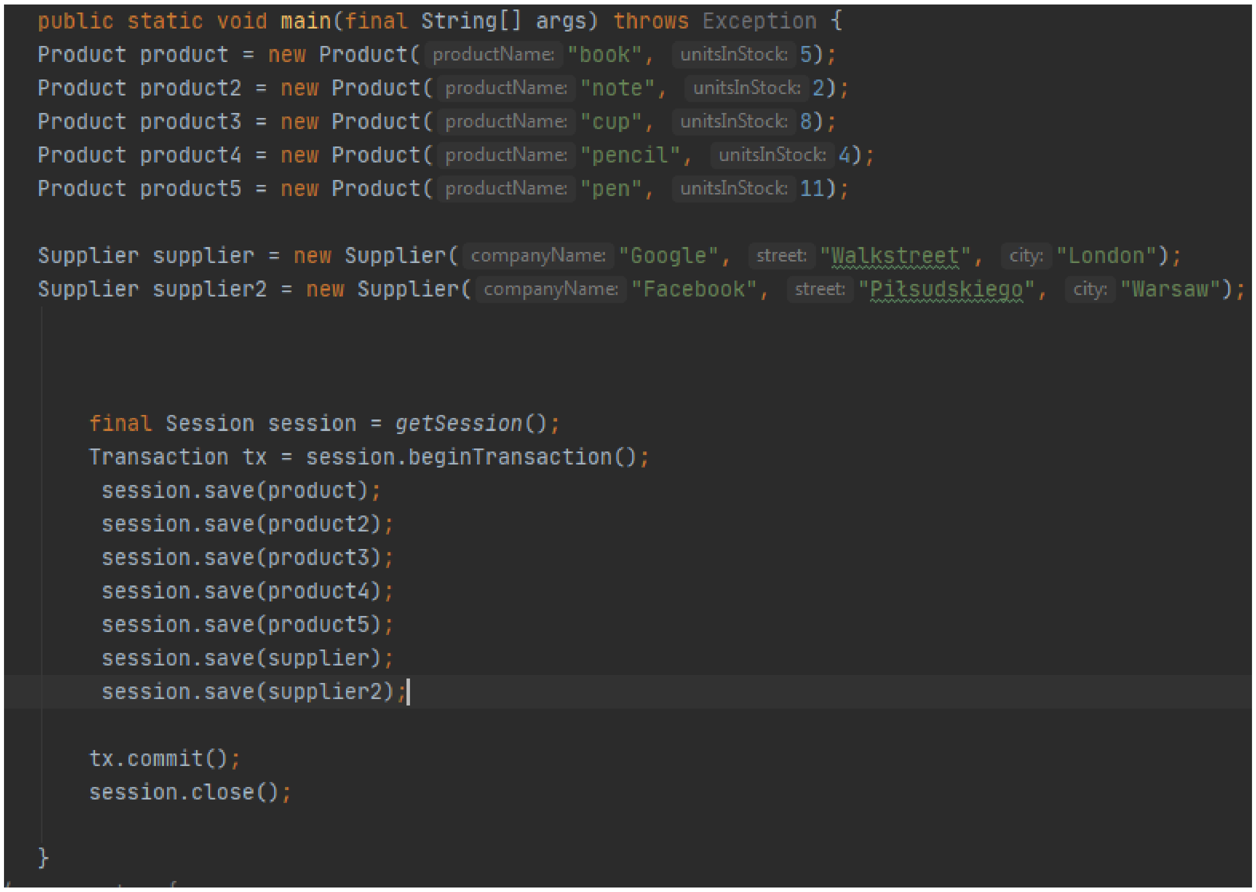
```java
import javax.persistence.*;
import java.util.List;

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

    private String CategoryName;

    @OneToMany
    private List<Product> Products;

    public Category(){};

    public Category(String categoryName) {
        CategoryName = categoryName;
    }

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

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

    public String getCategoryName() {
        return CategoryName;
    }

    @Override
    public String toString() {
        return "Category{" +
                "CategoryId=" + CategoryId +
                ", CategoryName='" + CategoryName + '\'' +
                '}';
    }
}
```

a. Zmodyfikuj produkty dodając wskazanie na kategorie do której należy.

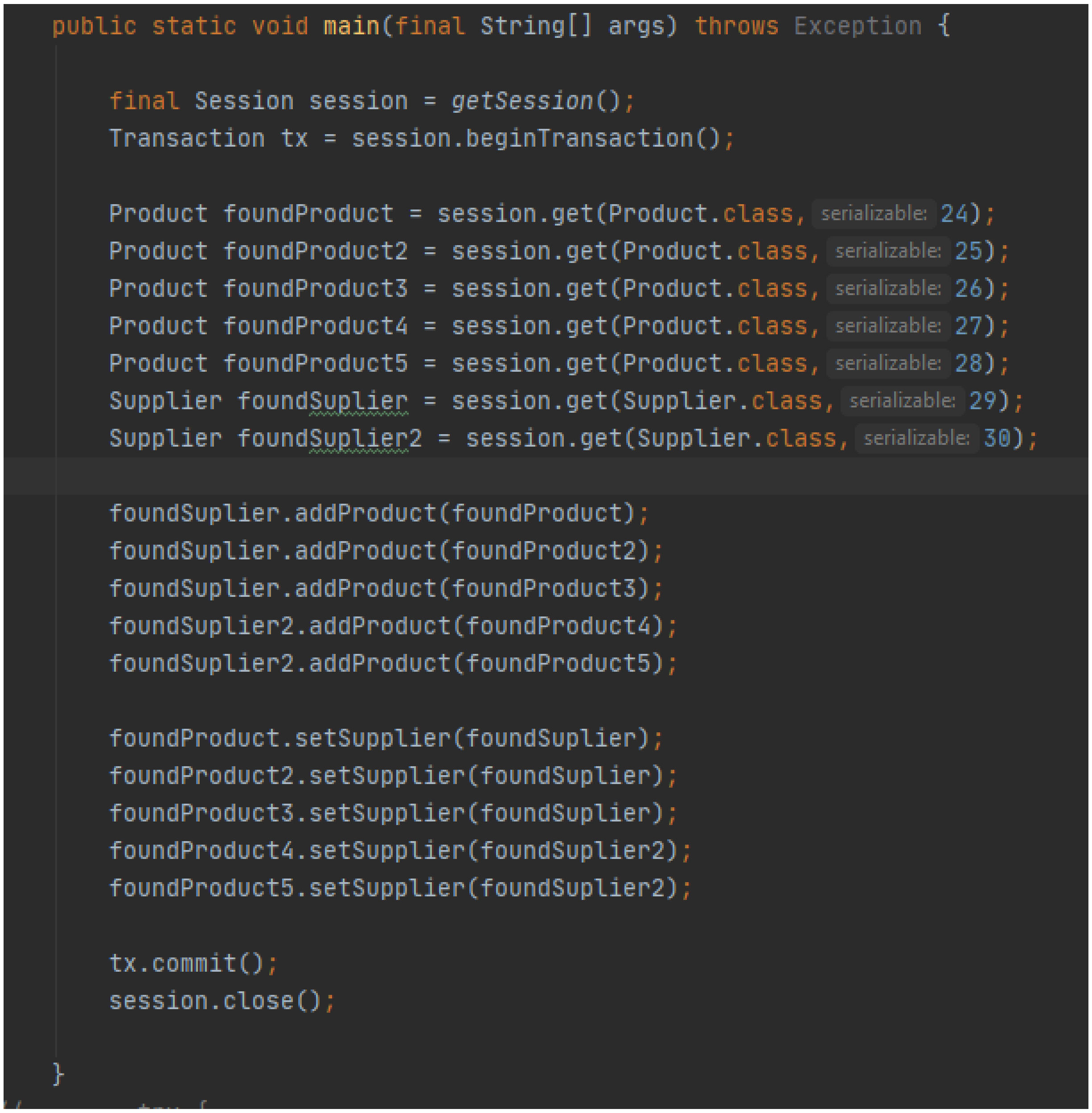


```java
import javax.persistence.*;

@Entity
public class Product {

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

    private String ProductName;
    private int UnitsInStock;

    @ManyToOne
    private Supplier Supplier;

    @ManyToOne
    private Category Category;

    public Product(){};

    public Product(String productName, int unitsInStock){
        this.ProductName = productName;
        this.UnitsInStock = unitsInStock;

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

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

    public Category getCategory() { return Category; }

    @Override
    public String toString() {
        return "Product{" +
                "productId=" + productId +
                ", ProductName='" + ProductName + '\'' +
                ", UnitsInStock=" + UnitsInStock +
                ", Supplier=" + Supplier +
                ", Category=" + Category +
                '}';
```

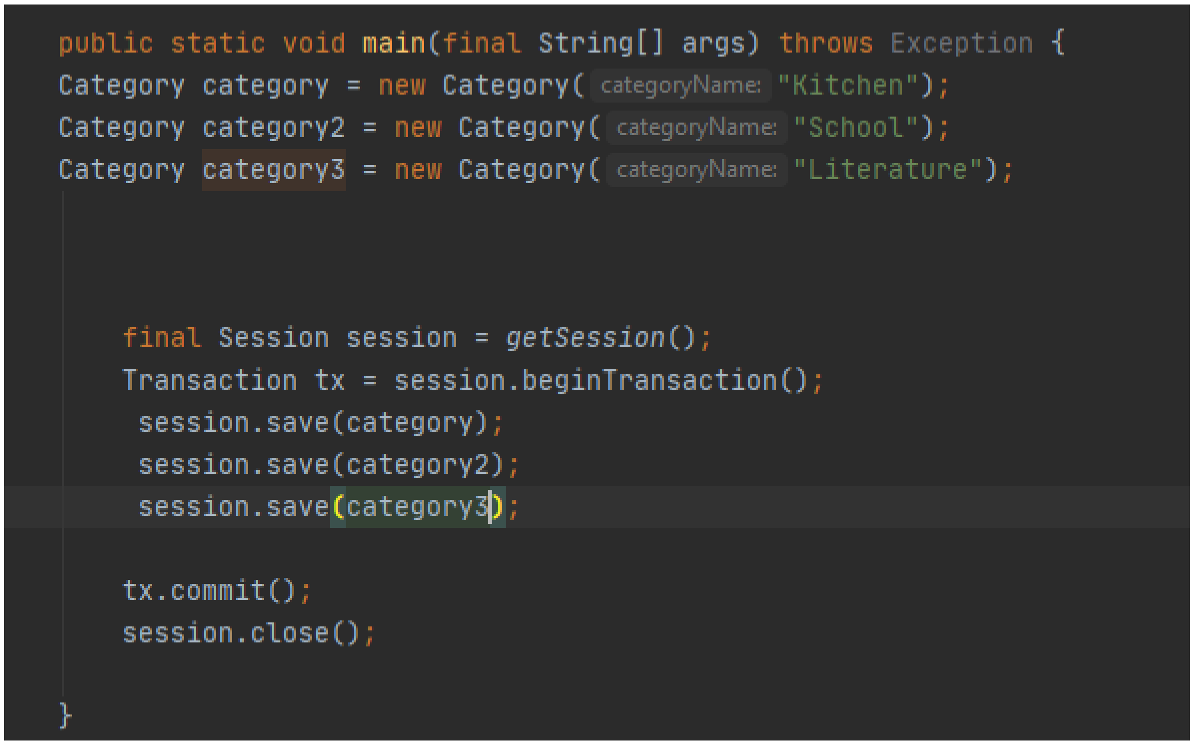
b. Stwórz kilka produktów i kilka kategorii



```
public static void main(final String[] args) throws Exception {
Product product = new Product( productName: "book",  unitsInStock: 5);
Product product2 = new Product( productName: "note",  unitsInStock: 2);
Product product3 = new Product( productName: "cup",  unitsInStock: 8);
Product product4 = new Product( productName: "pencil",  unitsInStock: 4);
Product product5 = new Product( productName: "pen",  unitsInStock: 11);

Supplier supplier = new Supplier( companyName: "Google",  street: "Walkstreet",  city: "London");
Supplier supplier2 = new Supplier( companyName: "Facebook",  street: "Piłsudskiego",  city: "Warsaw");



    final Session session = getSession();
    Transaction tx = session.beginTransaction();
     session.save(product);
     session.save(product2);
     session.save(product3);
     session.save(product4);
     session.save(product5);
     session.save(supplier);
     session.save(supplier2);

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

}
```

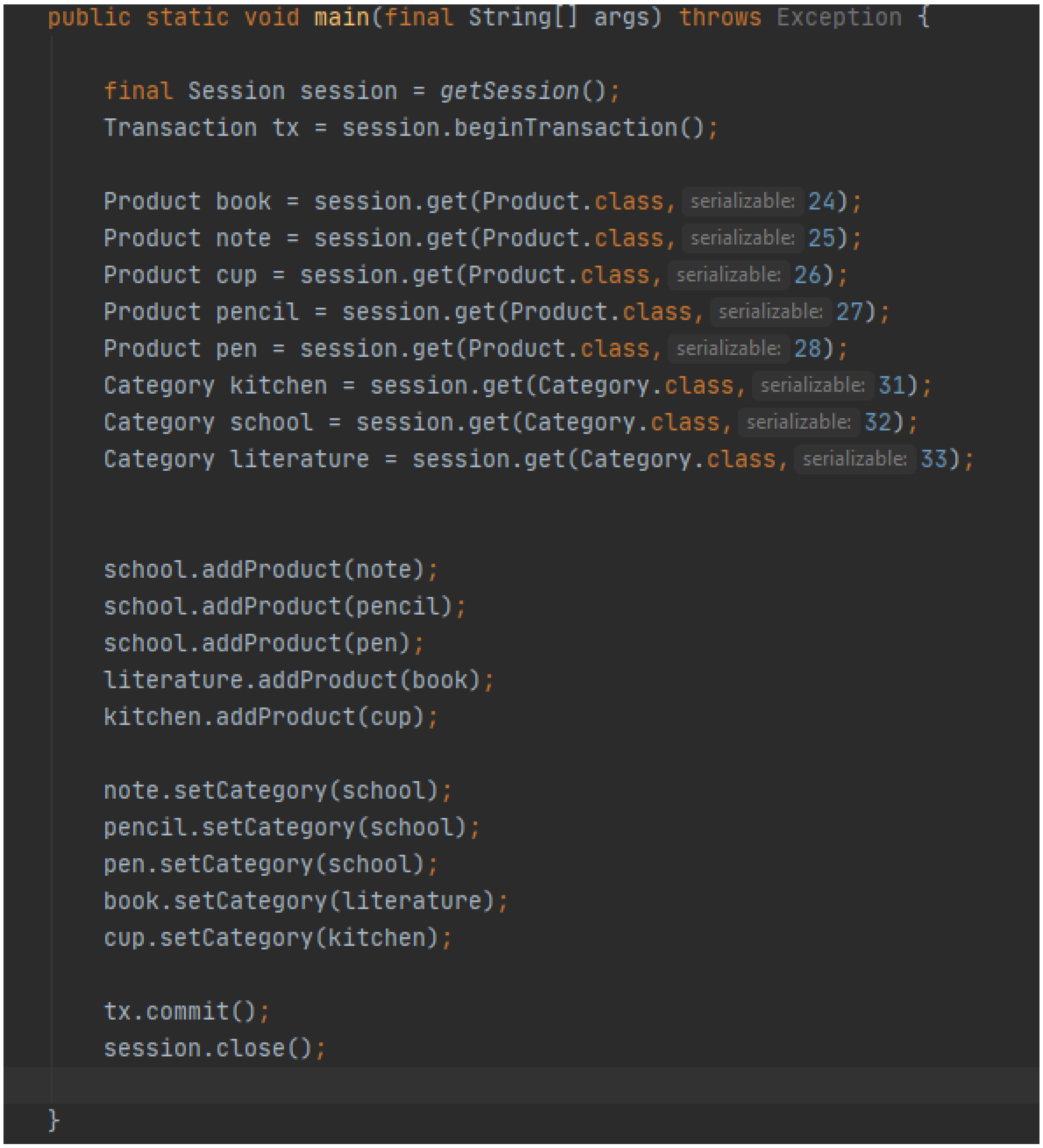


```
public static void main(final String[] args) throws Exception {

    final Session session = getSession();
    Transaction tx = session.beginTransaction();

    Product foundProduct = session.get(Product.class, serializable: 24);
    Product foundProduct2 = session.get(Product.class, serializable: 25);
    Product foundProduct3 = session.get(Product.class, serializable: 26);
    Product foundProduct4 = session.get(Product.class, serializable: 27);
    Product foundProduct5 = session.get(Product.class, serializable: 28);
    Supplier foundSuplier = session.get(Supplier.class, serializable: 29);
    Supplier foundSuplier2 = session.get(Supplier.class, serializable: 30);

    foundSuplier.addProduct(foundProduct);
    foundSuplier.addProduct(foundProduct2);
    foundSuplier.addProduct(foundProduct3);
    foundSuplier2.addProduct(foundProduct4);
    foundSuplier2.addProduct(foundProduct5);

    foundProduct.setSupplier(foundSuplier);
    foundProduct2.setSupplier(foundSuplier);
    foundProduct3.setSupplier(foundSuplier);
    foundProduct4.setSupplier(foundSuplier2);
    foundProduct5.setSupplier(foundSuplier2);

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

}
```

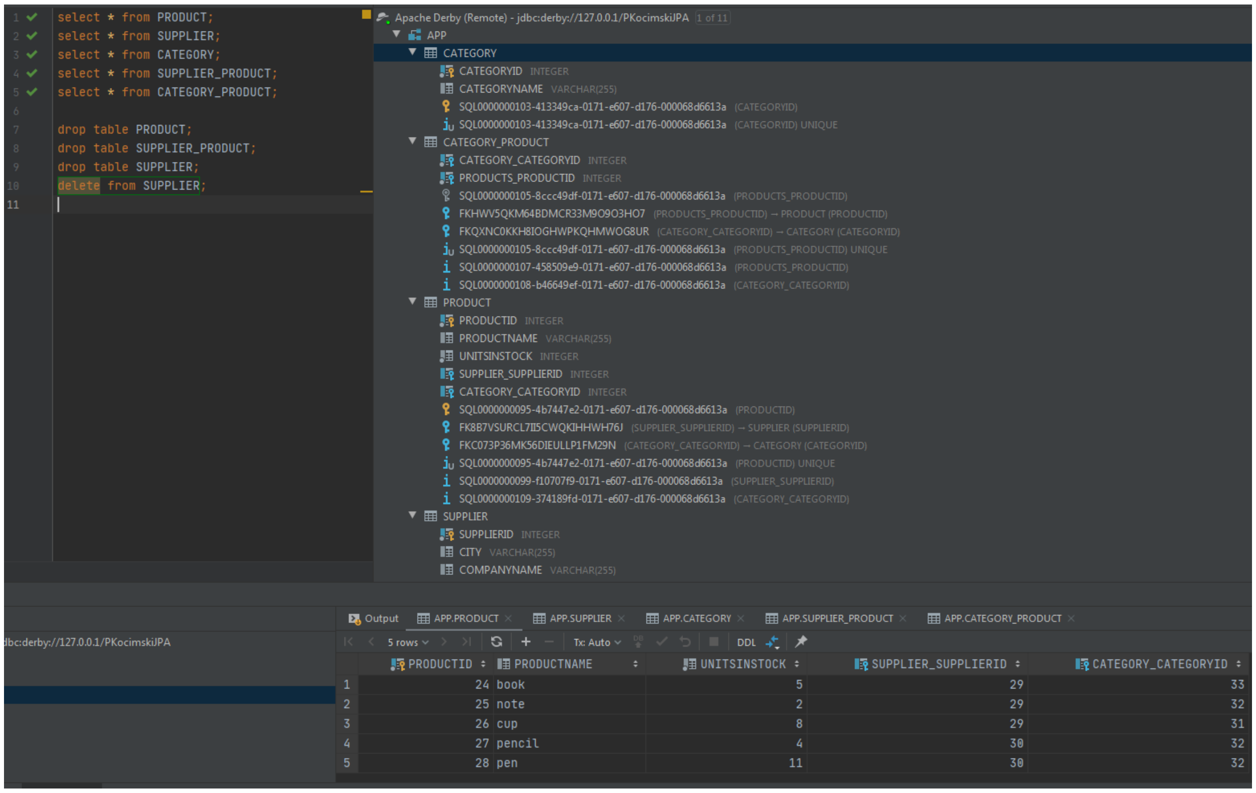


```
public static void main(final String[] args) throws Exception {
Category category = new Category( categoryName: "Kitchen");
Category category2 = new Category( categoryName: "School");
Category category3 = new Category( categoryName: "Literature");



    final Session session = getSession();
    Transaction tx = session.beginTransaction();
     session.save(category);
     session.save(category2);
     session.save(category3);

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

}
```

c. Dodaj kilka produktów do wybranej kategorii



```
public static void main(final String[] args) throws Exception {

    final Session session = getSession();
    Transaction tx = session.beginTransaction();

    Product book = session.get(Product.class, serializable: 24);
    Product note = session.get(Product.class, serializable: 25);
    Product cup = session.get(Product.class, serializable: 26);
    Product pencil = session.get(Product.class, serializable: 27);
    Product pen = session.get(Product.class, serializable: 28);
    Category kitchen = session.get(Category.class, serializable: 31);
    Category school = session.get(Category.class, serializable: 32);
    Category literature = session.get(Category.class, serializable: 33);


    school.addProduct(note);
    school.addProduct(pencil);
    school.addProduct(pen);
    literature.addProduct(book);
    kitchen.addProduct(cup);

    note.setCategory(school);
    pencil.setCategory(school);
    pen.setCategory(school);
    book.setCategory(literature);
    cup.setCategory(kitchen);

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

}
```



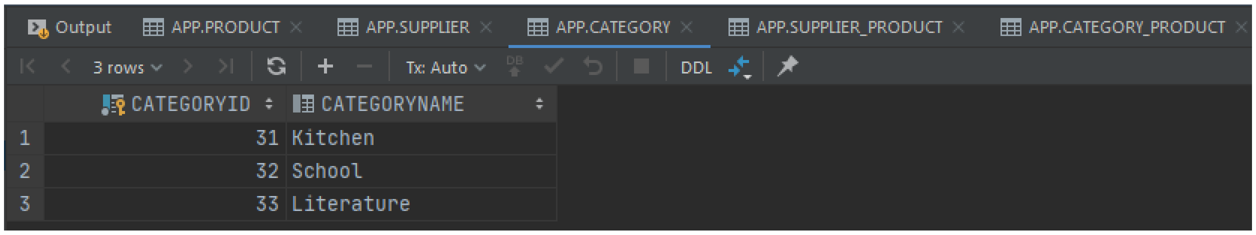
```sql
select * from PRODUCT;
select * from SUPPLIER;
select * from CATEGORY;
select * from SUPPLIER_PRODUCT;
select * from CATEGORY_PRODUCT;

drop table PRODUCT;
drop table SUPPLIER_PRODUCT;
drop table SUPPLIER;
delete from SUPPLIER;
```

```
Apache Derby (Remote) - jdbc:derby://127.0.0.1/PKocimskiJPA  1 of 11
APP
  CATEGORY
    CATEGORYID  INTEGER
    CATEGORYNAME  VARCHAR(255)
    SQL0000000103-413349ca-0171-e607-d176-000068d6613a  (CATEGORYID)
    SQL0000000103-413349ca-0171-e607-d176-000068d6613a  (CATEGORYID) UNIQUE
  CATEGORY_PRODUCT
    CATEGORY_CATEGORYID  INTEGER
    PRODUCTS_PRODUCTID  INTEGER
    SQL0000000105-8ccc49df-0171-e607-d176-000068d6613a  (PRODUCTS_PRODUCTID)
    FKHWV5QKM64BDMCR33M9O9O3HO7  (PRODUCTS_PRODUCTID) → PRODUCT (PRODUCTID)
    FKQXNC0KKH8IOGHWPKQHMWOG8UR  (CATEGORY_CATEGORYID) → CATEGORY (CATEGORYID)
    SQL0000000105-8ccc49df-0171-e607-d176-000068d6613a  (PRODUCTS_PRODUCTID) UNIQUE
    SQL0000000107-458509e9-0171-e607-d176-000068d6613a  (PRODUCTS_PRODUCTID)
    SQL0000000108-b46649ef-0171-e607-d176-000068d6613a  (CATEGORY_CATEGORYID)
  PRODUCT
    PRODUCTID  INTEGER
    PRODUCTNAME  VARCHAR(255)
    UNITSINSTOCK  INTEGER
    SUPPLIER_SUPPLIERID  INTEGER
    CATEGORY_CATEGORYID  INTEGER
    SQL0000000095-4b7447e2-0171-e607-d176-000068d6613a  (PRODUCTID)
    FK8B7VSURCL7II5CWQKIHHWH76J  (SUPPLIER_SUPPLIERID) → SUPPLIER (SUPPLIERID)
    FKC073P36MK56DIEULLP1FM29N  (CATEGORY_CATEGORYID) → CATEGORY (CATEGORYID)
    SQL0000000095-4b7447e2-0171-e607-d176-000068d6613a  (PRODUCTID) UNIQUE
    SQL0000000099-f10707f9-0171-e607-d176-000068d6613a  (SUPPLIER_SUPPLIERID)
    SQL0000000109-374189fd-0171-e607-d176-000068d6613a  (CATEGORY_CATEGORYID)
  SUPPLIER
    SUPPLIERID  INTEGER
    CITY  VARCHAR(255)
    COMPANYNAME  VARCHAR(255)
```

jdbc:derby://127.0.0.1/PKocimskiJPA

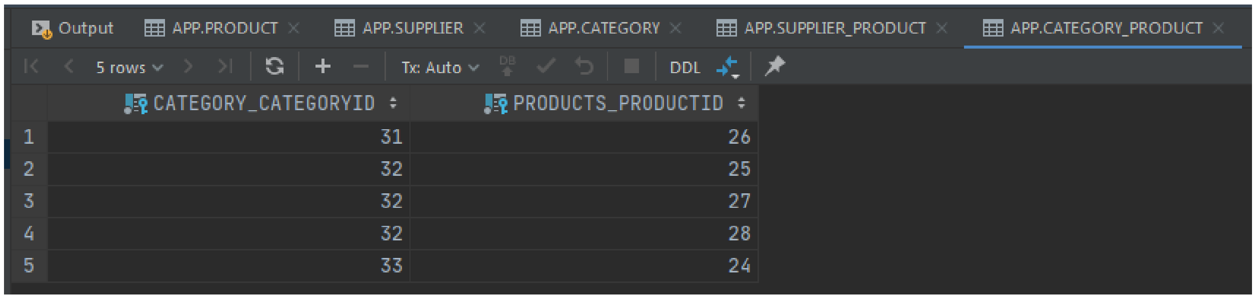
APP.PRODUCT — 5 rows

	PRODUCTID	PRODUCTNAME	UNITSINSTOCK	SUPPLIER_SUPPLIERID	CATEGORY_CATEGORYID
1	24	book	5	29	33
2	25	note	2	29	32
3	26	cup	8	29	31
4	27	pencil	4	30	32
5	28	pen	11	30	32



APP.CATEGORY — 3 rows

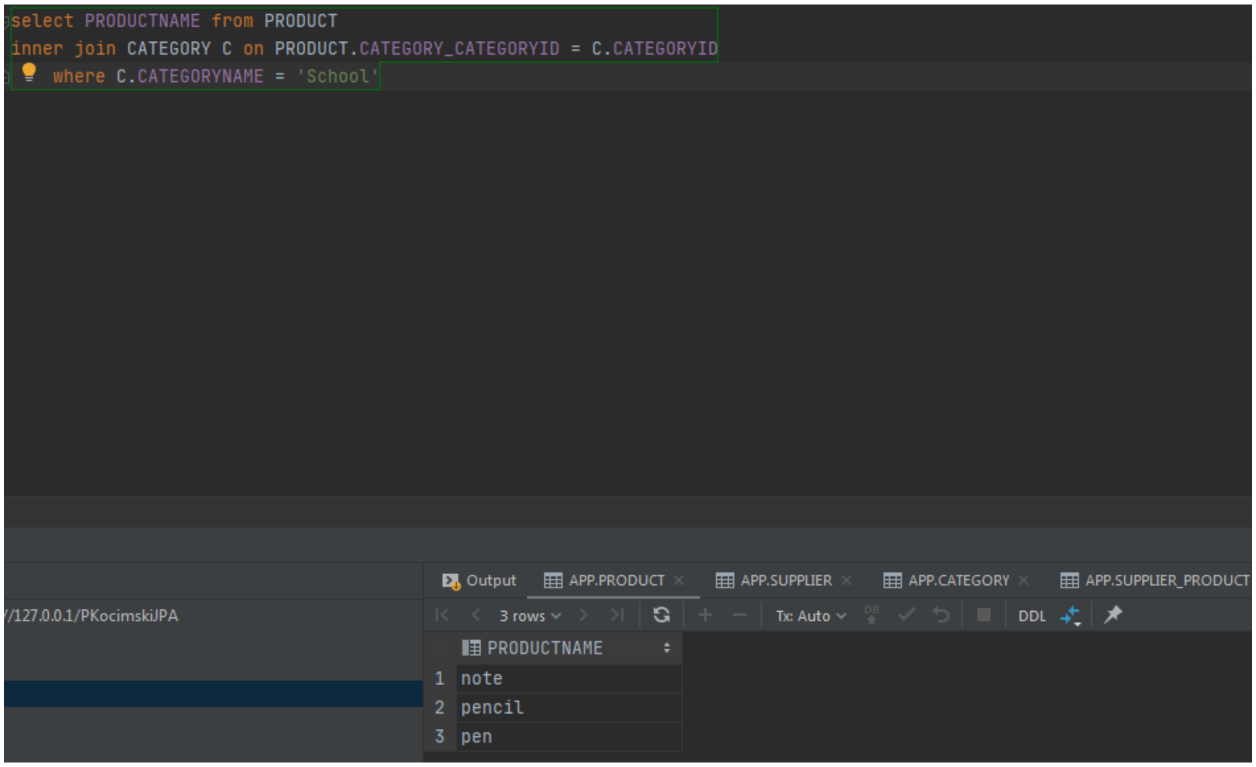
	CATEGORYID	CATEGORYNAME
1	31	Kitchen
2	32	School
3	33	Literature



APP.CATEGORY_PRODUCT — 5 rows

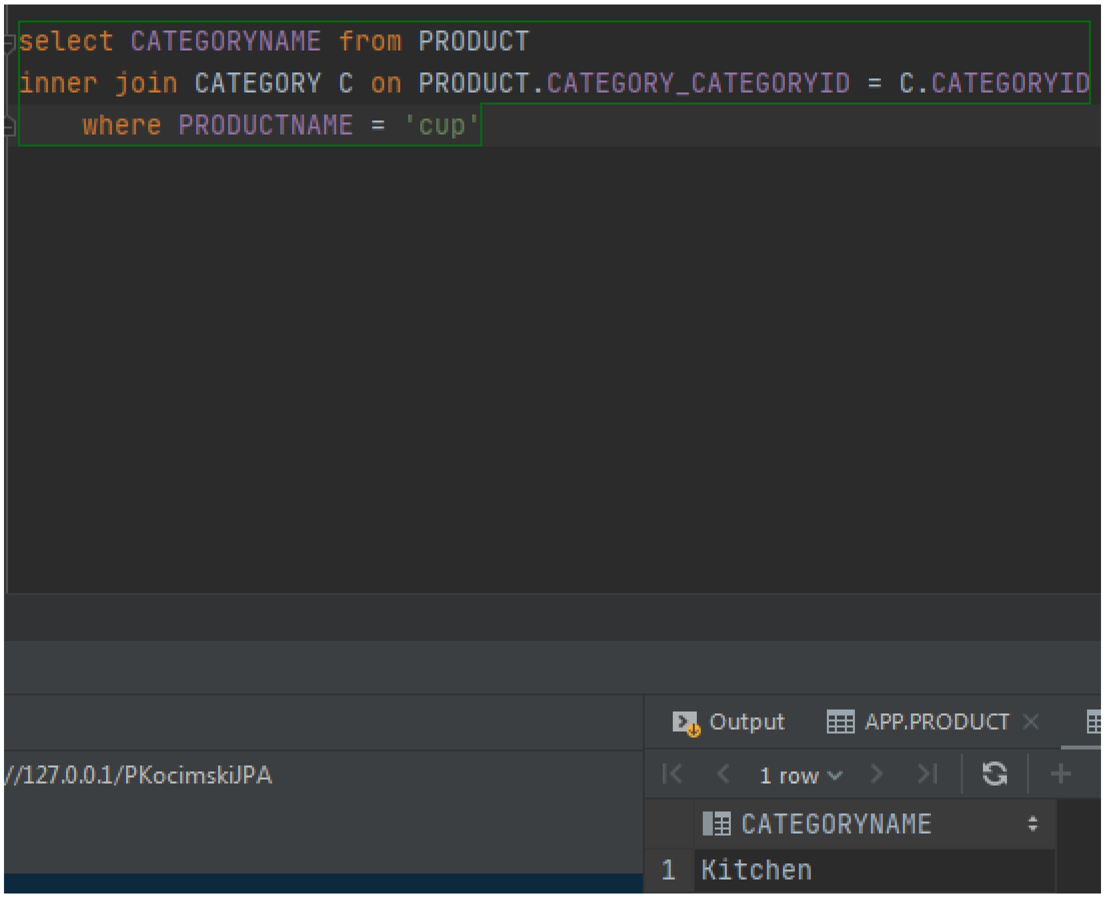
	CATEGORY_CATEGORYID	PRODUCTS_PRODUCTID
1	31	26
2	32	25
3	32	27
4	32	28
5	33	24

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



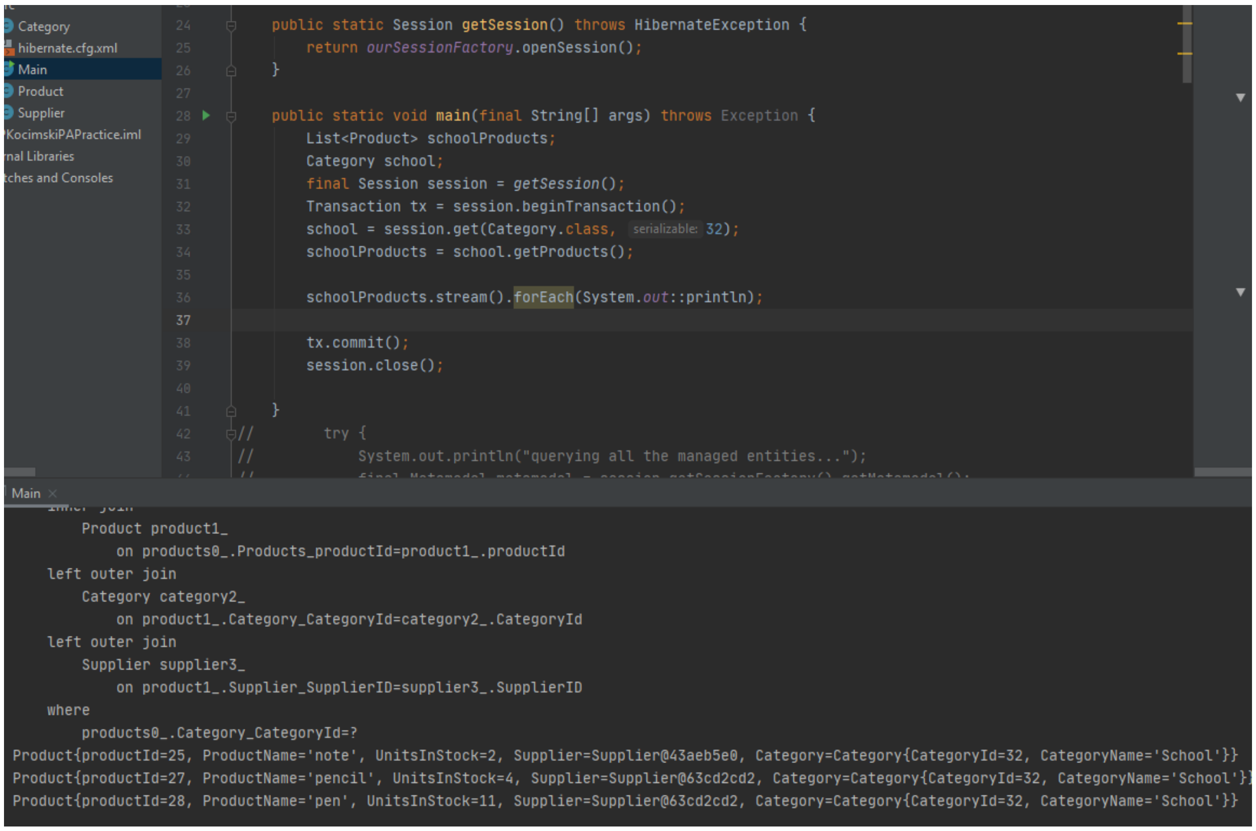
```
select PRODUCTNAME from PRODUCT
inner join CATEGORY C on PRODUCT.CATEGORY_CATEGORYID = C.CATEGORYID
  where C.CATEGORYNAME = 'School'
```

```
  PRODUCTNAME
1 note
2 pencil
3 pen
```



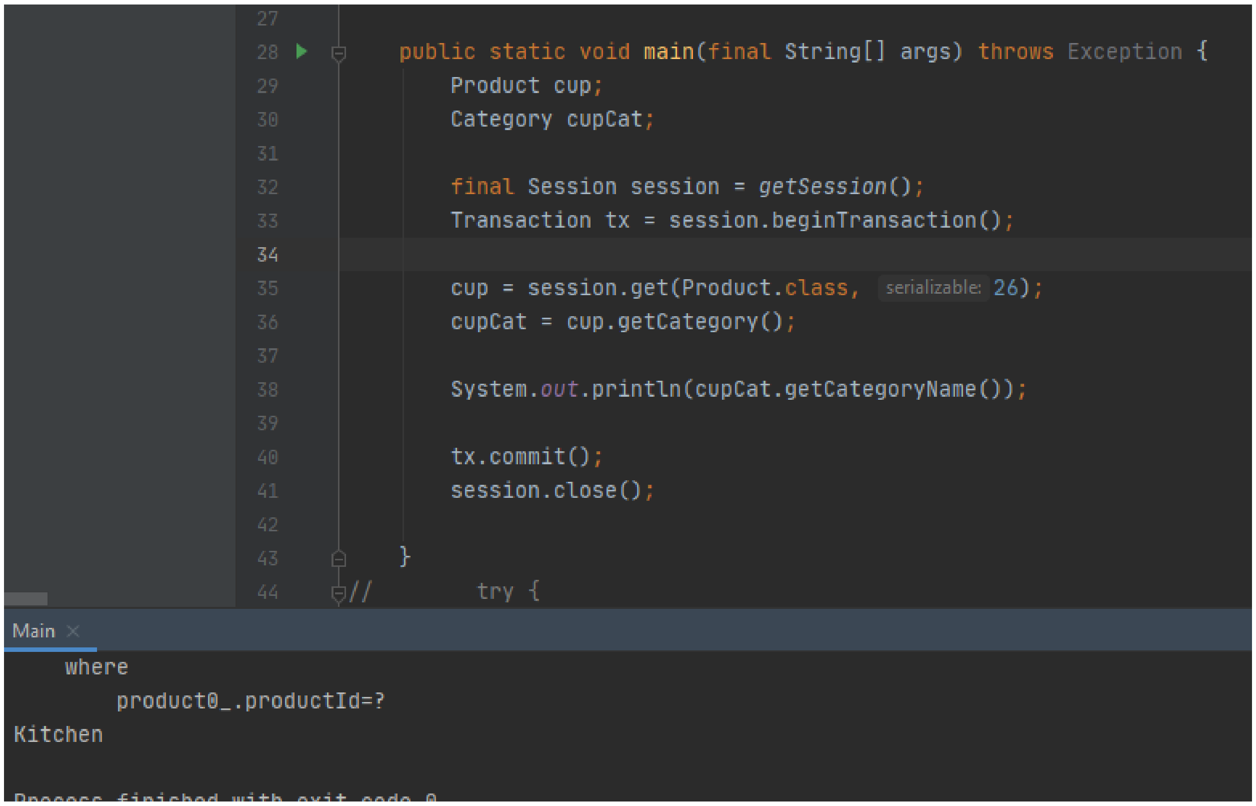
```
select CATEGORYNAME from PRODUCT
inner join CATEGORY C on PRODUCT.CATEGORY_CATEGORYID = C.CATEGORYID
    where PRODUCTNAME = 'cup'
```

```
  CATEGORYNAME
1 Kitchen
```



```java
    public static Session getSession() throws HibernateException {
        return ourSessionFactory.openSession();
    }

    public static void main(final String[] args) throws Exception {
        List<Product> schoolProducts;
        Category school;
        final Session session = getSession();
        Transaction tx = session.beginTransaction();
        school = session.get(Category.class, serializable: 32);
        schoolProducts = school.getProducts();

        schoolProducts.stream().forEach(System.out::println);

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

    }
//        try {
//            System.out.println("querying all the managed entities...");
```

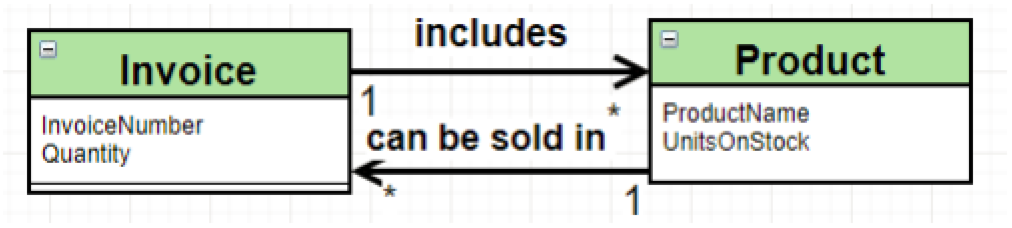
```
        Product product1_
            on products0_.Products_productId=product1_.productId
    left outer join
        Category category2_
            on product1_.Category_CategoryId=category2_.CategoryId
    left outer join
        Supplier supplier3_
            on product1_.Supplier_SupplierID=supplier3_.SupplierID
    where
        products0_.Category_CategoryId=?
Product{productId=25, ProductName='note', UnitsInStock=2, Supplier=Supplier@43aeb5e0, Category=Category{CategoryId=32, CategoryName='School'}}
Product{productId=27, ProductName='pencil', UnitsInStock=4, Supplier=Supplier@63cd2cd2, Category=Category{CategoryId=32, CategoryName='School'}}
Product{productId=28, ProductName='pen', UnitsInStock=11, Supplier=Supplier@63cd2cd2, Category=Category{CategoryId=32, CategoryName='School'}}
```

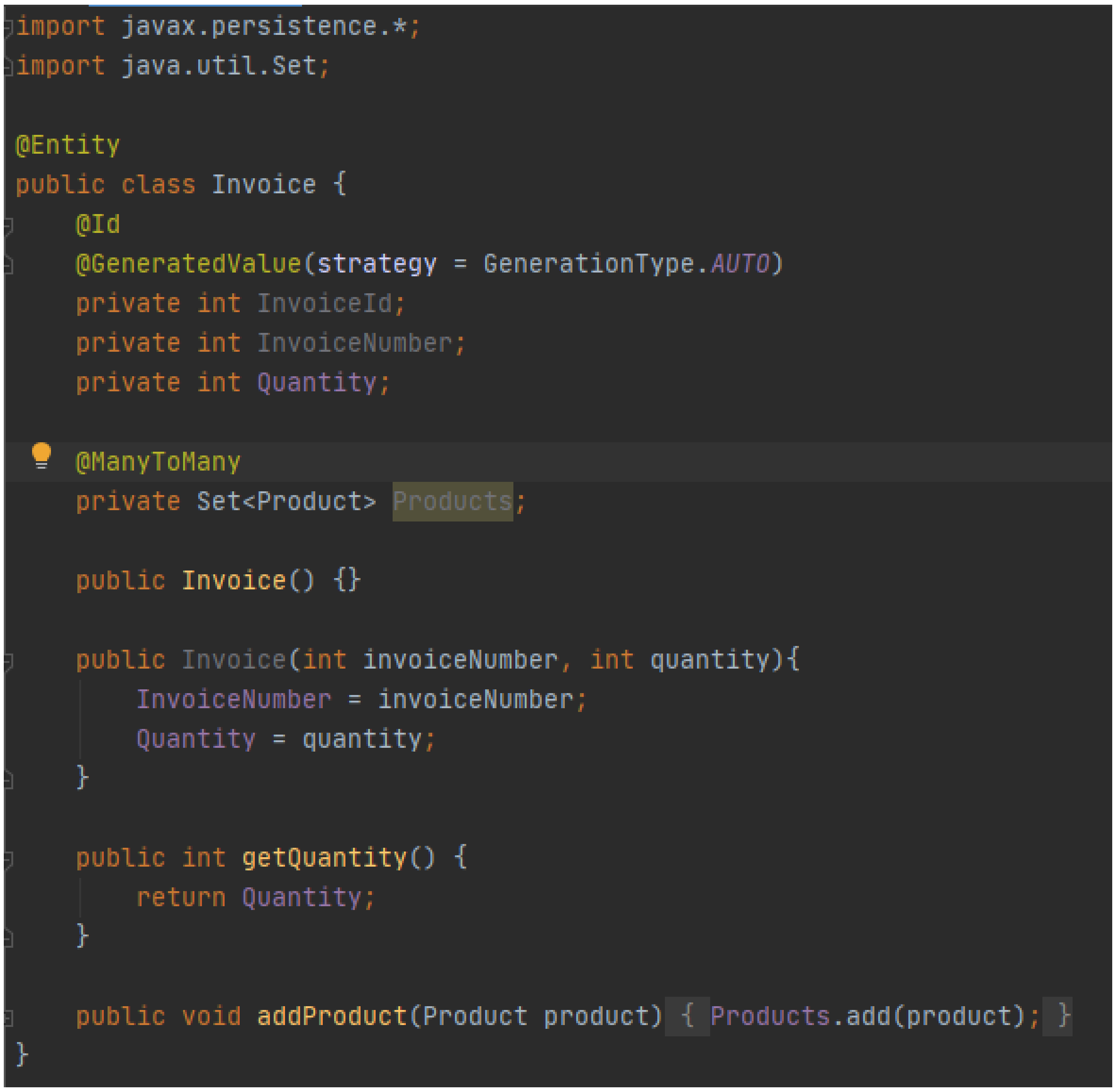


```java
    public static void main(final String[] args) throws Exception {
        Product cup;
        Category cupCat;

        final Session session = getSession();
        Transaction tx = session.beginTransaction();

        cup = session.get(Product.class, serializable: 26);
        cupCat = cup.getCategory();

        System.out.println(cupCat.getCategoryName());

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

    }
//        try {
```

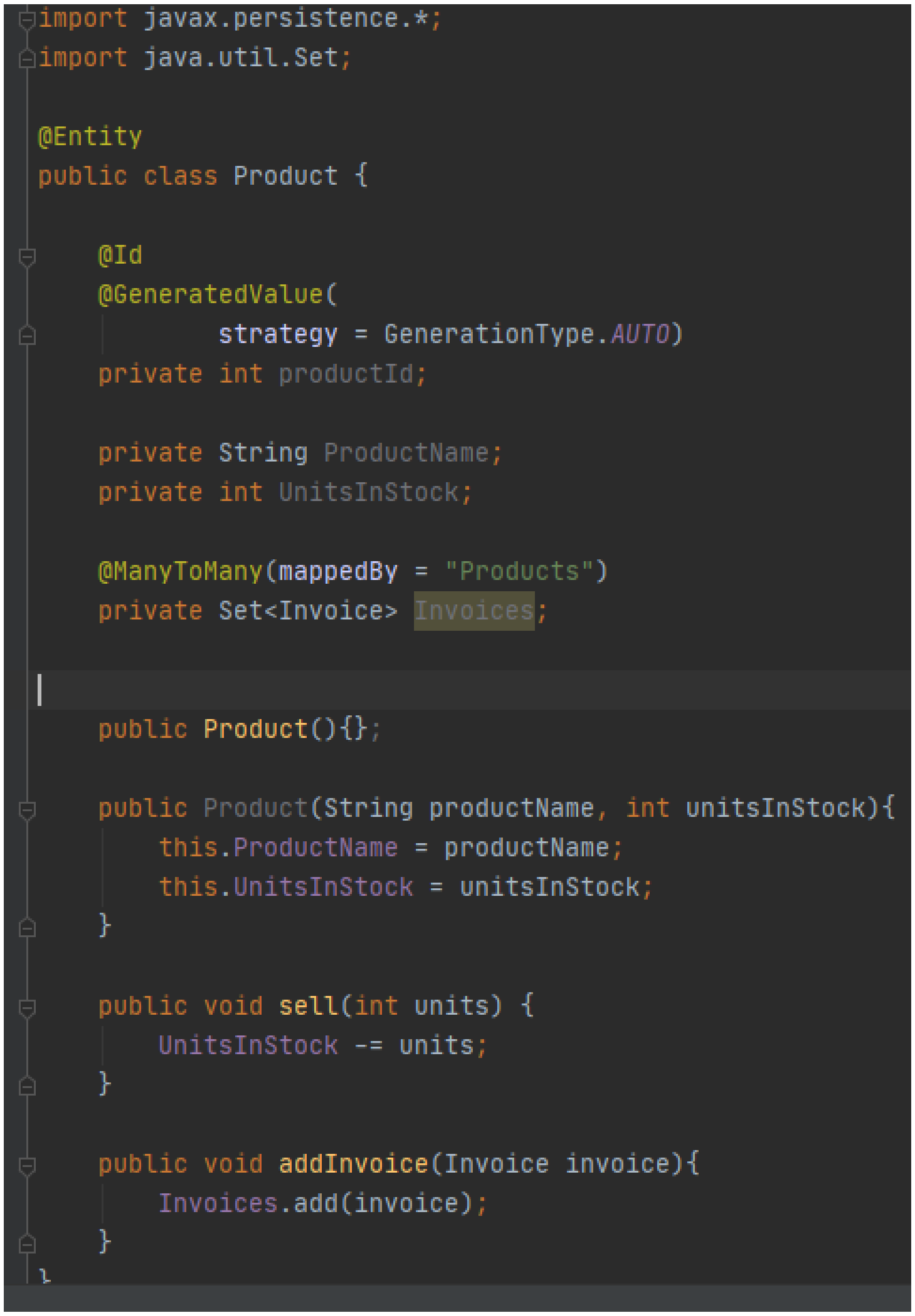
```
    where
        product0_.productId=?
Kitchen
```

VIII. Zamodeluj relacje wiele-do-wielu, jak poniżej:



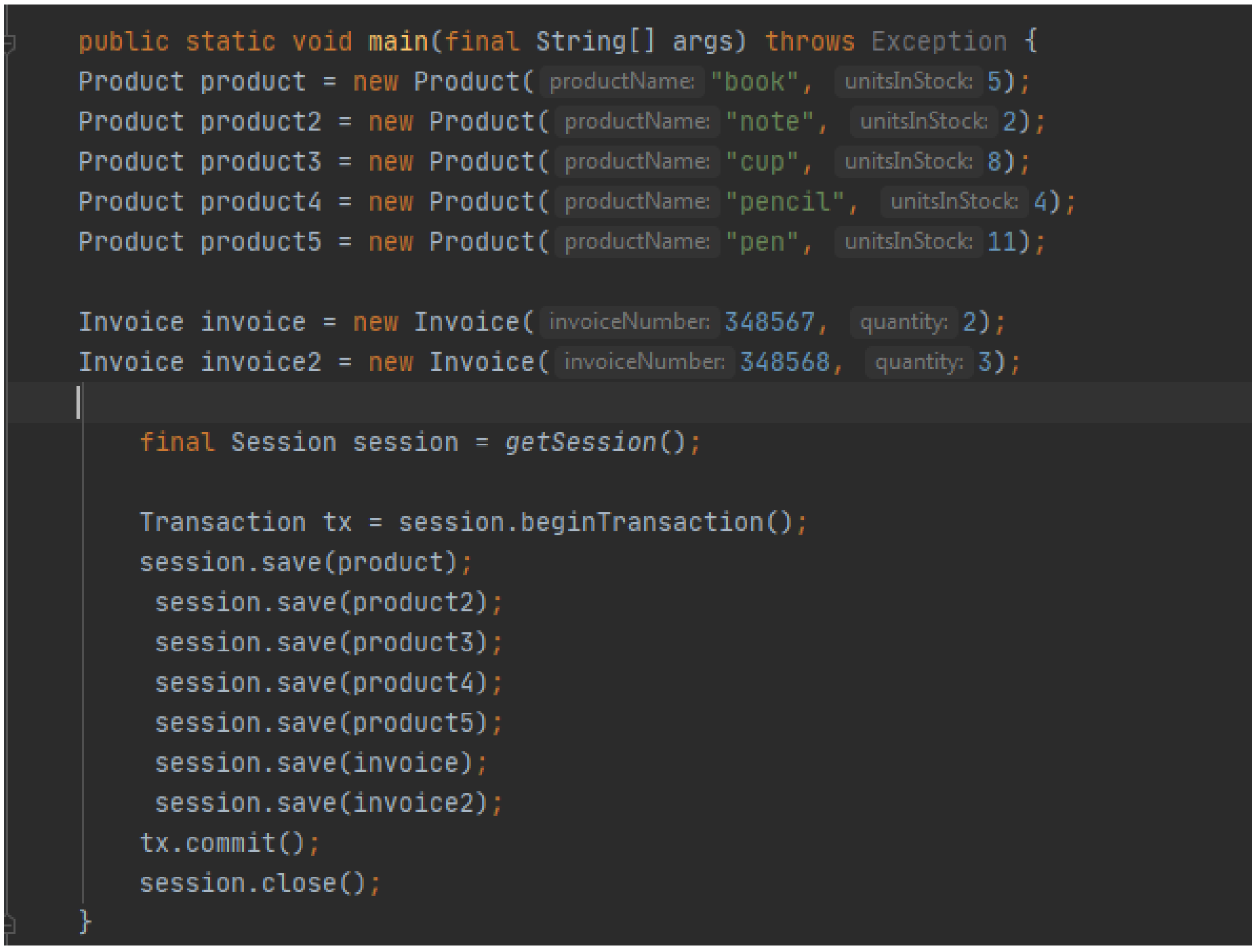

```java
import javax.persistence.*;
import java.util.Set;

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

    @ManyToMany
    private Set<Product> Products;

    public Invoice() {}

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

    public int getQuantity() {
        return Quantity;
    }

    public void addProduct(Product product) { Products.add(product); }
}
```

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

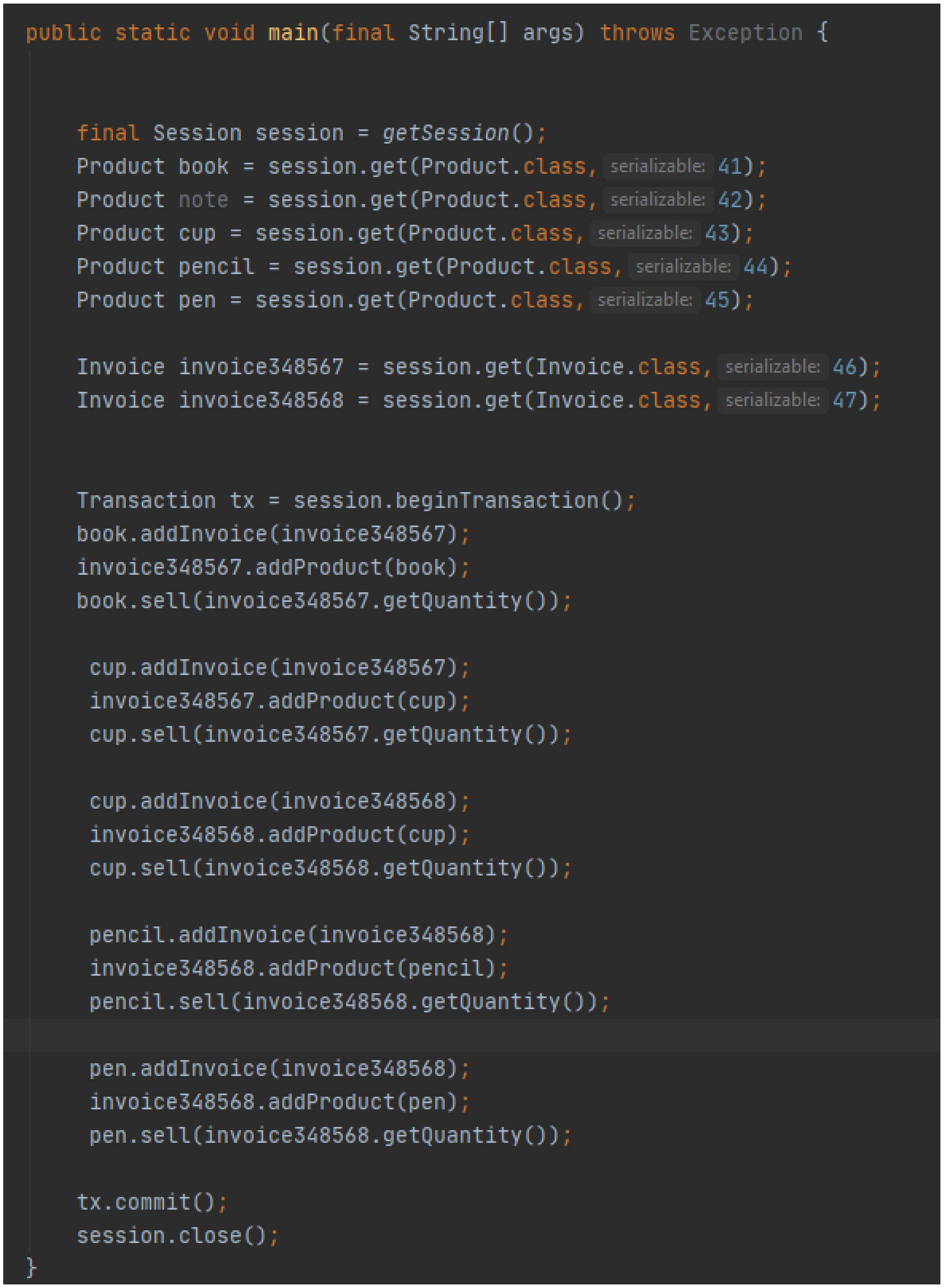


```
import javax.persistence.*;
import java.util.Set;

@Entity
public class Product {

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

    private String ProductName;
    private int UnitsInStock;

    @ManyToMany(mappedBy = "Products")
    private Set<Invoice> Invoices;


    public Product(){};

    public Product(String productName, int unitsInStock){
        this.ProductName = productName;
        this.UnitsInStock = unitsInStock;
    }

    public void sell(int units) {
        UnitsInStock -= units;
    }

    public void addInvoice(Invoice invoice){
        Invoices.add(invoice);
    }
}
```

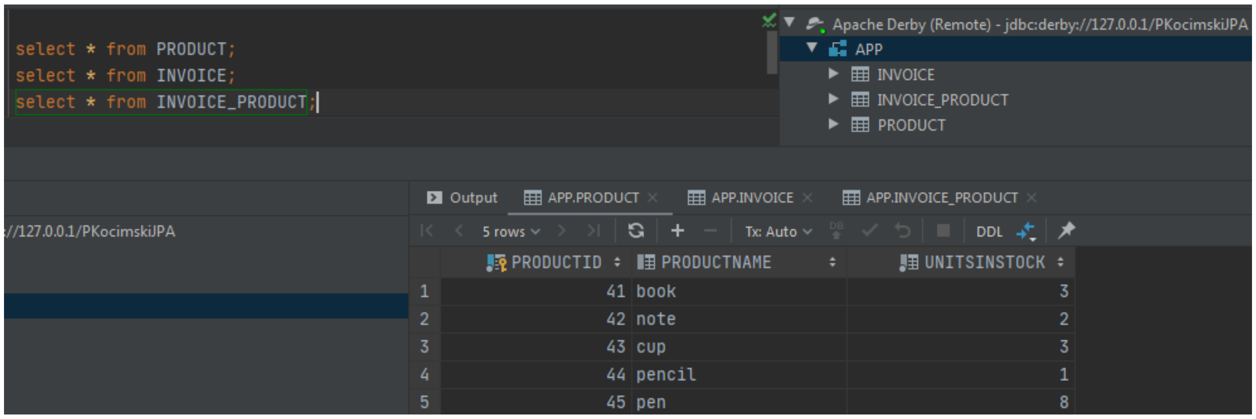
a. Stwórz kilka produktów I “sprzedaj” je na kilku transakcjach.



```java
public static void main(final String[] args) throws Exception {
Product product = new Product( productName: "book",  unitsInStock: 5);
Product product2 = new Product( productName: "note",  unitsInStock: 2);
Product product3 = new Product( productName: "cup",  unitsInStock: 8);
Product product4 = new Product( productName: "pencil",  unitsInStock: 4);
Product product5 = new Product( productName: "pen",  unitsInStock: 11);

Invoice invoice = new Invoice( invoiceNumber: 348567,  quantity: 2);
Invoice invoice2 = new Invoice( invoiceNumber: 348568,  quantity: 3);

    final Session session = getSession();

    Transaction tx = session.beginTransaction();
    session.save(product);
     session.save(product2);
     session.save(product3);
     session.save(product4);
     session.save(product5);
     session.save(invoice);
     session.save(invoice2);
    tx.commit();
    session.close();
}
```



```java
public static void main(final String[] args) throws Exception {


    final Session session = getSession();
    Product book = session.get(Product.class, serializable: 41);
    Product note = session.get(Product.class, serializable: 42);
    Product cup = session.get(Product.class, serializable: 43);
    Product pencil = session.get(Product.class, serializable: 44);
    Product pen = session.get(Product.class, serializable: 45);

    Invoice invoice348567 = session.get(Invoice.class, serializable: 46);
    Invoice invoice348568 = session.get(Invoice.class, serializable: 47);


    Transaction tx = session.beginTransaction();
    book.addInvoice(invoice348567);
    invoice348567.addProduct(book);
    book.sell(invoice348567.getQuantity());

     cup.addInvoice(invoice348567);
     invoice348567.addProduct(cup);
     cup.sell(invoice348567.getQuantity());

     cup.addInvoice(invoice348568);
     invoice348568.addProduct(cup);
     cup.sell(invoice348568.getQuantity());

     pencil.addInvoice(invoice348568);
     invoice348568.addProduct(pencil);
     pencil.sell(invoice348568.getQuantity());

     pen.addInvoice(invoice348568);
     invoice348568.addProduct(pen);
     pen.sell(invoice348568.getQuantity());

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



```
select * from PRODUCT;
select * from INVOICE;
select * from INVOICE_PRODUCT;
```

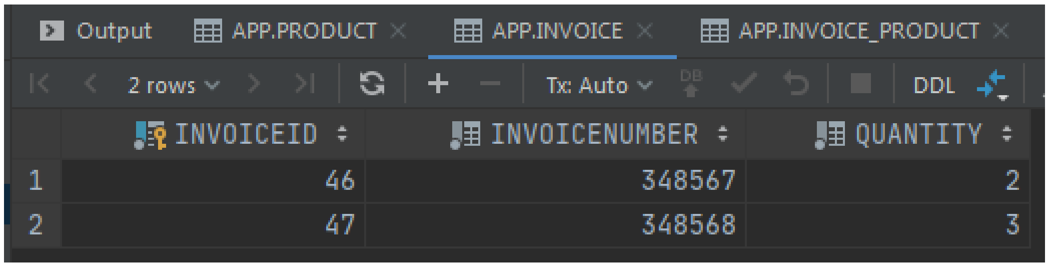
Apache Derby (Remote) - jdbc:derby://127.0.0.1/PKocimskiJPA
APP
INVOICE
INVOICE_PRODUCT
PRODUCT

//127.0.0.1/PKocimskiJPA

Output | APP.PRODUCT | APP.INVOICE | APP.INVOICE_PRODUCT

5 rows

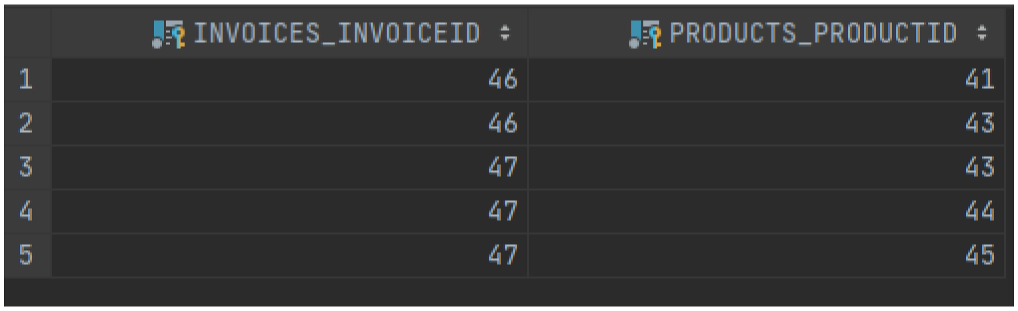
	PRODUCTID	PRODUCTNAME	UNITSINSTOCK
1	41	book	3
2	42	note	2
3	43	cup	3
4	44	pencil	1
5	45	pen	8



Output | APP.PRODUCT | APP.INVOICE | APP.INVOICE_PRODUCT

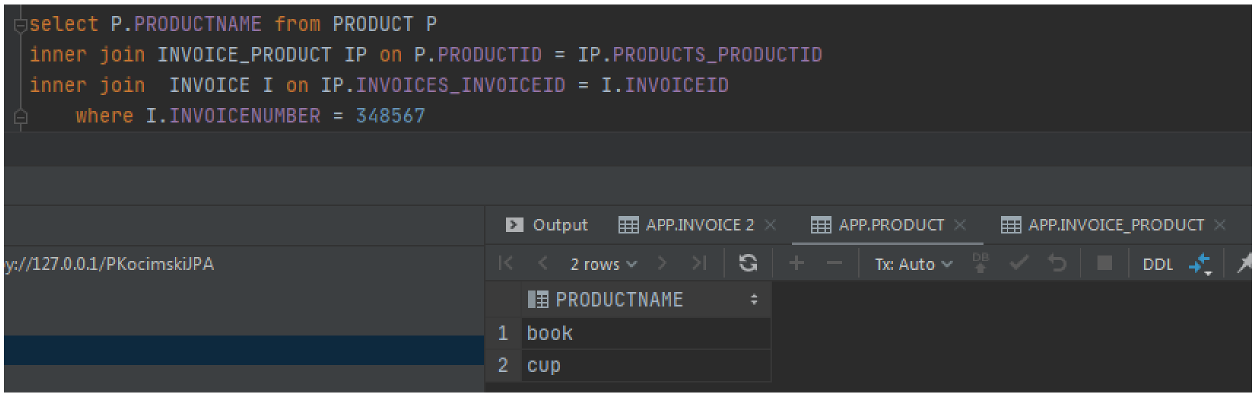
2 rows

	INVOICEID	INVOICENUMBER	QUANTITY
1	46	348567	2
2	47	348568	3



	INVOICES_INVOICEID	PRODUCTS_PRODUCTID
1	46	41
2	46	43
3	47	43
4	47	44
5	47	45

b. Pokaż produkty sprzedane w ramach wybranej faktury/transakcji



```
select P.PRODUCTNAME from PRODUCT P
inner join INVOICE_PRODUCT IP on P.PRODUCTID = IP.PRODUCTS_PRODUCTID
inner join  INVOICE I on IP.INVOICES_INVOICEID = I.INVOICEID
    where I.INVOICENUMBER = 348567
```

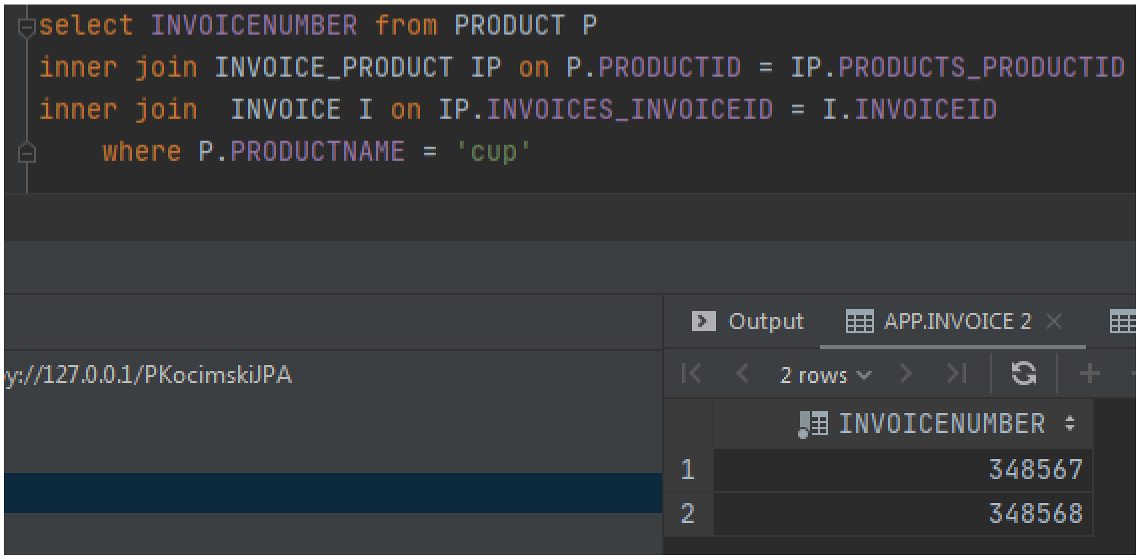
y://127.0.0.1/PKocimskiJPA

Output | APP.INVOICE 2 | APP.PRODUCT | APP.INVOICE_PRODUCT

2 rows

	PRODUCTNAME
1	book
2	cup

c. Pokaż faktury w ramach których był sprzedany wybrany produkt

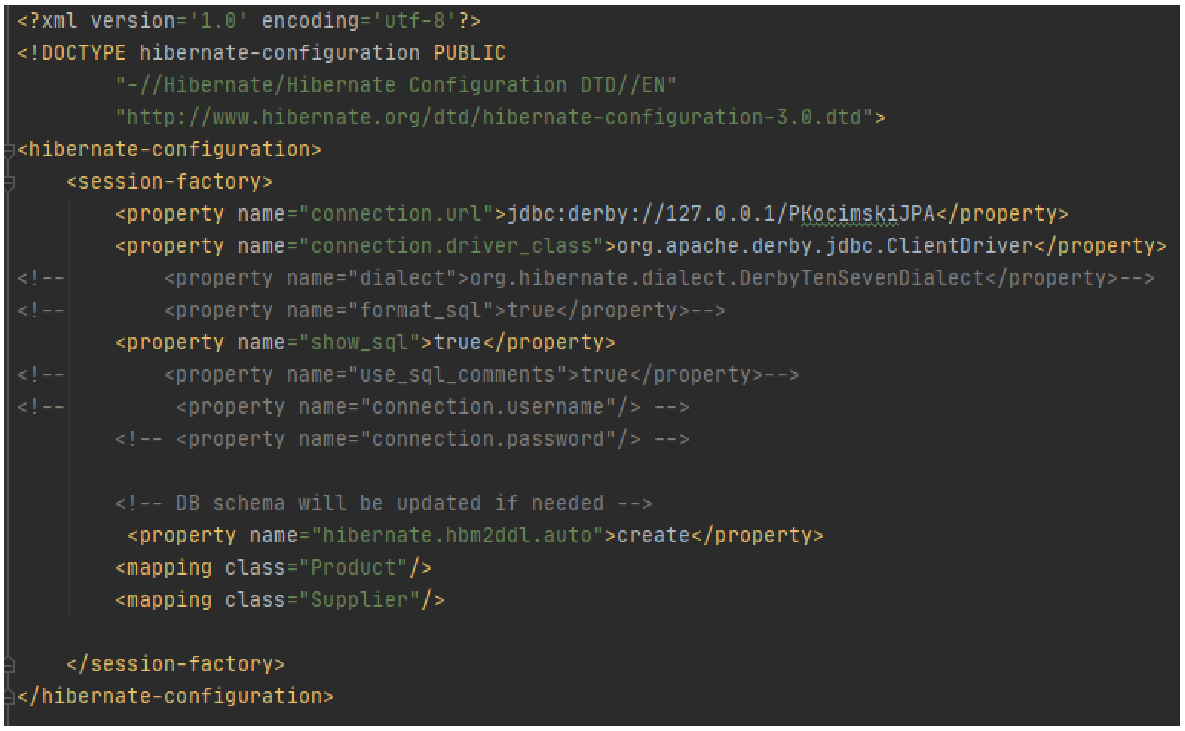


```
select INVOICENUMBER from PRODUCT P
inner join INVOICE_PRODUCT IP on P.PRODUCTID = IP.PRODUCTS_PRODUCTID
inner join  INVOICE I on IP.INVOICES_INVOICEID = I.INVOICEID
    where P.PRODUCTNAME = 'cup'
```

```
   INVOICENUMBER
1         348567
2         348568
```

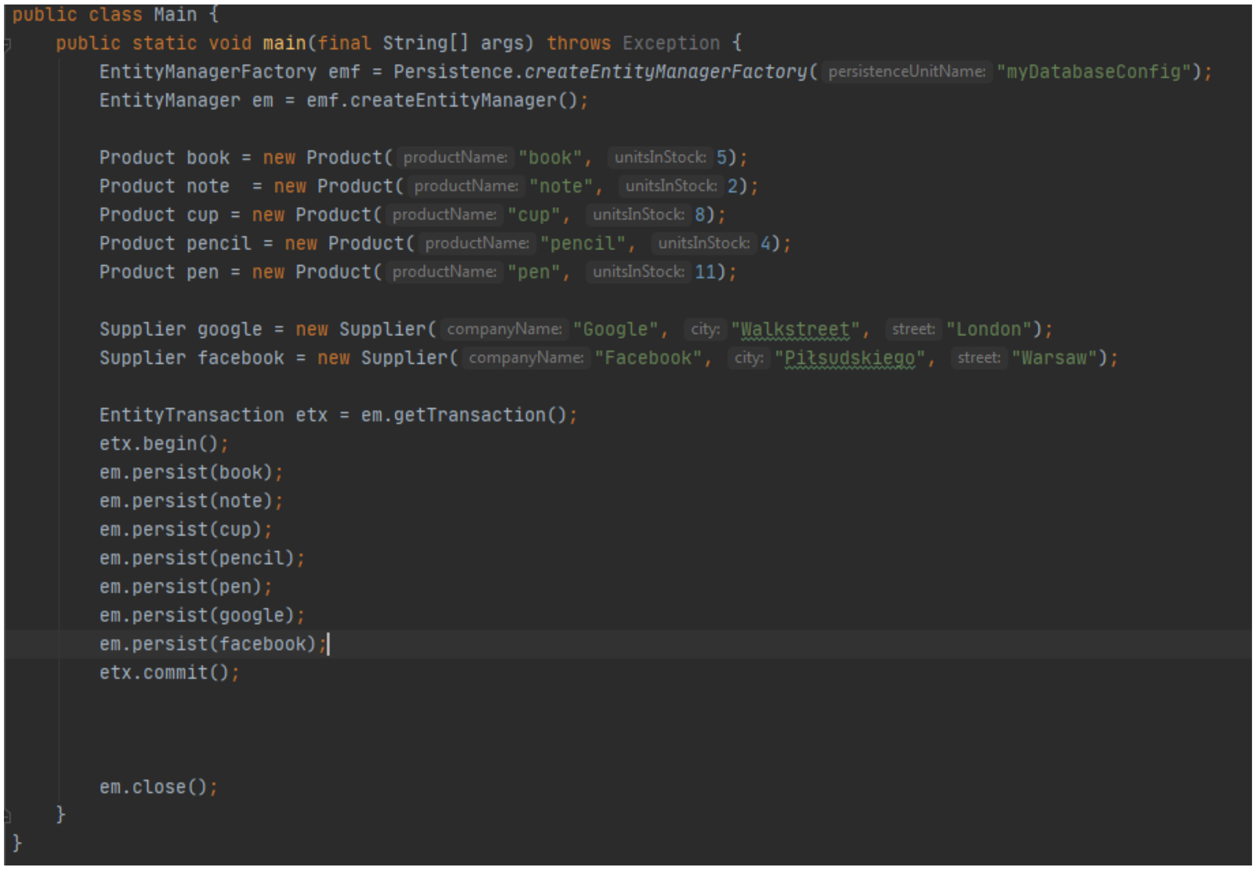
X. JPA

Dodałem plik konfiguracyjny persistence.xml

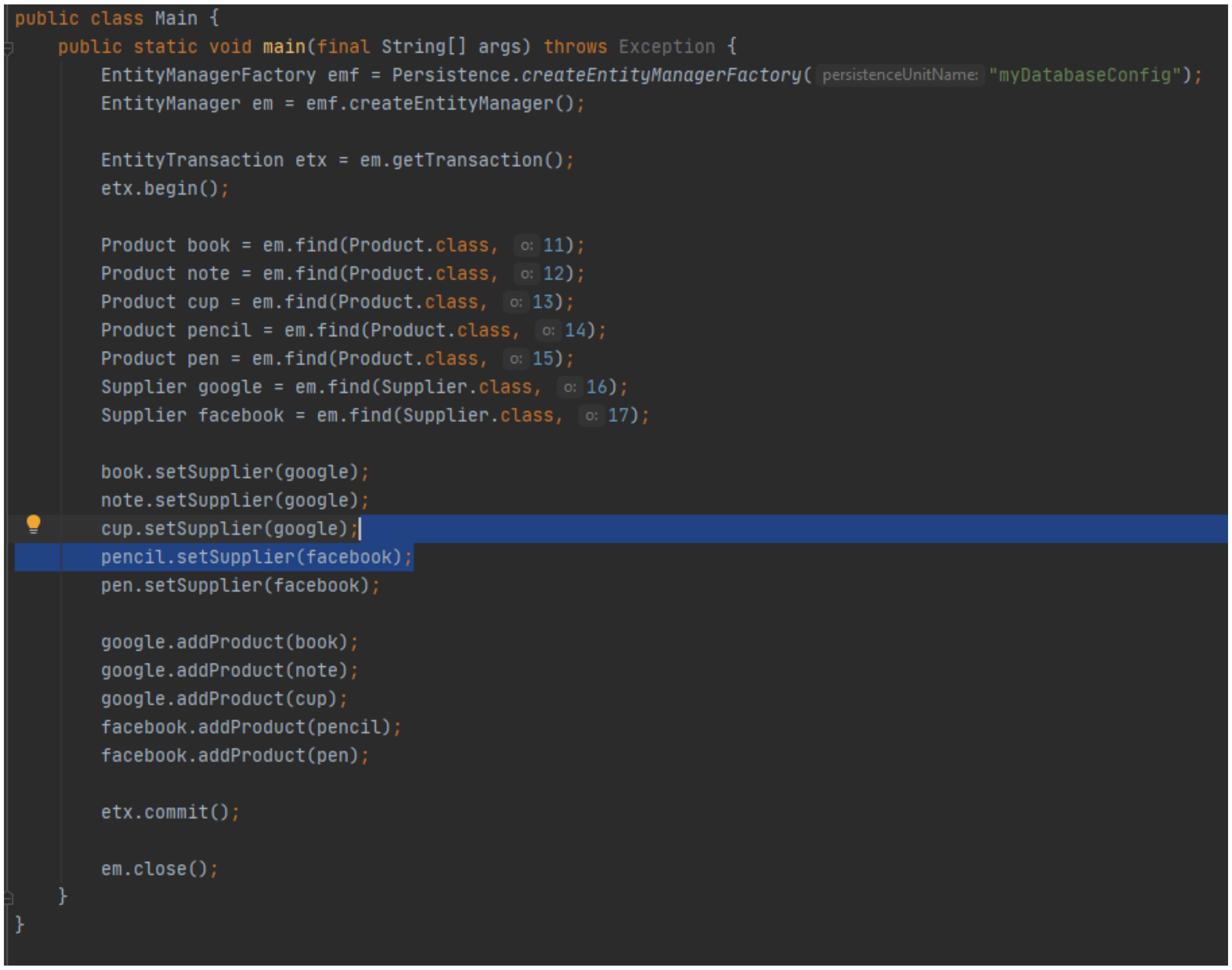


```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="connection.url">jdbc:derby://127.0.0.1/PKocimskiJPA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

        <!-- DB schema will be updated if needed -->
         <property name="hibernate.hbm2ddl.auto">create</property>
        <mapping class="Product"/>
        <mapping class="Supplier"/>

    </session-factory>
</hibernate-configuration>
```

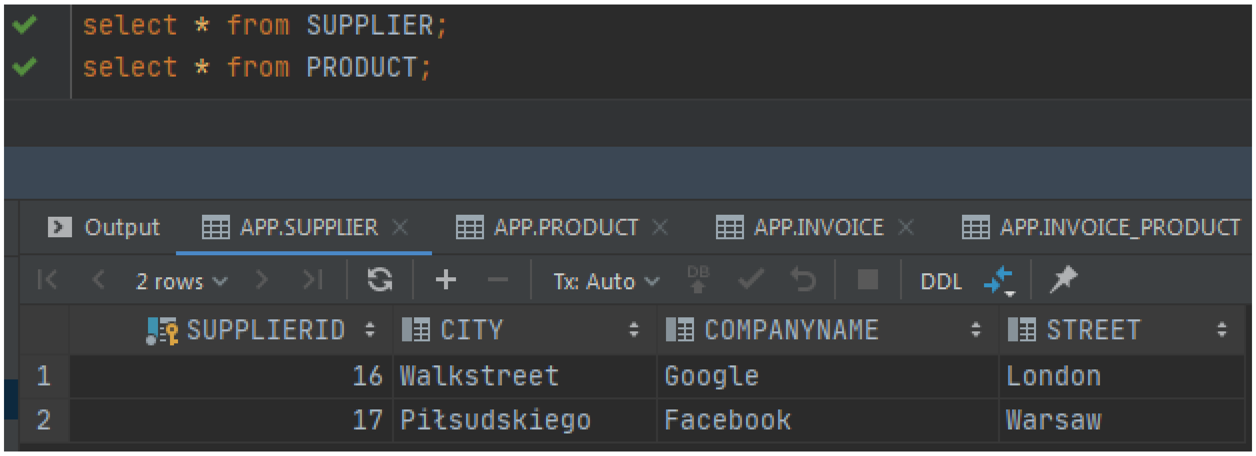
a. Stwórz nowego maina w którym zrobisz to samo co w punkcie VI ale z wykorzystaniem JPA



```java
public class Main {
    public static void main(final String[] args) throws Exception {
        EntityManagerFactory emf = Persistence.createEntityManagerFactory( persistenceUnitName: "myDatabaseConfig");
        EntityManager em = emf.createEntityManager();

        Product book = new Product( productName: "book",  unitsInStock: 5);
        Product note  = new Product( productName: "note",  unitsInStock: 2);
        Product cup = new Product( productName: "cup",  unitsInStock: 8);
        Product pencil = new Product( productName: "pencil",  unitsInStock: 4);
        Product pen = new Product( productName: "pen",  unitsInStock: 11);

        Supplier google = new Supplier( companyName: "Google",  city: "Walkstreet",  street: "London");
        Supplier facebook = new Supplier( companyName: "Facebook",  city: "Piłsudskiego",  street: "Warsaw");

        EntityTransaction etx = em.getTransaction();
        etx.begin();
        em.persist(book);
        em.persist(note);
        em.persist(cup);
        em.persist(pencil);
        em.persist(pen);
        em.persist(google);
        em.persist(facebook);
        etx.commit();



        em.close();
    }
}
```



```java
public class Main {
    public static void main(final String[] args) throws Exception {
        EntityManagerFactory emf = Persistence.createEntityManagerFactory( persistenceUnitName: "myDatabaseConfig");
        EntityManager em = emf.createEntityManager();

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

        Product book = em.find(Product.class,  o: 11);
        Product note = em.find(Product.class,  o: 12);
        Product cup = em.find(Product.class,  o: 13);
        Product pencil = em.find(Product.class,  o: 14);
        Product pen = em.find(Product.class,  o: 15);
        Supplier google = em.find(Supplier.class,  o: 16);
        Supplier facebook = em.find(Supplier.class,  o: 17);

        book.setSupplier(google);
        note.setSupplier(google);
        cup.setSupplier(google);
        pencil.setSupplier(facebook);
        pen.setSupplier(facebook);

        google.addProduct(book);
        google.addProduct(note);
        google.addProduct(cup);
        facebook.addProduct(pencil);
        facebook.addProduct(pen);

        etx.commit();

        em.close();
    }
}
```

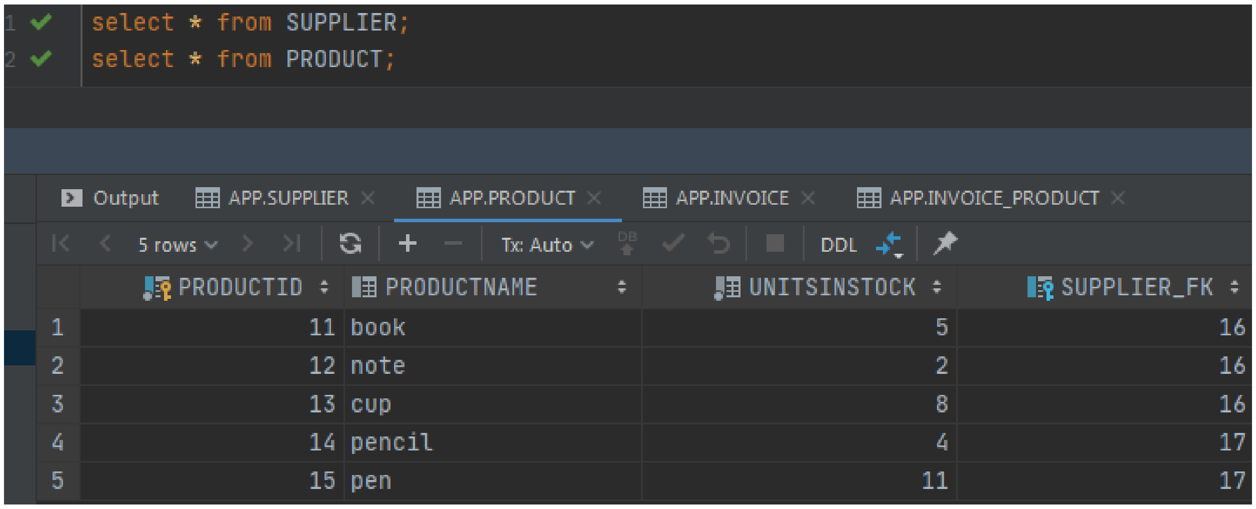


```sql
select * from SUPPLIER;
select * from PRODUCT;
```

Output | APP.SUPPLIER | APP.PRODUCT | APP.INVOICE | APP.INVOICE_PRODUCT

2 rows | Tx: Auto | DDL

	SUPPLIERID	CITY	COMPANYNAME	STREET
1	16	Walkstreet	Google	London
2	17	Piłsudskiego	Facebook	Warsaw

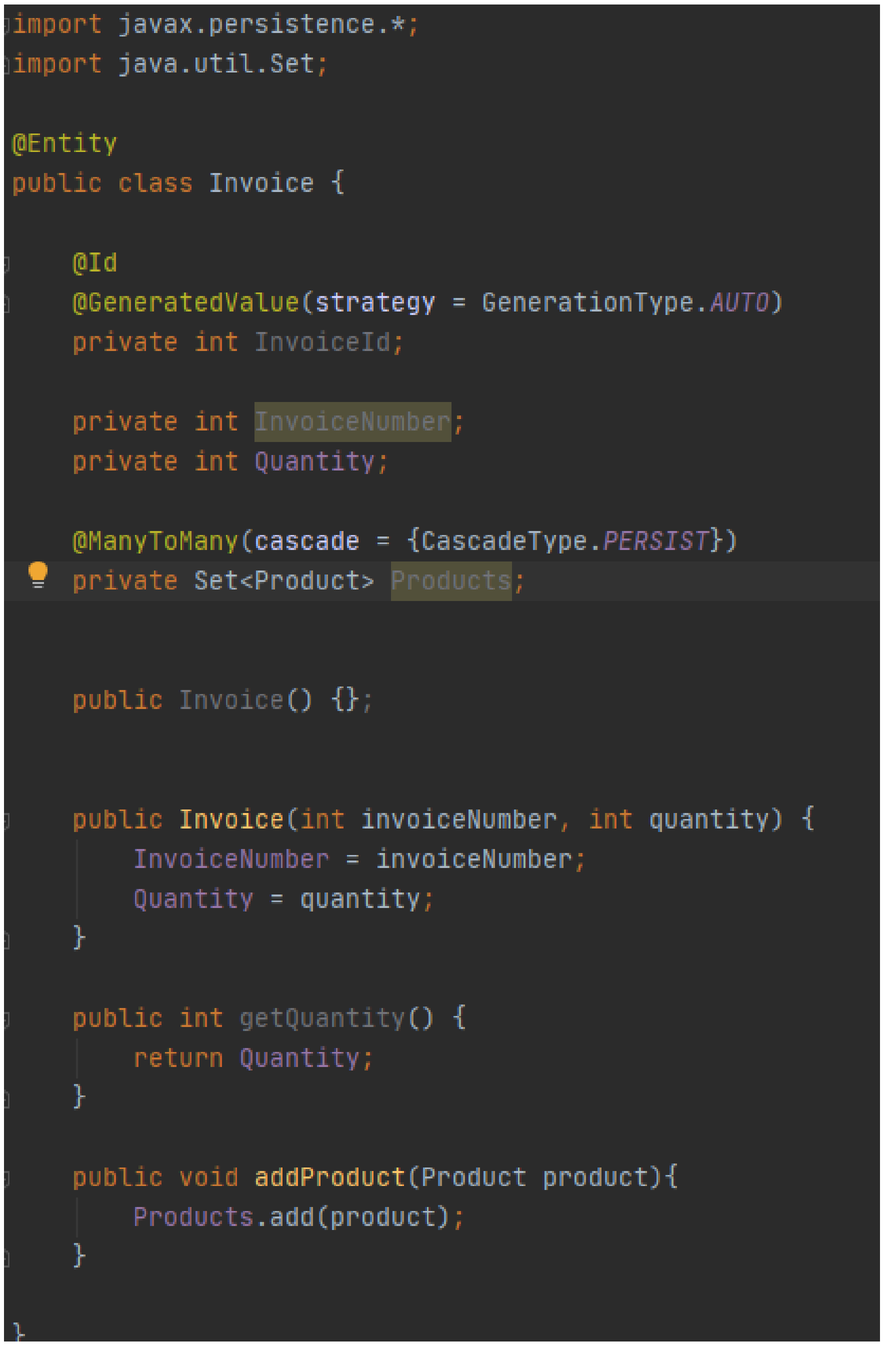


```
select * from SUPPLIER;
select * from PRODUCT;
```

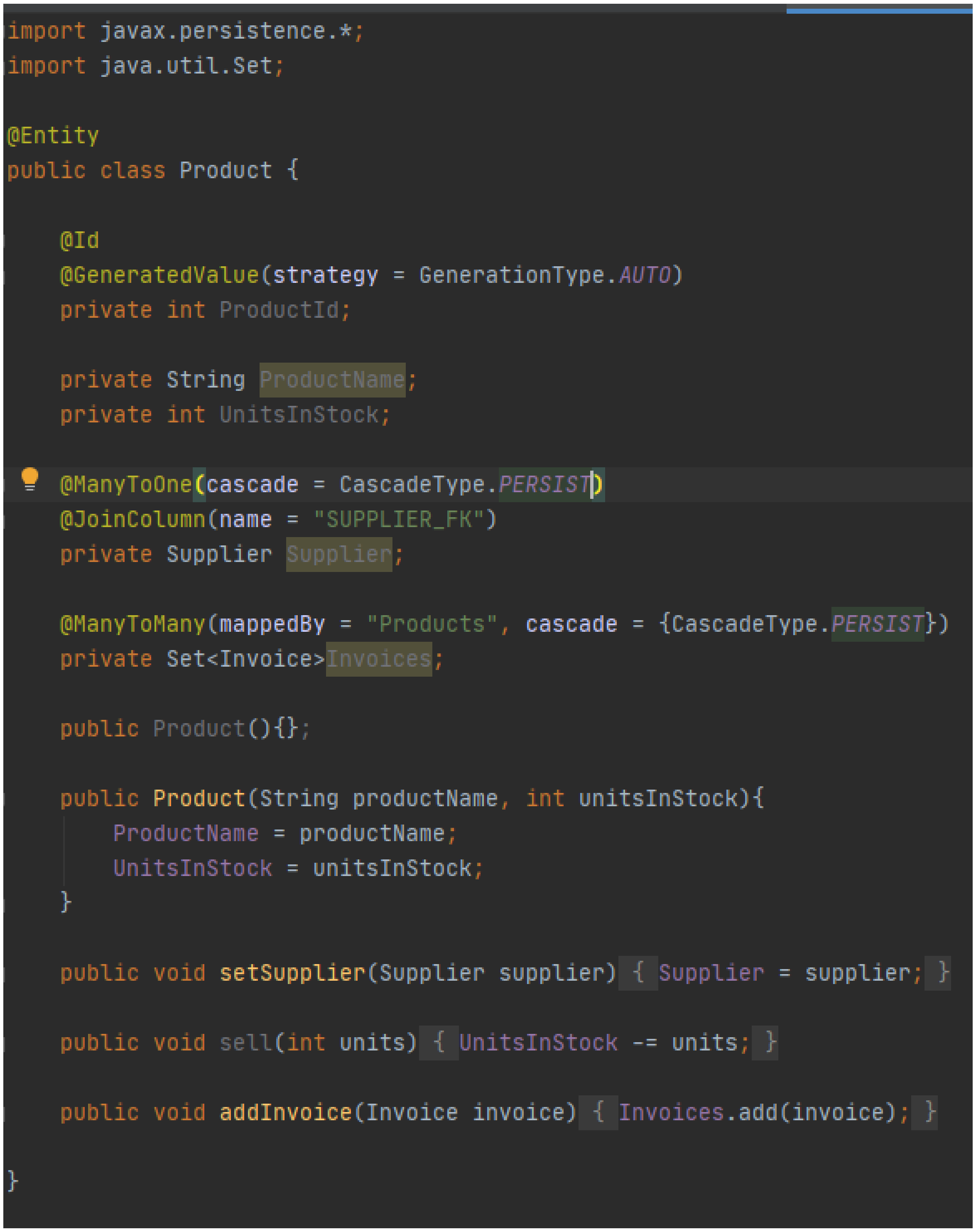
	PRODUCTID	PRODUCTNAME	UNITSINSTOCK	SUPPLIER_FK
1	11	book	5	16
2	12	note	2	16
3	13	cup	8	16
4	14	pencil	4	17
5	15	pen	11	17

XI. Kaskady

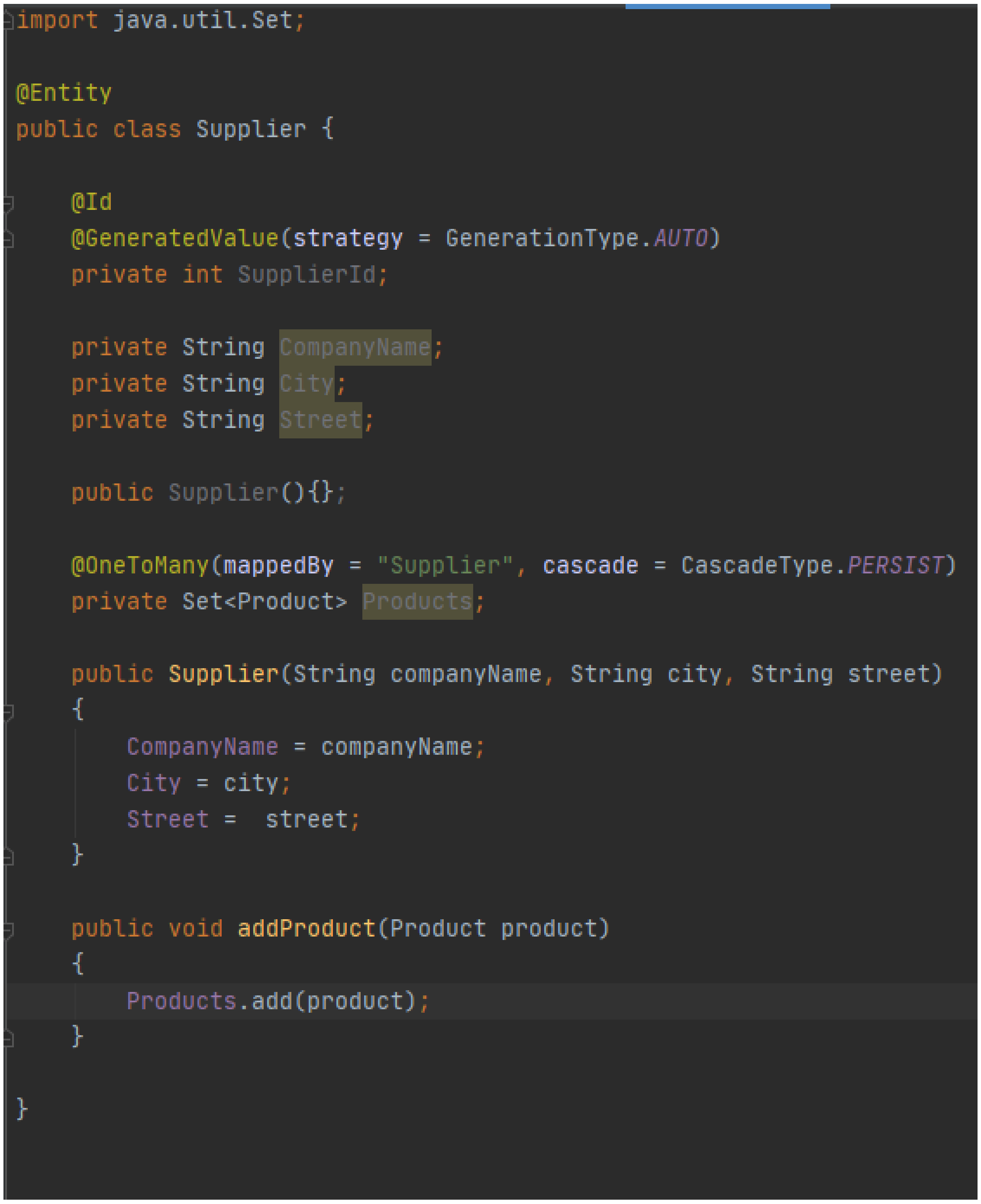
a. Zmodyfikuj model w taki sposób aby było możliwe kaskadowe tworzenie faktur wraz z nowymi produktami, oraz produktów wraz z nową fakturą



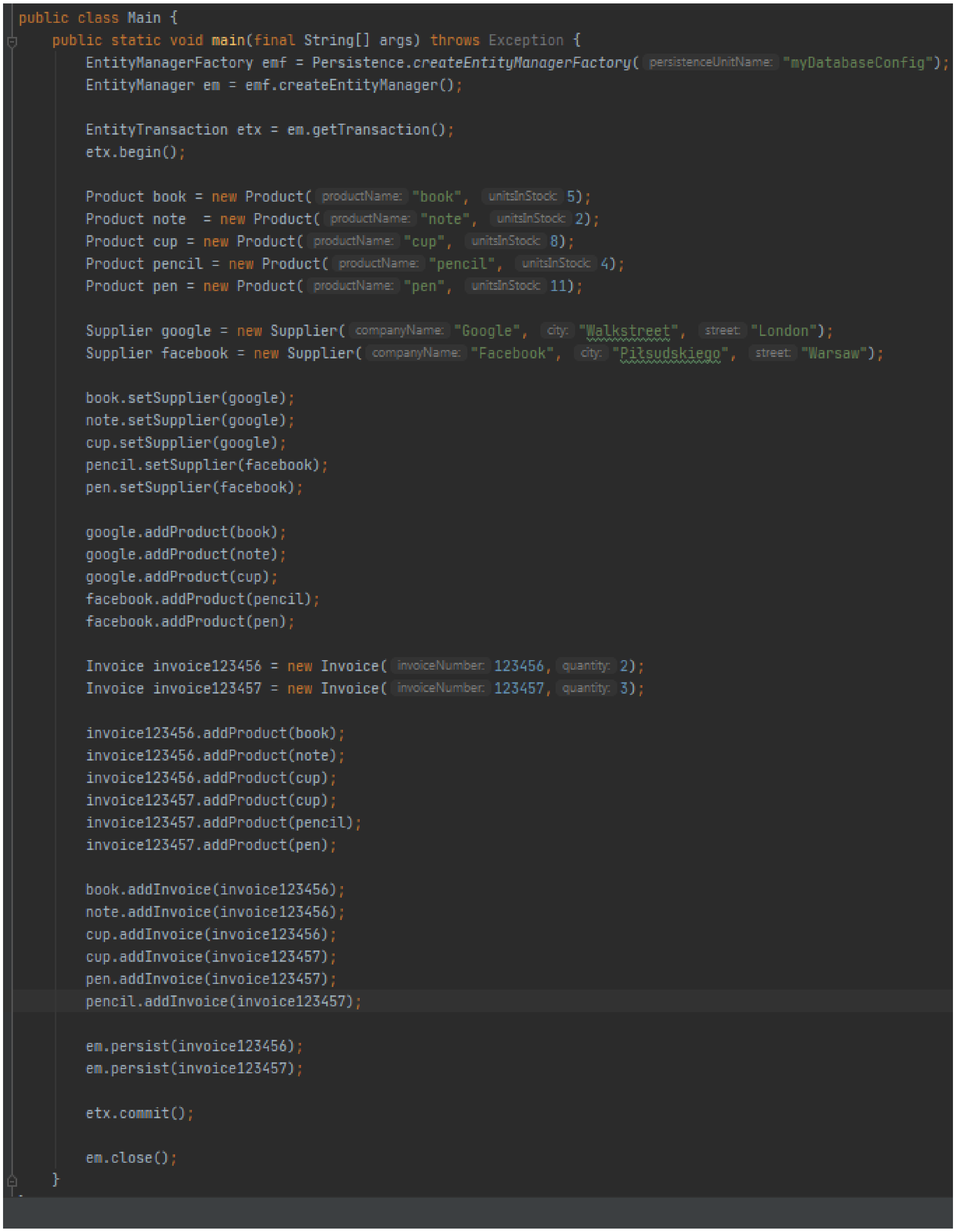
```java
import javax.persistence.*;
import java.util.Set;

@Entity
public class Invoice {

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

    private int InvoiceNumber;
    private int Quantity;

    @ManyToMany(cascade = {CascadeType.PERSIST})
    private Set<Product> Products;


    public Invoice() {};


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

    public int getQuantity() {
        return Quantity;
    }

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

}
```



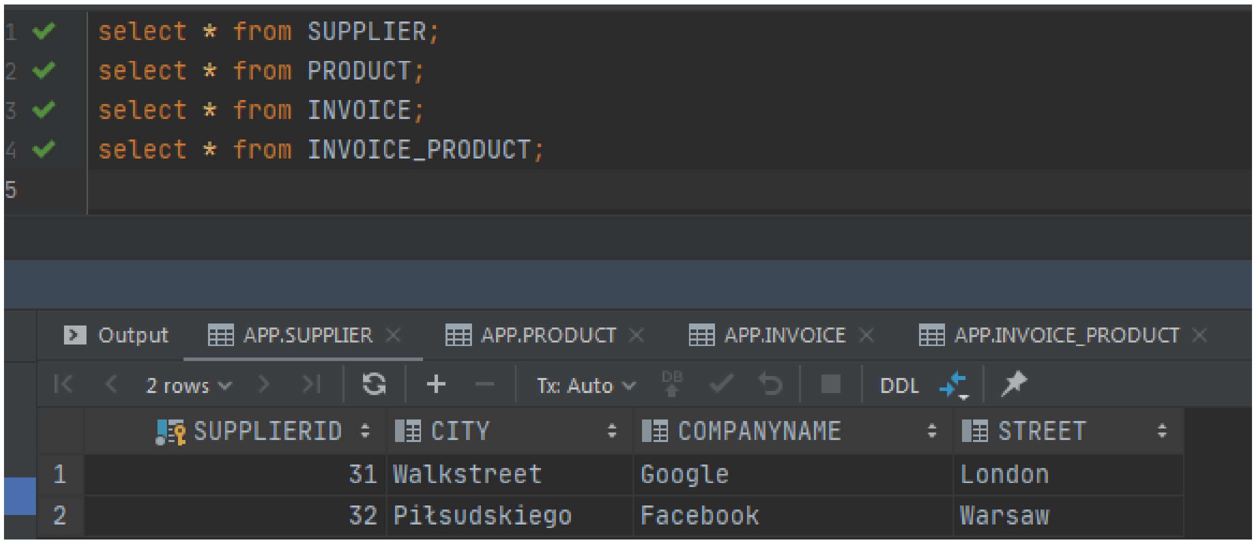
```java
import javax.persistence.*;
import java.util.Set;

@Entity
public class Product {

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

    private String ProductName;
    private int UnitsInStock;

    @ManyToOne(cascade = CascadeType.PERSIST)
    @JoinColumn(name = "SUPPLIER_FK")
    private Supplier Supplier;

    @ManyToMany(mappedBy = "Products", cascade = {CascadeType.PERSIST})
    private Set<Invoice>Invoices;

    public Product(){};

    public Product(String productName, int unitsInStock){
        ProductName = productName;
        UnitsInStock = unitsInStock;
    }

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

    public void sell(int units) { UnitsInStock -= units; }

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

}
```



```java
import java.util.Set;

@Entity
public class Supplier {

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

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

    public Supplier(){};

    @OneToMany(mappedBy = "Supplier", cascade = CascadeType.PERSIST)
    private Set<Product> Products;

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

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

}
```

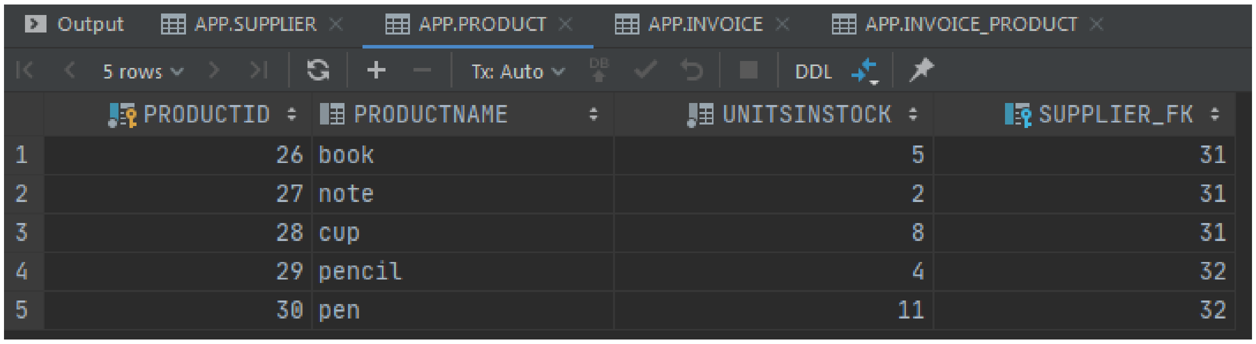


```java
public class Main {
    public static void main(final String[] args) throws Exception {
        EntityManagerFactory emf = Persistence.createEntityManagerFactory( persistenceUnitName: "myDatabaseConfig");
        EntityManager em = emf.createEntityManager();

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

        Product book = new Product( productName: "book",  unitsInStock: 5);
        Product note  = new Product( productName: "note",  unitsInStock: 2);
        Product cup = new Product( productName: "cup",  unitsInStock: 8);
        Product pencil = new Product( productName: "pencil",  unitsInStock: 4);
        Product pen = new Product( productName: "pen",  unitsInStock: 11);

        Supplier google = new Supplier( companyName: "Google",  city: "Walkstreet",  street: "London");
        Supplier facebook = new Supplier( companyName: "Facebook",  city: "Piłsudskiego",  street: "Warsaw");

        book.setSupplier(google);
        note.setSupplier(google);
        cup.setSupplier(google);
        pencil.setSupplier(facebook);
        pen.setSupplier(facebook);

        google.addProduct(book);
        google.addProduct(note);
        google.addProduct(cup);
        facebook.addProduct(pencil);
        facebook.addProduct(pen);

        Invoice invoice123456 = new Invoice( invoiceNumber: 123456, quantity: 2);
        Invoice invoice123457 = new Invoice( invoiceNumber: 123457, quantity: 3);

        invoice123456.addProduct(book);
        invoice123456.addProduct(note);
        invoice123456.addProduct(cup);
        invoice123457.addProduct(cup);
        invoice123457.addProduct(pencil);
        invoice123457.addProduct(pen);

        book.addInvoice(invoice123456);
        note.addInvoice(invoice123456);
        cup.addInvoice(invoice123456);
        cup.addInvoice(invoice123457);
        pen.addInvoice(invoice123457);
        pencil.addInvoice(invoice123457);

        em.persist(invoice123456);
        em.persist(invoice123457);

        etx.commit();

        em.close();
    }
```

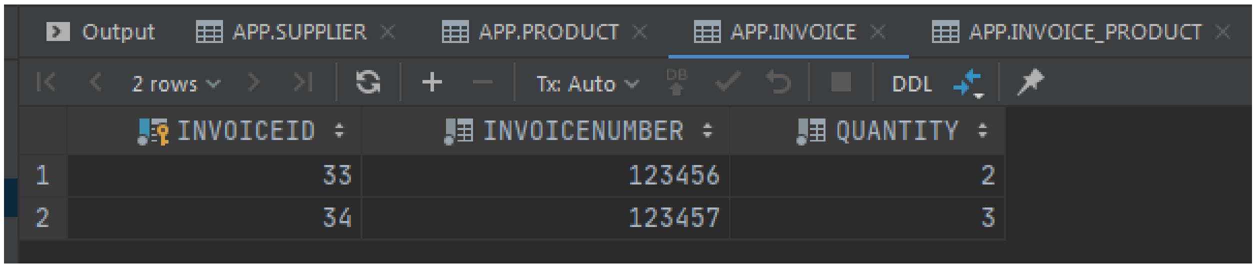


```
select * from SUPPLIER;
select * from PRODUCT;
select * from INVOICE;
select * from INVOICE_PRODUCT;
```

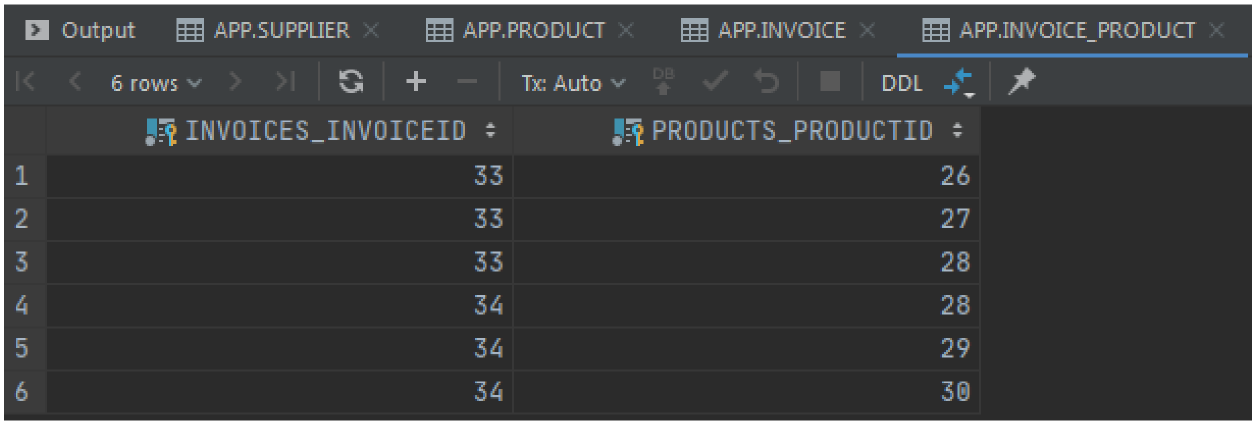
	SUPPLIERID	CITY	COMPANYNAME	STREET
1	31	Walkstreet	Google	London
2	32	Piłsudskiego	Facebook	Warsaw



	PRODUCTID	PRODUCTNAME	UNITSINSTOCK	SUPPLIER_FK
1	26	book	5	31
2	27	note	2	31
3	28	cup	8	31
4	29	pencil	4	32
5	30	pen	11	32



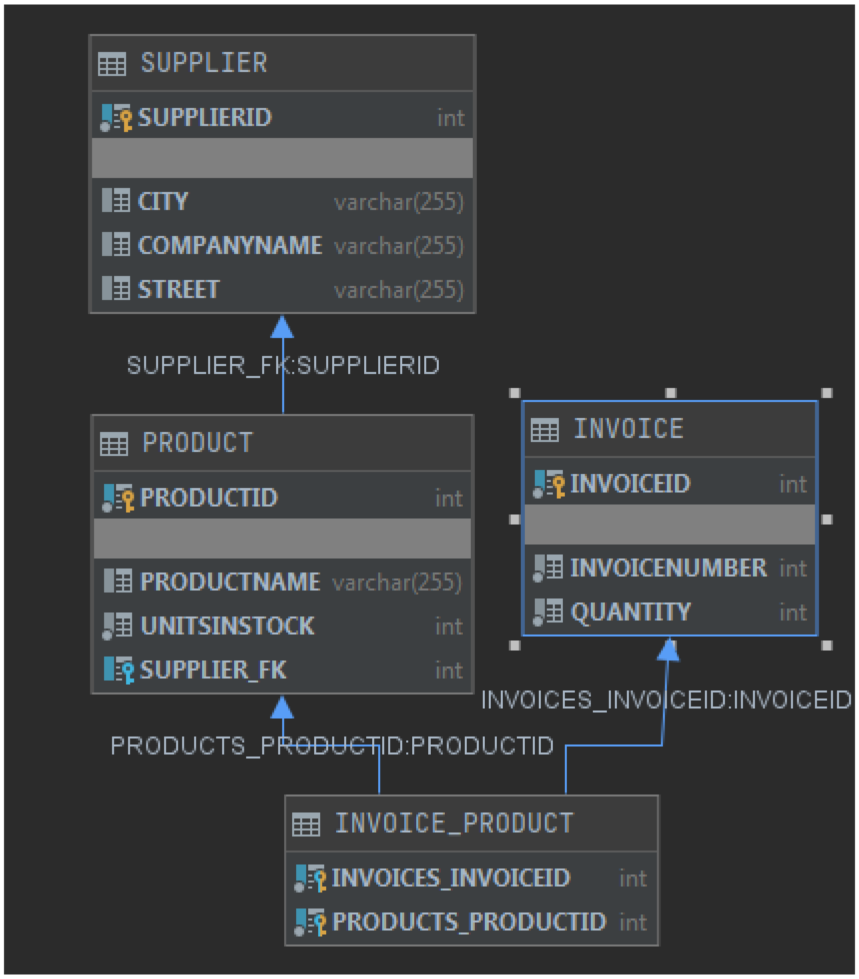
	INVOICEID	INVOICENUMBER	QUANTITY
1	33	123456	2
2	34	123457	3



Output | APP.SUPPLIER | APP.PRODUCT | APP.INVOICE | APP.INVOICE_PRODUCT

6 rows | Tx: Auto | DDL

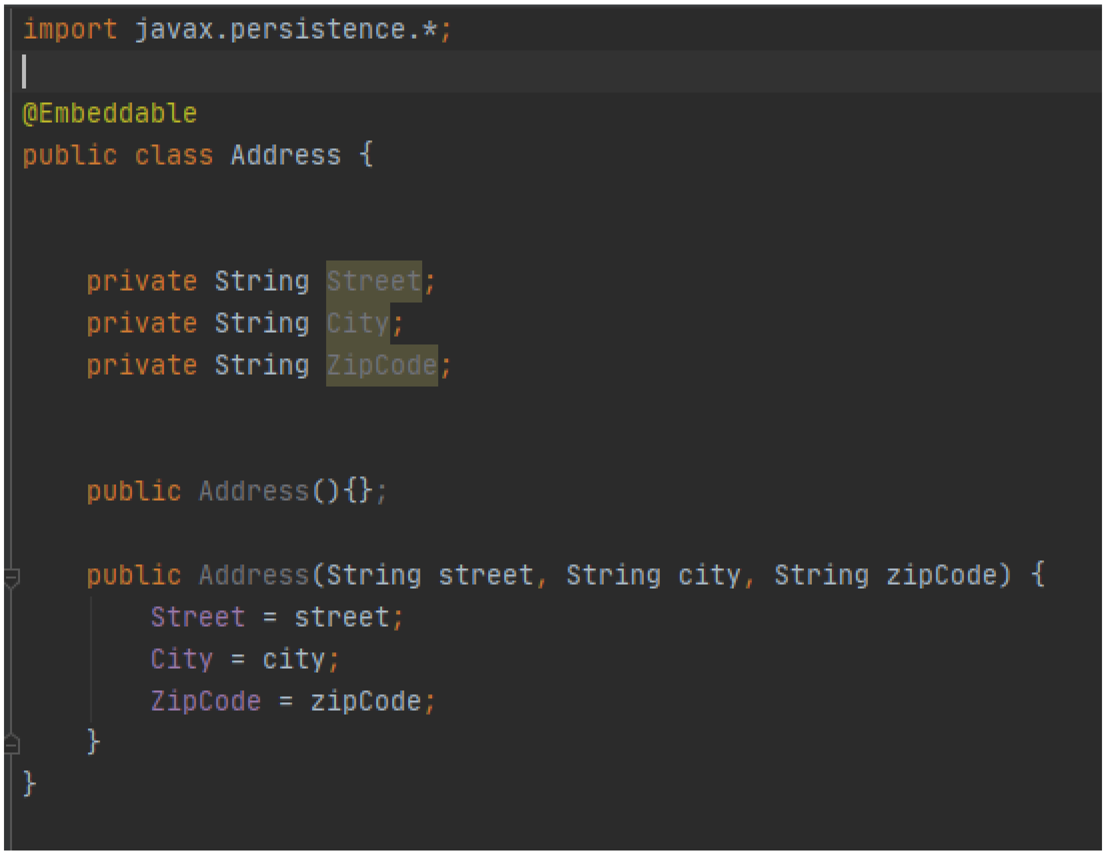
	INVOICES_INVOICEID	PRODUCTS_PRODUCTID
1	33	26
2	33	27
3	33	28
4	34	28
5	34	29
6	34	30



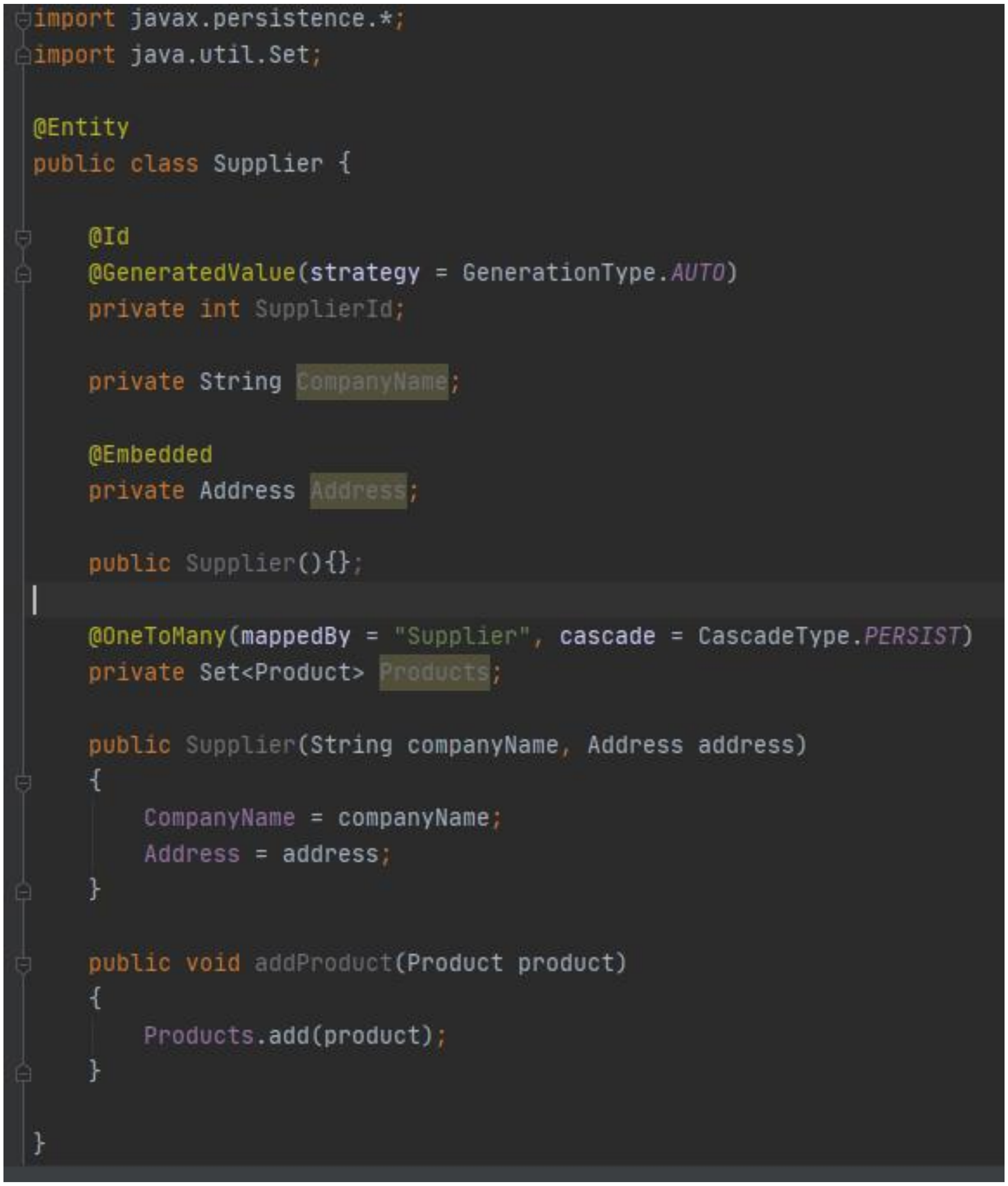
```
SUPPLIER
SUPPLIERID      int
CITY            varchar(255)
COMPANYNAME     varchar(255)
STREET          varchar(255)

PRODUCT
PRODUCTID       int
PRODUCTNAME     varchar(255)
UNITSINSTOCK    int
SUPPLIER_FK     int

INVOICE
INVOICEID       int
INVOICENUMBER   int
QUANTITY        int

INVOICE_PRODUCT
INVOICES_INVOICEID  int
PRODUCTS_PRODUCTID  int
```

XII. Embedded class

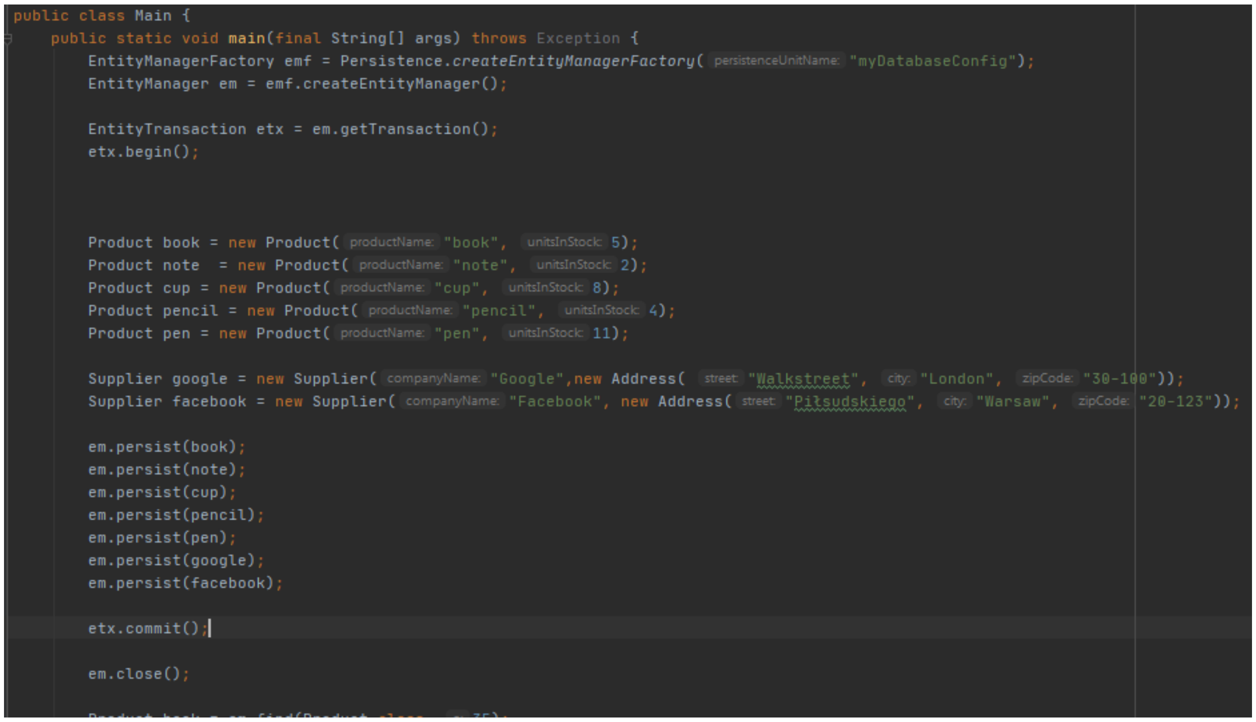
a. Dodaj do modelu klase adres. „Wbuduj” ją do tabeli Dostawców.



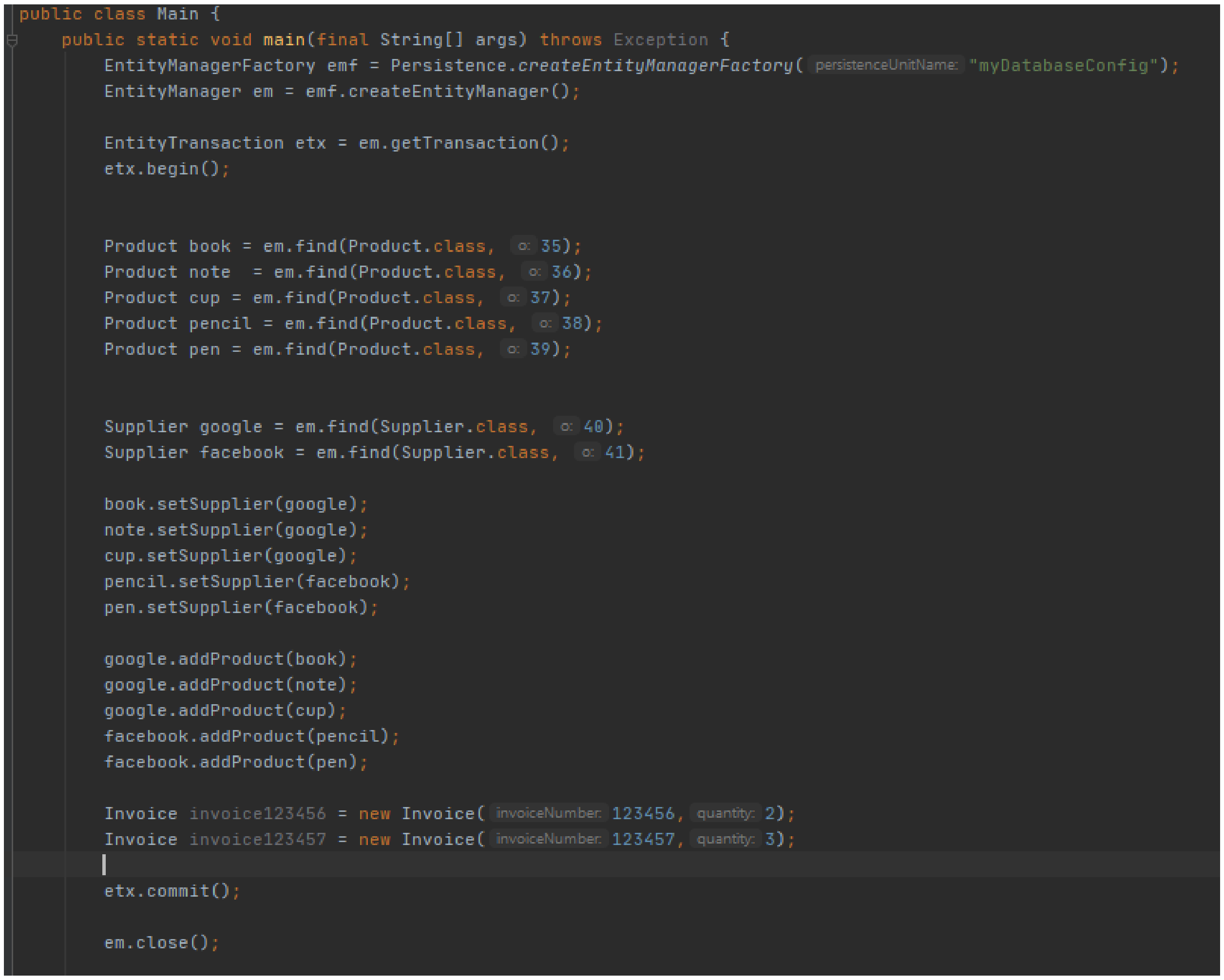
```java
import javax.persistence.*;

@Embeddable
public class Address {


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


    public Address(){};

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



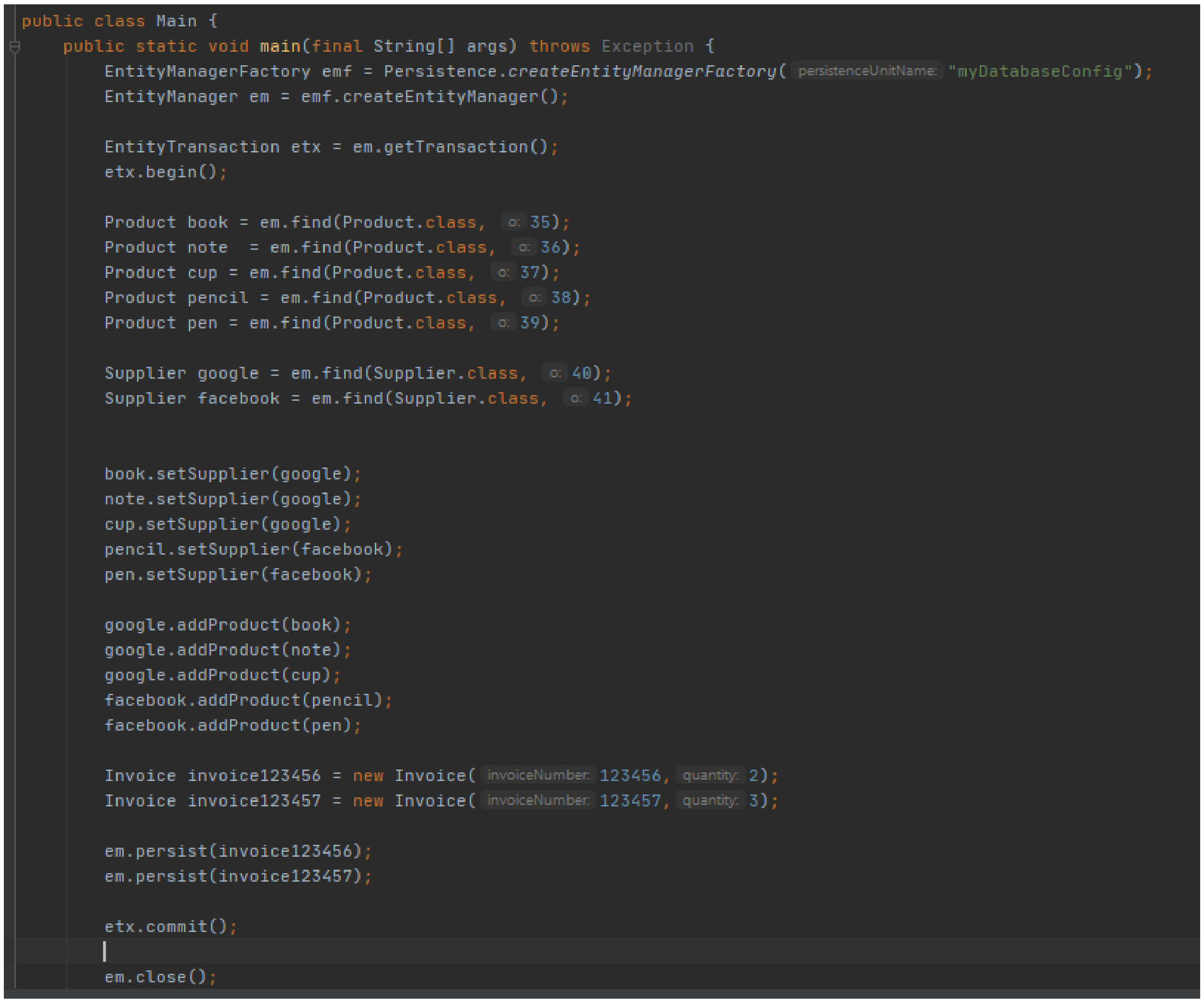
```java
import javax.persistence.*;
import java.util.Set;

@Entity
public class Supplier {

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

    private String CompanyName;

    @Embedded
    private Address Address;

    public Supplier(){};

    @OneToMany(mappedBy = "Supplier", cascade = CascadeType.PERSIST)
    private Set<Product> Products;

    public Supplier(String companyName, Address address)
    {
        CompanyName = companyName;
        Address = address;
    }

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

}
```



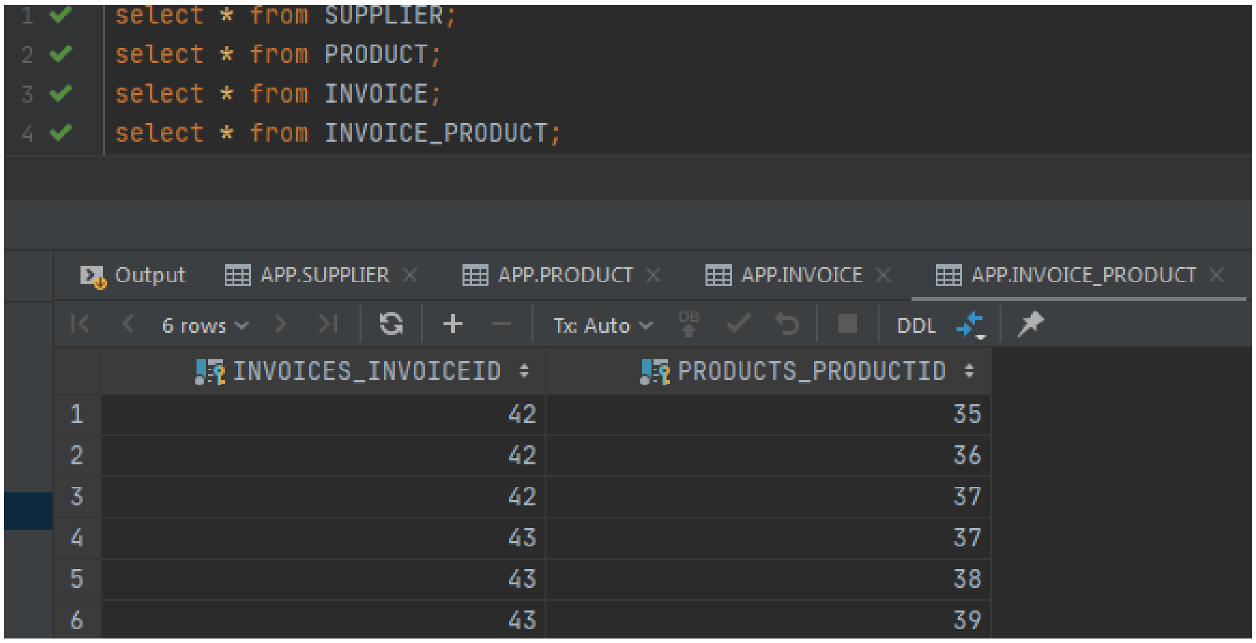
```java
public class Main {
    public static void main(final String[] args) throws Exception {
        EntityManagerFactory emf = Persistence.createEntityManagerFactory( persistenceUnitName: "myDatabaseConfig");
        EntityManager em = emf.createEntityManager();

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



        Product book = new Product( productName: "book",  unitsInStock: 5);
        Product note  = new Product( productName: "note",  unitsInStock: 2);
        Product cup = new Product( productName: "cup",  unitsInStock: 8);
        Product pencil = new Product( productName: "pencil",  unitsInStock: 4);
        Product pen = new Product( productName: "pen",  unitsInStock: 11);

        Supplier google = new Supplier( companyName: "Google",new Address(  street: "Walkstreet",  city: "London",  zipCode: "30-100"));
        Supplier facebook = new Supplier( companyName: "Facebook", new Address( street: "Piłsudskiego",  city: "Warsaw",  zipCode: "20-123"));

        em.persist(book);
        em.persist(note);
        em.persist(cup);
        em.persist(pencil);
        em.persist(pen);
        em.persist(google);
        em.persist(facebook);

        etx.commit();

        em.close();
```



```java
public class Main {
    public static void main(final String[] args) throws Exception {
        EntityManagerFactory emf = Persistence.createEntityManagerFactory( persistenceUnitName: "myDatabaseConfig");
        EntityManager em = emf.createEntityManager();

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


        Product book = em.find(Product.class,  o: 35);
        Product note  = em.find(Product.class,  o: 36);
        Product cup = em.find(Product.class,  o: 37);
        Product pencil = em.find(Product.class,  o: 38);
        Product pen = em.find(Product.class,  o: 39);


        Supplier google = em.find(Supplier.class,  o: 40);
        Supplier facebook = em.find(Supplier.class,  o: 41);

        book.setSupplier(google);
        note.setSupplier(google);
        cup.setSupplier(google);
        pencil.setSupplier(facebook);
        pen.setSupplier(facebook);

        google.addProduct(book);
        google.addProduct(note);
        google.addProduct(cup);
        facebook.addProduct(pencil);
        facebook.addProduct(pen);

        Invoice invoice123456 = new Invoice( invoiceNumber: 123456, quantity: 2);
        Invoice invoice123457 = new Invoice( invoiceNumber: 123457, quantity: 3);

        etx.commit();

        em.close();
```

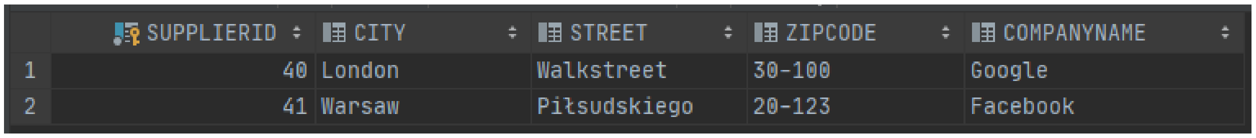


```java
public class Main {
    public static void main(final String[] args) throws Exception {
        EntityManagerFactory emf = Persistence.createEntityManagerFactory( persistenceUnitName: "myDatabaseConfig");
        EntityManager em = emf.createEntityManager();

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

        Product book = em.find(Product.class,  o: 35);
        Product note  = em.find(Product.class,  o: 36);
        Product cup = em.find(Product.class,  o: 37);
        Product pencil = em.find(Product.class,  o: 38);
        Product pen = em.find(Product.class,  o: 39);

        Supplier google = em.find(Supplier.class,  o: 40);
        Supplier facebook = em.find(Supplier.class,  o: 41);


        book.setSupplier(google);
        note.setSupplier(google);
        cup.setSupplier(google);
        pencil.setSupplier(facebook);
        pen.setSupplier(facebook);

        google.addProduct(book);
        google.addProduct(note);
        google.addProduct(cup);
        facebook.addProduct(pencil);
        facebook.addProduct(pen);

        Invoice invoice123456 = new Invoice( invoiceNumber: 123456,  quantity: 2);
        Invoice invoice123457 = new Invoice( invoiceNumber: 123457,  quantity: 3);

        em.persist(invoice123456);
        em.persist(invoice123457);

        etx.commit();

        em.close();
```

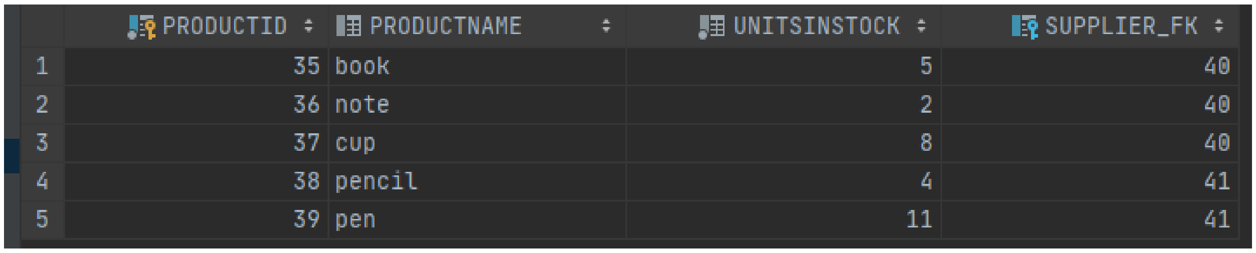


```sql
select * from SUPPLIER;
select * from PRODUCT;
select * from INVOICE;
select * from INVOICE_PRODUCT;
```

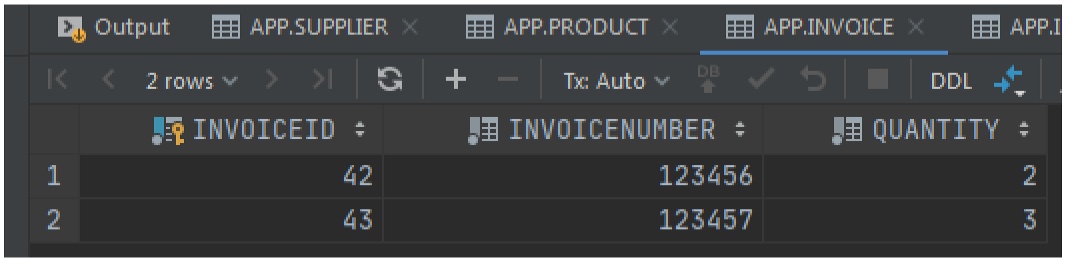
	INVOICES_INVOICEID	PRODUCTS_PRODUCTID
1	42	35
2	42	36
3	42	37
4	43	37
5	43	38
6	43	39



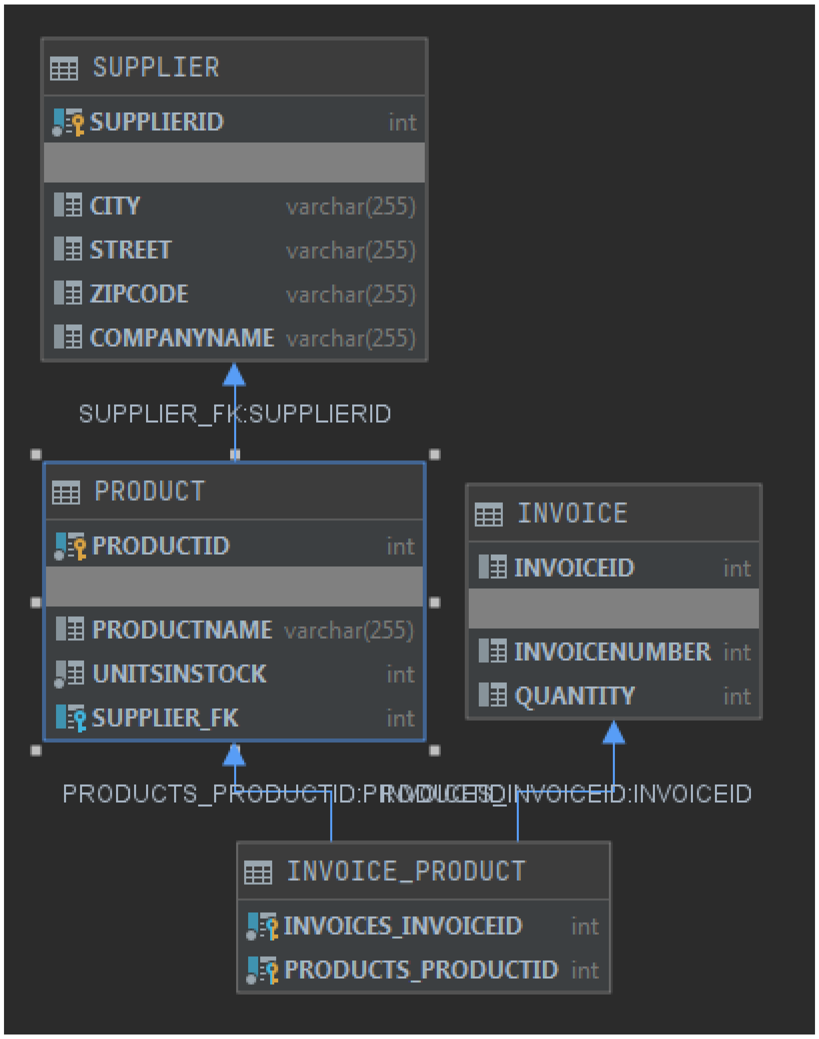
	SUPPLIERID	CITY	STREET	ZIPCODE	COMPANYNAME
1	40	London	Walkstreet	30-100	Google
2	41	Warsaw	Piłsudskiego	20-123	Facebook



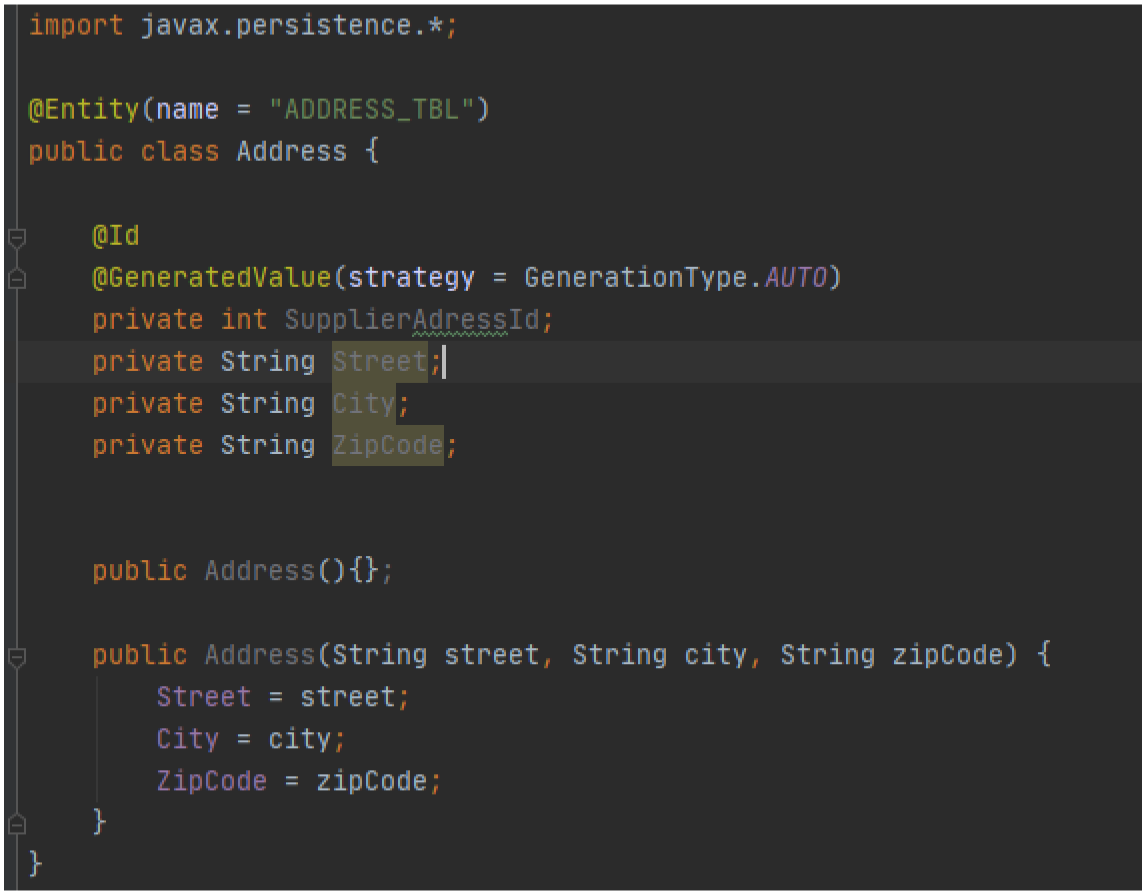
	PRODUCTID	PRODUCTNAME	UNITSINSTOCK	SUPPLIER_FK
1	35	book	5	40
2	36	note	2	40
3	37	cup	8	40
4	38	pencil	4	41
5	39	pen	11	41



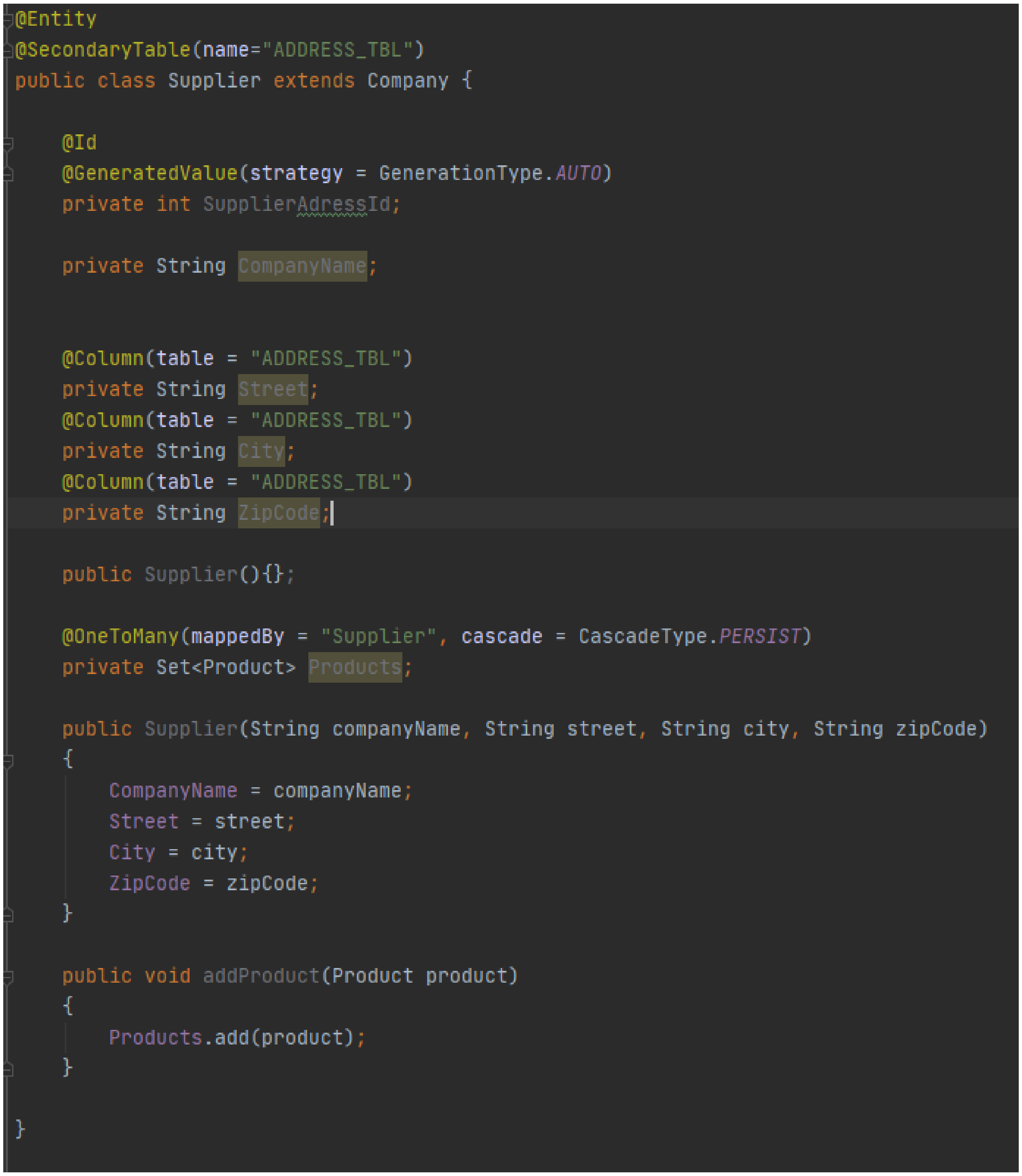
	INVOICEID	INVOICENUMBER	QUANTITY
1	42	123456	2
2	43	123457	3


SUPPLIER
SUPPLIERID int
CITY varchar(255)
STREET varchar(255)
ZIPCODE varchar(255)
COMPANYNAME varchar(255)
SUPPLIER_FK:SUPPLIERID
PRODUCT
PRODUCTID int
PRODUCTNAME varchar(255)
UNITSINSTOCK int
SUPPLIER_FK int
INVOICE
INVOICEID int
INVOICENUMBER int
QUANTITY int
PRODUCTS_PRODUCTID:PRODUCTID
INVOICES_INVOICEID:INVOICEID
INVOICE_PRODUCT
INVOICES_INVOICEID int
PRODUCTS_PRODUCTID int

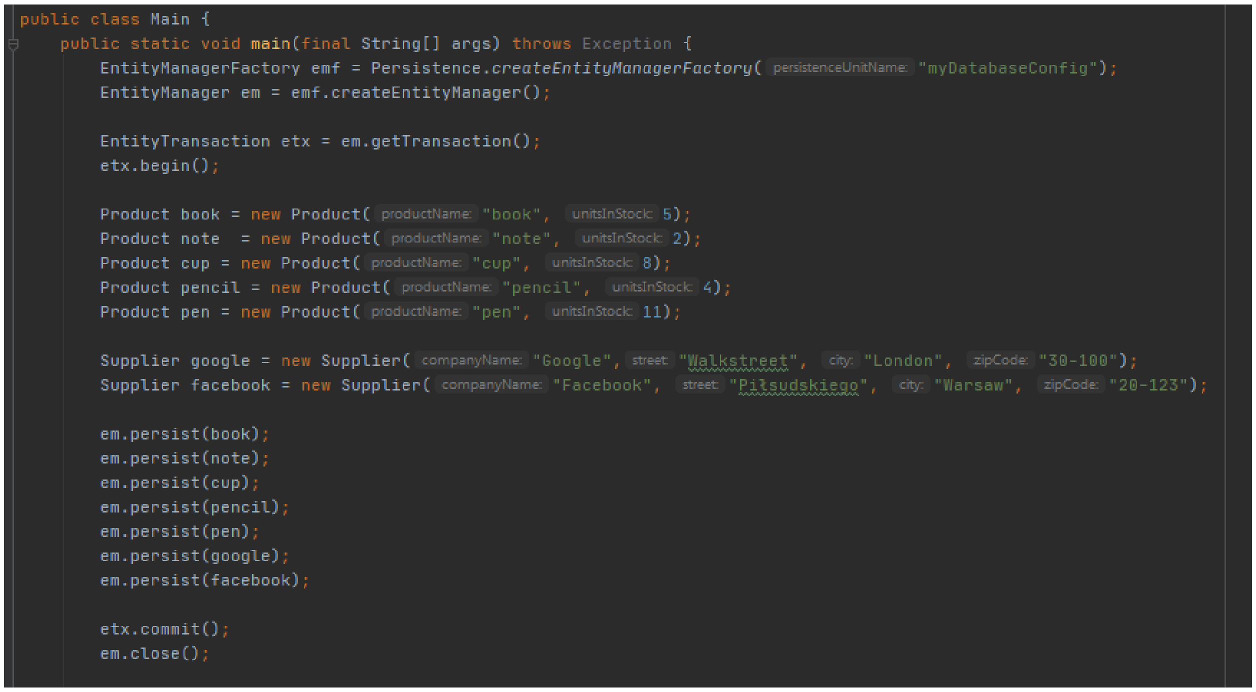
c. Zmodyfikuj model w taki sposób, że dane adresowe znajdują się w klasie dostawców. Zmapuj to do dwóch osobnych tabel.



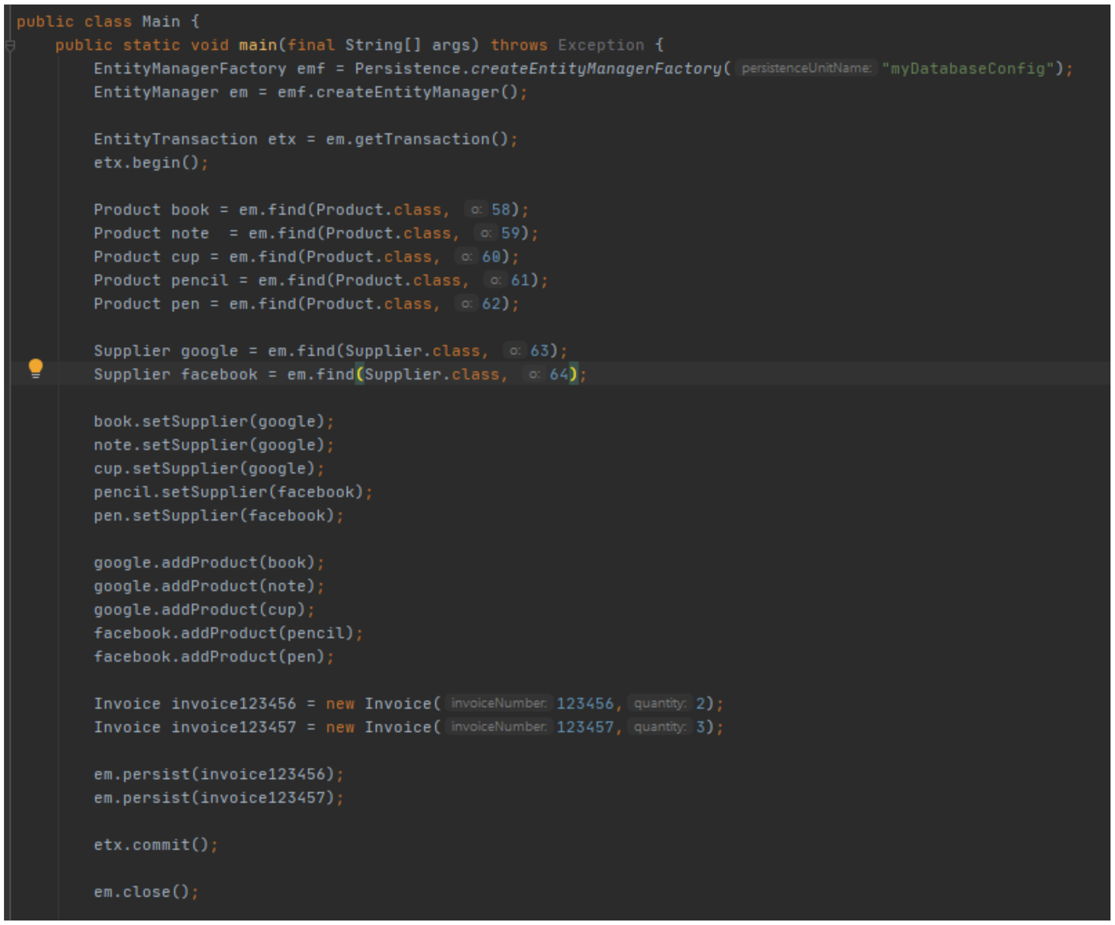
```java
import javax.persistence.*;

@Entity(name = "ADDRESS_TBL")
public class Address {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int SupplierAdressId;
    private String Street;
    private String City;
    private String ZipCode;


    public Address(){};

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



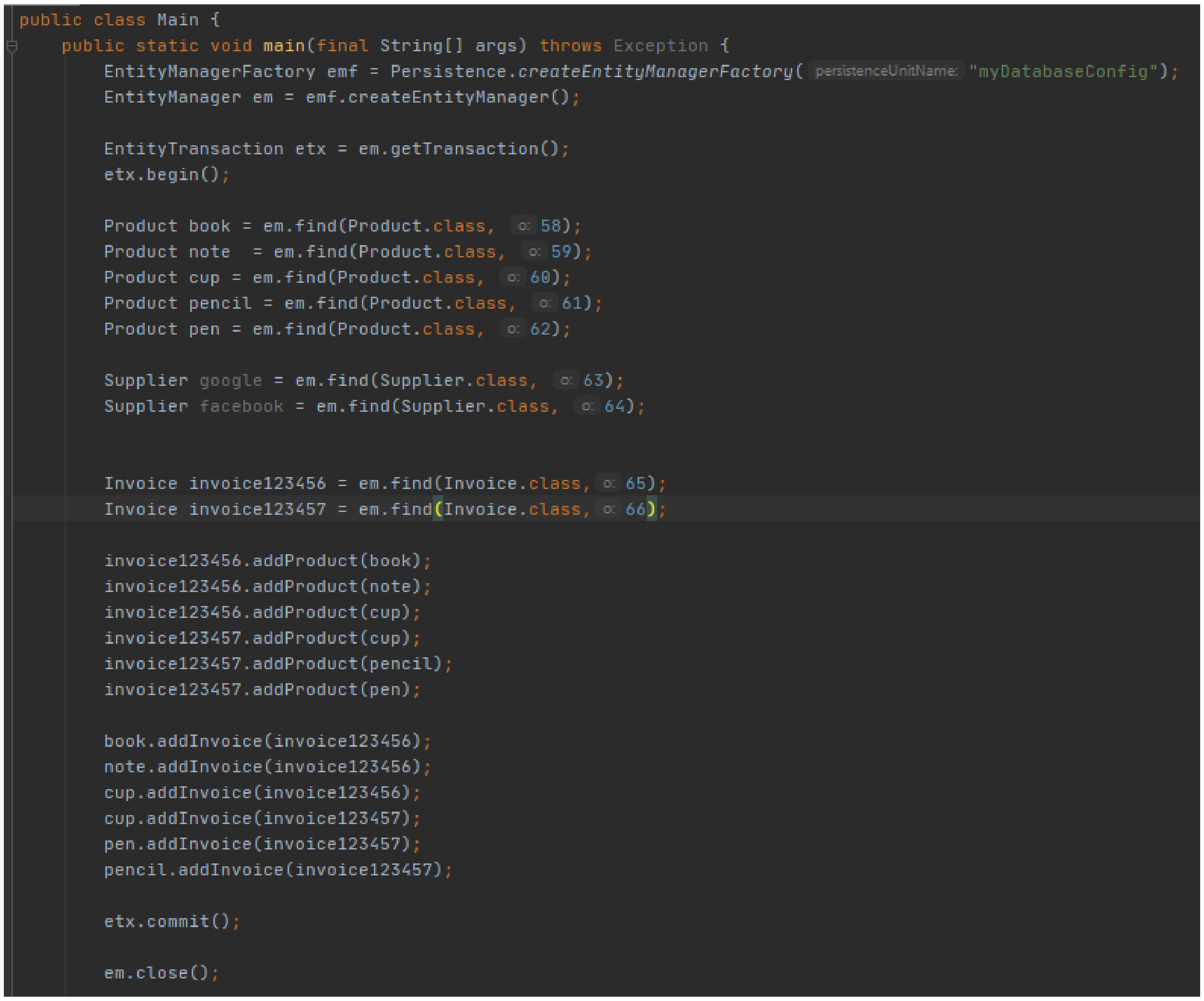
```java
@Entity
@SecondaryTable(name="ADDRESS_TBL")
public class Supplier extends Company {

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

    private String CompanyName;


    @Column(table = "ADDRESS_TBL")
    private String Street;
    @Column(table = "ADDRESS_TBL")
    private String City;
    @Column(table = "ADDRESS_TBL")
    private String ZipCode;

    public Supplier(){};

    @OneToMany(mappedBy = "Supplier", cascade = CascadeType.PERSIST)
    private Set<Product> Products;

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

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

}
```



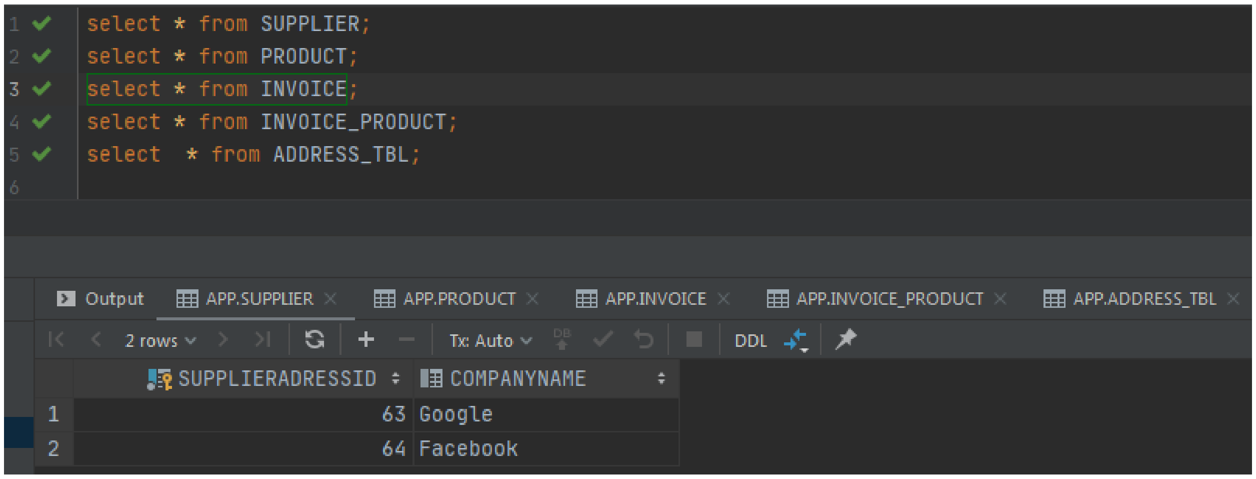
```java
public class Main {
    public static void main(final String[] args) throws Exception {
        EntityManagerFactory emf = Persistence.createEntityManagerFactory( persistenceUnitName: "myDatabaseConfig");
        EntityManager em = emf.createEntityManager();

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

        Product book = new Product( productName: "book",  unitsInStock: 5);
        Product note  = new Product( productName: "note",  unitsInStock: 2);
        Product cup = new Product( productName: "cup",  unitsInStock: 8);
        Product pencil = new Product( productName: "pencil",  unitsInStock: 4);
        Product pen = new Product( productName: "pen",  unitsInStock: 11);

        Supplier google = new Supplier( companyName: "Google", street: "Walkstreet",  city: "London",  zipCode: "30-100");
        Supplier facebook = new Supplier( companyName: "Facebook",  street: "Piłsudskiego",  city: "Warsaw",  zipCode: "20-123");

        em.persist(book);
        em.persist(note);
        em.persist(cup);
        em.persist(pencil);
        em.persist(pen);
        em.persist(google);
        em.persist(facebook);

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



```java
public class Main {
    public static void main(final String[] args) throws Exception {
        EntityManagerFactory emf = Persistence.createEntityManagerFactory( persistenceUnitName: "myDatabaseConfig");
        EntityManager em = emf.createEntityManager();

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

        Product book = em.find(Product.class,  o: 58);
        Product note  = em.find(Product.class,  o: 59);
        Product cup = em.find(Product.class,  o: 60);
        Product pencil = em.find(Product.class,  o: 61);
        Product pen = em.find(Product.class,  o: 62);

        Supplier google = em.find(Supplier.class,  o: 63);
        Supplier facebook = em.find(Supplier.class,  o: 64);

        book.setSupplier(google);
        note.setSupplier(google);
        cup.setSupplier(google);
        pencil.setSupplier(facebook);
        pen.setSupplier(facebook);

        google.addProduct(book);
        google.addProduct(note);
        google.addProduct(cup);
        facebook.addProduct(pencil);
        facebook.addProduct(pen);

        Invoice invoice123456 = new Invoice( invoiceNumber: 123456, quantity: 2);
        Invoice invoice123457 = new Invoice( invoiceNumber: 123457, quantity: 3);

        em.persist(invoice123456);
        em.persist(invoice123457);

        etx.commit();

        em.close();
```

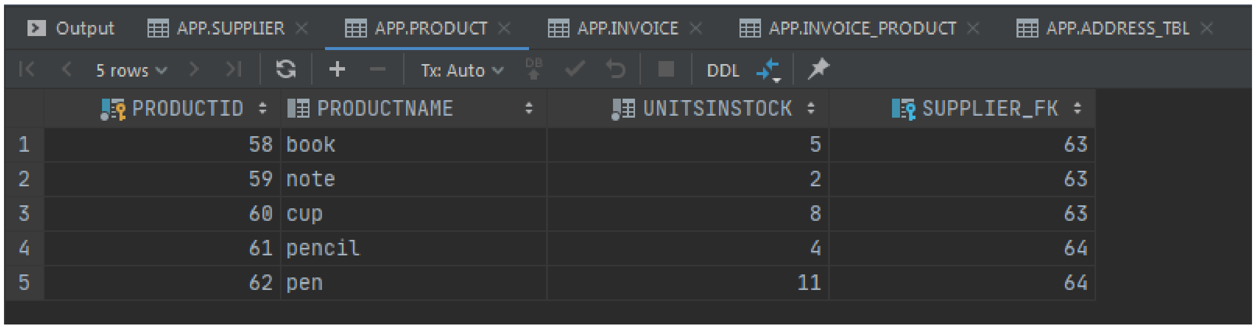


```java
public class Main {
    public static void main(final String[] args) throws Exception {
        EntityManagerFactory emf = Persistence.createEntityManagerFactory( persistenceUnitName: "myDatabaseConfig");
        EntityManager em = emf.createEntityManager();

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

        Product book = em.find(Product.class,  o: 58);
        Product note  = em.find(Product.class,  o: 59);
        Product cup = em.find(Product.class,  o: 60);
        Product pencil = em.find(Product.class,  o: 61);
        Product pen = em.find(Product.class,  o: 62);

        Supplier google = em.find(Supplier.class,  o: 63);
        Supplier facebook = em.find(Supplier.class,  o: 64);


        Invoice invoice123456 = em.find(Invoice.class, o: 65);
        Invoice invoice123457 = em.find(Invoice.class, o: 66);

        invoice123456.addProduct(book);
        invoice123456.addProduct(note);
        invoice123456.addProduct(cup);
        invoice123457.addProduct(cup);
        invoice123457.addProduct(pencil);
        invoice123457.addProduct(pen);

        book.addInvoice(invoice123456);
        note.addInvoice(invoice123456);
        cup.addInvoice(invoice123456);
        cup.addInvoice(invoice123457);
        pen.addInvoice(invoice123457);
        pencil.addInvoice(invoice123457);

        etx.commit();

        em.close();
```

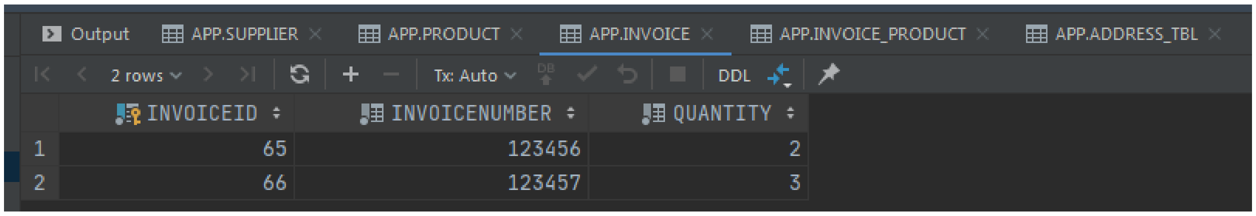


```sql
select * from SUPPLIER;
select * from PRODUCT;
select * from INVOICE;
select * from INVOICE_PRODUCT;
select  * from ADDRESS_TBL;
```

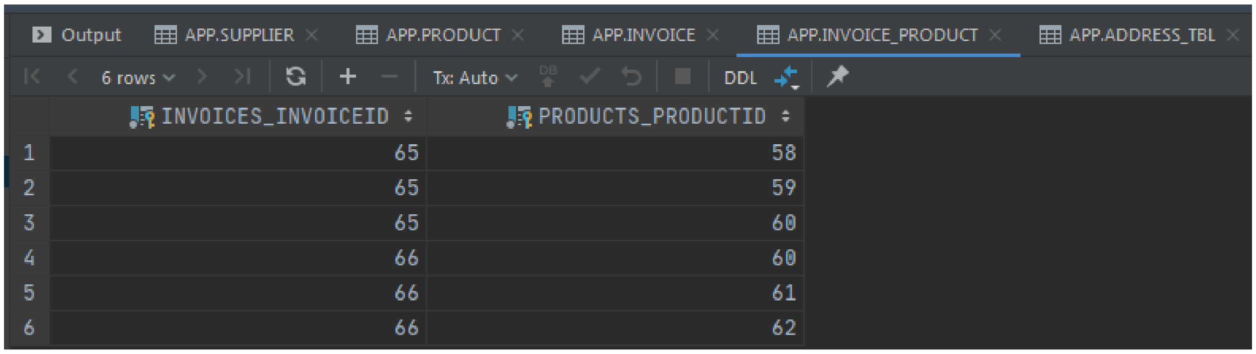
	SUPPLIERADRESSID	COMPANYNAME
1	63	Google
2	64	Facebook



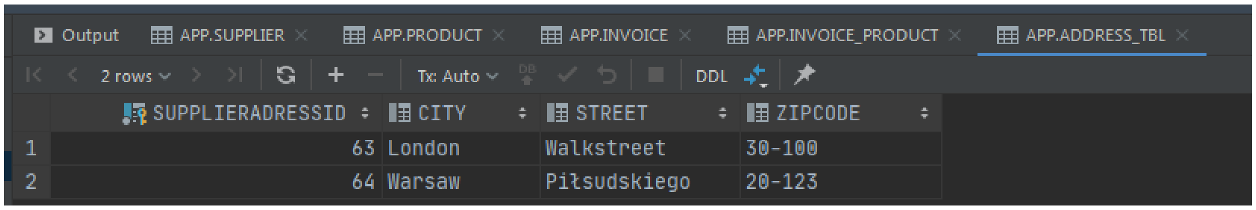
	PRODUCTID	PRODUCTNAME	UNITSINSTOCK	SUPPLIER_FK
1	58	book	5	63
2	59	note	2	63
3	60	cup	8	63
4	61	pencil	4	64
5	62	pen	11	64



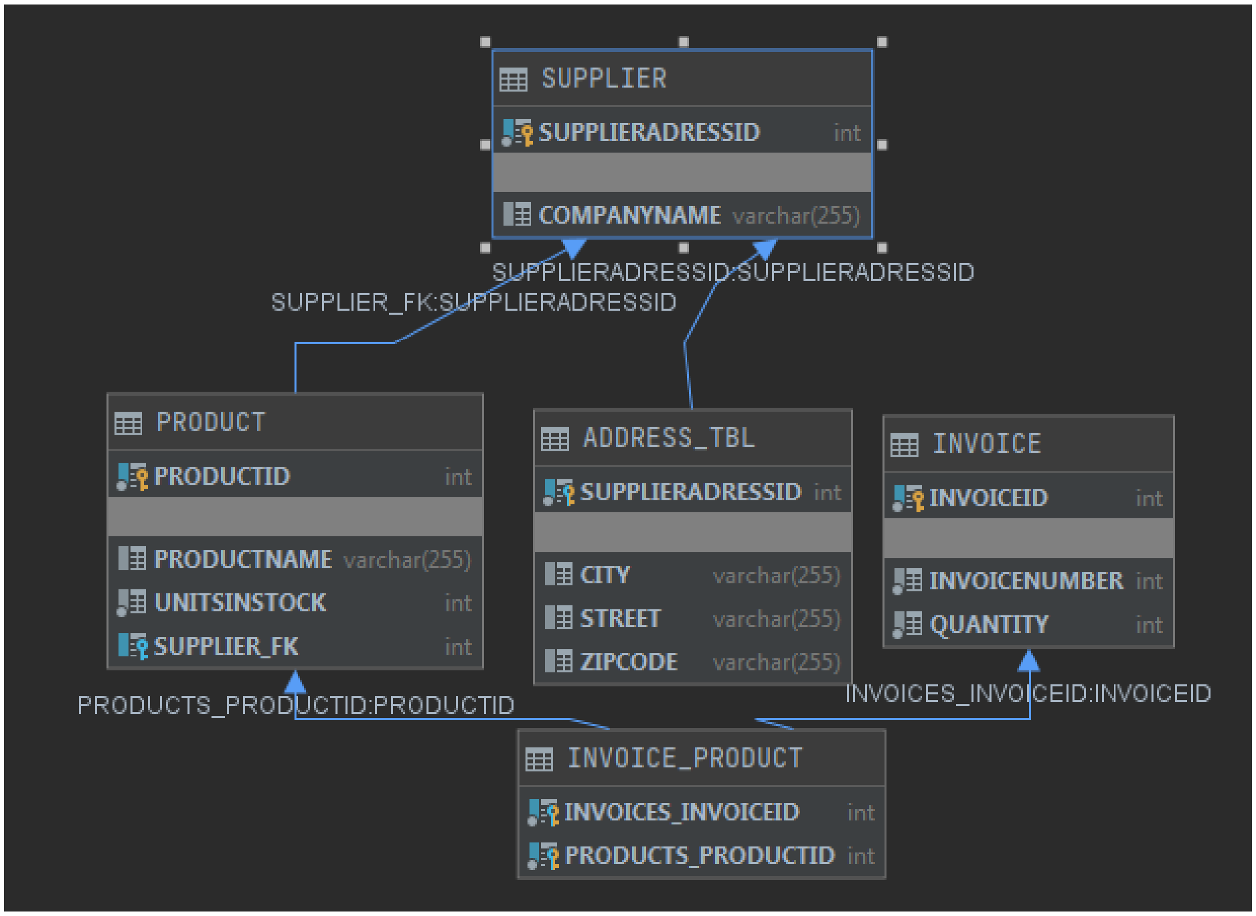
	INVOICEID	INVOICENUMBER	QUANTITY
1	65	123456	2
2	66	123457	3



	INVOICES_INVOICEID	PRODUCTS_PRODUCTID
1	65	58
2	65	59
3	65	60
4	66	60
5	66	61
6	66	62

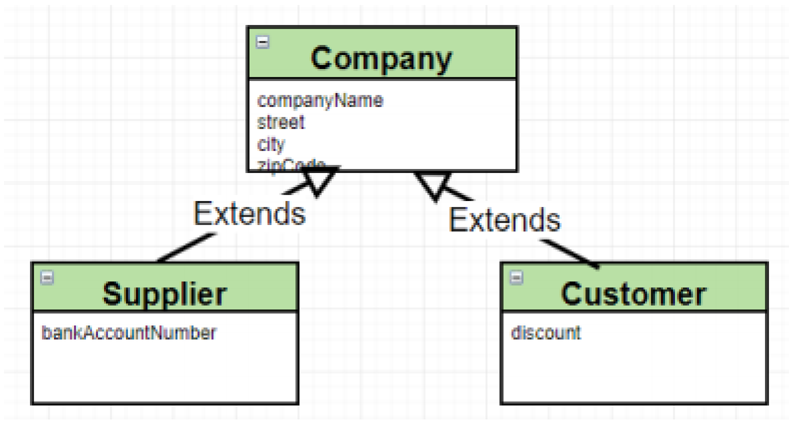


	SUPPLIERADRESSID	CITY	STREET	ZIPCODE
1	63	London	Walkstreet	30-100
2	64	Warsaw	Piłsudskiego	20-123

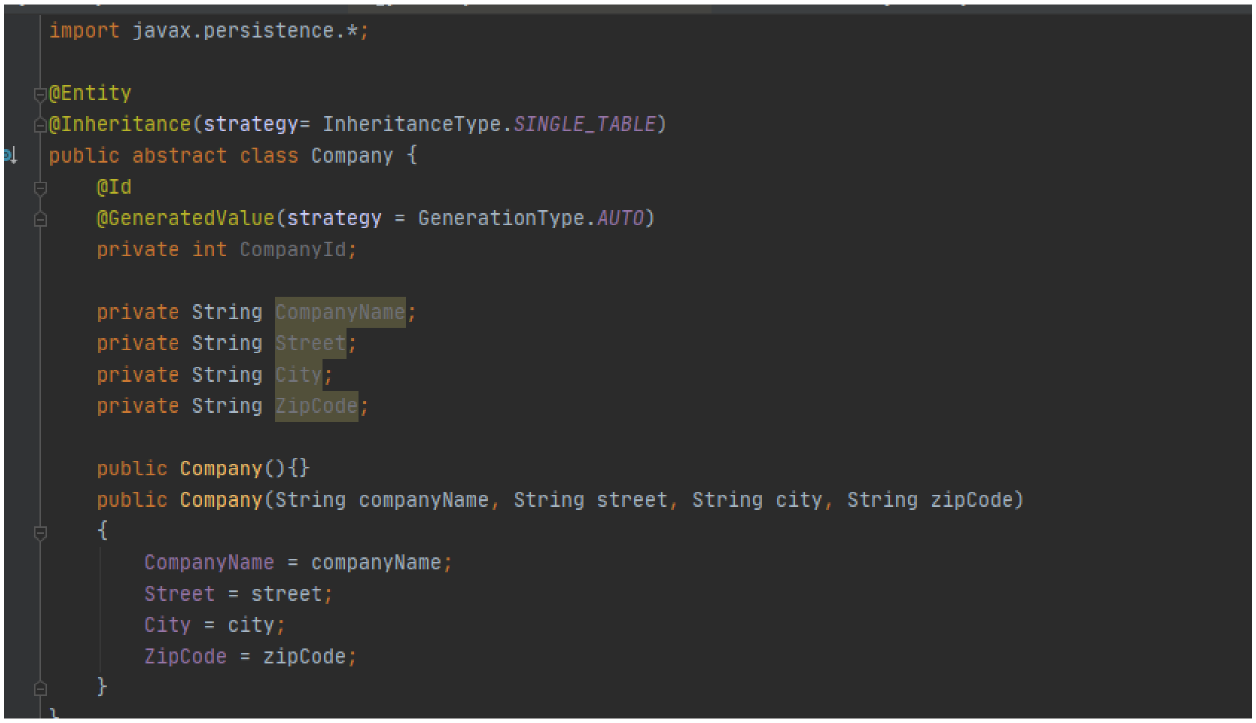


XIII. Dziedziczenie

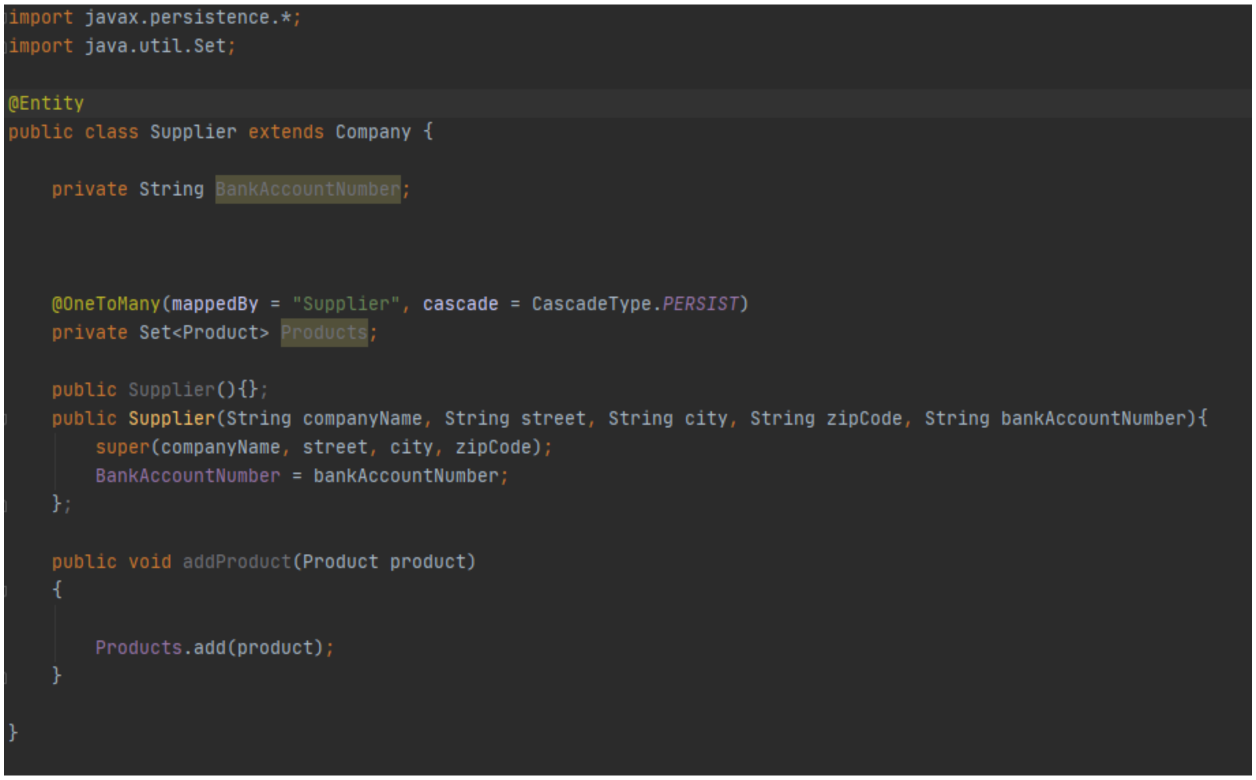
a. Wprowadź do modelu następującą hierarchie:



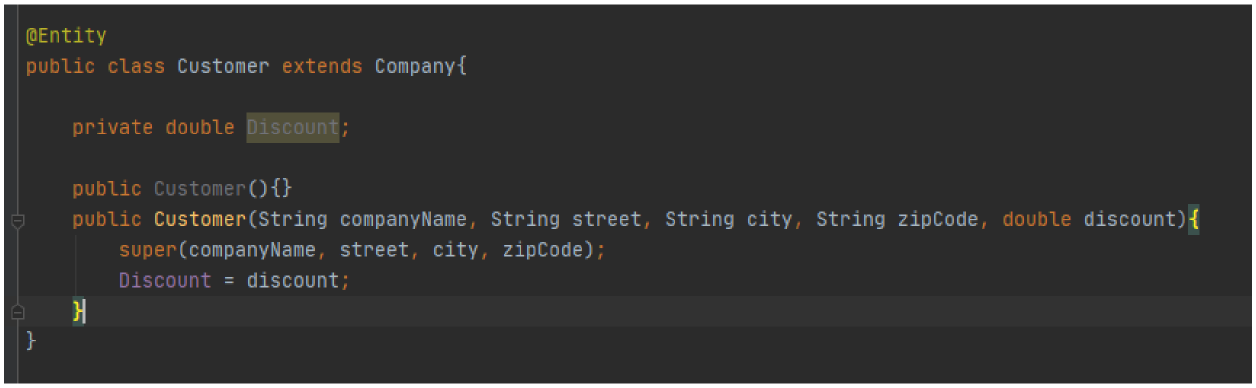
Jedna tabela na całą hierarchię



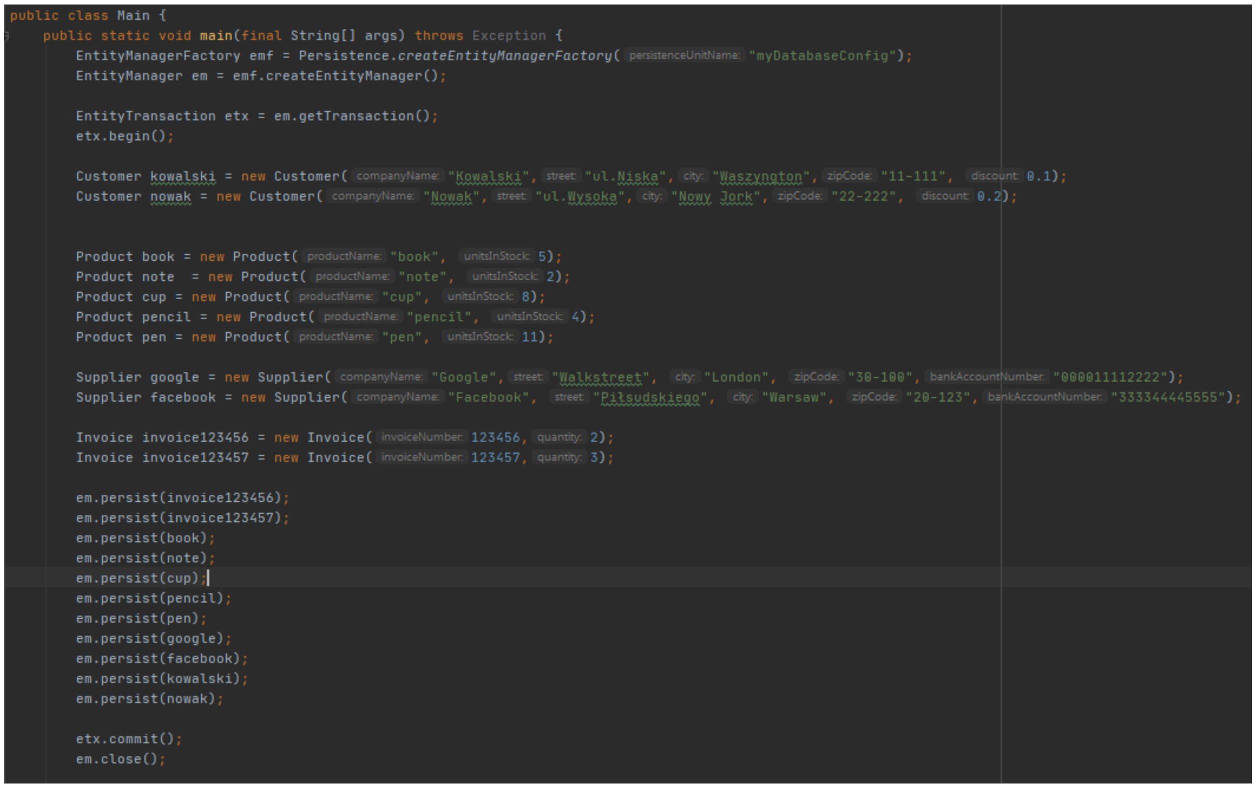
```java
import javax.persistence.*;

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



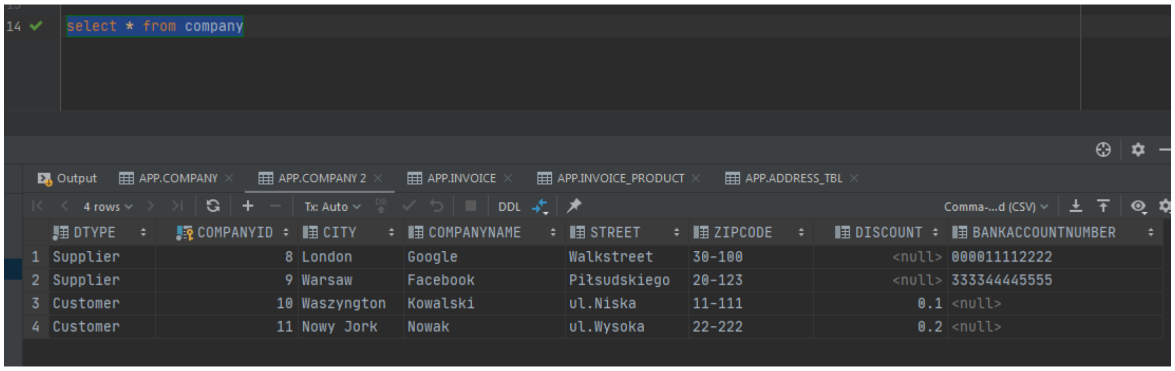
```java
import javax.persistence.*;
import java.util.Set;

@Entity
public class Supplier extends Company {

    private String BankAccountNumber;



    @OneToMany(mappedBy = "Supplier", cascade = CascadeType.PERSIST)
    private Set<Product> Products;

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

    public void addProduct(Product product)
    {

        Products.add(product);
    }

}
```



```java
@Entity
public class Customer extends Company{

    private double Discount;

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

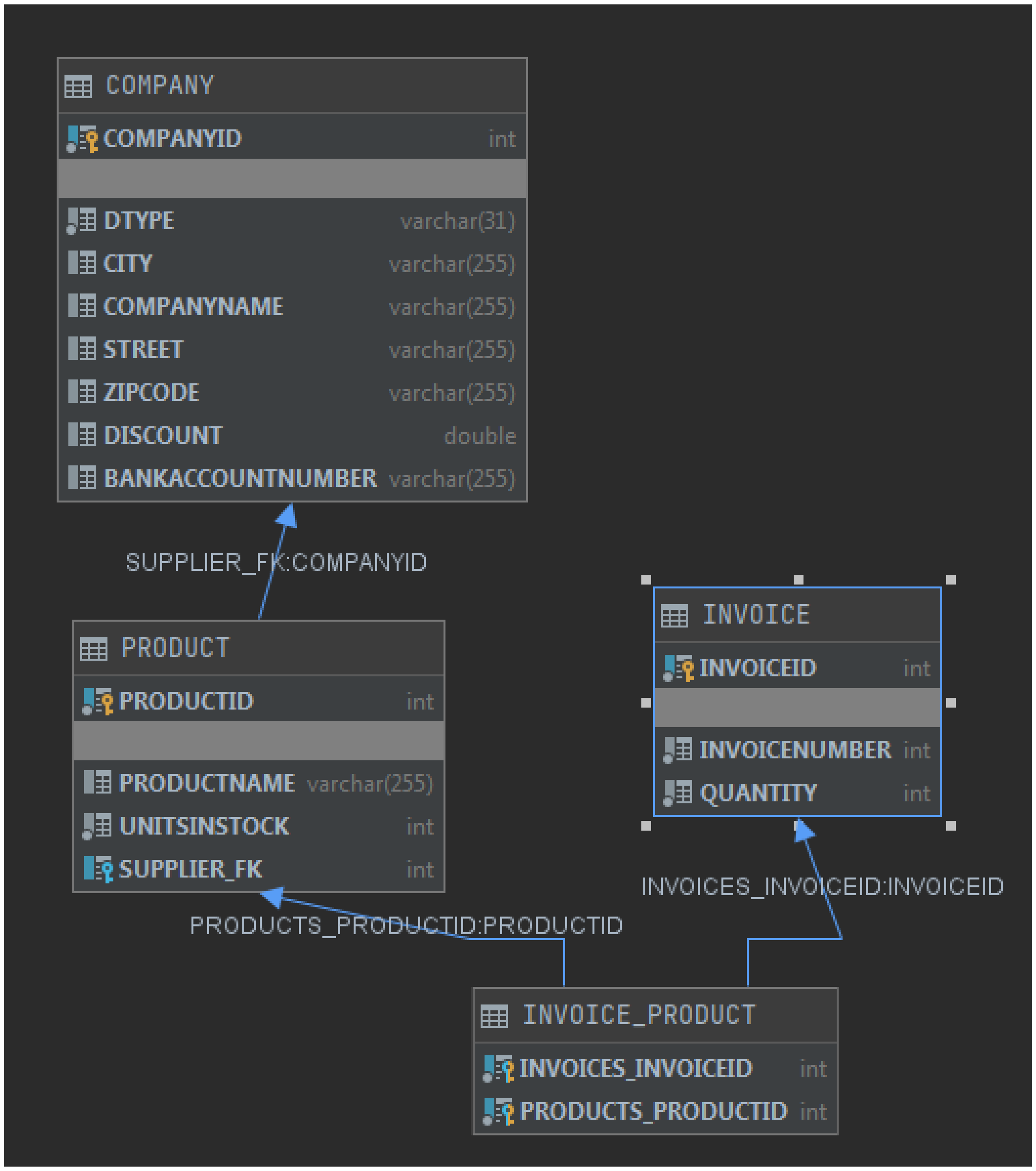


```java
public class Main {
    public static void main(final String[] args) throws Exception {
        EntityManagerFactory emf = Persistence.createEntityManagerFactory( persistenceUnitName: "myDatabaseConfig");
        EntityManager em = emf.createEntityManager();

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

        Customer kowalski = new Customer( companyName: "Kowalski", street: "ul.Niska", city: "Waszyngton", zipCode: "11-111",  discount: 0.1);
        Customer nowak = new Customer( companyName: "Nowak", street: "ul.Wysoka", city: "Nowy Jork", zipCode: "22-222",  discount: 0.2);


        Product book = new Product( productName: "book",  unitsInStock: 5);
        Product note  = new Product( productName: "note",  unitsInStock: 2);
        Product cup = new Product( productName: "cup",  unitsInStock: 8);
        Product pencil = new Product( productName: "pencil",  unitsInStock: 4);
        Product pen = new Product( productName: "pen",  unitsInStock: 11);

        Supplier google = new Supplier( companyName: "Google", street: "Walkstreet",  city: "London",  zipCode: "30-100", bankAccountNumber: "000011112222");
        Supplier facebook = new Supplier( companyName: "Facebook",  street: "Piłsudskiego",  city: "Warsaw",  zipCode: "20-123", bankAccountNumber: "333344445555");

        Invoice invoice123456 = new Invoice( invoiceNumber: 123456, quantity: 2);
        Invoice invoice123457 = new Invoice( invoiceNumber: 123457, quantity: 3);

        em.persist(invoice123456);
        em.persist(invoice123457);
        em.persist(book);
        em.persist(note);
        em.persist(cup);
        em.persist(pencil);
        em.persist(pen);
        em.persist(google);
        em.persist(facebook);
        em.persist(kowalski);
        em.persist(nowak);

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

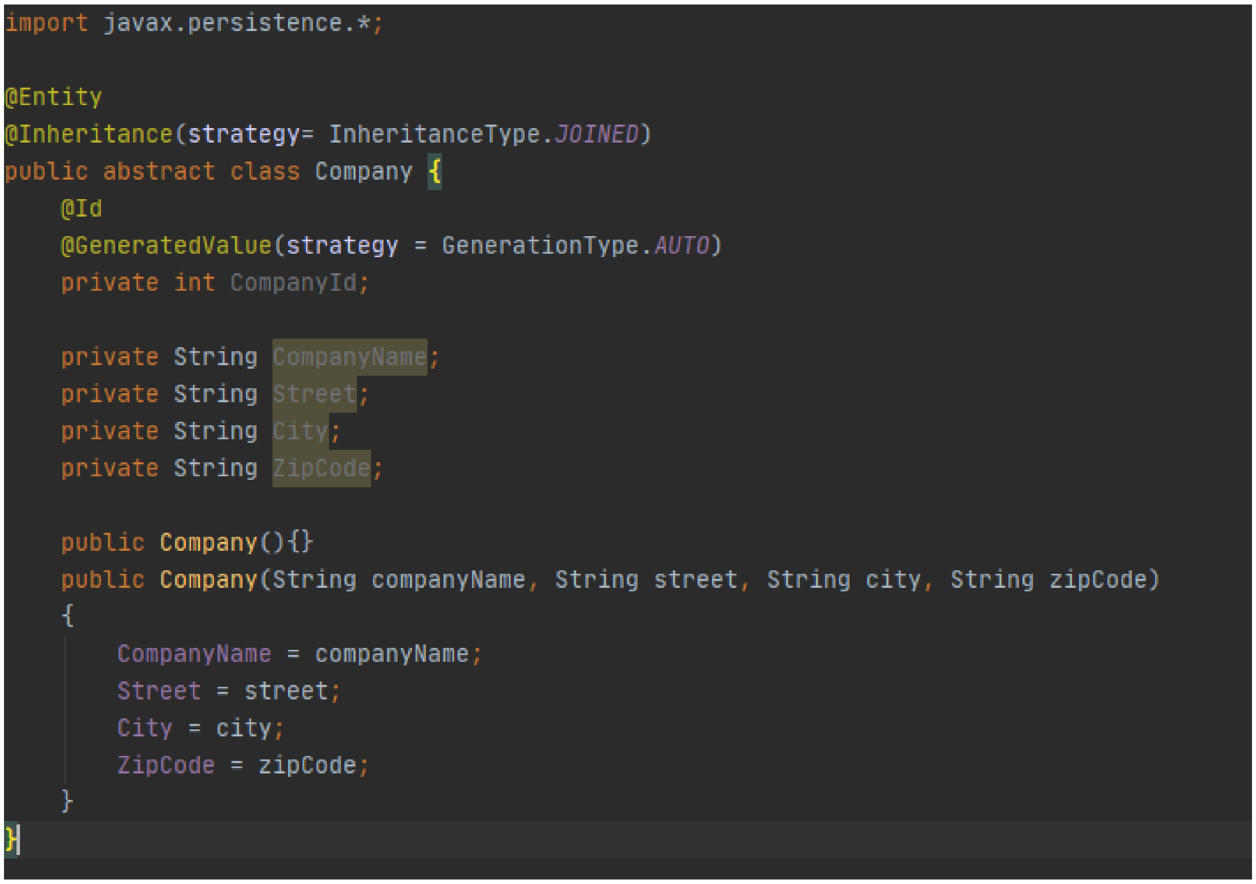


```sql
select * from company
```

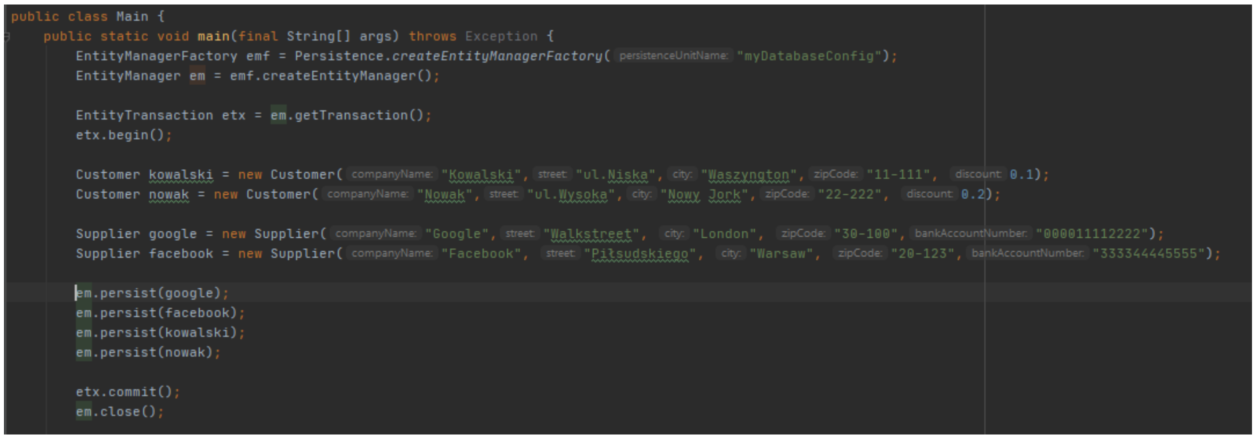
	DTYPE	COMPANYID	CITY	COMPANYNAME	STREET	ZIPCODE	DISCOUNT	BANKACCOUNTNUMBER
1	Supplier	8	London	Google	Walkstreet	30-100	<null>	000011112222
2	Supplier	9	Warsaw	Facebook	Piłsudskiego	20-123	<null>	333344445555
3	Customer	10	Waszyngton	Kowalski	ul.Niska	11-111	0.1	<null>
4	Customer	11	Nowy Jork	Nowak	ul.Wysoka	22-222	0.2	<null>


COMPANY
COMPANYID int
DTYPE varchar(31)
CITY varchar(255)
COMPANYNAME varchar(255)
STREET varchar(255)
ZIPCODE varchar(255)
DISCOUNT double
BANKACCOUNTNUMBER varchar(255)
SUPPLIER_FK:COMPANYID
PRODUCT
PRODUCTID int
PRODUCTNAME varchar(255)
UNITSINSTOCK int
SUPPLIER_FK int
INVOICE
INVOICEID int
INVOICENUMBER int
QUANTITY int
INVOICES_INVOICEID:INVOICEID
PRODUCTS_PRODUCTID:PRODUCTID
INVOICE_PRODUCT
INVOICES_INVOICEID int
PRODUCTS_PRODUCTID int

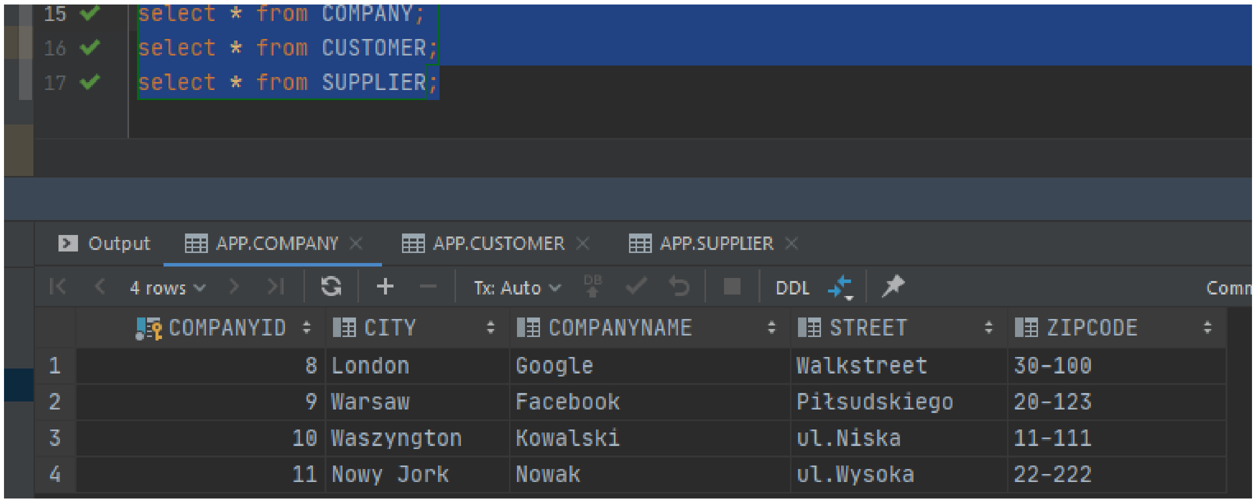
2.Tabele łączone



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

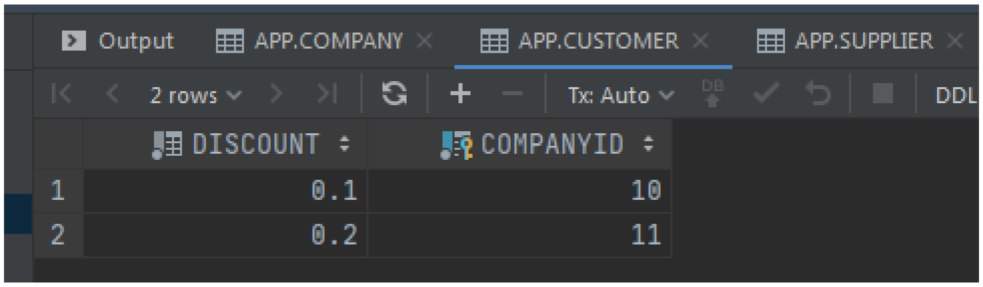


```java
public class Main {
    public static void main(final String[] args) throws Exception {
        EntityManagerFactory emf = Persistence.createEntityManagerFactory( persistenceUnitName: "myDatabaseConfig");
        EntityManager em = emf.createEntityManager();

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

        Customer kowalski = new Customer( companyName: "Kowalski", street: "ul.Niska", city: "Waszyngton", zipCode: "11-111",  discount: 0.1);
        Customer nowak = new Customer( companyName: "Nowak", street: "ul.Wysoka", city: "Nowy Jork", zipCode: "22-222",  discount: 0.2);

        Supplier google = new Supplier( companyName: "Google", street: "Walkstreet",  city: "London",  zipCode: "30-100", bankAccountNumber: "000011112222");
        Supplier facebook = new Supplier( companyName: "Facebook",  street: "Piłsudskiego",  city: "Warsaw",  zipCode: "20-123", bankAccountNumber: "333344445555");

        em.persist(google);
        em.persist(facebook);
        em.persist(kowalski);
        em.persist(nowak);

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

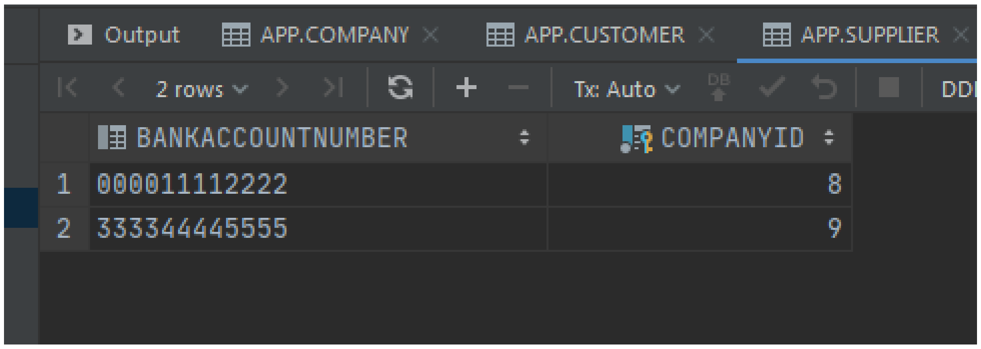


```
select * from COMPANY;
select * from CUSTOMER;
select * from SUPPLIER;
```

	COMPANYID	CITY	COMPANYNAME	STREET	ZIPCODE
1	8	London	Google	Walkstreet	30-100
2	9	Warsaw	Facebook	Piłsudskiego	20-123
3	10	Waszyngton	Kowalski	ul.Niska	11-111
4	11	Nowy Jork	Nowak	ul.Wysoka	22-222

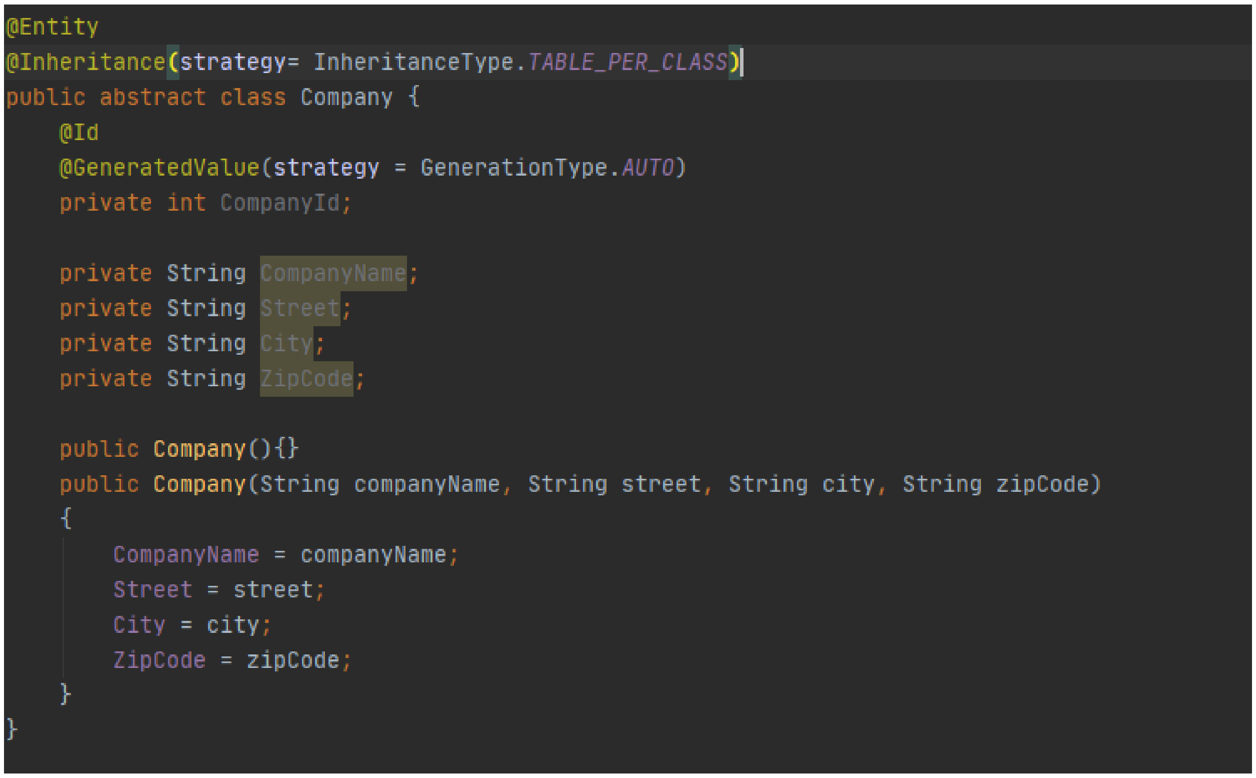


	DISCOUNT	COMPANYID
1	0.1	10
2	0.2	11

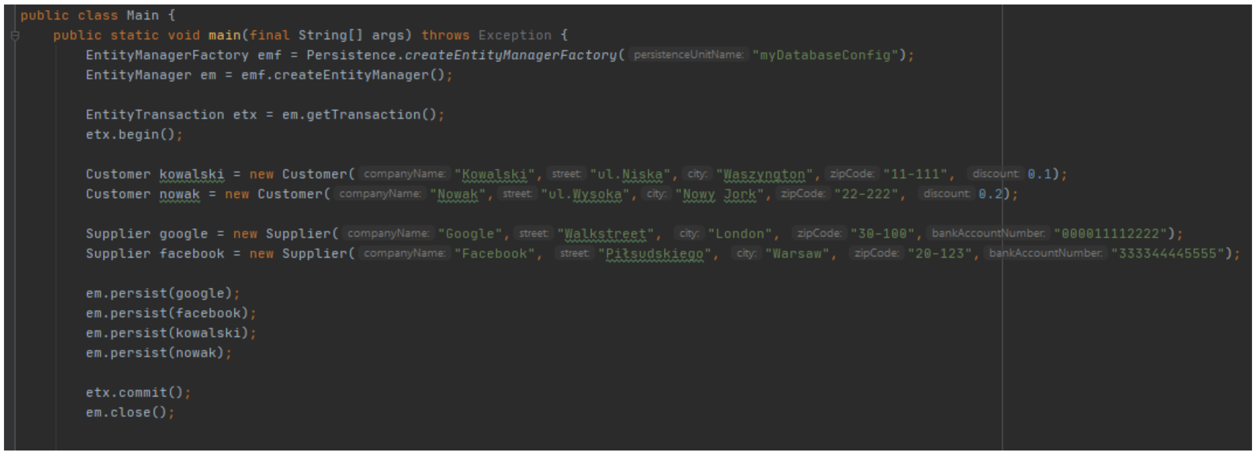


	BANKACCOUNTNUMBER	COMPANYID
1	000011112222	8
2	333344445555	9

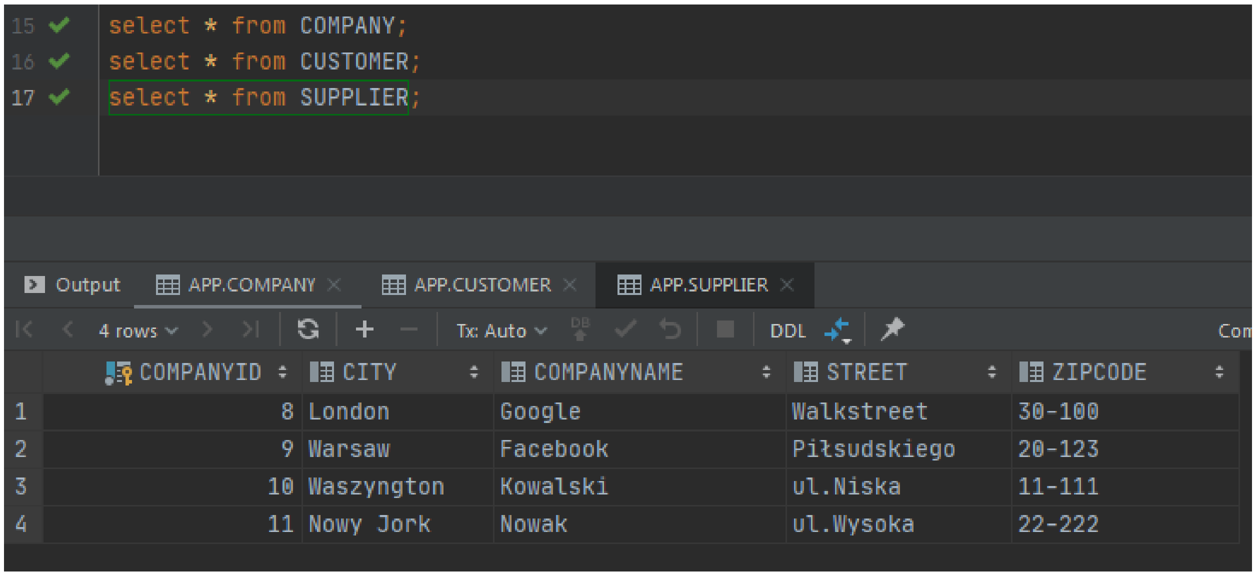
3.Jedna tabela na konkretną klasę



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

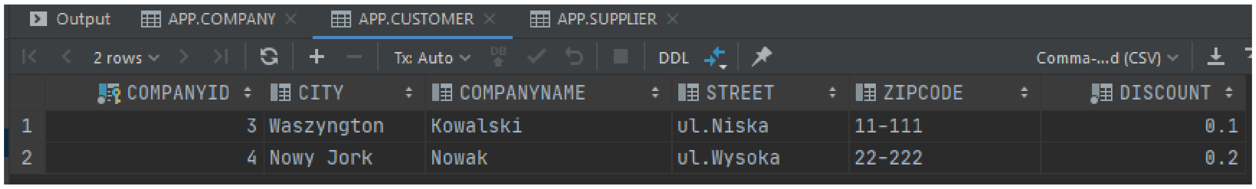


```java
public class Main {
    public static void main(final String[] args) throws Exception {
        EntityManagerFactory emf = Persistence.createEntityManagerFactory( persistenceUnitName: "myDatabaseConfig");
        EntityManager em = emf.createEntityManager();

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

        Customer kowalski = new Customer( companyName: "Kowalski", street: "ul.Niska", city: "Waszyngton", zipCode: "11-111",  discount: 0.1);
        Customer nowak = new Customer( companyName: "Nowak", street: "ul.Wysoka", city: "Nowy Jork", zipCode: "22-222",  discount: 0.2);

        Supplier google = new Supplier( companyName: "Google", street: "Walkstreet",  city: "London",  zipCode: "30-100", bankAccountNumber: "000011112222");
        Supplier facebook = new Supplier( companyName: "Facebook",  street: "Piłsudskiego",  city: "Warsaw",  zipCode: "20-123", bankAccountNumber: "333344445555");

        em.persist(google);
        em.persist(facebook);
        em.persist(kowalski);
        em.persist(nowak);

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

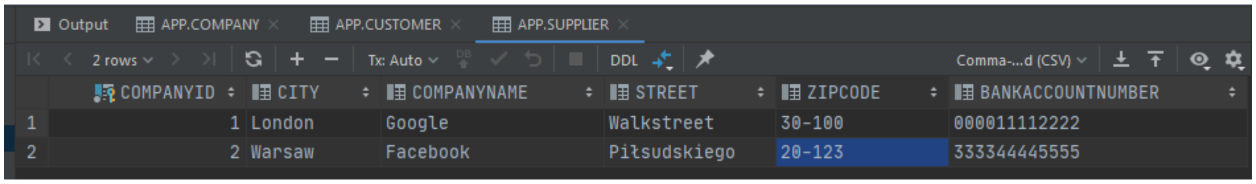


```sql
select * from COMPANY;
select * from CUSTOMER;
select * from SUPPLIER;
```

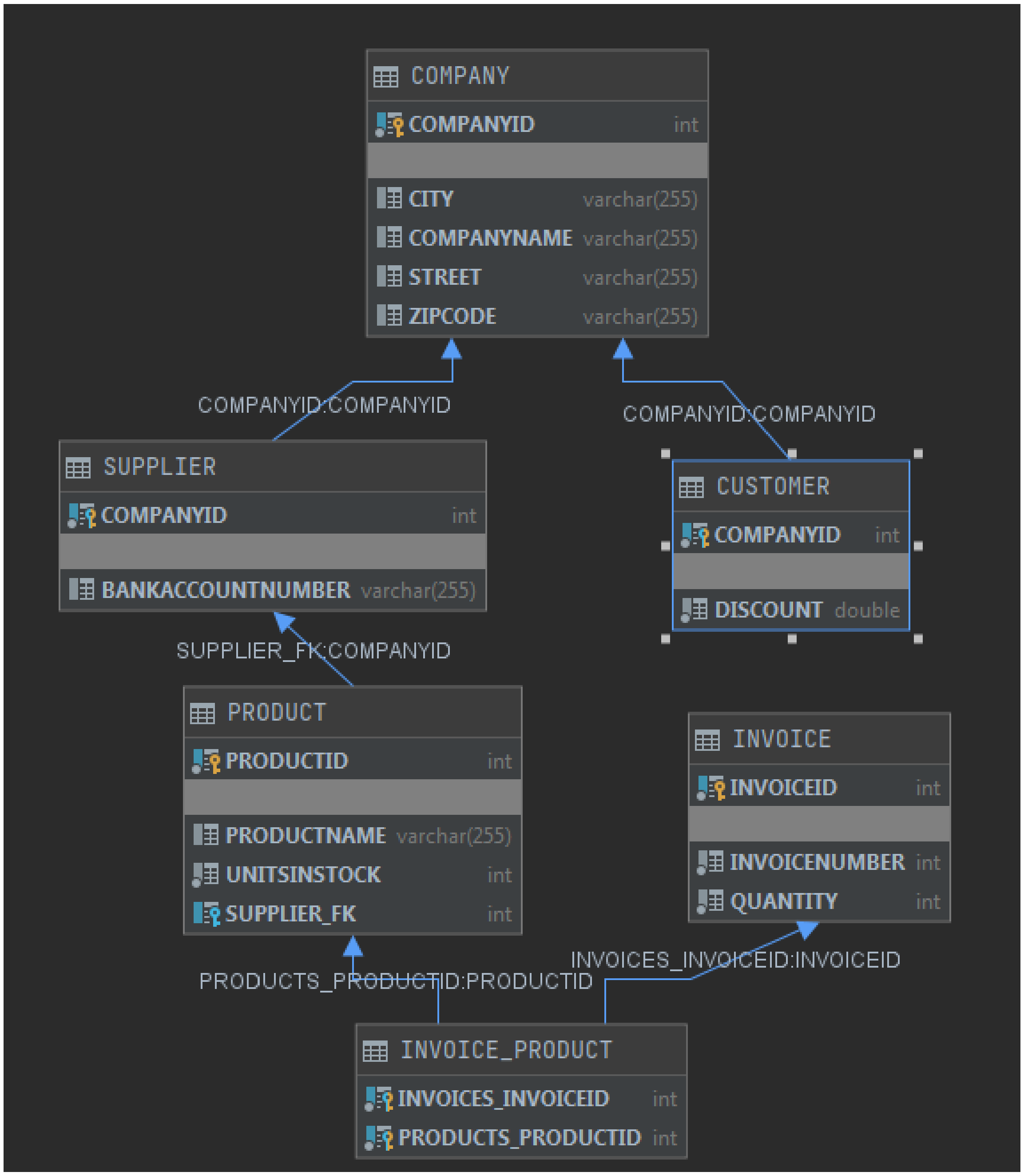
	COMPANYID	CITY	COMPANYNAME	STREET	ZIPCODE
1	8	London	Google	Walkstreet	30-100
2	9	Warsaw	Facebook	Piłsudskiego	20-123
3	10	Waszyngton	Kowalski	ul.Niska	11-111
4	11	Nowy Jork	Nowak	ul.Wysoka	22-222



	COMPANYID	CITY	COMPANYNAME	STREET	ZIPCODE	DISCOUNT
1	3	Waszyngton	Kowalski	ul.Niska	11-111	0.1
2	4	Nowy Jork	Nowak	ul.Wysoka	22-222	0.2



	COMPANYID	CITY	COMPANYNAME	STREET	ZIPCODE	BANKACCOUNTNUMBER
1	1	London	Google	Walkstreet	30-100	000011112222
2	2	Warsaw	Facebook	Piłsudskiego	20-123	333344445555


COMPANY
COMPANYID int
CITY varchar(255)
COMPANYNAME varchar(255)
STREET varchar(255)
ZIPCODE varchar(255)
COMPANYID:COMPANYID
COMPANYID:COMPANYID
SUPPLIER
COMPANYID int
BANKACCOUNTNUMBER varchar(255)
CUSTOMER
COMPANYID int
DISCOUNT double
SUPPLIER_FK:COMPANYID
PRODUCT
PRODUCTID int
PRODUCTNAME varchar(255)
UNITSINSTOCK int
SUPPLIER_FK int
INVOICE
INVOICEID int
INVOICENUMBER int
QUANTITY int
PRODUCTS_PRODUCTID:PRODUCTID
INVOICES_INVOICEID:INVOICEID
INVOICE_PRODUCT
INVOICES_INVOICEID int
PRODUCTS_PRODUCTID int