Dawid Majchrowski

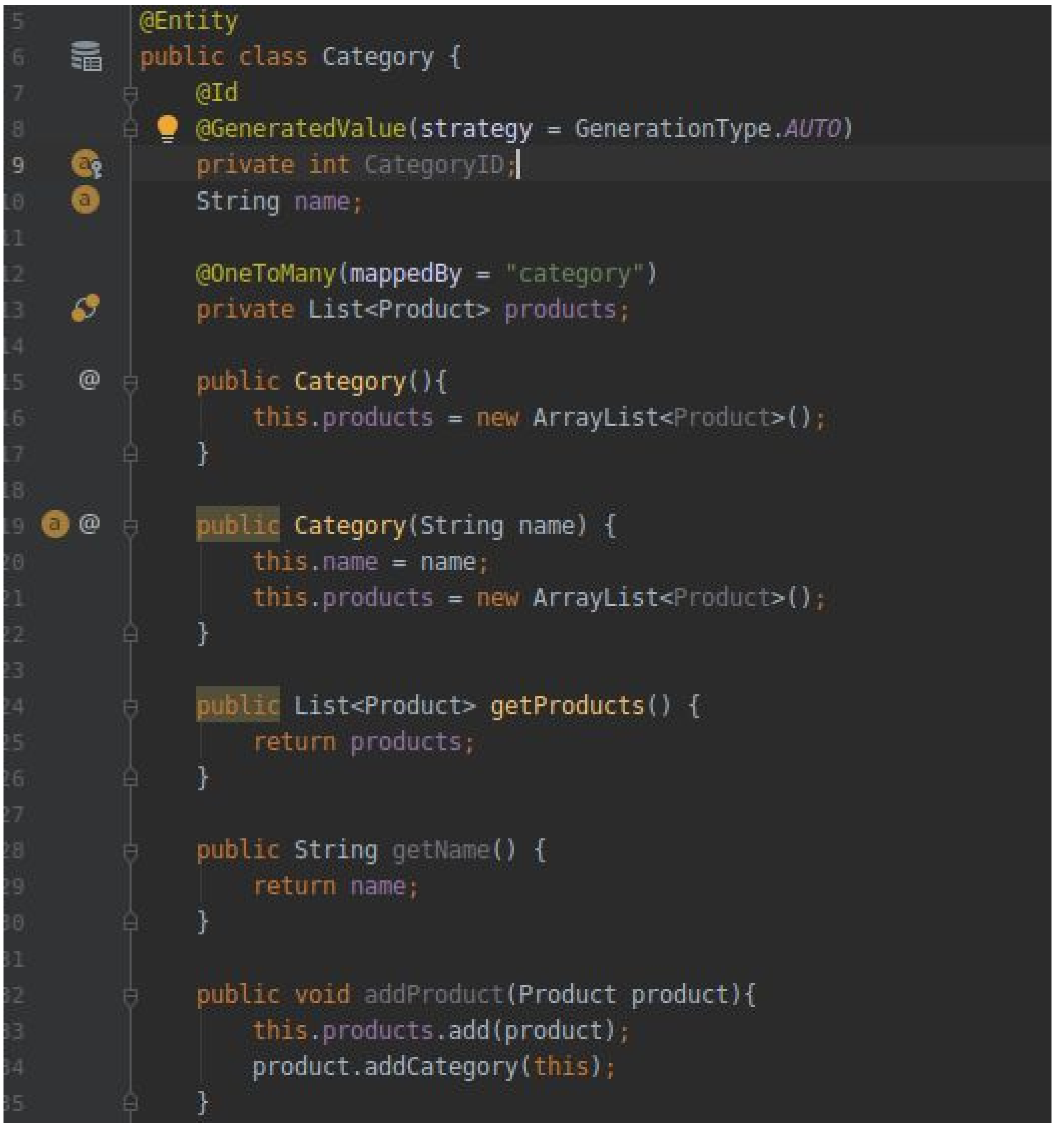
Hibernate, JPA

Sprawozdanie - 24.11.2019

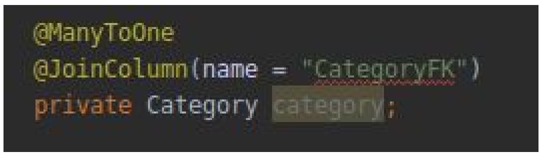
Sprawozdanie kontynuujemy od miejsca miejsca zakończenia ćwiczeń (zad 6):

6 Nowa klasa Category

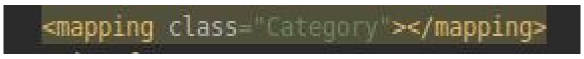
Dodajemy klase Category z relacją 1 do wielu po obu stronach.



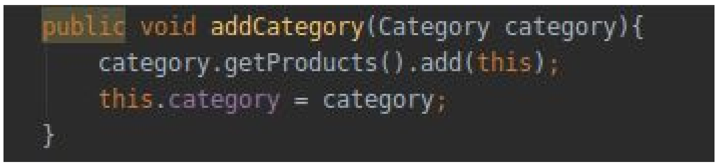
```java
@Entity
public class Category {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int CategoryID;
    String name;

    @OneToMany(mappedBy = "category")
    private List<Product> products;

    public Category(){
        this.products = new ArrayList<Product>();
    }

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

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

    public String getName() {
        return name;
    }

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



```java
@ManyToOne
@JoinColumn(name = "CategoryFK")
private Category category;
```

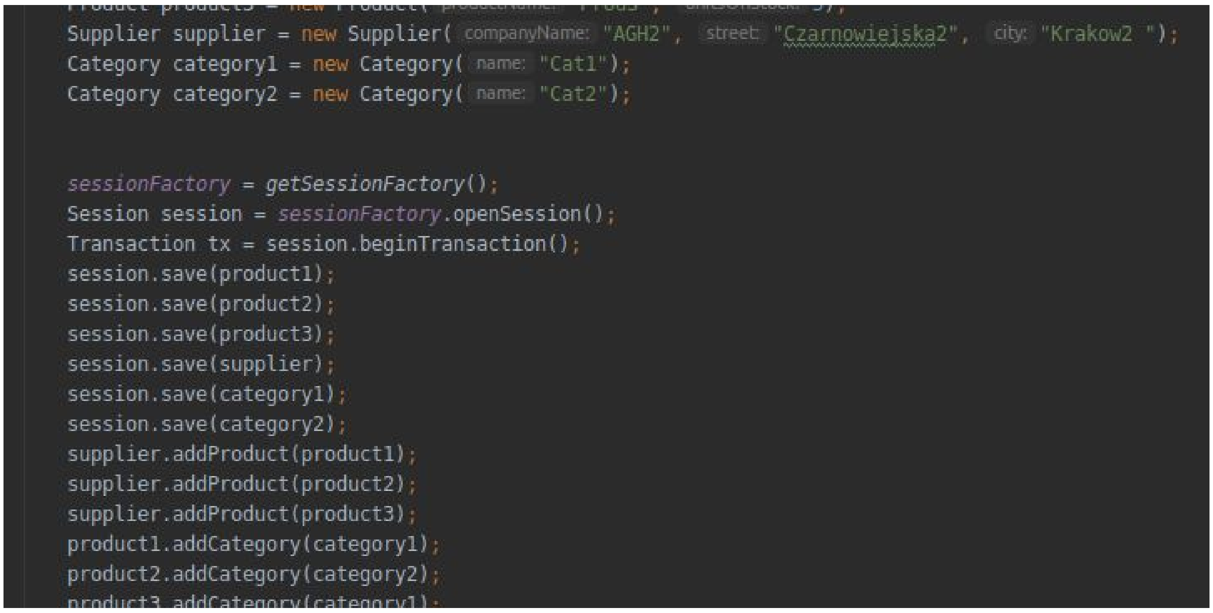


```xml
<mapping class="Category"></mapping>
```

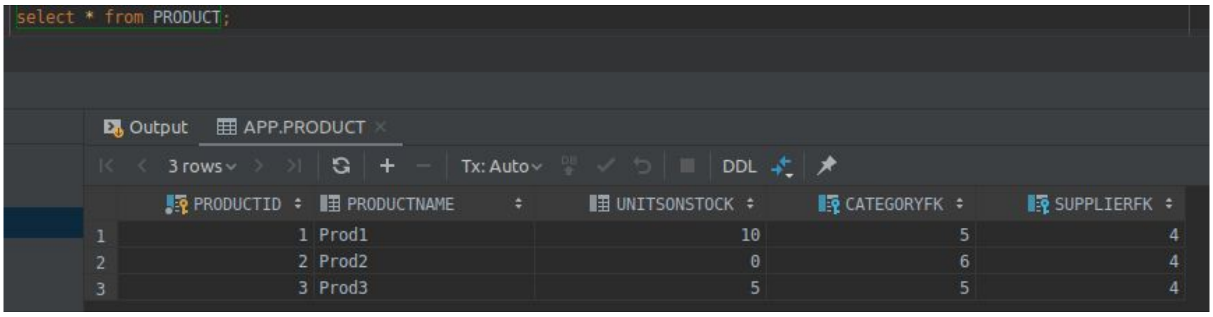


```java
public void addCategory(Category category){
    category.getProducts().add(this);
    this.category = category;
}
```

W Mainie dodajemy kilka produktów oraz kategori.

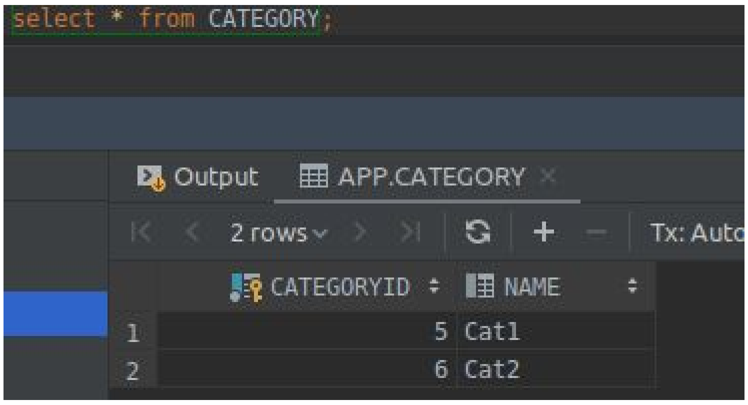


```java
        Supplier supplier = new Supplier( companyName: "AGH2",  street: "Czarnowiejska2",  city: "Krakow2 ");
        Category category1 = new Category( name: "Cat1");
        Category category2 = new Category( name: "Cat2");


        sessionFactory = getSessionFactory();
        Session session = sessionFactory.openSession();
        Transaction tx = session.beginTransaction();
        session.save(product1);
        session.save(product2);
        session.save(product3);
        session.save(supplier);
        session.save(category1);
        session.save(category2);
        supplier.addProduct(product1);
        supplier.addProduct(product2);
        supplier.addProduct(product3);
        product1.addCategory(category1);
        product2.addCategory(category2);
        product3.addCategory(category1);
```



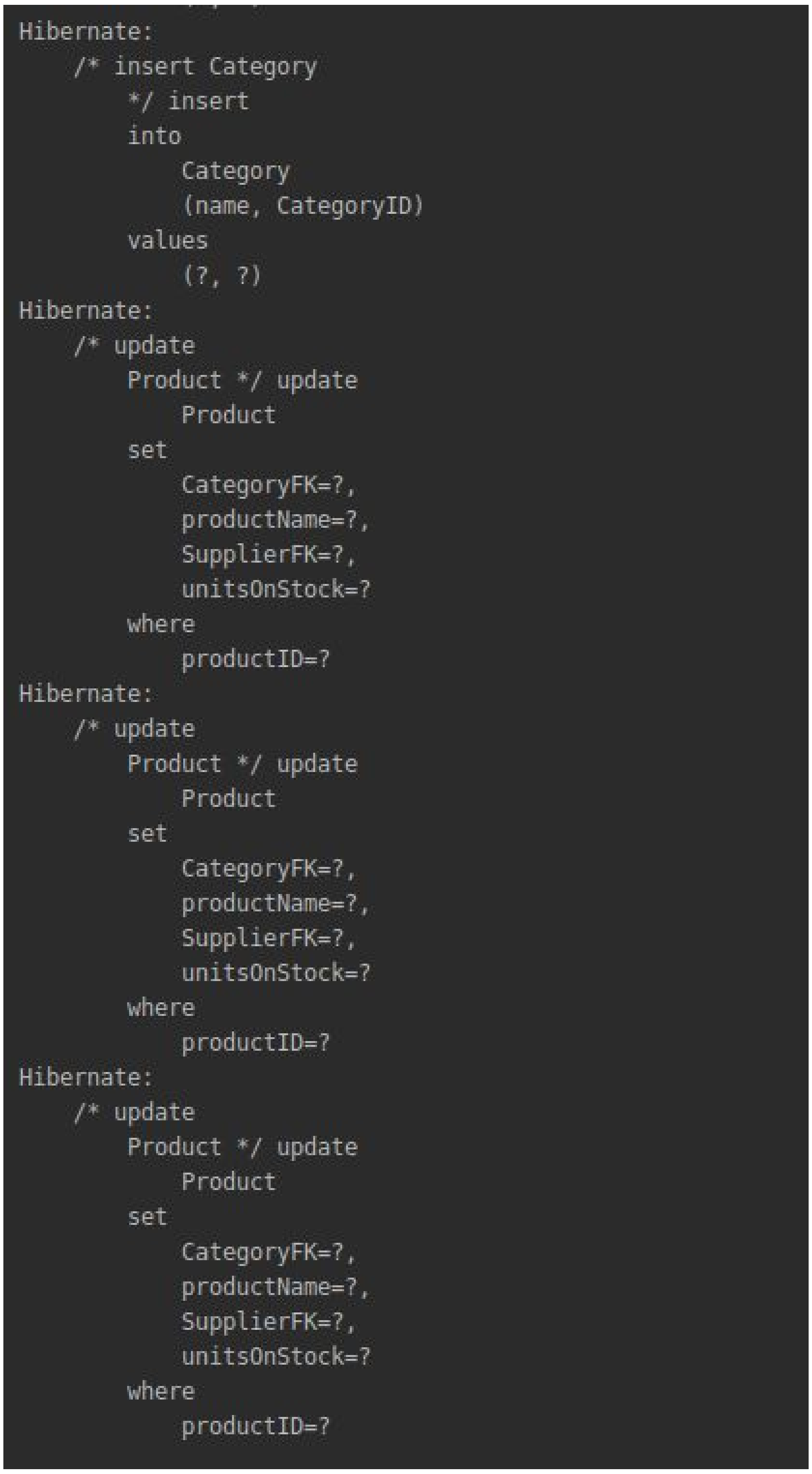
```sql
select * from PRODUCT;
```

```
    PRODUCTID   PRODUCTNAME   UNITSONSTOCK   CATEGORYFK   SUPPLIERFK
1           1   Prod1                   10            5            4
2           2   Prod2                    0            6            4
3           3   Prod3                    5            5            4
```



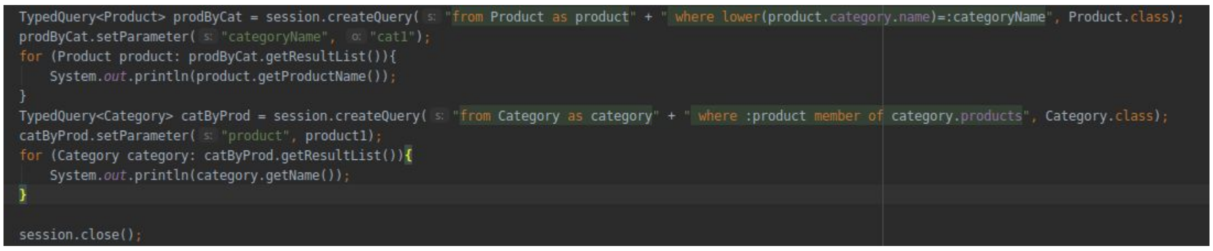
```sql
select * from CATEGORY;
```

```
    CATEGORYID   NAME
1            5   Cat1
2            6   Cat2
```

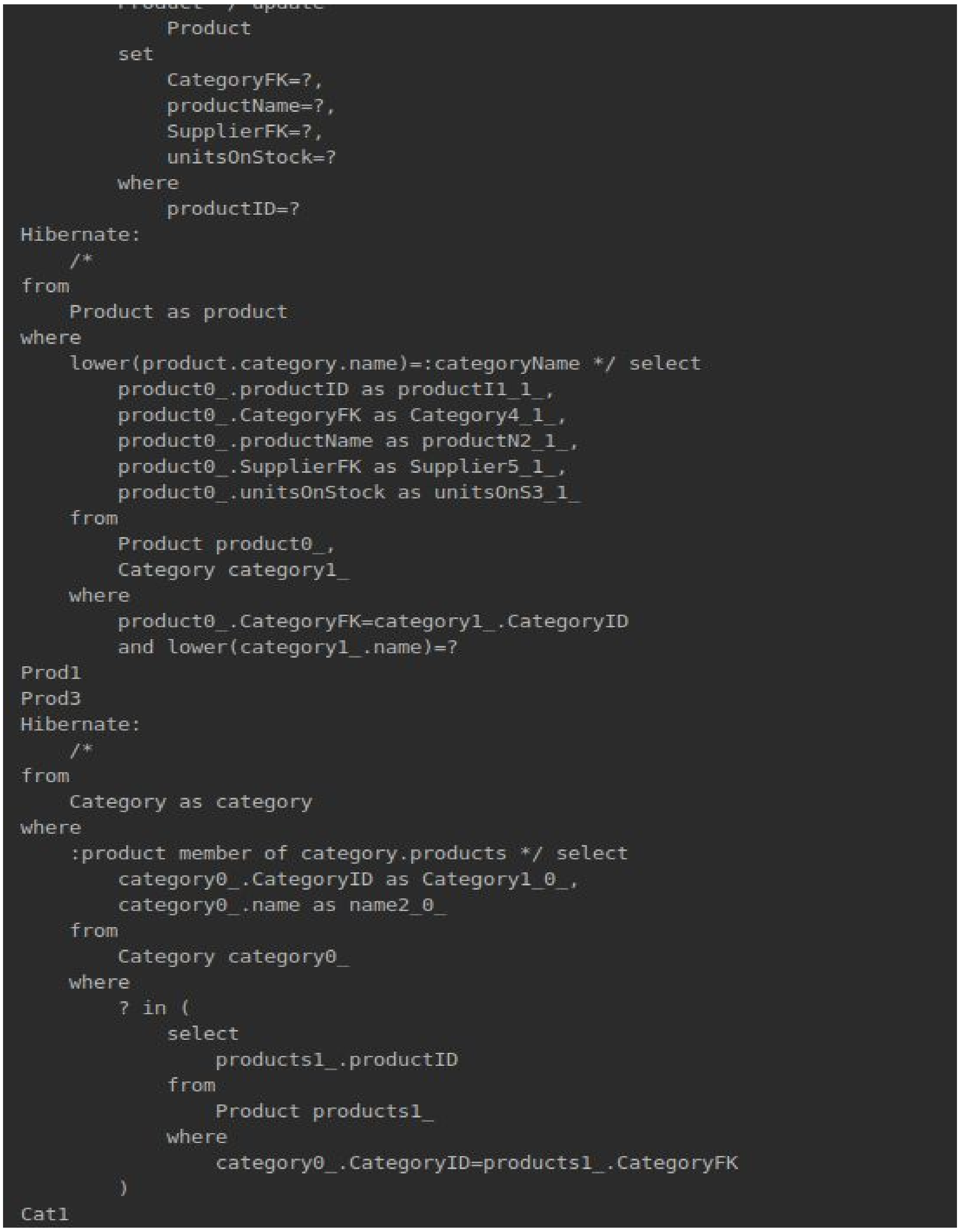


```
Hibernate:
    /* insert Category
        */ insert
        into
            Category
            (name, CategoryID)
        values
            (?, ?)
Hibernate:
    /* update
        Product */ update
            Product
        set
            CategoryFK=?,
            productName=?,
            SupplierFK=?,
            unitsOnStock=?
        where
            productID=?
Hibernate:
    /* update
        Product */ update
            Product
        set
            CategoryFK=?,
            productName=?,
            SupplierFK=?,
            unitsOnStock=?
        where
            productID=?
Hibernate:
    /* update
        Product */ update
            Product
        set
            CategoryFK=?,
            productName=?,
            SupplierFK=?,
            unitsOnStock=?
        where
            productID=?
```

Wyszukiwanie produktu z danej kategorii i kategorii dla której należy dany produkt.



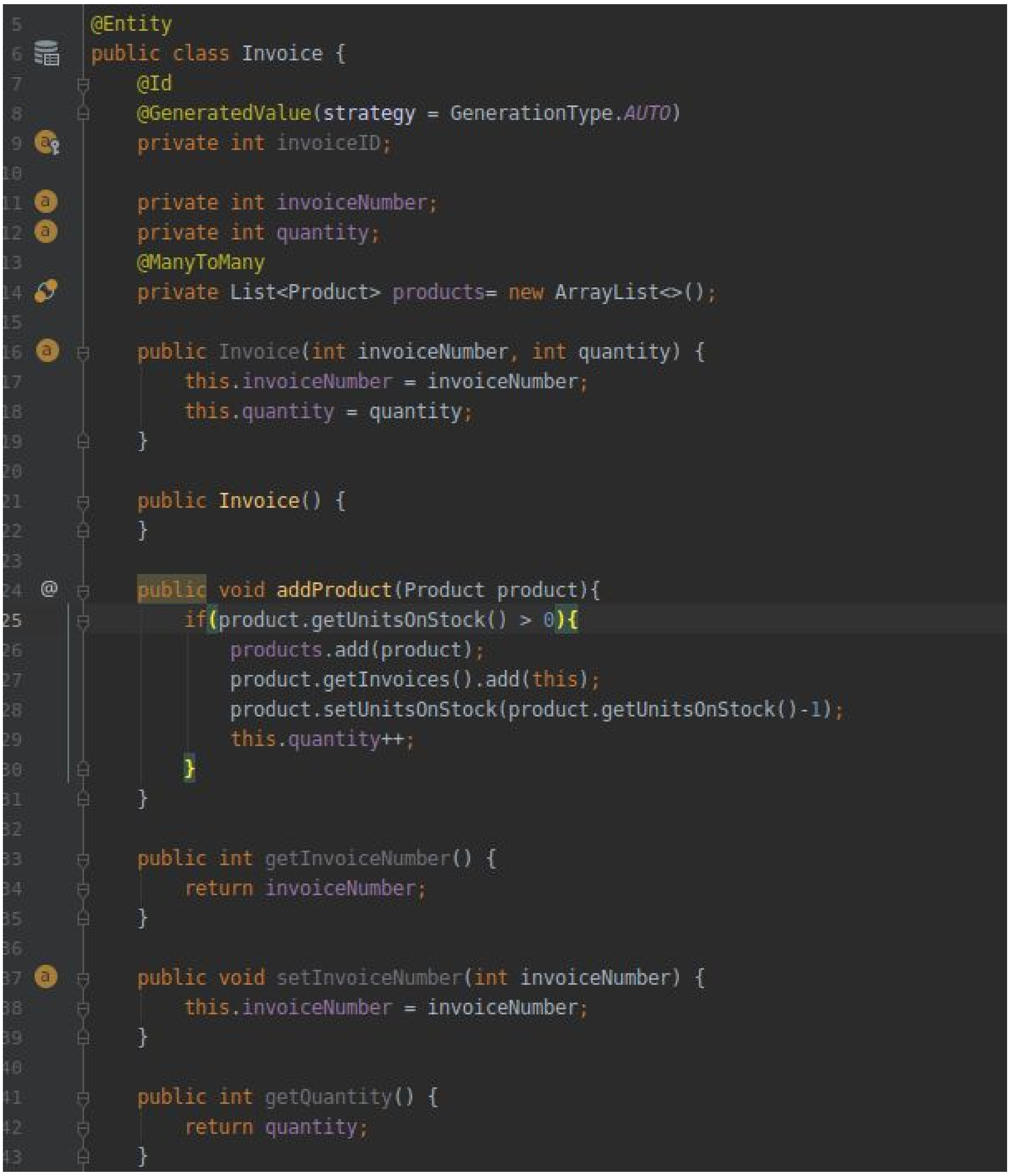
```java
TypedQuery<Product> prodByCat = session.createQuery( "from Product as product" + " where lower(product.category.name)=:categoryName", Product.class);
prodByCat.setParameter( "categoryName",  "cat1");
for (Product product: prodByCat.getResultList()){
    System.out.println(product.getProductName());
}
TypedQuery<Category> catByProd = session.createQuery( "from Category as category" + " where :product member of category.products", Category.class);
catByProd.setParameter( "product", product1);
for (Category category: catByProd.getResultList()){
    System.out.println(category.getName());
}

session.close();
```



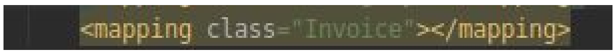
```
            Product
        set
            CategoryFK=?,
            productName=?,
            SupplierFK=?,
            unitsOnStock=?
        where
            productID=?
Hibernate: 
    /* 
from
    Product as product 
where
    lower(product.category.name)=:categoryName */ select
        product0_.productID as productI1_1_,
        product0_.CategoryFK as Category4_1_,
        product0_.productName as productN2_1_,
        product0_.SupplierFK as Supplier5_1_,
        product0_.unitsOnStock as unitsOnS3_1_ 
    from
        Product product0_,
        Category category1_ 
    where
        product0_.CategoryFK=category1_.CategoryID 
        and lower(category1_.name)=?
Prod1
Prod3
Hibernate: 
    /* 
from
    Category as category 
where
    :product member of category.products */ select
        category0_.CategoryID as Category1_0_,
        category0_.name as name2_0_ 
    from
        Category category0_ 
    where
        ? in (
            select
                products1_.productID 
            from
                Product products1_ 
            where
                category0_.CategoryID=products1_.CategoryFK
        )
Cat1
```

7. Relacja wiele do wielu

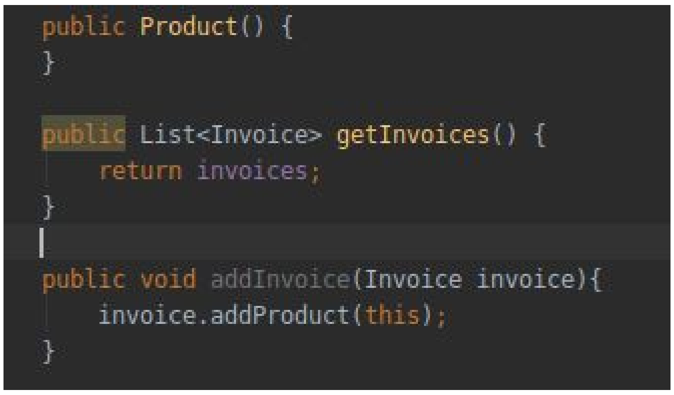
Dodajemy klase Invoce i mapujemy relacje wiele do wielu z klasą Product.



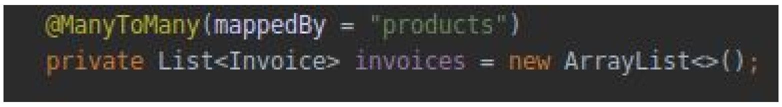
```java
@Entity
public class Invoice {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int invoiceID;

    private int invoiceNumber;
    private int quantity;
    @ManyToMany
    private List<Product> products= new ArrayList<>();

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

    public Invoice() {
    }

    public void addProduct(Product product){
        if(product.getUnitsOnStock() > 0){
            products.add(product);
            product.getInvoices().add(this);
            product.setUnitsOnStock(product.getUnitsOnStock()-1);
            this.quantity++;
        }
    }

    public int getInvoiceNumber() {
        return invoiceNumber;
    }

    public void setInvoiceNumber(int invoiceNumber) {
        this.invoiceNumber = invoiceNumber;
    }

    public int getQuantity() {
        return quantity;
    }
```



```xml
<mapping class="Invoice"></mapping>
```

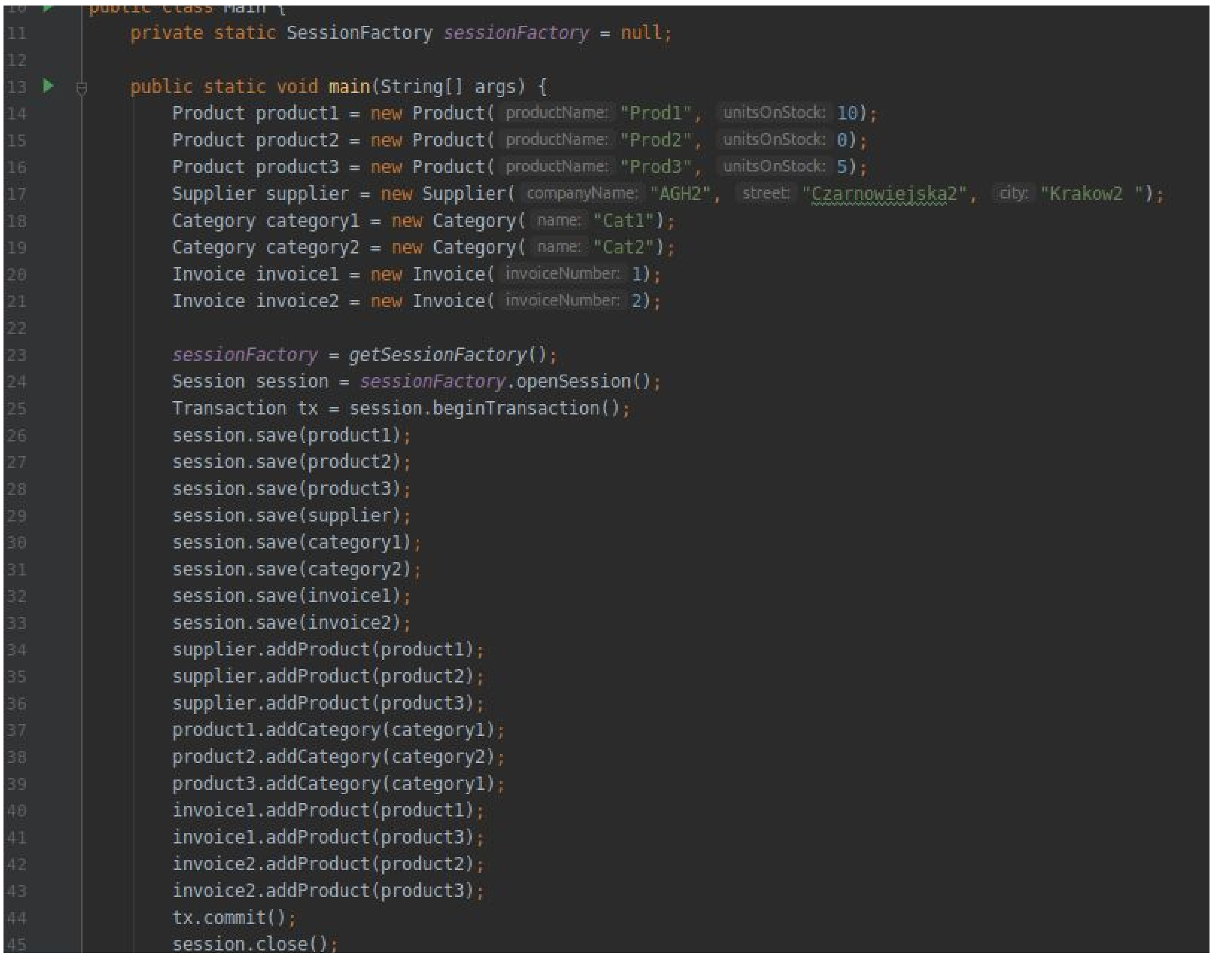


```java
public Product() {
}

public List<Invoice> getInvoices() {
    return invoices;
}

public void addInvoice(Invoice invoice){
    invoice.addProduct(this);
}
```

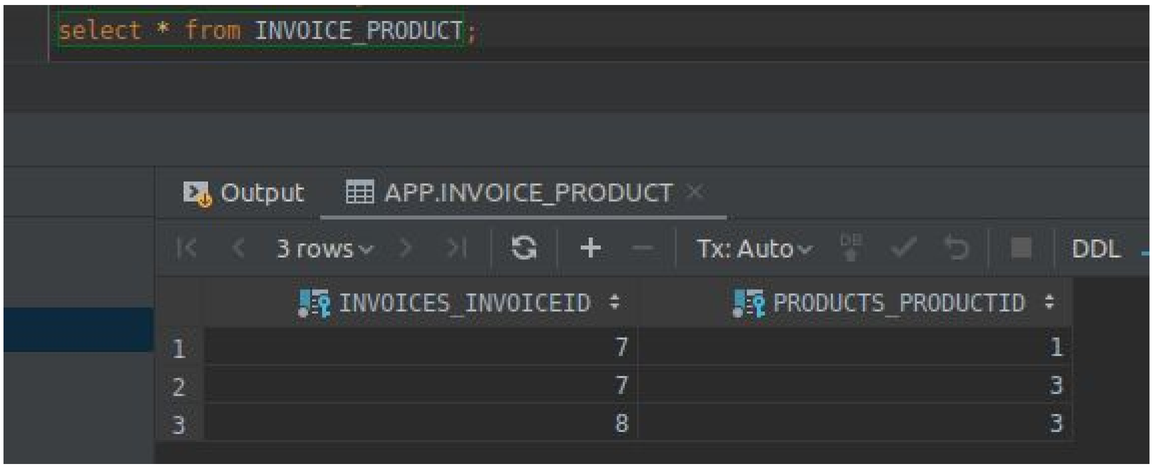


```java
@ManyToMany(mappedBy = "products")
private List<Invoice> invoices = new ArrayList<>();
```

Dodajemy do starego maina, 2 faktury i dodajemy “sprzedajemy” produkty na odpowiednich transakcjach. Zauważmy, że wykonujemy 4 transakcje sprzedaży, natomiast product2 ma wartość 0 unitOnStock, dlatego sprzedaż produktu 2 nie powinna się powieść, a w bazie powinny pojawić się tylko 3 rekordy.



```java
public class Main {
    private static SessionFactory sessionFactory = null;

    public static void main(String[] args) {
        Product product1 = new Product( productName: "Prod1",  unitsOnStock: 10);
        Product product2 = new Product( productName: "Prod2",  unitsOnStock: 0);
        Product product3 = new Product( productName: "Prod3",  unitsOnStock: 5);
        Supplier supplier = new Supplier( companyName: "AGH2",  street: "Czarnowiejska2",  city: "Krakow2 ");
        Category category1 = new Category( name: "Cat1");
        Category category2 = new Category( name: "Cat2");
        Invoice invoice1 = new Invoice( invoiceNumber: 1);
        Invoice invoice2 = new Invoice( invoiceNumber: 2);

        sessionFactory = getSessionFactory();
        Session session = sessionFactory.openSession();
        Transaction tx = session.beginTransaction();
        session.save(product1);
        session.save(product2);
        session.save(product3);
        session.save(supplier);
        session.save(category1);
        session.save(category2);
        session.save(invoice1);
        session.save(invoice2);
        supplier.addProduct(product1);
        supplier.addProduct(product2);
        supplier.addProduct(product3);
        product1.addCategory(category1);
        product2.addCategory(category2);
        product3.addCategory(category1);
        invoice1.addProduct(product1);
        invoice1.addProduct(product3);
        invoice2.addProduct(product2);
        invoice2.addProduct(product3);
        tx.commit();
        session.close();
```

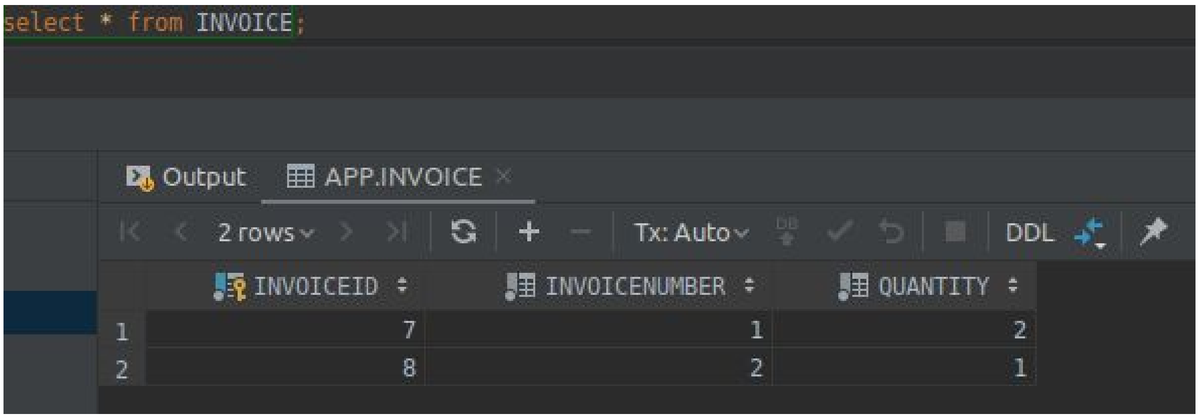


```
select * from INVOICE_PRODUCT;
```

Output | APP.INVOICE_PRODUCT

3 rows | Tx: Auto | DDL

	INVOICES_INVOICEID	PRODUCTS_PRODUCTID
1	7	1
2	7	3
3	8	3

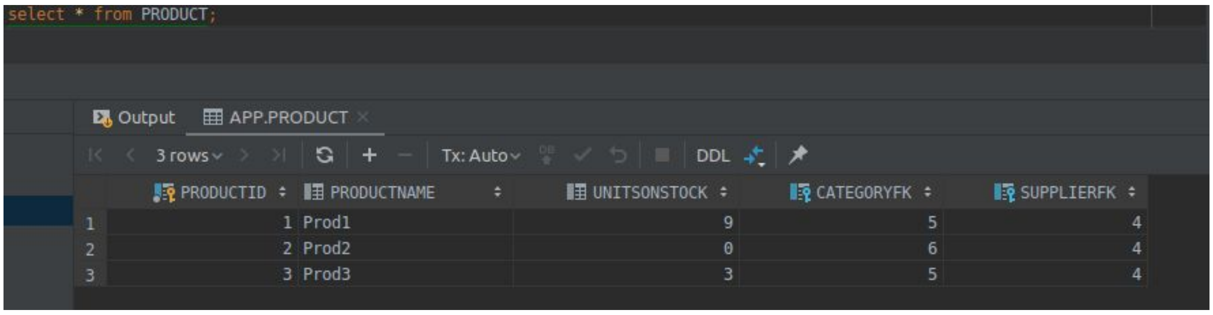


```
select * from INVOICE;
```

Output | APP.INVOICE

2 rows | Tx: Auto | DDL

	INVOICEID	INVOICENUMBER	QUANTITY
1	7	1	2
2	8	2	1

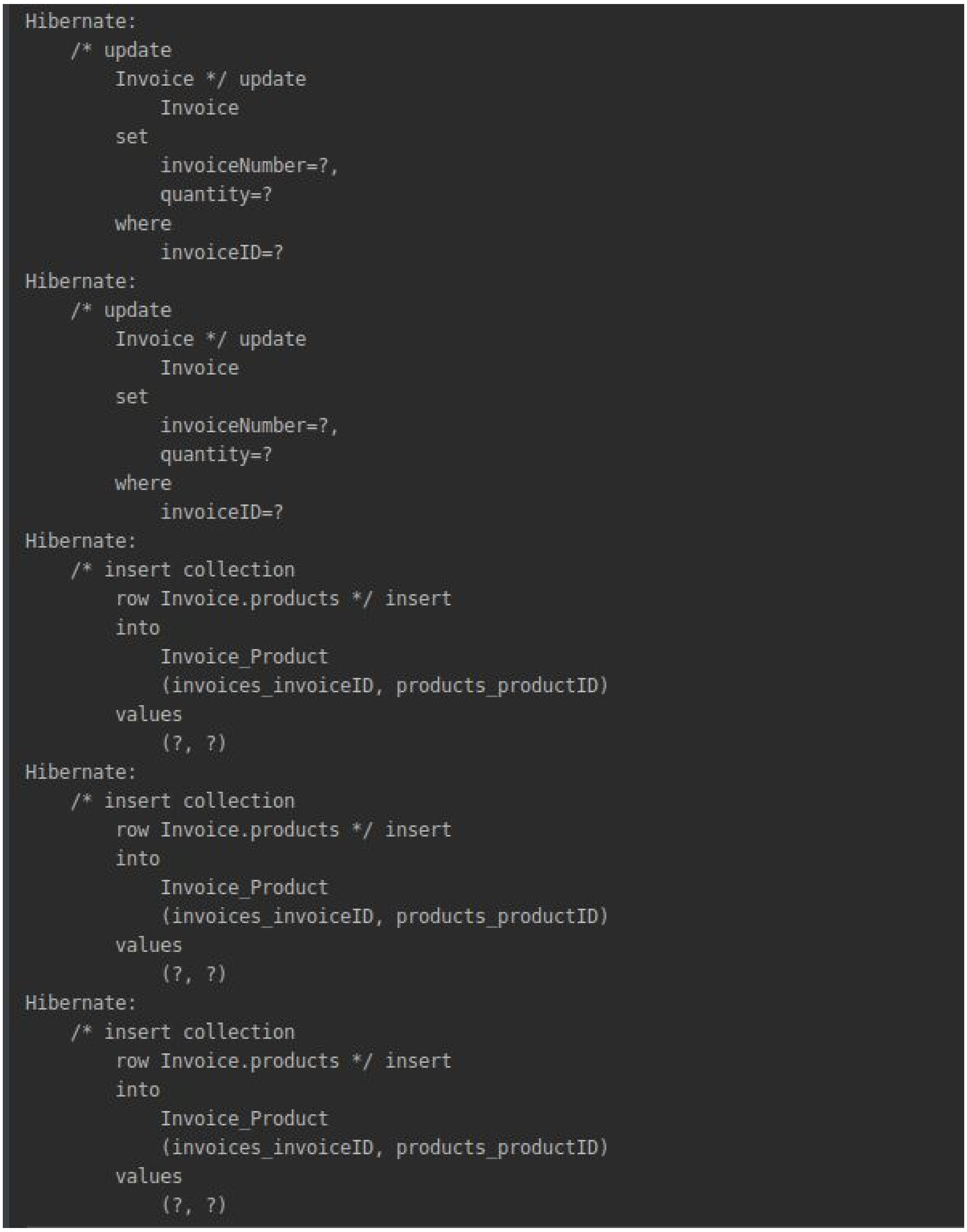


```
select * from PRODUCT;
```

Output | APP.PRODUCT

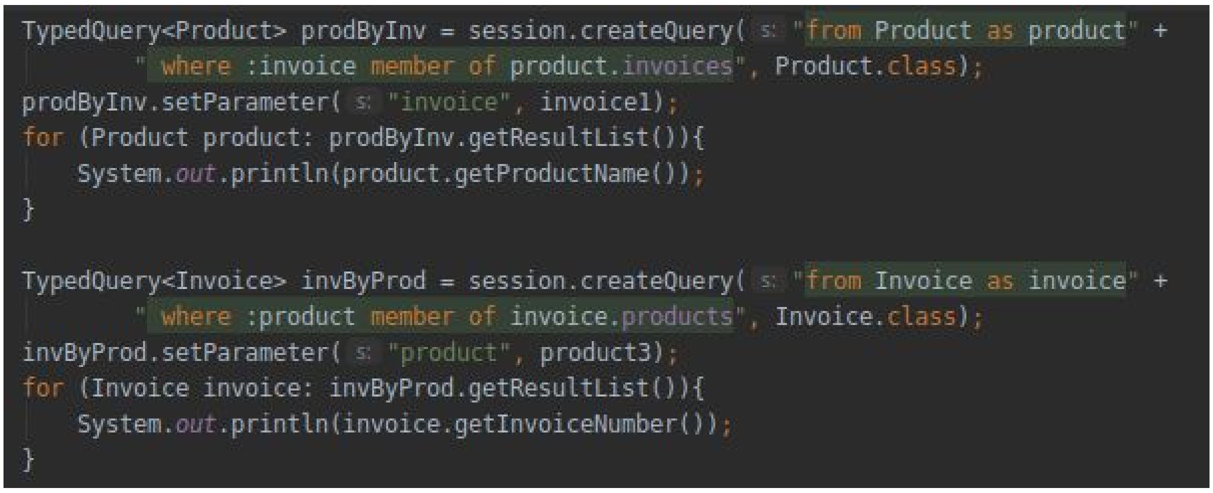
3 rows | Tx: Auto | DDL

	PRODUCTID	PRODUCTNAME	UNITSONSTOCK	CATEGORYFK	SUPPLIERFK
1	1	Prod1	9	5	4
2	2	Prod2	0	6	4
3	3	Prod3	3	5	4

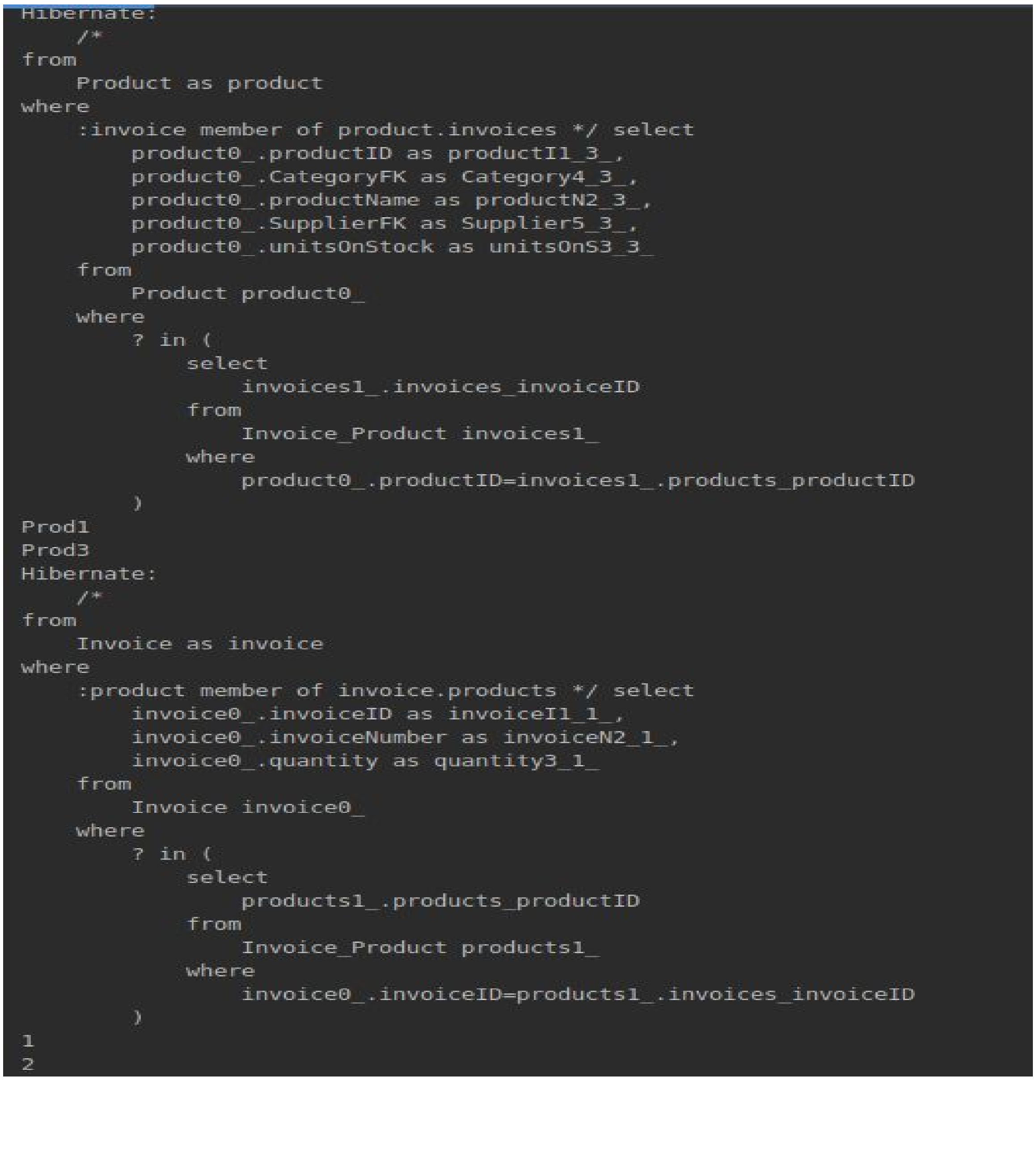


```
Hibernate:
    /* update
        Invoice */ update
            Invoice
        set
            invoiceNumber=?,
            quantity=?
        where
            invoiceID=?
Hibernate:
    /* update
        Invoice */ update
            Invoice
        set
            invoiceNumber=?,
            quantity=?
        where
            invoiceID=?
Hibernate:
    /* insert collection
        row Invoice.products */ insert
        into
            Invoice_Product
            (invoices_invoiceID, products_productID)
        values
            (?, ?)
Hibernate:
    /* insert collection
        row Invoice.products */ insert
        into
            Invoice_Product
            (invoices_invoiceID, products_productID)
        values
            (?, ?)
Hibernate:
    /* insert collection
        row Invoice.products */ insert
        into
            Invoice_Product
            (invoices_invoiceID, products_productID)
        values
            (?, ?)
```

Faktury w ramach produktów i produkty w ramach faktur:



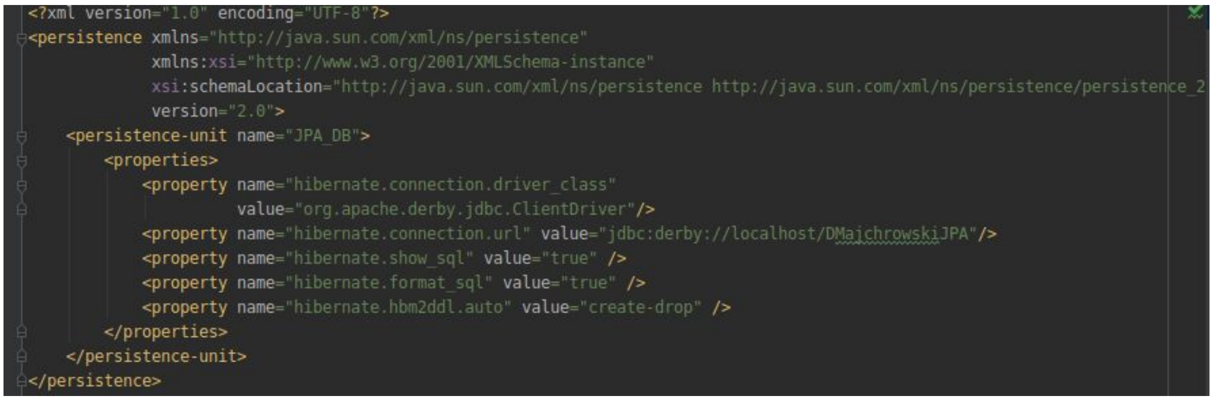
```java
TypedQuery<Product> prodByInv = session.createQuery( s: "from Product as product" +
        " where :invoice member of product.invoices", Product.class);
prodByInv.setParameter( s: "invoice", invoice1);
for (Product product: prodByInv.getResultList()){
    System.out.println(product.getProductName());
}

TypedQuery<Invoice> invByProd = session.createQuery( s: "from Invoice as invoice" +
        " where :product member of invoice.products", Invoice.class);
invByProd.setParameter( s: "product", product3);
for (Invoice invoice: invByProd.getResultList()){
    System.out.println(invoice.getInvoiceNumber());
}
```



```
Hibernate:
    /*
from
    Product as product
where
    :invoice member of product.invoices */ select
        product0_.productID as productI1_3_,
        product0_.CategoryFK as Category4_3_,
        product0_.productName as productN2_3_,
        product0_.SupplierFK as Supplier5_3_,
        product0_.unitsOnStock as unitsOnS3_3_
    from
        Product product0_
    where
        ? in (
            select
                invoices1_.invoices_invoiceID
            from
                Invoice_Product invoices1_
            where
                product0_.productID=invoices1_.products_productID
        )
Prod1
Prod3
Hibernate:
    /*
from
    Invoice as invoice
where
    :product member of invoice.products */ select
        invoice0_.invoiceID as invoiceI1_1_,
        invoice0_.invoiceNumber as invoiceN2_1_,
        invoice0_.quantity as quantity3_1_
    from
        Invoice invoice0_
    where
        ? in (
            select
                products1_.products_productID
            from
                Invoice_Product products1_
            where
                invoice0_.invoiceID=products1_.invoices_invoiceID
        )
1
2
```

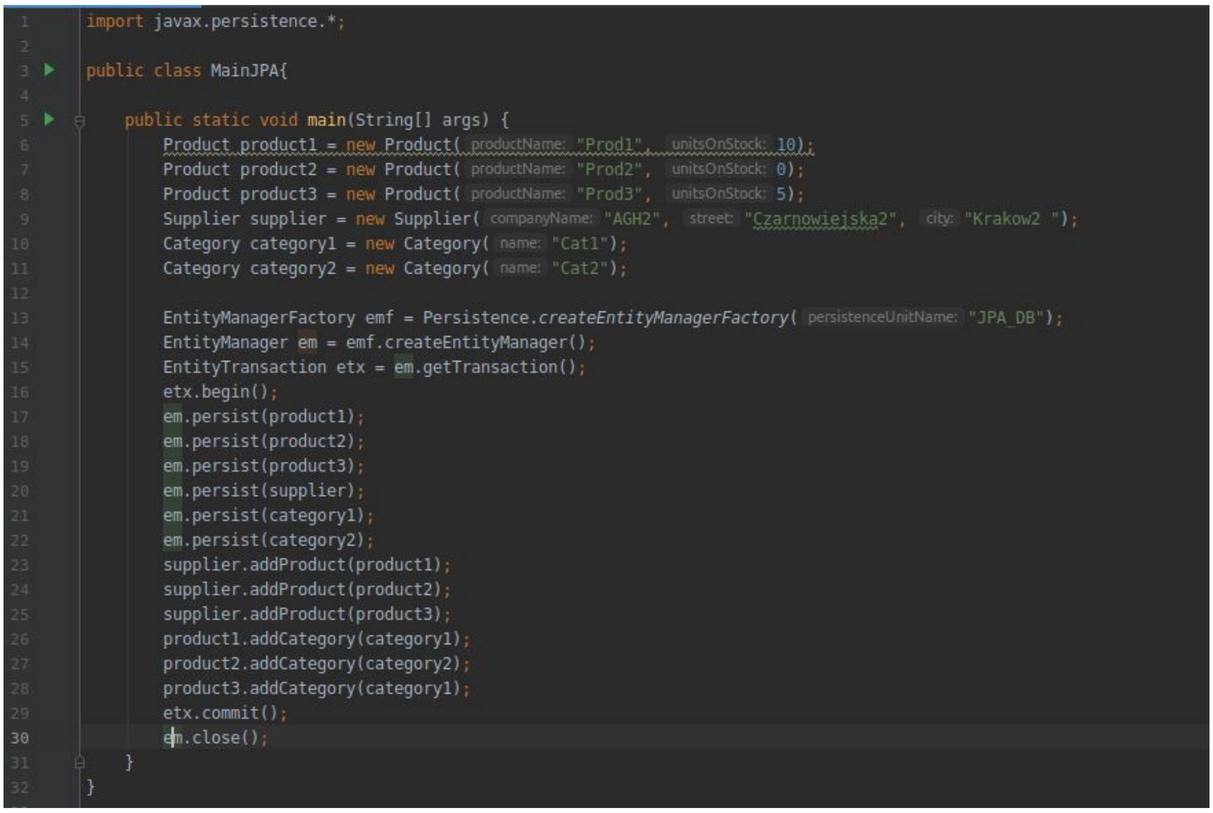
9. JPA

Do folderu src dodajemy METAINF/persistance.xml, który wygląda następująco, prersistance-unit name=”JPA_DB”, więc tak będziemy się odwoływać.

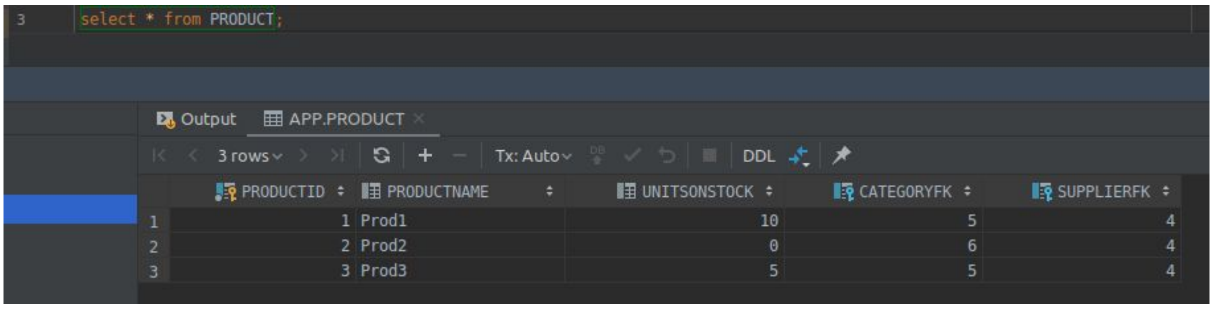


```xml
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.com/xml/ns/persistence/persistence_2
             version="2.0">
    <persistence-unit name="JPA_DB">
        <properties>
            <property name="hibernate.connection.driver_class"
                      value="org.apache.derby.jdbc.ClientDriver"/>
            <property name="hibernate.connection.url" value="jdbc:derby://localhost/DMajchrowskiJPA"/>
            <property name="hibernate.show_sql" value="true" />
            <property name="hibernate.format_sql" value="true" />
            <property name="hibernate.hbm2ddl.auto" value="create-drop" />
        </properties>
    </persistence-unit>
</persistence>
```

Dodajemy klasę MainJPA, w ktorym edytujemy punkt VI, dodawanie do bazy wygląda następująco:

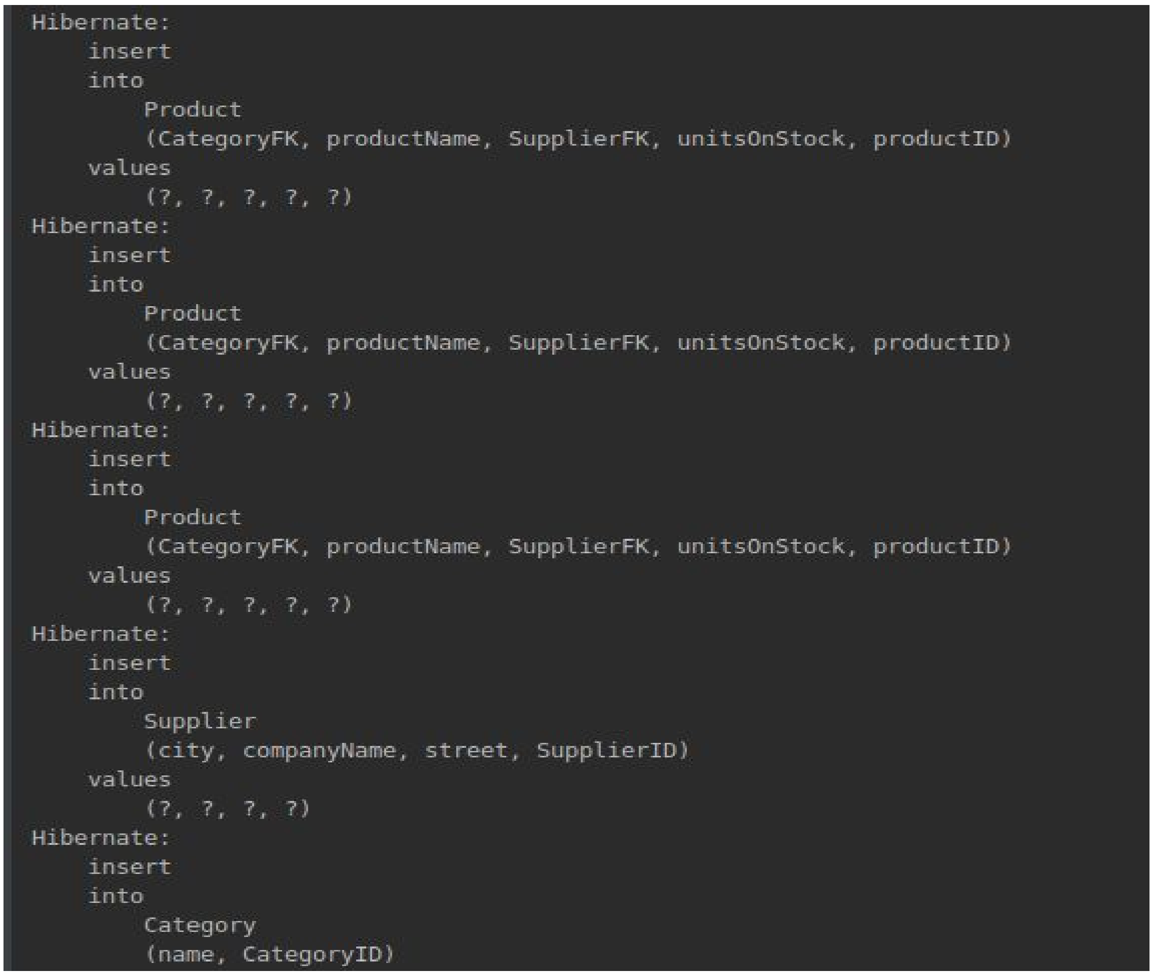


```java
import javax.persistence.*;

public class MainJPA{

    public static void main(String[] args) {
        Product product1 = new Product( productName: "Prod1",  unitsOnStock: 10);
        Product product2 = new Product( productName: "Prod2",  unitsOnStock: 0);
        Product product3 = new Product( productName: "Prod3",  unitsOnStock: 5);
        Supplier supplier = new Supplier( companyName: "AGH2",  street: "Czarnowiejska2",  city: "Krakow2 ");
        Category category1 = new Category( name: "Cat1");
        Category category2 = new Category( name: "Cat2");

        EntityManagerFactory emf = Persistence.createEntityManagerFactory( persistenceUnitName: "JPA_DB");
        EntityManager em = emf.createEntityManager();
        EntityTransaction etx = em.getTransaction();
        etx.begin();
        em.persist(product1);
        em.persist(product2);
        em.persist(product3);
        em.persist(supplier);
        em.persist(category1);
        em.persist(category2);
        supplier.addProduct(product1);
        supplier.addProduct(product2);
        supplier.addProduct(product3);
        product1.addCategory(category1);
        product2.addCategory(category2);
        product3.addCategory(category1);
        etx.commit();
        em.close();
    }
}
```



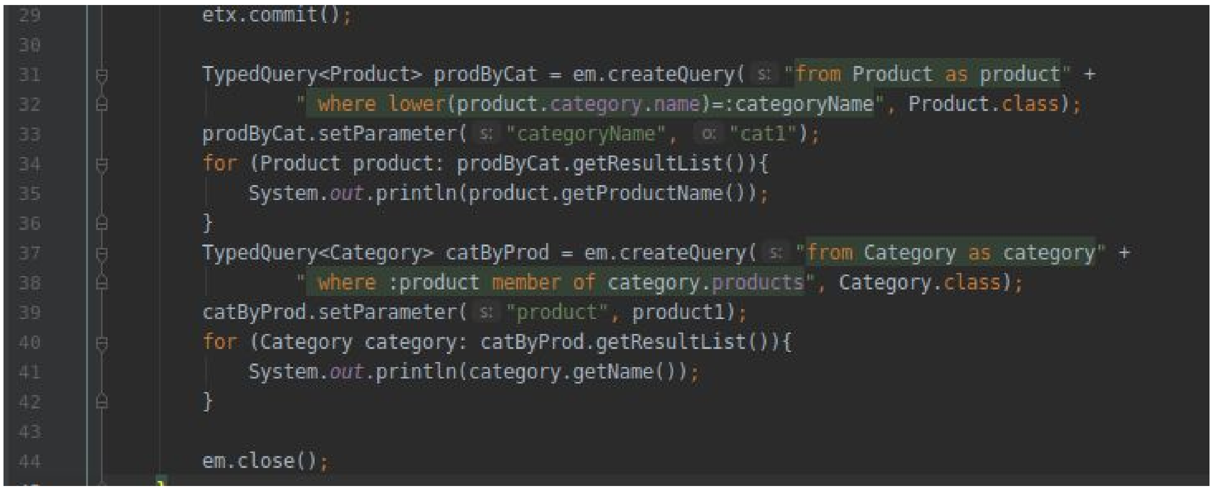
```sql
select * from PRODUCT;
```

	PRODUCTID	PRODUCTNAME	UNITSONSTOCK	CATEGORYFK	SUPPLIERFK
1	1	Prod1	10	5	4
2	2	Prod2	0	6	4
3	3	Prod3	5	5	4

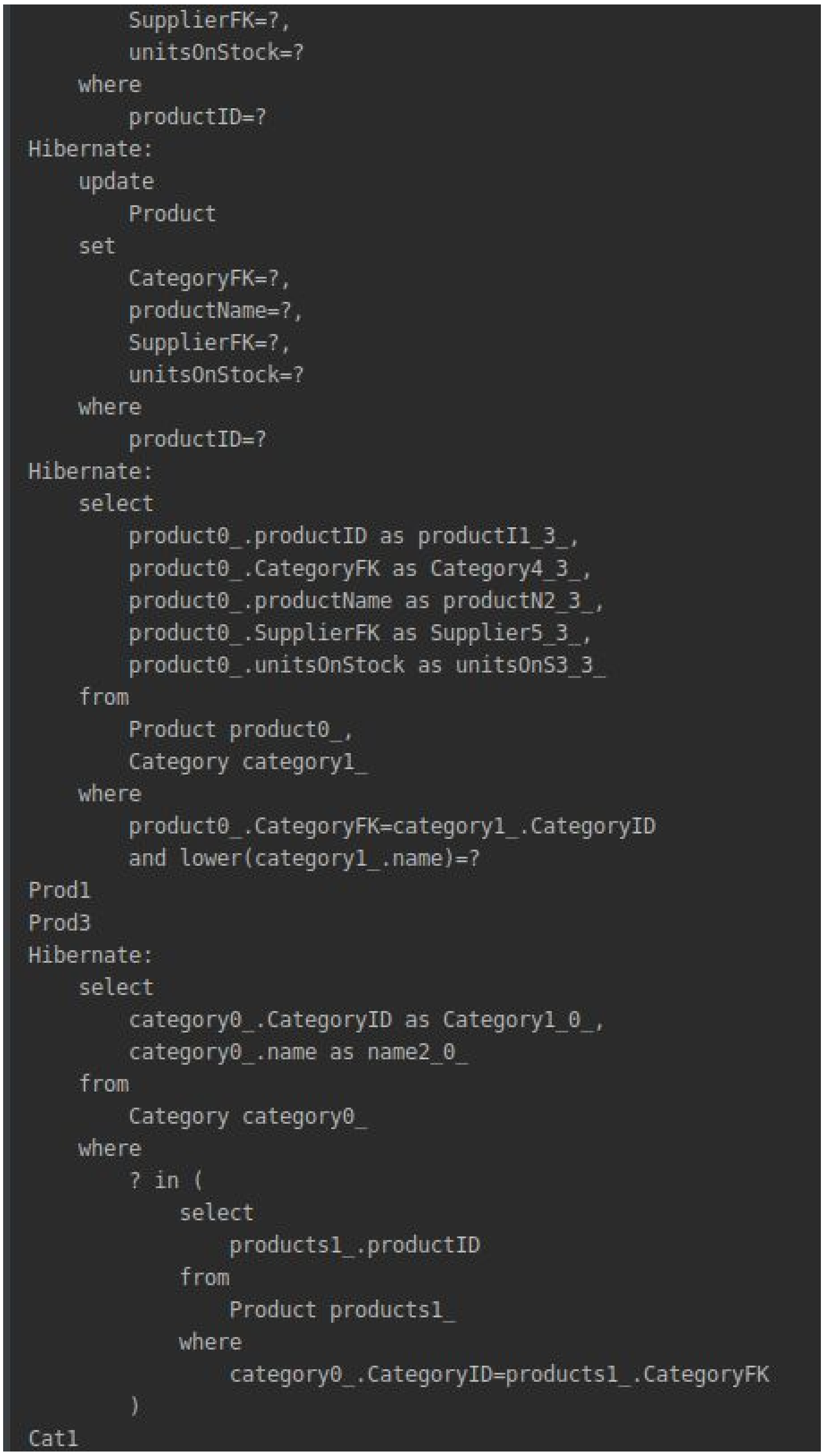


```
Hibernate:
    insert
    into
        Product
        (CategoryFK, productName, SupplierFK, unitsOnStock, productID)
    values
        (?, ?, ?, ?, ?)
Hibernate:
    insert
    into
        Product
        (CategoryFK, productName, SupplierFK, unitsOnStock, productID)
    values
        (?, ?, ?, ?, ?)
Hibernate:
    insert
    into
        Product
        (CategoryFK, productName, SupplierFK, unitsOnStock, productID)
    values
        (?, ?, ?, ?, ?)
Hibernate:
    insert
    into
        Supplier
        (city, companyName, street, SupplierID)
    values
        (?, ?, ?, ?)
Hibernate:
    insert
    into
        Category
        (name, CategoryID)
```

Natomiast pobieranie kategorii i produktów następująco



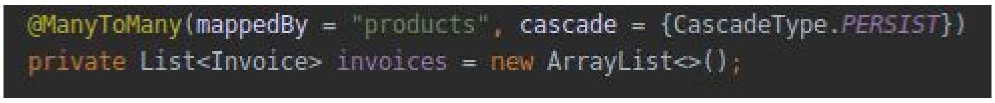
```java
        etx.commit();

        TypedQuery<Product> prodByCat = em.createQuery( s: "from Product as product" +
                " where lower(product.category.name)=:categoryName", Product.class);
        prodByCat.setParameter( s: "categoryName", o: "cat1");
        for (Product product: prodByCat.getResultList()){
            System.out.println(product.getProductName());
        }
        TypedQuery<Category> catByProd = em.createQuery( s: "from Category as category" +
                " where :product member of category.products", Category.class);
        catByProd.setParameter( s: "product", product1);
        for (Category category: catByProd.getResultList()){
            System.out.println(category.getName());
        }

        em.close();
```



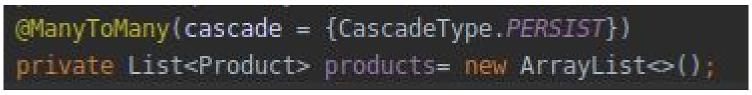
```
        SupplierFK=?,
        unitsOnStock=?
    where
        productID=?
Hibernate:
    update
        Product
    set
        CategoryFK=?,
        productName=?,
        SupplierFK=?,
        unitsOnStock=?
    where
        productID=?
Hibernate:
    select
        product0_.productID as productI1_3_,
        product0_.CategoryFK as Category4_3_,
        product0_.productName as productN2_3_,
        product0_.SupplierFK as Supplier5_3_,
        product0_.unitsOnStock as unitsOnS3_3_
    from
        Product product0_,
        Category category1_
    where
        product0_.CategoryFK=category1_.CategoryID
        and lower(category1_.name)=?
Prod1
Prod3
Hibernate:
    select
        category0_.CategoryID as Category1_0_,
        category0_.name as name2_0_
    from
        Category category0_
    where
        ? in (
            select
                products1_.productID
            from
                Product products1_
            where
                category0_.CategoryID=products1_.CategoryFK
        )
Cat1
```

Możemy zaobserowwać, że róznica między HIberneta, a JPA to plik konfiguracyjny, który już nie wymaga mapowania. Natomiast w mainie, to inne nazwy metod oraz sesji, natomiast pozostała część kodu pozostaje bez zmian.

10. Kaskady
Dodajmy kaskady dla produktów oraz faktów.

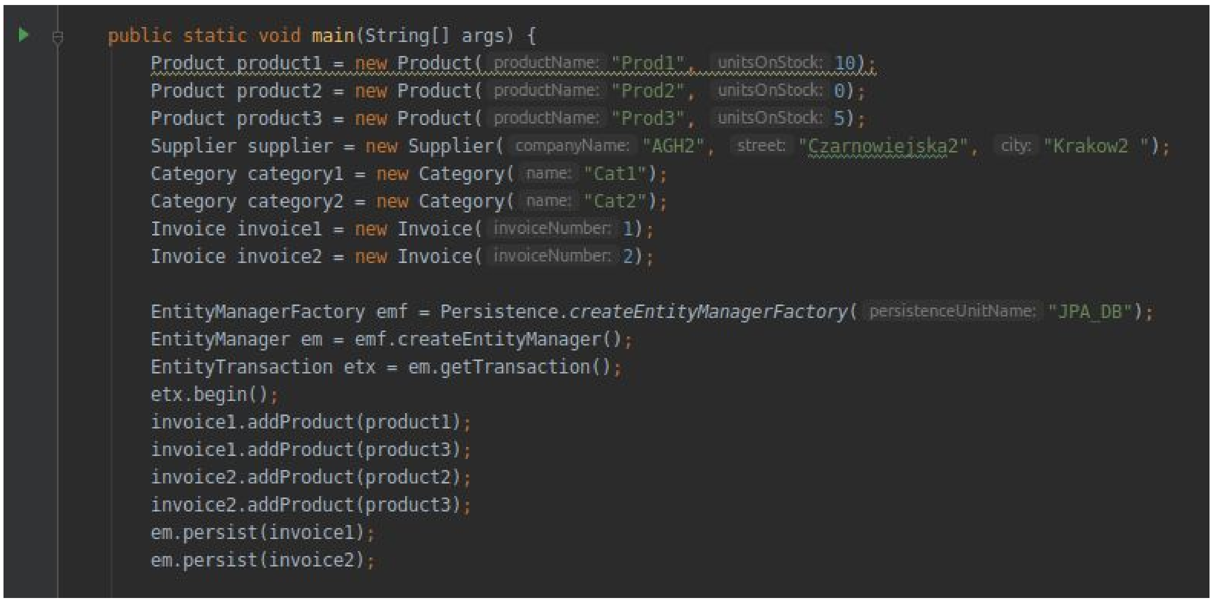


```
@ManyToMany(mappedBy = "products", cascade = {CascadeType.PERSIST})
private List<Invoice> invoices = new ArrayList<>();
```

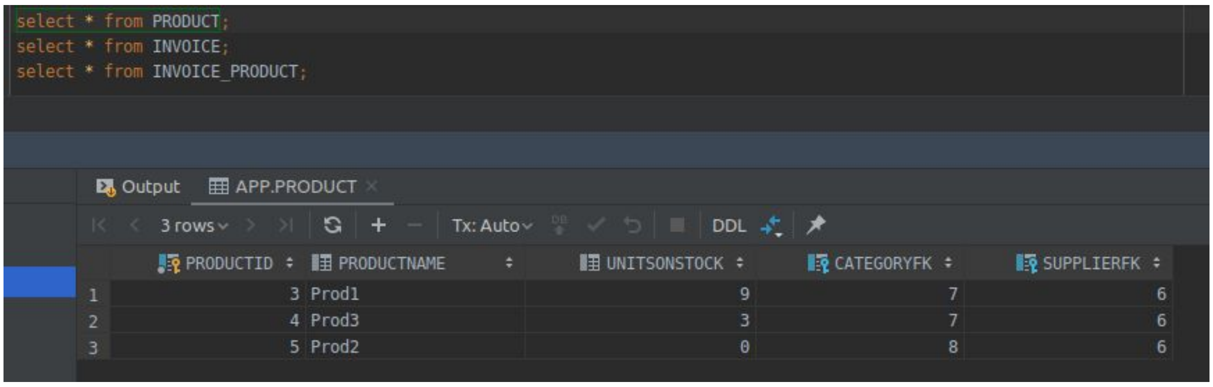


```
@ManyToMany(cascade = {CascadeType.PERSIST})
private List<Product> products= new ArrayList<>();
```

W mainie widzimy(w porównaniu do rozwiązania bez kaskady), że nie musimy już pisać em.persist({produkt{(), jeżeli zrobimy to dla faktów, persystencja zostanie wykonana kaskadowo.



```
    public static void main(String[] args) {
        Product product1 = new Product( productName: "Prod1",  unitsOnStock: 10);
        Product product2 = new Product( productName: "Prod2",  unitsOnStock: 0);
        Product product3 = new Product( productName: "Prod3",  unitsOnStock: 5);
        Supplier supplier = new Supplier( companyName: "AGH2",  street: "Czarnowiejska2",  city: "Krakow2 ");
        Category category1 = new Category( name: "Cat1");
        Category category2 = new Category( name: "Cat2");
        Invoice invoice1 = new Invoice( invoiceNumber: 1);
        Invoice invoice2 = new Invoice( invoiceNumber: 2);

        EntityManagerFactory emf = Persistence.createEntityManagerFactory( persistenceUnitName: "JPA_DB");
        EntityManager em = emf.createEntityManager();
        EntityTransaction etx = em.getTransaction();
        etx.begin();
        invoice1.addProduct(product1);
        invoice1.addProduct(product3);
        invoice2.addProduct(product2);
        invoice2.addProduct(product3);
        em.persist(invoice1);
        em.persist(invoice2);
```



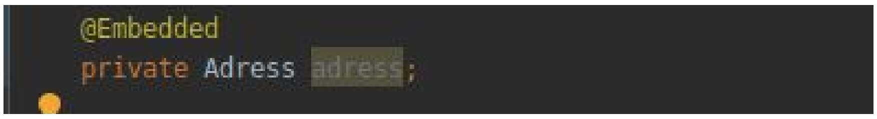
```
select * from PRODUCT;
select * from INVOICE;
select * from INVOICE_PRODUCT;
```

Output | APP.PRODUCT
3 rows | Tx: Auto | DDL

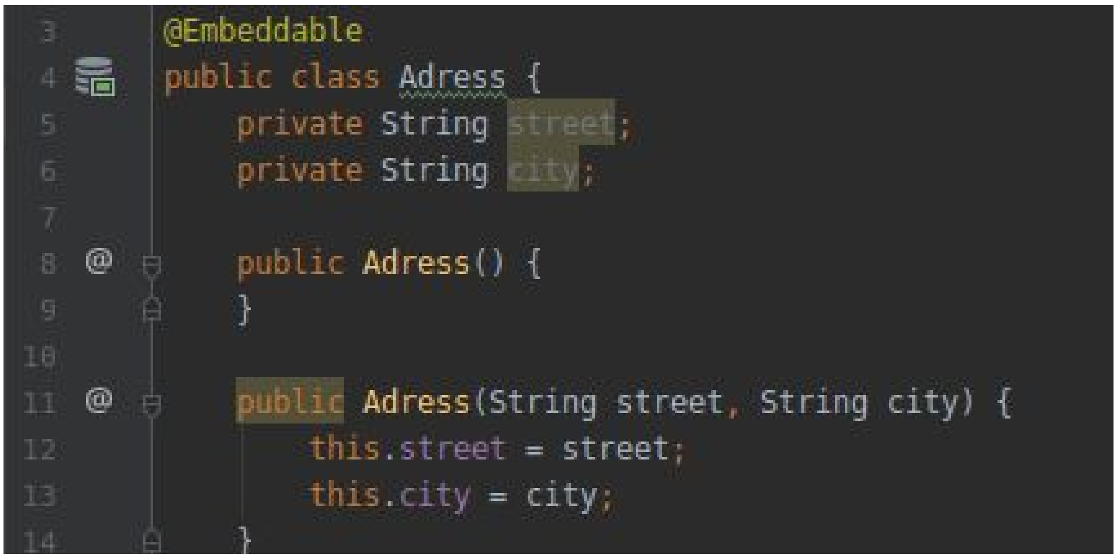
	PRODUCTID	PRODUCTNAME	UNITSONSTOCK	CATEGORYFK	SUPPLIERFK
1	3	Prod1	9	7	6
2	4	Prod3	3	7	6
3	5	Prod2	0	8	6

11. Klasy wbudowane

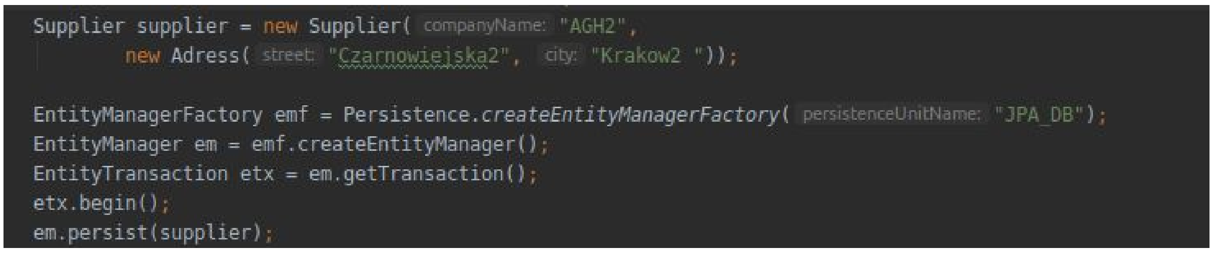
- Klasa Address zostaje wbudowana do Supplie



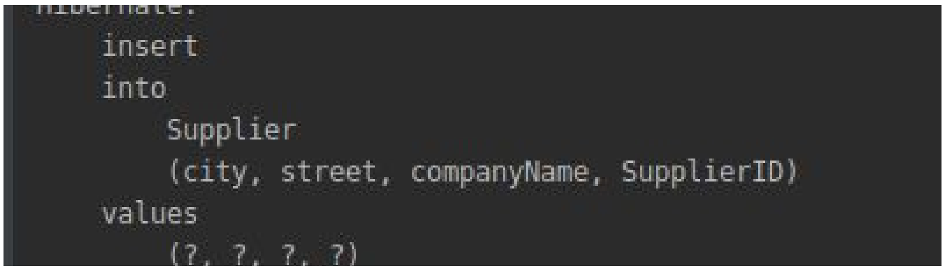
```
@Embedded
private Adress adress;
```



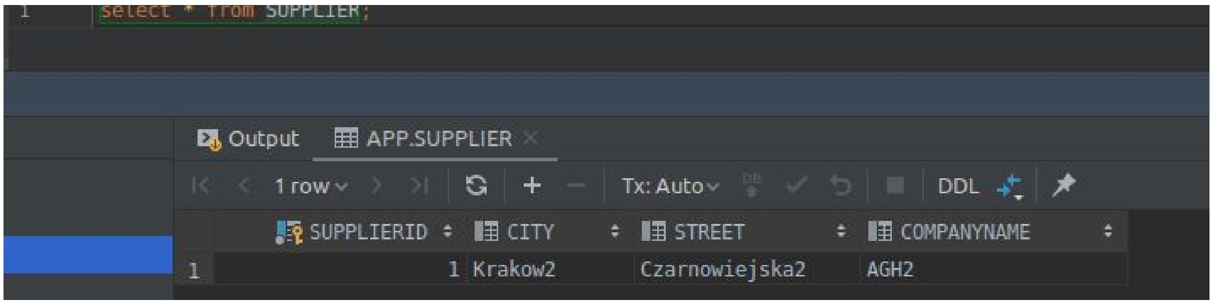
```java
@Embeddable
public class Adress {
    private String street;
    private String city;

    public Adress() {
    }

    public Adress(String street, String city) {
        this.street = street;
        this.city = city;
    }
```



```java
Supplier supplier = new Supplier( companyName: "AGH2",
        new Adress( street: "Czarnowiejska2",  city: "Krakow2 "));

EntityManagerFactory emf = Persistence.createEntityManagerFactory( persistenceUnitName: "JPA_DB");
EntityManager em = emf.createEntityManager();
EntityTransaction etx = em.getTransaction();
etx.begin();
em.persist(supplier);
```



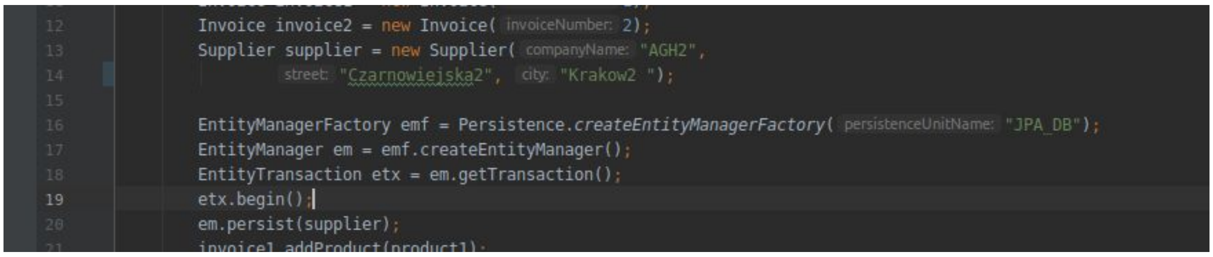
```
Hibernate:
    insert
    into
        Supplier
        (city, street, companyName, SupplierID)
    values
        (?, ?, ?, ?)
```



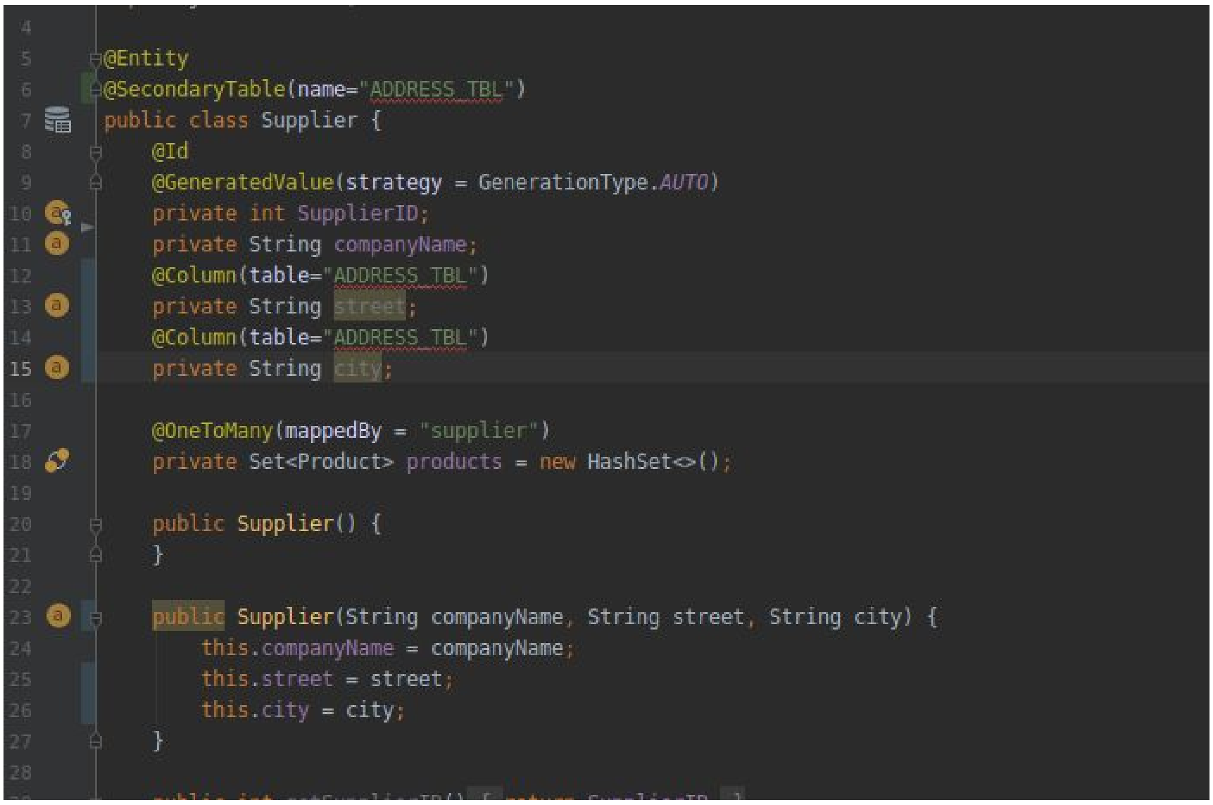
```sql
select * from SUPPLIER;
```

	SUPPLIERID	CITY	STREET	COMPANYNAME
1	1	Krakow2	Czarnowiejska2	AGH2

- Dane adresowe znajdują się w tabeli dostawców

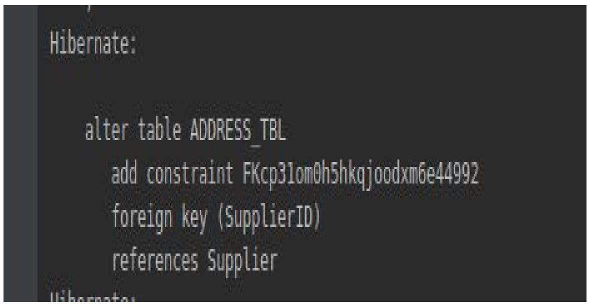


```java
Invoice invoice2 = new Invoice( invoiceNumber: 2);
Supplier supplier = new Supplier( companyName: "AGH2",
        street: "Czarnowiejska2",  city: "Krakow2 ");

EntityManagerFactory emf = Persistence.createEntityManagerFactory( persistenceUnitName: "JPA_DB");
EntityManager em = emf.createEntityManager();
EntityTransaction etx = em.getTransaction();
etx.begin();
em.persist(supplier);
invoice1.addProduct(product1);
```

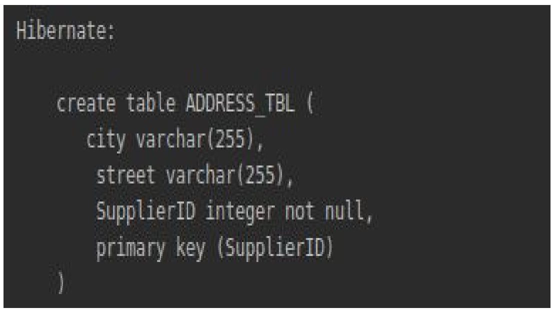


```java
@Entity
@SecondaryTable(name="ADDRESS_TBL")
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int SupplierID;
    private String companyName;
    @Column(table="ADDRESS_TBL")
    private String street;
    @Column(table="ADDRESS_TBL")
    private String city;

    @OneToMany(mappedBy = "supplier")
    private Set<Product> products = new HashSet<>();

    public Supplier() {
    }

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

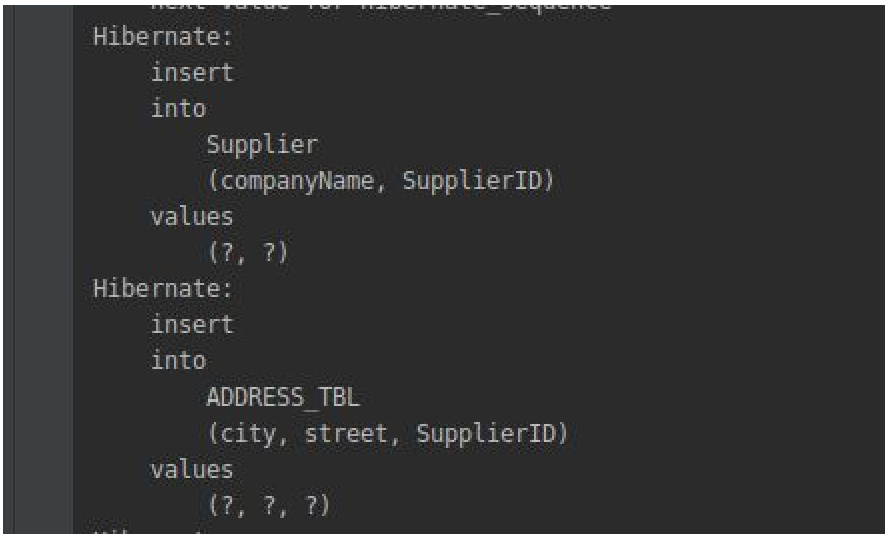
Obserwujemy logi, widzimy dodanie drugiej tabeli



```
Hibernate:

    alter table ADDRESS_TBL
       add constraint FKcp31om0h5hkqjoodxm6e44992
       foreign key (SupplierID)
       references Supplier
```

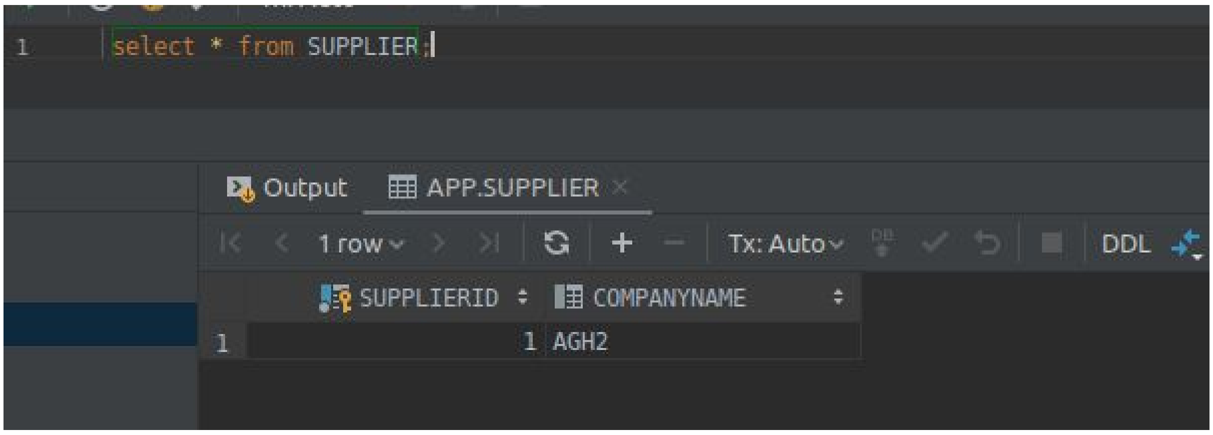


```
Hibernate:

    create table ADDRESS_TBL (
       city varchar(255),
        street varchar(255),
        SupplierID integer not null,
        primary key (SupplierID)
    )
```



```
Hibernate:
    insert
    into
        Supplier
        (companyName, SupplierID)
    values
        (?, ?)
Hibernate:
    insert
    into
        ADDRESS_TBL
        (city, street, SupplierID)
    values
        (?, ?, ?)
```

Następnie dla potwierdzenia sprawdzamy zawartość bazy danych

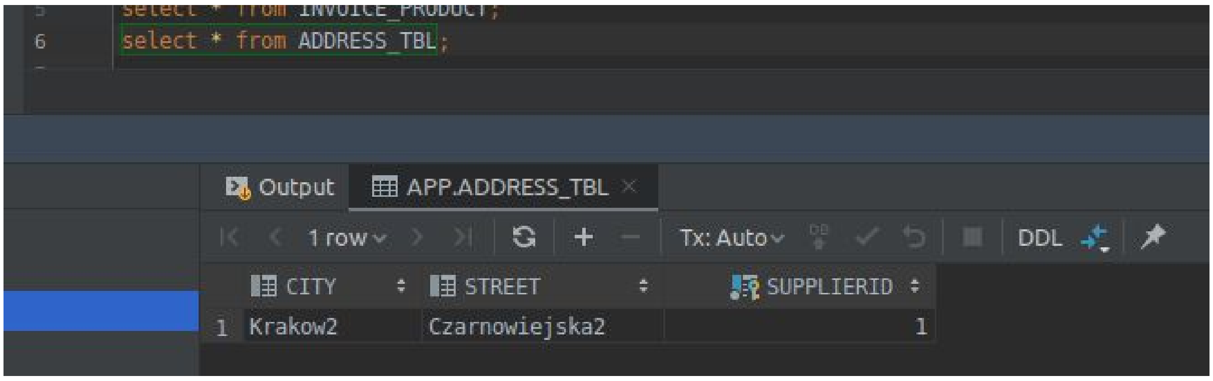


```
select * from SUPPLIER;
```

Output | APP.SUPPLIER

1 row | Tx: Auto | DDL

	SUPPLIERID	COMPANYNAME
1	1	AGH2



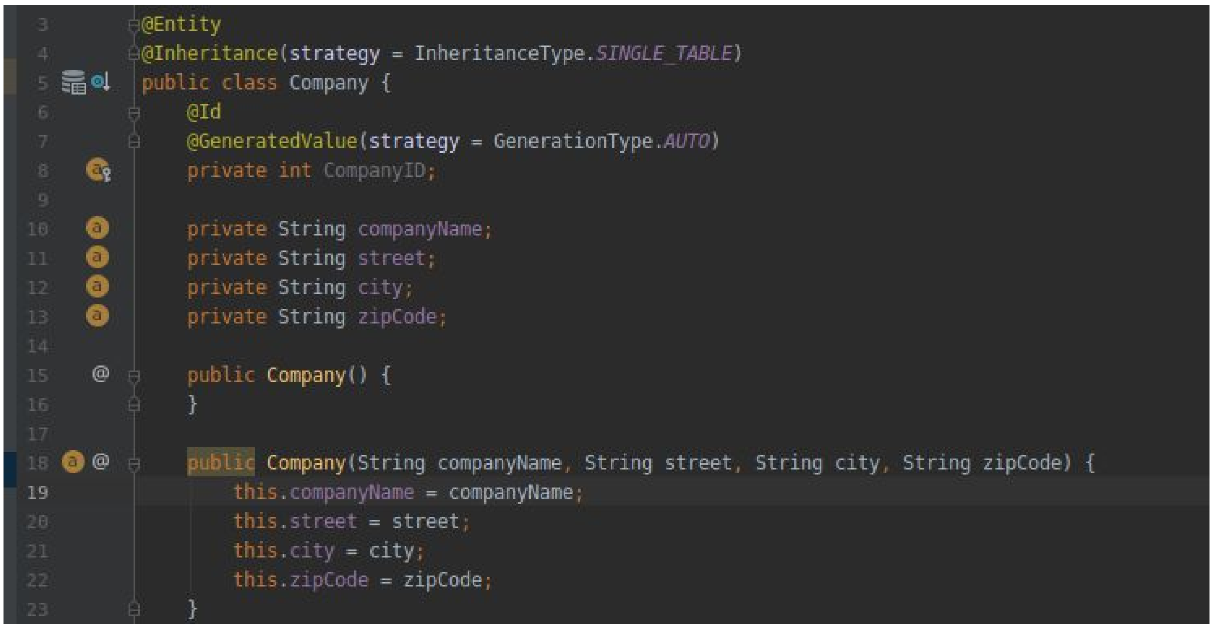
```
select * from ADDRESS_TBL;
```

Output | APP.ADDRESS_TBL

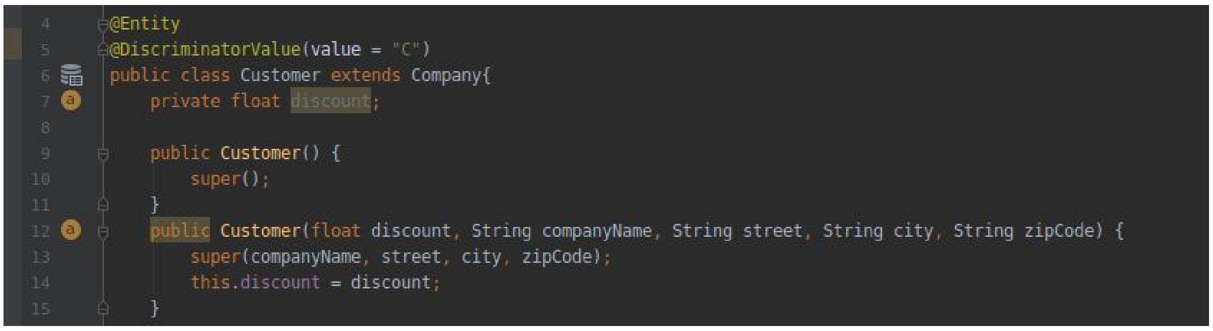
1 row | Tx: Auto | DDL

	CITY	STREET	SUPPLIERID
1	Krakow2	Czarnowiejska2	1

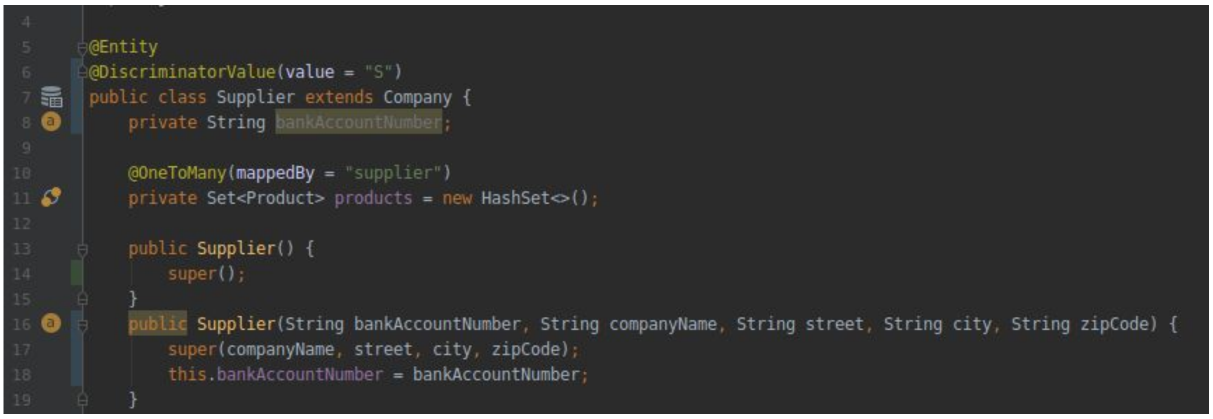
12. Dziedziczenie
    - Jedna tabela



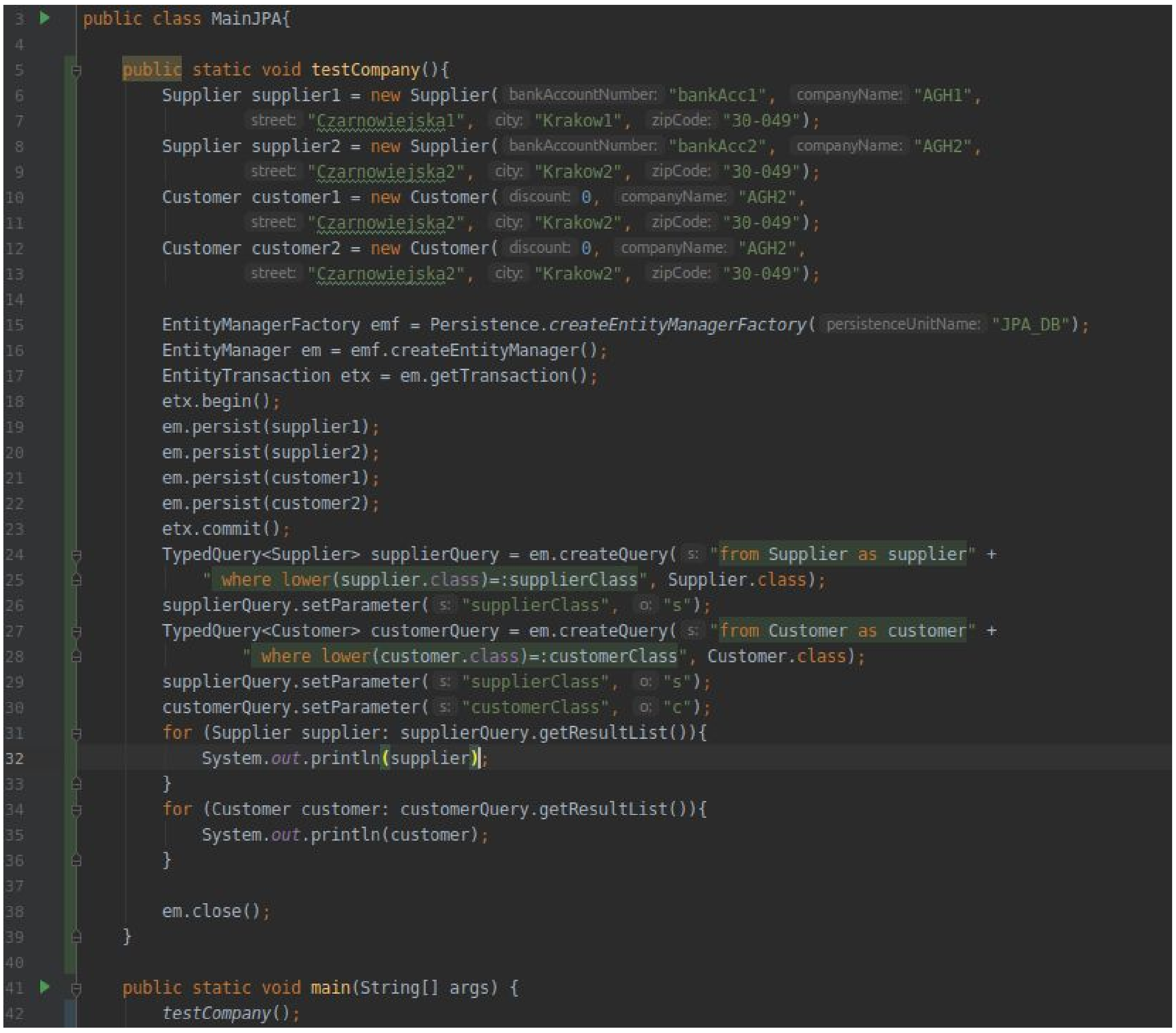
```java
@Entity
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
public class Company {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int CompanyID;

    private String companyName;
    private String street;
    private String city;
    private String zipCode;

    public Company() {
    }

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



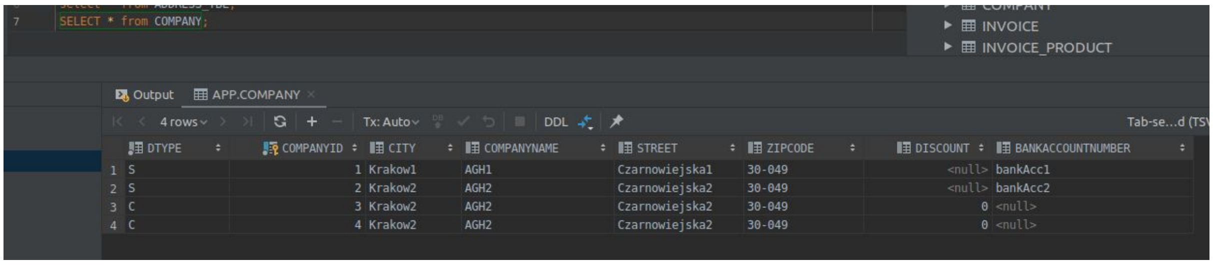
```java
@Entity
@DiscriminatorValue(value = "C")
public class Customer extends Company{
    private float discount;

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



```java

@Entity
@DiscriminatorValue(value = "S")
public class Supplier extends Company {
    private String bankAccountNumber;

    @OneToMany(mappedBy = "supplier")
    private Set<Product> products = new HashSet<>();

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

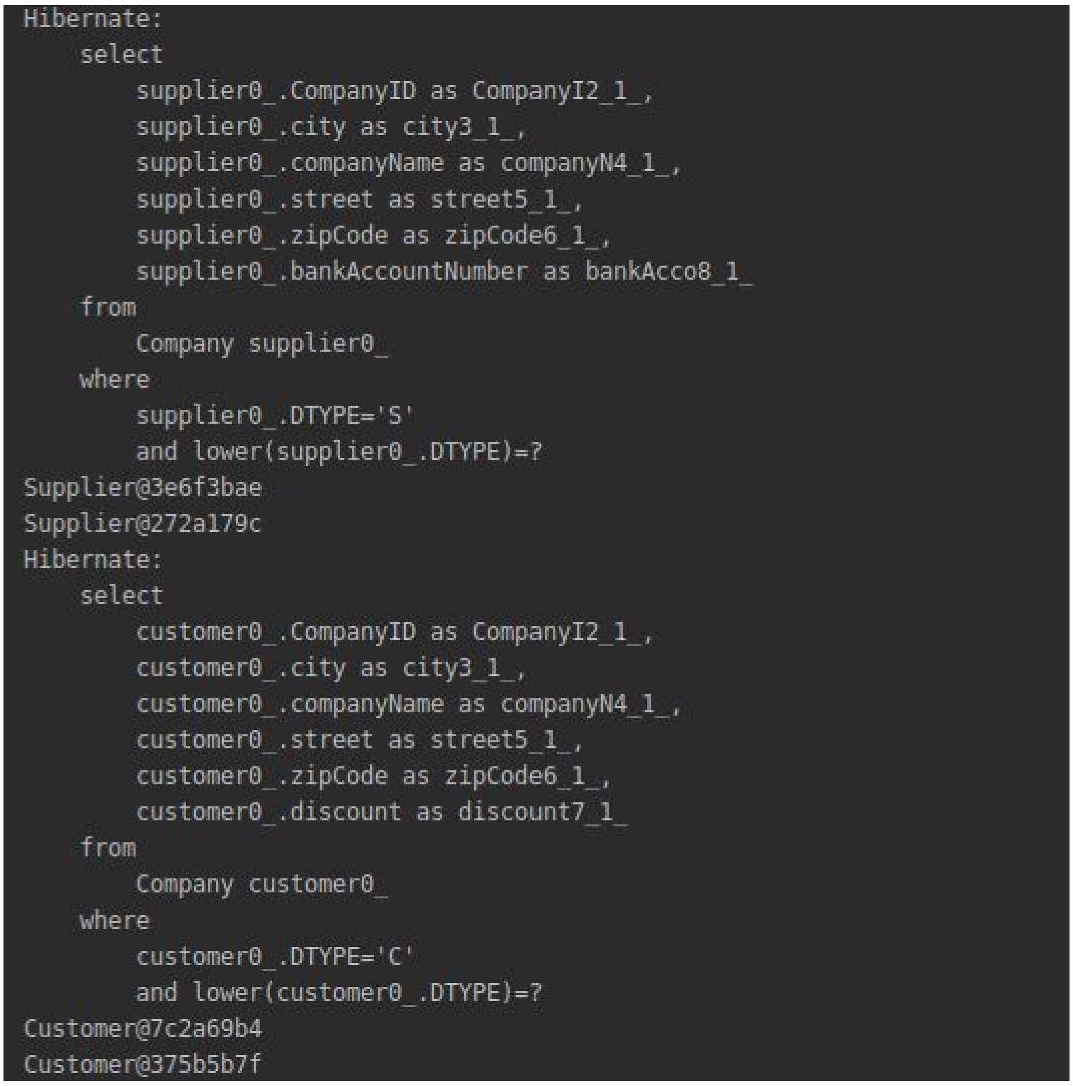


```java
public class MainJPA{

    public static void testCompany(){
        Supplier supplier1 = new Supplier( bankAccountNumber: "bankAcc1",  companyName: "AGH1",
                 street: "Czarnowiejska1",  city: "Krakow1",  zipCode: "30-049");
        Supplier supplier2 = new Supplier( bankAccountNumber: "bankAcc2",  companyName: "AGH2",
                 street: "Czarnowiejska2",  city: "Krakow2",  zipCode: "30-049");
        Customer customer1 = new Customer( discount: 0,  companyName: "AGH2",
                 street: "Czarnowiejska2",  city: "Krakow2",  zipCode: "30-049");
        Customer customer2 = new Customer( discount: 0,  companyName: "AGH2",
                 street: "Czarnowiejska2",  city: "Krakow2",  zipCode: "30-049");

        EntityManagerFactory emf = Persistence.createEntityManagerFactory( persistenceUnitName: "JPA_DB");
        EntityManager em = emf.createEntityManager();
        EntityTransaction etx = em.getTransaction();
        etx.begin();
        em.persist(supplier1);
        em.persist(supplier2);
        em.persist(customer1);
        em.persist(customer2);
        etx.commit();
        TypedQuery<Supplier> supplierQuery = em.createQuery( s: "from Supplier as supplier" +
                " where lower(supplier.class)=:supplierClass", Supplier.class);
        supplierQuery.setParameter( s: "supplierClass",  o: "s");
        TypedQuery<Customer> customerQuery = em.createQuery( s: "from Customer as customer" +
                    " where lower(customer.class)=:customerClass", Customer.class);
        supplierQuery.setParameter( s: "supplierClass",  o: "s");
        customerQuery.setParameter( s: "customerClass",  o: "c");
        for (Supplier supplier: supplierQuery.getResultList()){
            System.out.println(supplier);
        }
        for (Customer customer: customerQuery.getResultList()){
            System.out.println(customer);
        }

        em.close();
    }

    public static void main(String[] args) {
        testCompany();
```



```
SELECT * from COMPANY;
```

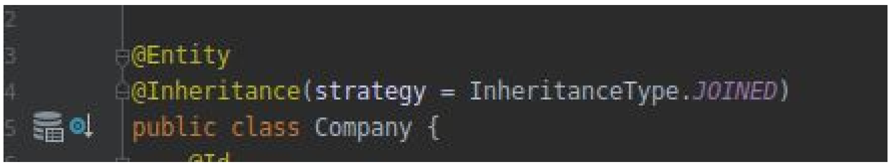
	DTYPE	COMPANYID	CITY	COMPANYNAME	STREET	ZIPCODE	DISCOUNT	BANKACCOUNTNUMBER
1	S	1	Krakow1	AGH1	Czarnowiejska1	30-049	<null>	bankAcc1
2	S	2	Krakow2	AGH2	Czarnowiejska2	30-049	<null>	bankAcc2
3	C	3	Krakow2	AGH2	Czarnowiejska2	30-049	0	<null>
4	C	4	Krakow2	AGH2	Czarnowiejska2	30-049	0	<null>



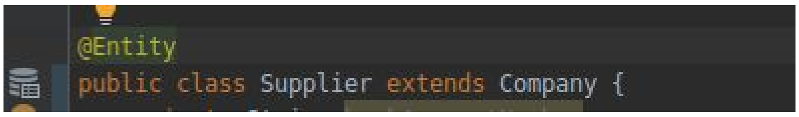
```
Hibernate: 
    select
        supplier0_.CompanyID as CompanyI2_1_,
        supplier0_.city as city3_1_,
        supplier0_.companyName as companyN4_1_,
        supplier0_.street as street5_1_,
        supplier0_.zipCode as zipCode6_1_,
        supplier0_.bankAccountNumber as bankAcco8_1_ 
    from
        Company supplier0_ 
    where
        supplier0_.DTYPE='S' 
        and lower(supplier0_.DTYPE)=?
Supplier@3e6f3bae
Supplier@272a179c
Hibernate: 
    select
        customer0_.CompanyID as CompanyI2_1_,
        customer0_.city as city3_1_,
        customer0_.companyName as companyN4_1_,
        customer0_.street as street5_1_,
        customer0_.zipCode as zipCode6_1_,
        customer0_.discount as discount7_1_ 
    from
        Company customer0_ 
    where
        customer0_.DTYPE='C' 
        and lower(customer0_.DTYPE)=?
Customer@7c2a69b4
Customer@375b5b7f
```

- Tabele Łączone

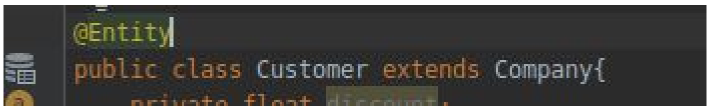
(W stosunku do joinów, zmiana strategy w Company, oraz usunięcie DiscriminatorValue)



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

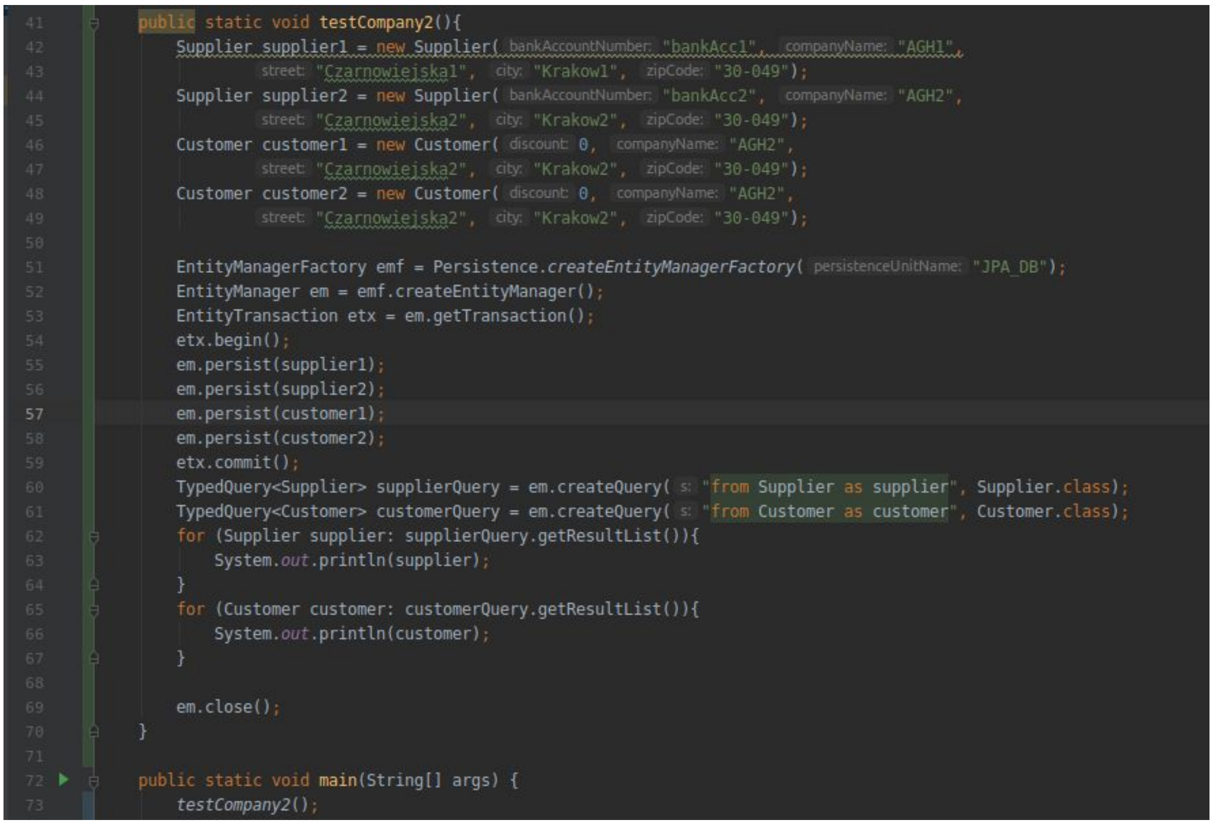


```java
@Entity
public class Supplier extends Company {
```

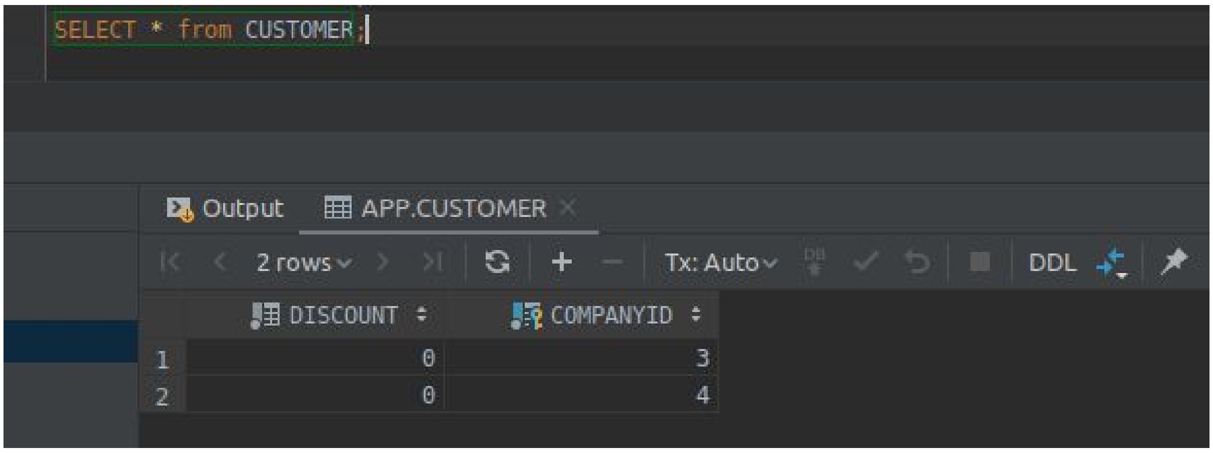


```java
@Entity
public class Customer extends Company{
```

Main dodający oraz wyciągający rekordy z bazy.

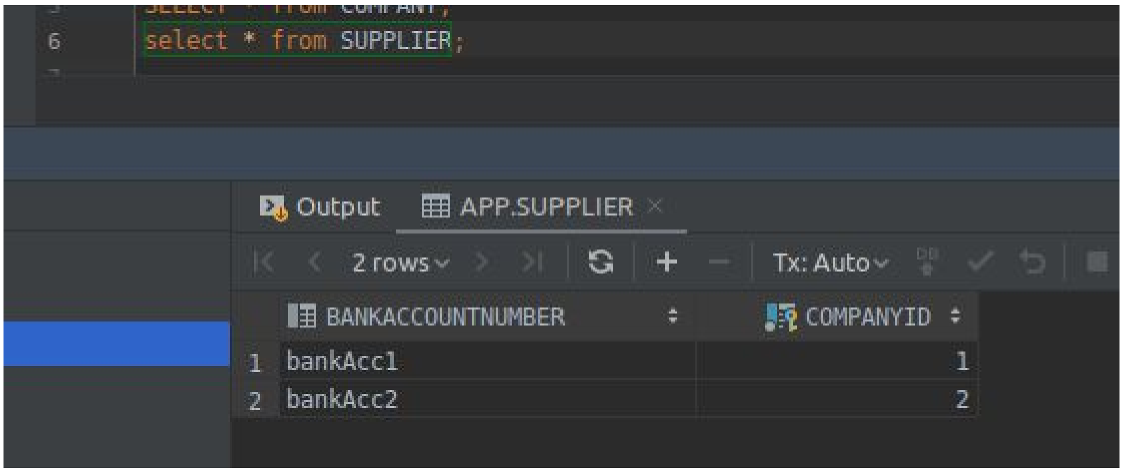


```java
    public static void testCompany2(){
        Supplier supplier1 = new Supplier( bankAccountNumber: "bankAcc1",  companyName: "AGH1",
                street: "Czarnowiejska1",  city: "Krakow1",  zipCode: "30-049");
        Supplier supplier2 = new Supplier( bankAccountNumber: "bankAcc2",  companyName: "AGH2",
                street: "Czarnowiejska2",  city: "Krakow2",  zipCode: "30-049");
        Customer customer1 = new Customer( discount: 0,  companyName: "AGH2",
                street: "Czarnowiejska2",  city: "Krakow2",  zipCode: "30-049");
        Customer customer2 = new Customer( discount: 0,  companyName: "AGH2",
                street: "Czarnowiejska2",  city: "Krakow2",  zipCode: "30-049");

        EntityManagerFactory emf = Persistence.createEntityManagerFactory( persistenceUnitName: "JPA_DB");
        EntityManager em = emf.createEntityManager();
        EntityTransaction etx = em.getTransaction();
        etx.begin();
        em.persist(supplier1);
        em.persist(supplier2);
        em.persist(customer1);
        em.persist(customer2);
        etx.commit();
        TypedQuery<Supplier> supplierQuery = em.createQuery( s: "from Supplier as supplier", Supplier.class);
        TypedQuery<Customer> customerQuery = em.createQuery( s: "from Customer as customer", Customer.class);
        for (Supplier supplier: supplierQuery.getResultList()){
            System.out.println(supplier);
        }
        for (Customer customer: customerQuery.getResultList()){
            System.out.println(customer);
        }

        em.close();
    }

    public static void main(String[] args) {
        testCompany2();
```



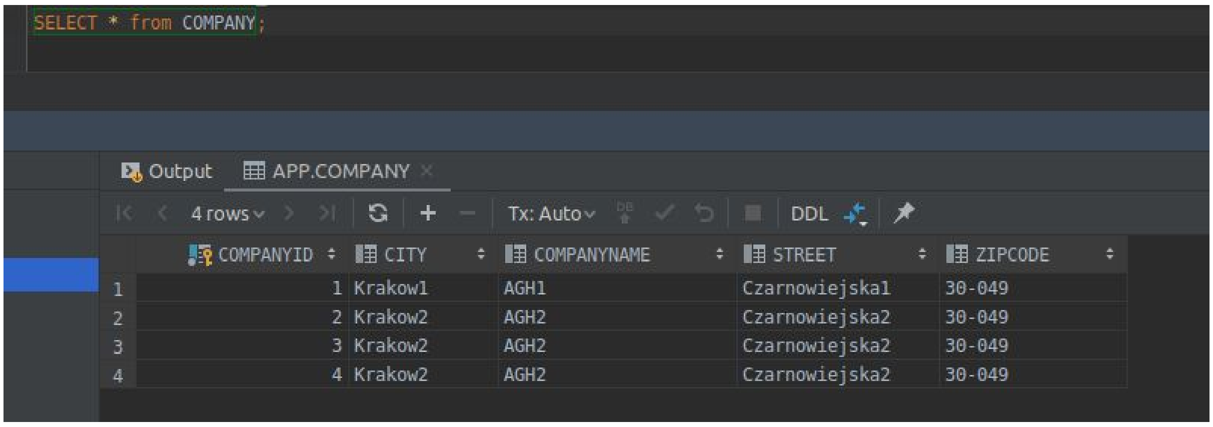
```sql
SELECT * from CUSTOMER;
```

	DISCOUNT	COMPANYID
1	0	3
2	0	4



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

	BANKACCOUNTNUMBER	COMPANYID
1	bankAcc1	1
2	bankAcc2	2

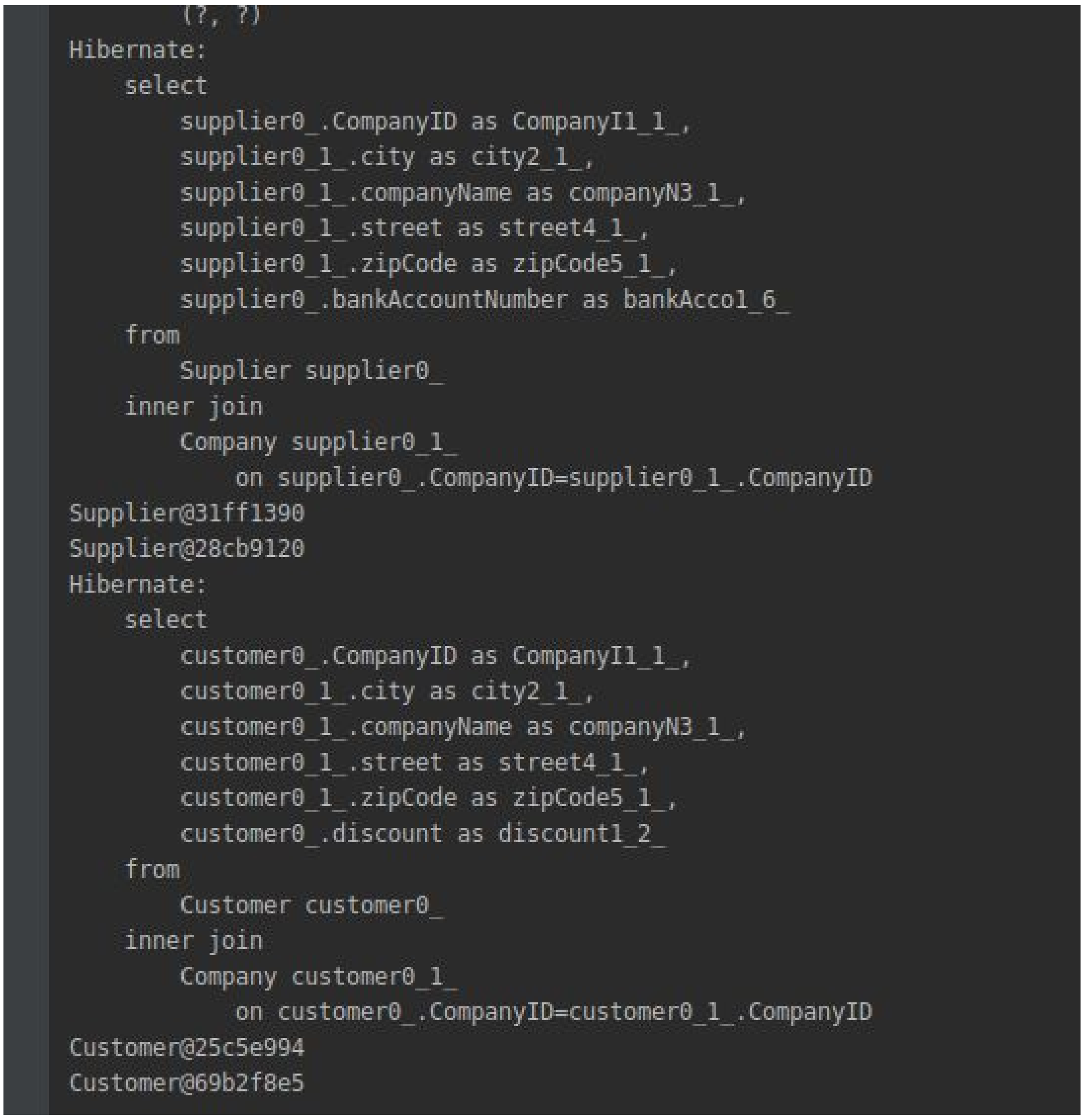


```
SELECT * from COMPANY;
```

Output | APP.COMPANY

4 rows | Tx: Auto | DDL

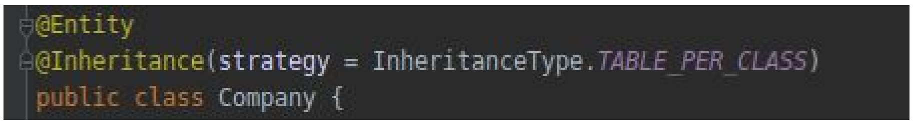
	COMPANYID	CITY	COMPANYNAME	STREET	ZIPCODE
1	1	Krakow1	AGH1	Czarnowiejska1	30-049
2	2	Krakow2	AGH2	Czarnowiejska2	30-049
3	3	Krakow2	AGH2	Czarnowiejska2	30-049
4	4	Krakow2	AGH2	Czarnowiejska2	30-049



```
            (?, ?)
Hibernate: 
    select
        supplier0_.CompanyID as CompanyI1_1_,
        supplier0_1_.city as city2_1_,
        supplier0_1_.companyName as companyN3_1_,
        supplier0_1_.street as street4_1_,
        supplier0_1_.zipCode as zipCode5_1_,
        supplier0_.bankAccountNumber as bankAcco1_6_
    from
        Supplier supplier0_
    inner join
        Company supplier0_1_
            on supplier0_.CompanyID=supplier0_1_.CompanyID
Supplier@31ff1390
Supplier@28cb9120
Hibernate: 
    select
        customer0_.CompanyID as CompanyI1_1_,
        customer0_1_.city as city2_1_,
        customer0_1_.companyName as companyN3_1_,
        customer0_1_.street as street4_1_,
        customer0_1_.zipCode as zipCode5_1_,
        customer0_.discount as discount1_2_
    from
        Customer customer0_
    inner join
        Company customer0_1_
            on customer0_.CompanyID=customer0_1_.CompanyID
Customer@25c5e994
Customer@69b2f8e5
```

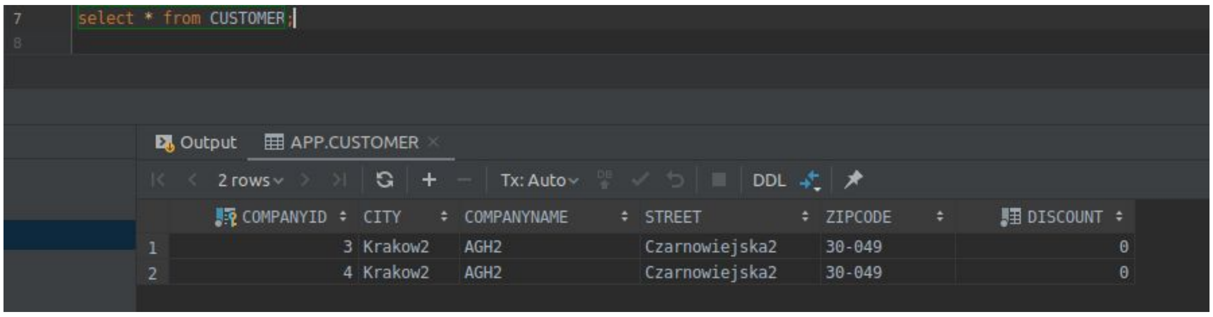
- Tabela na klasę

(Jedyna zmiana to zmiana strategii w stosunku do tabeli łączonych)



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

(Main bez zmian w stosunku do tabeli łączonych)

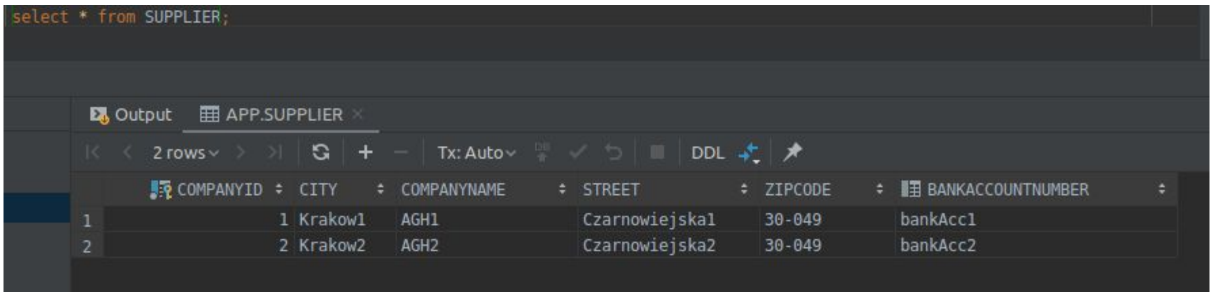


```sql
select * from CUSTOMER;
```

Output | APP.CUSTOMER

2 rows | Tx: Auto | DDL

	COMPANYID	CITY	COMPANYNAME	STREET	ZIPCODE	DISCOUNT
1	3	Krakow2	AGH2	Czarnowiejska2	30-049	0
2	4	Krakow2	AGH2	Czarnowiejska2	30-049	0



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

Output | APP.SUPPLIER

2 rows | Tx: Auto | DDL

	COMPANYID	CITY	COMPANYNAME	STREET	ZIPCODE	BANKACCOUNTNUMBER
1	1	Krakow1	AGH1	Czarnowiejska1	30-049	bankAcc1
2	2	Krakow2	AGH2	Czarnowiejska2	30-049	bankAcc2

Tym samym wszystkie zadania laboratoryjne, jako zadanie domowe tworzymy aplikację Webową do zamawiania produktów.

Aplikacja do zamawiania produktów