Raport z wykonania ćwiczenia Hibernate

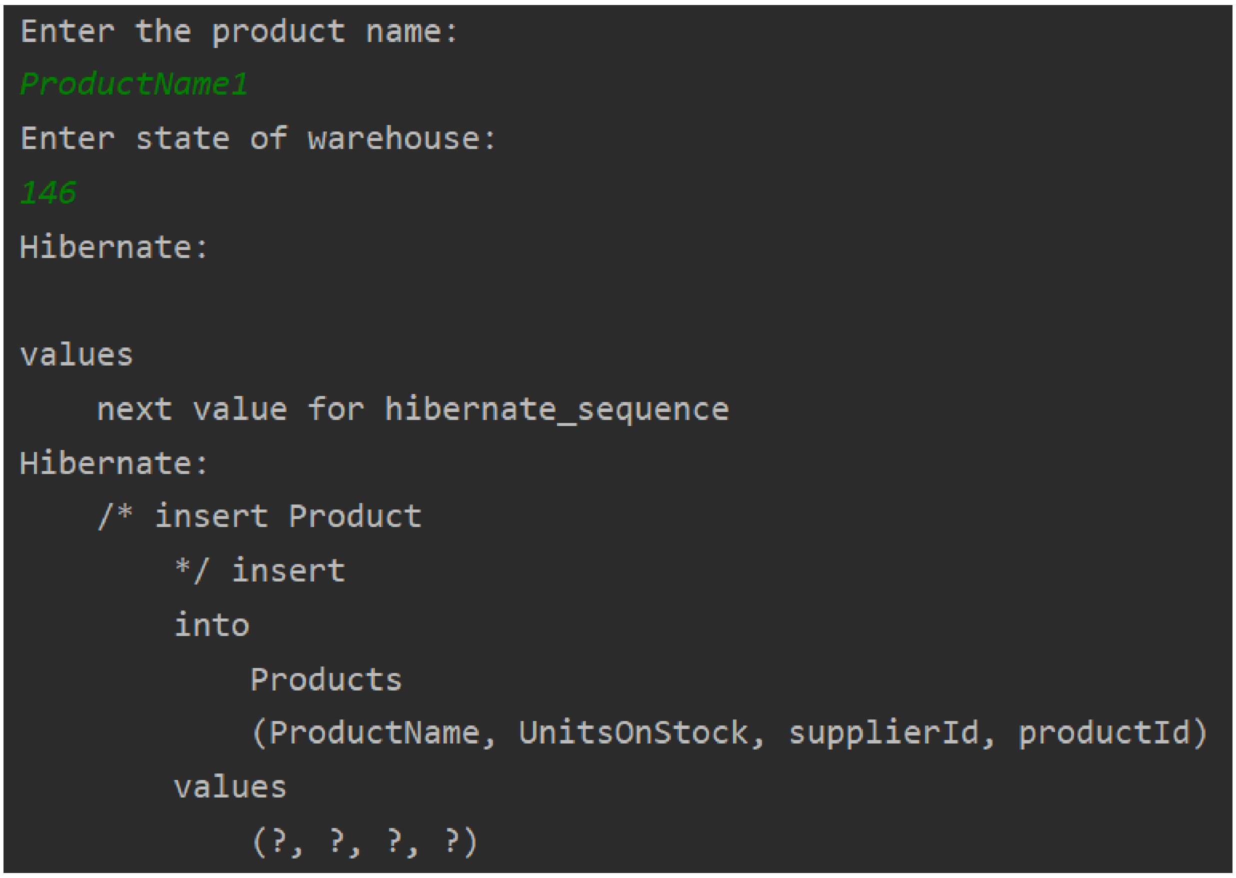
Jakub Płotnikowski

Listopad 2019r.

Spis treści

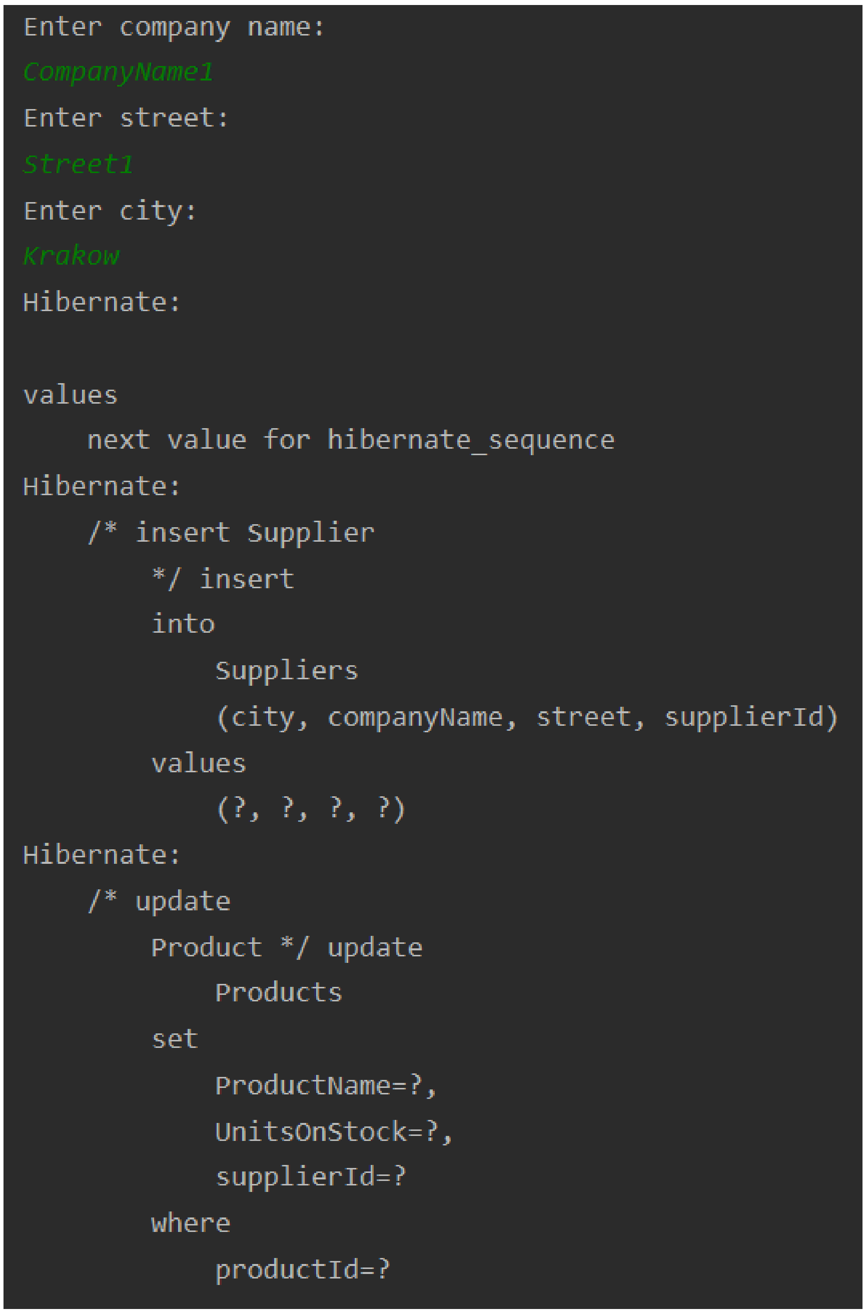
1 III. Modyfikacja modelu - wprowadzenie Dostawcy (dokończenie z zajęć)

1.1 Moment dodawania produktu



```
Enter the product name:
ProductName1
Enter state of warehouse:
146
Hibernate:

values
    next value for hibernate_sequence
Hibernate:
    /* insert Product
        */ insert
        into
            Products
            (ProductName, UnitsOnStock, supplierId, productId)
        values
            (?, ?, ?, ?)
```

1.2 Moment dodawania dostawcy



```
Enter company name:
CompanyName1
Enter street:
Street1
Enter city:
Krakow
Hibernate:

values
    next value for hibernate_sequence
Hibernate:
    /* insert Supplier
        */ insert
        into
            Suppliers
            (city, companyName, street, supplierId)
        values
            (?, ?, ?, ?)
Hibernate:
    /* update
        Product */ update
            Products
        set
            ProductName=?,
            UnitsOnStock=?,
            supplierId=?
        where
            productId=?
```

1.3 Kod klasy Product

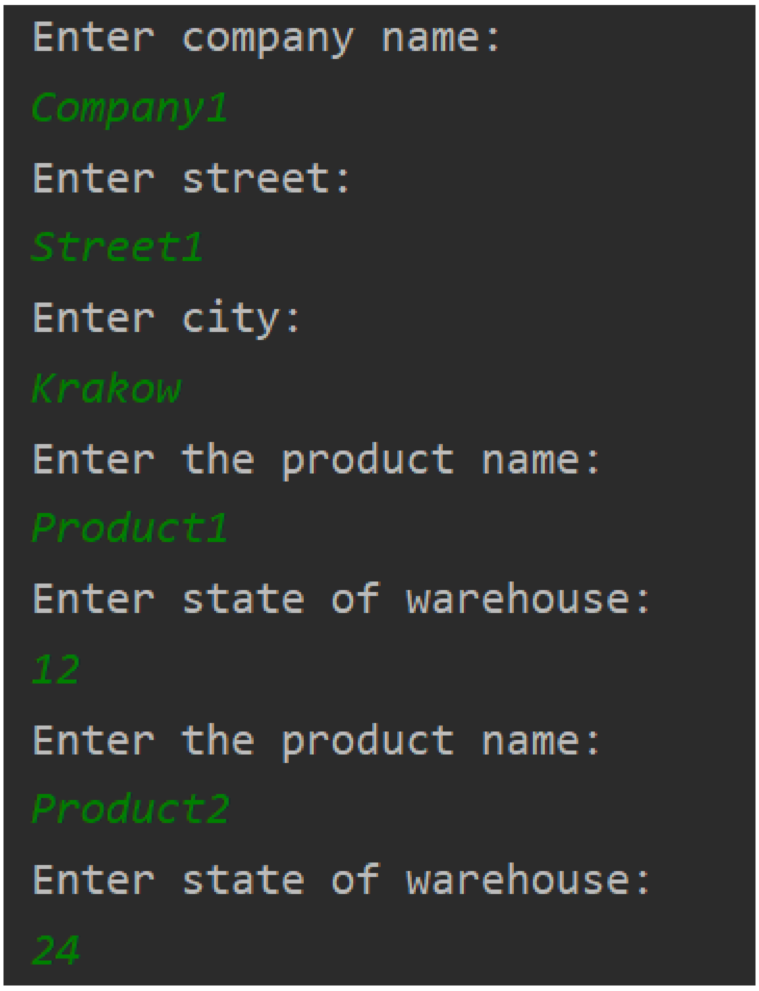
```
import lombok.*;

import javax.persistence.*;

@Entity(name = "Products")
@Setter
@NoArgsConstructor
public class Product {

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

    private String ProductName;
    private int UnitsOnStock;

    @ManyToOne
    @JoinColumn(name = "supplierId")
    Supplier supplier;

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

1.4 Kod klasy Supplier

```java
import lombok.AllArgsConstructor;
import lombok.NoArgsConstructor;

import javax.persistence.*;

@Entity
@Table(name = "Suppliers")
@Getter
@Setter
@NoArgsConstructor
public class Supplier {

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

    private String companyName;

    private String street;

    private String city;

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

*/
```

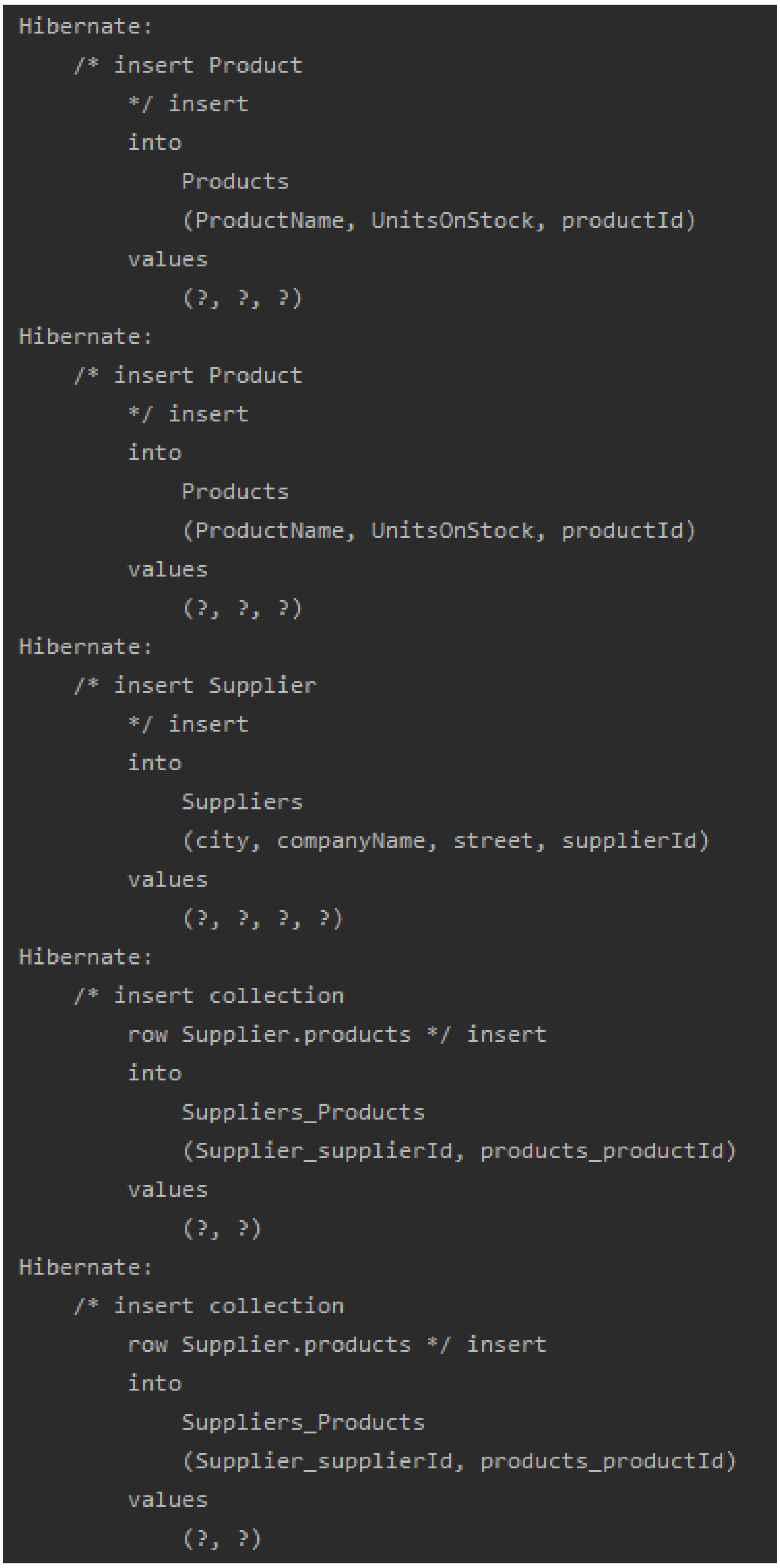
2 IVa. Odwrócenie relacji - wykorzystanie tabeli łącznikowej

2.1 Pobranie informacji o dostawcy i produktach oraz dodanie ich do bazy



```
Enter company name:
Company1
Enter street:
Street1
Enter city:
Krakow
Enter the product name:
Product1
Enter state of warehouse:
12
Enter the product name:
Product2
Enter state of warehouse:
24
```

2.2 Zapytania wykonane przez Hibernate



```
Hibernate:
    /* insert Product
        */ insert
        into
            Products
            (ProductName, UnitsOnStock, productId)
        values
            (?, ?, ?)
Hibernate:
    /* insert Product
        */ insert
        into
            Products
            (ProductName, UnitsOnStock, productId)
        values
            (?, ?, ?)
Hibernate:
    /* insert Supplier
        */ insert
        into
            Suppliers
            (city, companyName, street, supplierId)
        values
            (?, ?, ?, ?)
Hibernate:
    /* insert collection
        row Supplier.products */ insert
        into
            Suppliers_Products
            (Supplier_supplierId, products_productId)
        values
            (?, ?)
Hibernate:
    /* insert collection
        row Supplier.products */ insert
        into
            Suppliers_Products
            (Supplier_supplierId, products_productId)
        values
            (?, ?)
```

2.3 Kod klasy Product

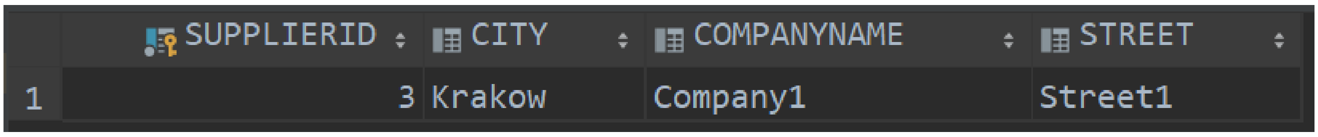
```
import lombok.*;

import javax.persistence.*;

@Entity(name = "Products")
@Getter
@Setter
@NoArgsConstructor
public class Product {

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

    private String ProductName;
    private int UnitsOnStock;

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

*/
```

2.4 Kod klasy Supplier

```java
import lombok.NoArgsConstructor;

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

@Entity
@Table(name = "Suppliers")
@Getter
@Setter
@NoArgsConstructor
public class Supplier {

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

    private String companyName;

    private String street;

    private String city;

    @OneToMany
    Set<Product> products = new HashSet<>();

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

*/
```

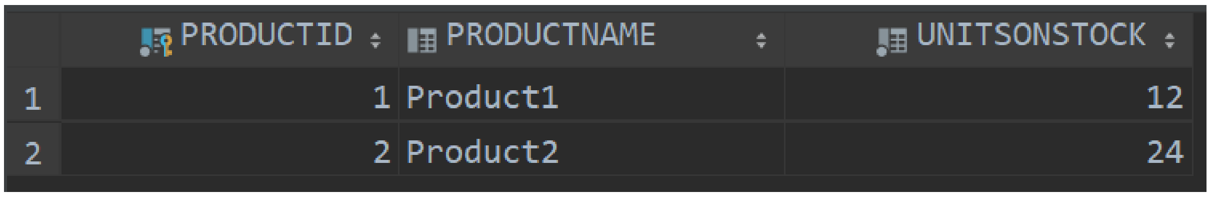
2.5 Wyniki SELECT * z poszczególnych tabel

2.5.1 Tabela Suppliers



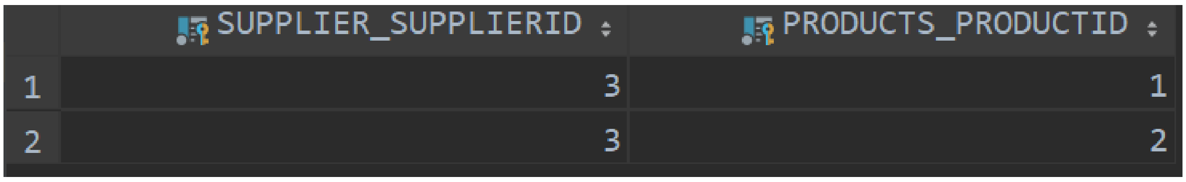
	SUPPLIERID	CITY	COMPANYNAME	STREET
1	3	Krakow	Company1	Street1

2.5.2 Tabela Products



	PRODUCTID	PRODUCTNAME	UNITSONSTOCK
1	1	Product1	12
2	2	Product2	24

2.5.3 Tabela łącznikowa



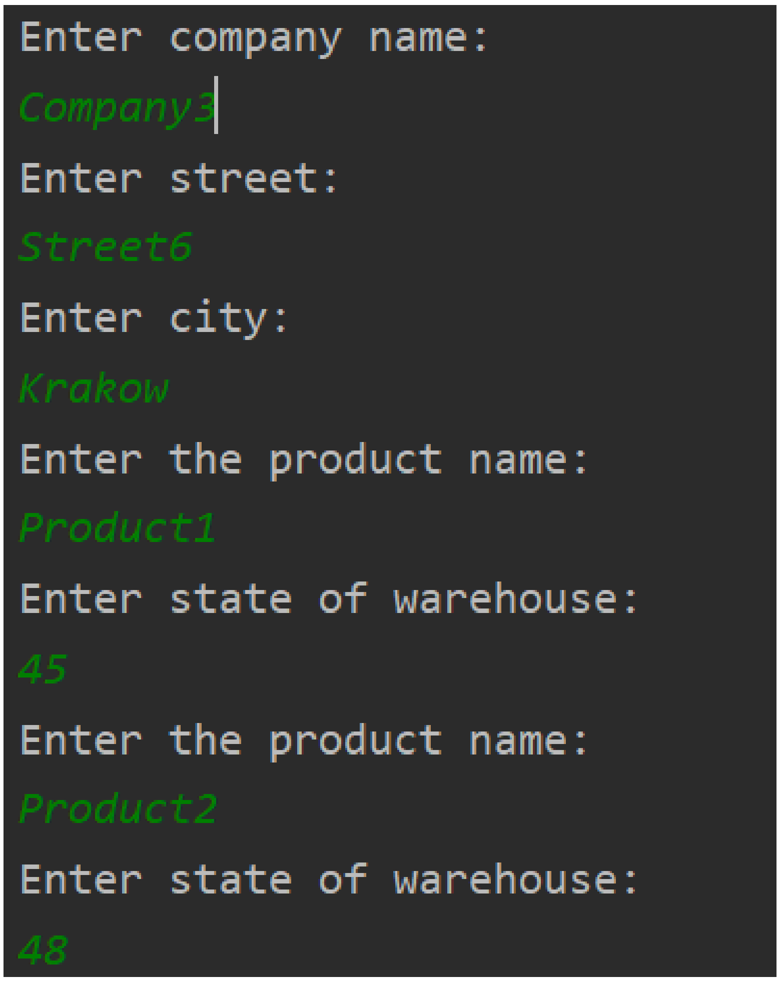
	SUPPLIER_SUPPLIERID	PRODUCTS_PRODUCTID
1	3	1
2	3	2

2.6 Kod DDL

```sql
create schema APP;

create table PRODUCTS
(
   PRODUCTID INTEGER not null
      primary key,
   PRODUCTNAME VARCHAR(255),
   UNITSONSTOCK INTEGER not null
);

create table SUPPLIERS
(
   SUPPLIERID INTEGER not null
      primary key,
   CITY VARCHAR(255),
   COMPANYNAME VARCHAR(255),
   STREET VARCHAR(255)
);

create table SUPPLIERS_PRODUCTS
(
   SUPPLIER_SUPPLIERID INTEGER not null
      constraint FK1LCEBGBKYY9X3G5OVUJTBFLFA
         references SUPPLIERS,
   PRODUCTS_PRODUCTID INTEGER not null
      unique
      constraint FKFKUOWRIMPXYNTIT2EBU11K0JF
         references PRODUCTS,
   primary key (SUPPLIER_SUPPLIERID, PRODUCTS_PRODUCTID)
);
```

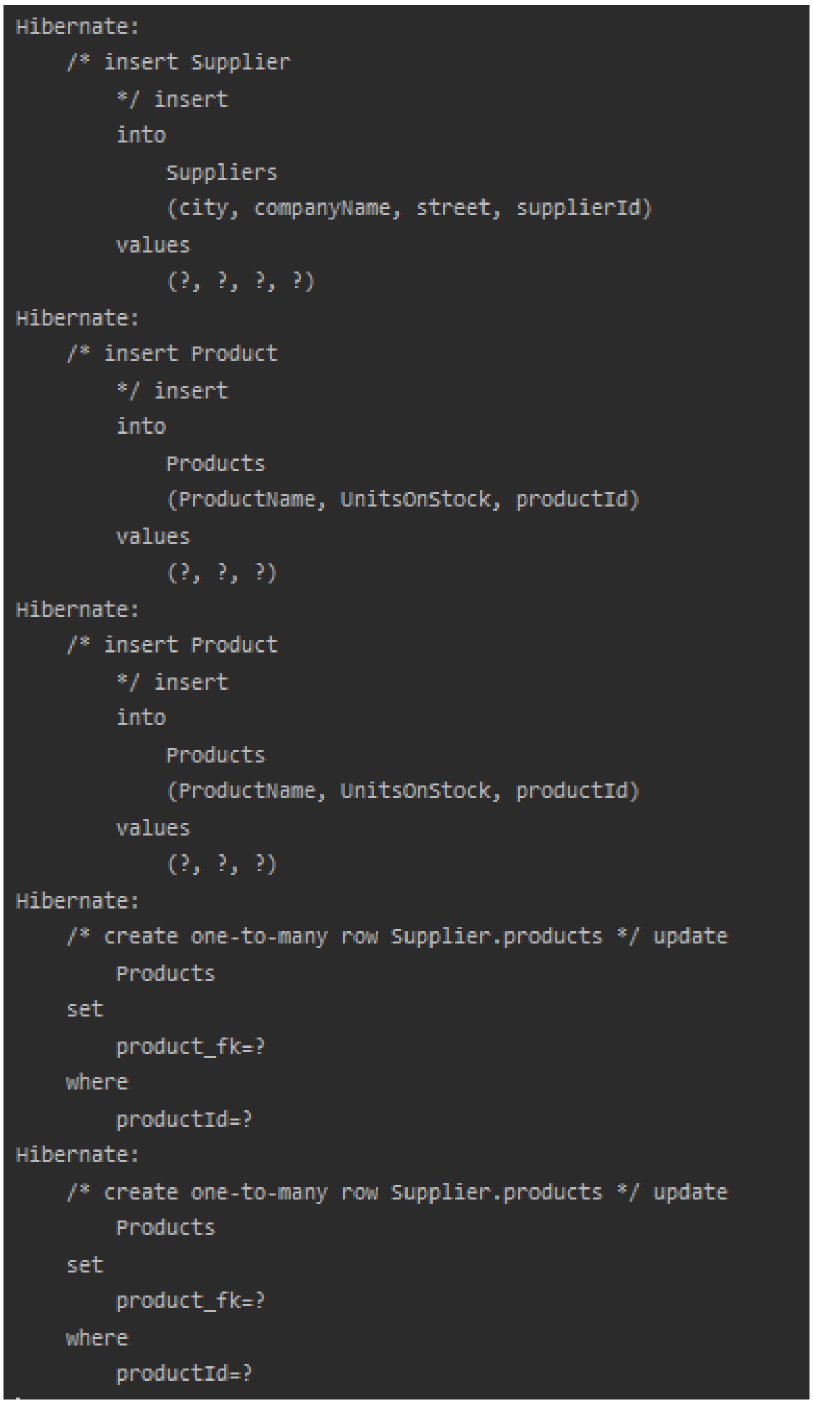
3 IVb. Odwrócenie relacji - brak tabeli łącznikowej

3.1 Pobranie informacji o dostawcy i produktach oraz dodanie ich do bazy



```
Enter company name:
Company3
Enter street:
Street6
Enter city:
Krakow
Enter the product name:
Product1
Enter state of warehouse:
45
Enter the product name:
Product2
Enter state of warehouse:
48
```

3.2 Zapytania wykonane przez Hibernate



```
Hibernate:
    /* insert Supplier
        */ insert
        into
            Suppliers
            (city, companyName, street, supplierId)
        values
            (?, ?, ?, ?)
Hibernate:
    /* insert Product
        */ insert
        into
            Products
            (ProductName, UnitsOnStock, productId)
        values
            (?, ?, ?)
Hibernate:
    /* insert Product
        */ insert
        into
            Products
            (ProductName, UnitsOnStock, productId)
        values
            (?, ?, ?)
Hibernate:
    /* create one-to-many row Supplier.products */ update
        Products
    set
        product_fk=?
    where
        productId=?
Hibernate:
    /* create one-to-many row Supplier.products */ update
        Products
    set
        product_fk=?
    where
        productId=?
```

3.3 Kod klasy Product

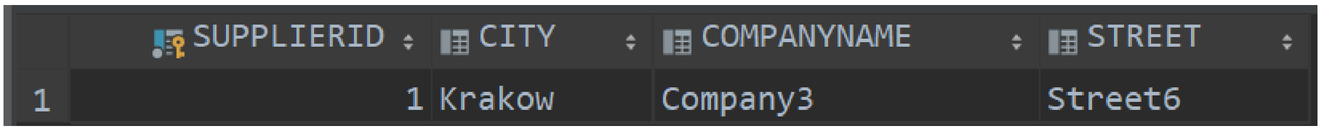
```
import lombok.*;

import javax.persistence.*;

@Entity(name = "Products")
@Getter
@Setter
@NoArgsConstructor
public class Product {

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

    private String ProductName;
    private int UnitsOnStock;

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

*/
```

3.4 Kod klasy Supplier

```java
import lombok.NoArgsConstructor;

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

@Entity
@Table(name = "Suppliers")
@Getter
@Setter
@NoArgsConstructor
public class Supplier {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    public int supplierId;

    private String companyName;

    private String street;

    private String city;

    @OneToMany
    @JoinColumn(name = "product_fk")
    Set<Product> products = new HashSet<>();

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

*/
```

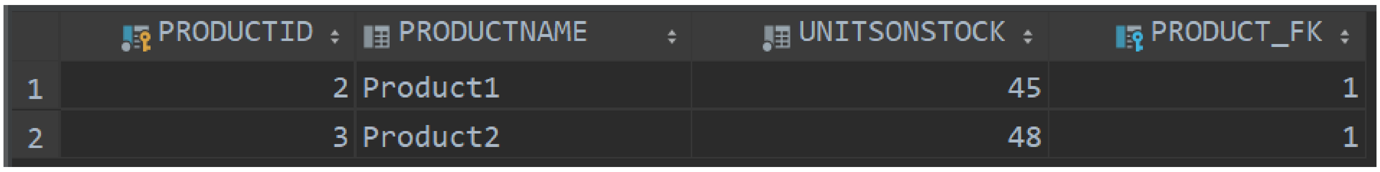
3.5 Wyniki SELECT * z poszczególnych tabel

3.5.1 Tabela Suppliers



	SUPPLIERID	CITY	COMPANYNAME	STREET
1	1	Krakow	Company3	Street6

3.5.2 Tabela Products



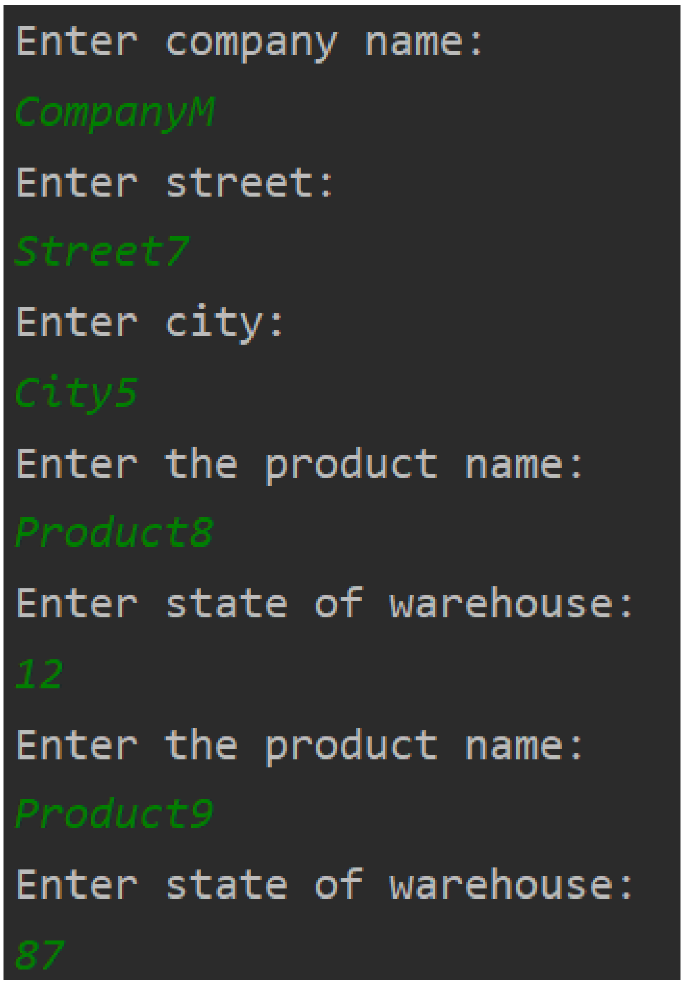
	PRODUCTID	PRODUCTNAME	UNITSONSTOCK	PRODUCT_FK
1	2	Product1	45	1
2	3	Product2	48	1

3.6 Kod DDL

```
create schema APP;

create table SUPPLIERS
(
   SUPPLIERID INTEGER not null
      primary key,
   CITY VARCHAR(255),
   COMPANYNAME VARCHAR(255),
   STREET VARCHAR(255)
);

create table PRODUCTS
(
   PRODUCTID INTEGER not null
      primary key,
   PRODUCTNAME VARCHAR(255),
   UNITSONSTOCK INTEGER not null,
   PRODUCT_FK INTEGER
      constraint FKACCSFELAACV8O55RWW68FG7KF
         references SUPPLIERS
);
```

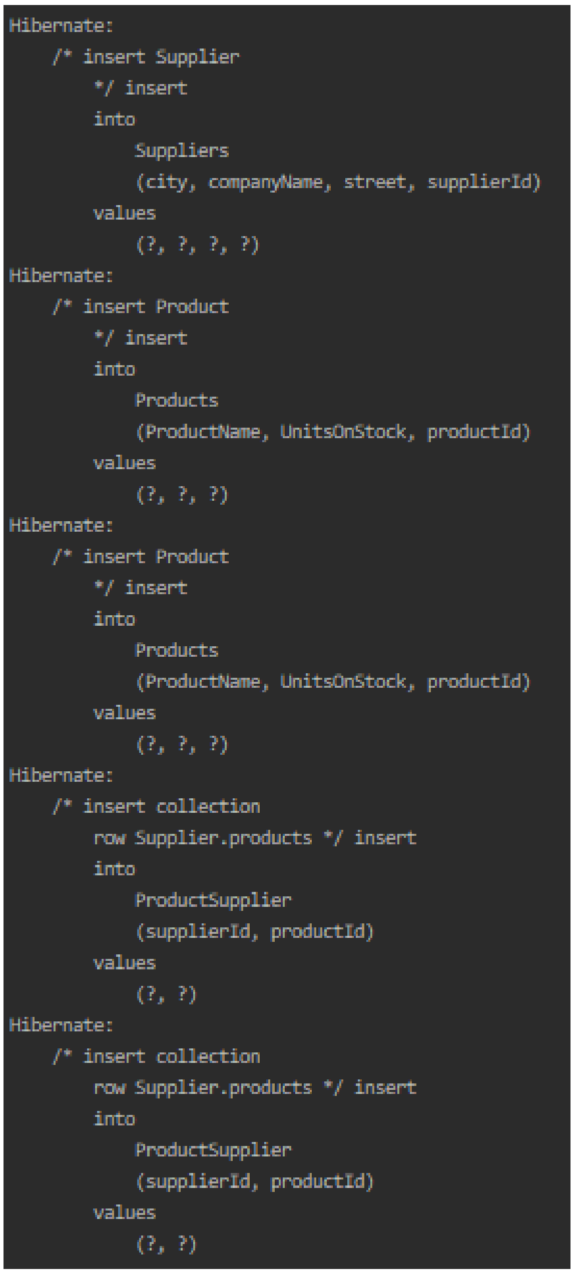
4 V. Relacja dwustronna. Dostawca - producent

4.1 Pobranie informacji o dostawcy i produktach oraz dodanie ich do bazy



```
Enter company name:
CompanyM
Enter street:
Street7
Enter city:
City5
Enter the product name:
Product8
Enter state of warehouse:
12
Enter the product name:
Product9
Enter state of warehouse:
87
```

4.2 Zapytania wykonane przez Hibernate



```
Hibernate:
    /* insert Supplier
        */ insert
        into
            Suppliers
            (city, companyName, street, supplierId)
        values
            (?, ?, ?, ?)
Hibernate:
    /* insert Product
        */ insert
        into
            Products
            (ProductName, UnitsOnStock, productId)
        values
            (?, ?, ?)
Hibernate:
    /* insert Product
        */ insert
        into
            Products
            (ProductName, UnitsOnStock, productId)
        values
            (?, ?, ?)
Hibernate:
    /* insert collection
        row Supplier.products */ insert
        into
            ProductSupplier
            (supplierId, productId)
        values
            (?, ?)
Hibernate:
    /* insert collection
        row Supplier.products */ insert
        into
            ProductSupplier
            (supplierId, productId)
        values
            (?, ?)
```

4.3 Kod klasy Product

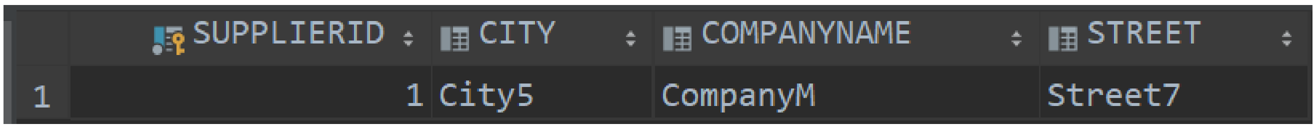
```
import lombok.*;

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

@Entity(name = "Products")
@Getter
@Setter
@NoArgsConstructor
public class Product {

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

    private String ProductName;
    private int UnitsOnStock;

    @ManyToMany(mappedBy = "products")
    Set<Supplier> suppliers;

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

 */
```

4.4 Kod klasy Supplier

```java
import lombok.NoArgsConstructor;

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

@Entity
@Table(name = "Suppliers")
@Getter
@Setter
@NoArgsConstructor
public class Supplier {

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

    private String companyName;

    private String street;

    private String city;

    @ManyToMany
    @JoinTable(
            name = "ProductSupplier",
            joinColumns = {@JoinColumn(name = "supplierId")},
            inverseJoinColumns = {@JoinColumn(name = "productId")}
    )
    Set<Product> products = new HashSet<>();

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

*/
```

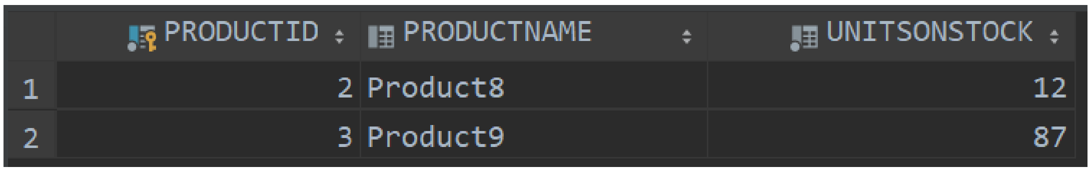
4.5 Wyniki SELECT * z poszczególnych tabel

4.5.1 Tabela Suppliers



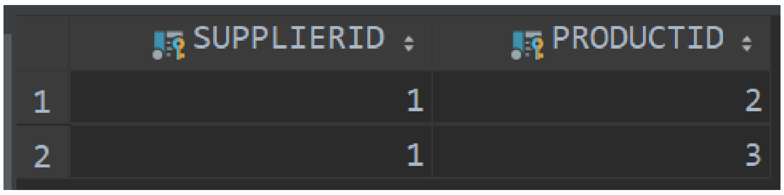
	SUPPLIERID	CITY	COMPANYNAME	STREET
1	1	City5	CompanyM	Street7

4.5.2 Tabela Products



	PRODUCTID	PRODUCTNAME	UNITSONSTOCK
1	2	Product8	12
2	3	Product9	87

4.5.3 Tabela łącznikowa



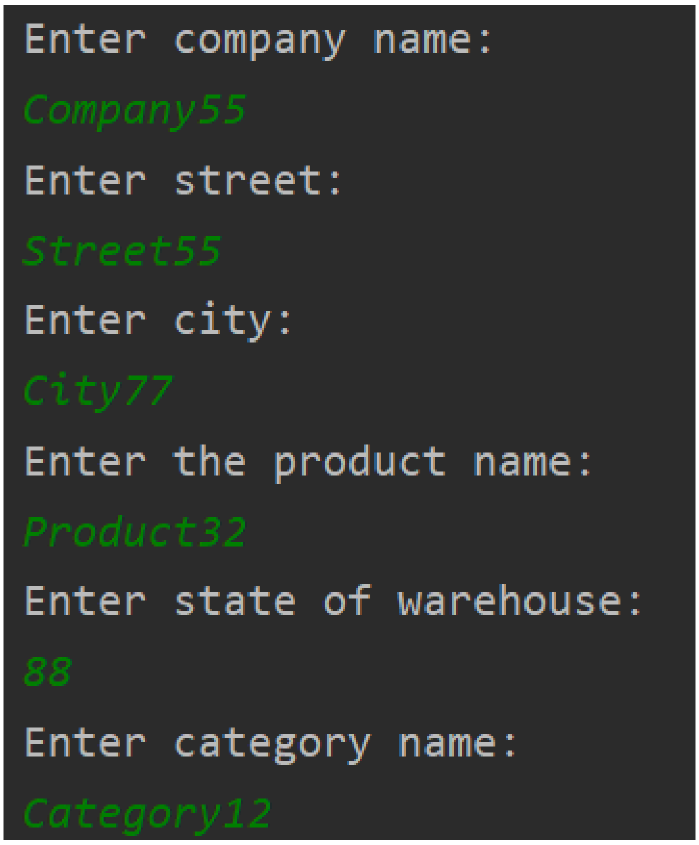
	SUPPLIERID	PRODUCTID
1	1	2
2	1	3

4.6 Kod DDL

```
create schema APP;

create table PRODUCTS
(
   PRODUCTID INTEGER not null
      primary key,
   PRODUCTNAME VARCHAR(255),
   UNITSONSTOCK INTEGER not null
);

create table SUPPLIERS
(
   SUPPLIERID INTEGER not null
      primary key,
   CITY VARCHAR(255),
   COMPANYNAME VARCHAR(255),
   STREET VARCHAR(255)
);

create table PRODUCTSUPPLIER
(
   SUPPLIERID INTEGER not null
      constraint FKHVXVPX6PJND3K0VD8TEG53M0S
         references SUPPLIERS,
   PRODUCTID INTEGER not null
      constraint FKSYI1HWRQIKCD1IS3N7V9F2J5A
         references PRODUCTS,
   primary key (SUPPLIERID, PRODUCTID)
);
```

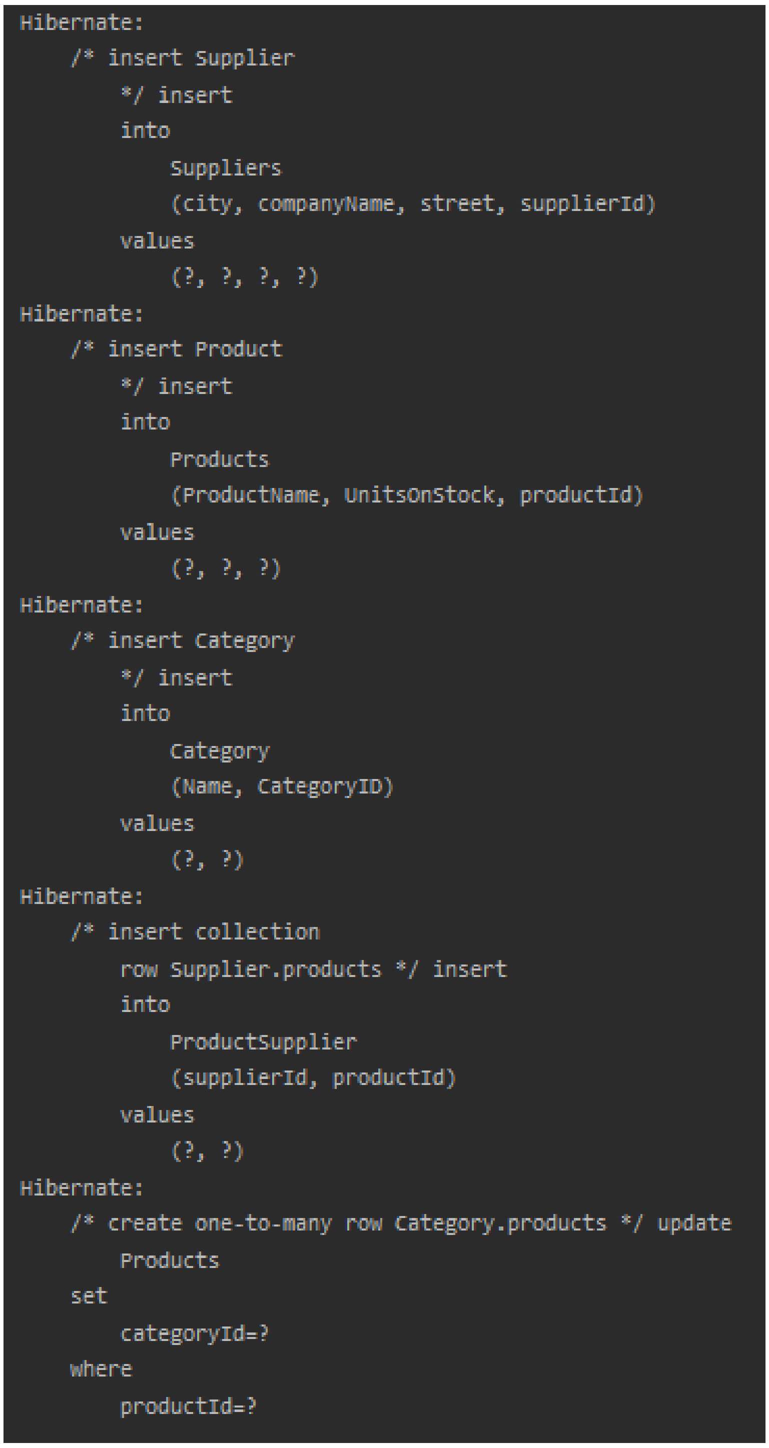
5 VI. Obsługa klasy Category

5.1 Pobranie informacji o dostawcy, produkcie i kategorii



```
Enter company name:
Company55
Enter street:
Street55
Enter city:
City77
Enter the product name:
Product32
Enter state of warehouse:
88
Enter category name:
Category12
```

5.2 Zapytania wykonane przez Hibernate



```
Hibernate:
    /* insert Supplier
        */ insert
        into
            Suppliers
            (city, companyName, street, supplierId)
        values
            (?, ?, ?, ?)
Hibernate:
    /* insert Product
        */ insert
        into
            Products
            (ProductName, UnitsOnStock, productId)
        values
            (?, ?, ?)
Hibernate:
    /* insert Category
        */ insert
        into
            Category
            (Name, CategoryID)
        values
            (?, ?)
Hibernate:
    /* insert collection
        row Supplier.products */ insert
        into
            ProductSupplier
            (supplierId, productId)
        values
            (?, ?)
Hibernate:
    /* create one-to-many row Category.products */ update
        Products
    set
        categoryId=?
    where
        productId=?
```

5.3 Kod klasy Product

```
import lombok.*;

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

@Entity(name = "Products")
@Getter
@Setter
@NoArgsConstructor
public class Product {

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

    private String ProductName;
    private int UnitsOnStock;

    @ManyToMany(mappedBy = "products")
    Set<Supplier> suppliers;

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

 */
```

5.4 Kod klasy Supplier

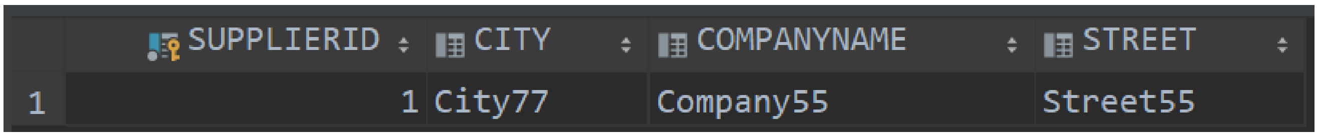
```
import lombok.NoArgsConstructor;

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

@Entity
@Table(name = "Suppliers")
@Getter
@Setter
@NoArgsConstructor
public class Supplier {

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

    private String companyName;

    private String street;

    private String city;

    @ManyToMany
    @JoinTable(
            name = "ProductSupplier",
            joinColumns = {@JoinColumn(name = "supplierId")},
            inverseJoinColumns = {@JoinColumn(name = "productId")}
    )
    Set<Product> products = new HashSet<>();

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

*/
```

5.5 Kod klasy Category

```java
import lombok.Getter;
import lombok.Setter;

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

@Entity
@Setter
@Getter
public class Category {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int CategoryID;
    private String Name;

    @OneToMany
    @JoinColumn(name = "categoryId")
    List<Product> products = new ArrayList<>();

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

 */
```

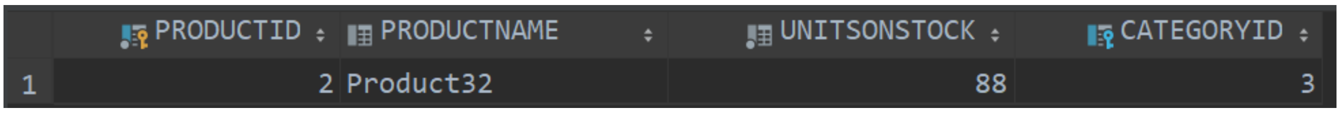
5.6 Wyniki SELECT * z poszczególnych tabel

5.6.1 Tabela Suppliers



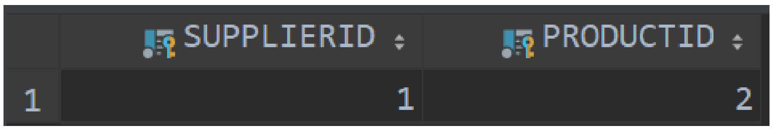
	SUPPLIERID	CITY	COMPANYNAME	STREET
1	1	City77	Company55	Street55

5.6.2 Tabela Products



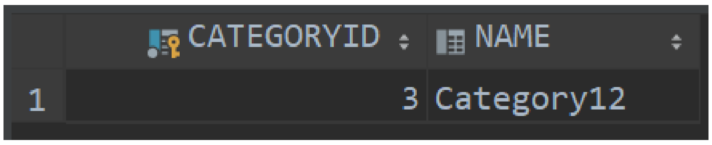
	PRODUCTID	PRODUCTNAME	UNITSONSTOCK	CATEGORYID
1	2	Product32	88	3

5.6.3 Tabela ProductsSupplier



	SUPPLIERID	PRODUCTID
1	1	2

5.6.4 Tabela Category



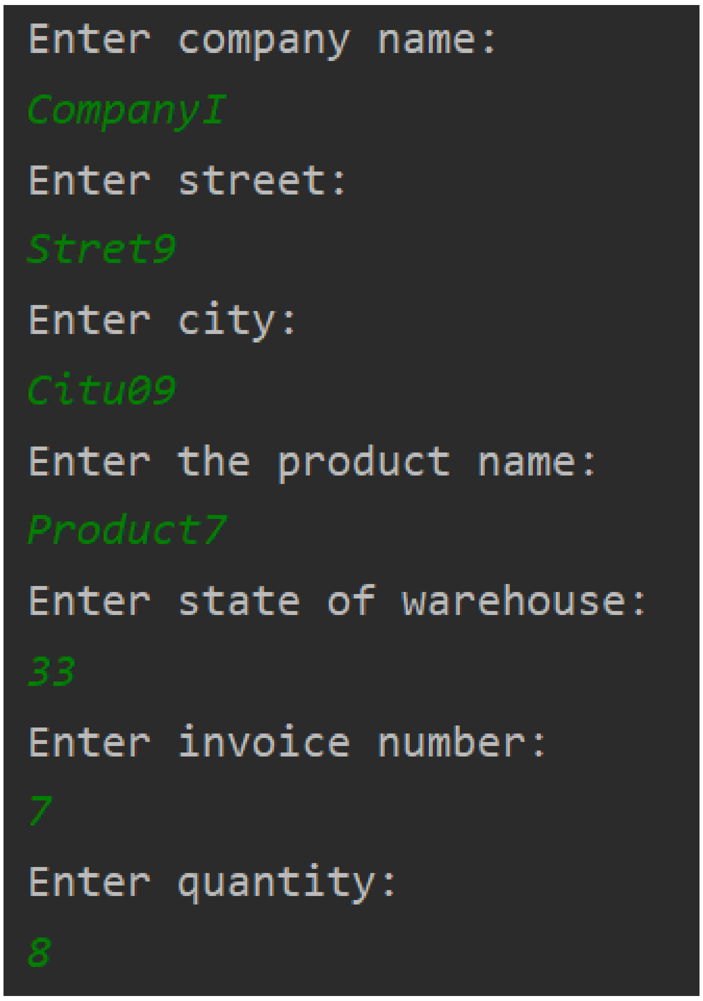
	CATEGORYID	NAME
1	3	Category12

5.7 Kod DDL

```
create schema APP;

create table CATEGORY
(
   CATEGORYID INTEGER not null
      primary key,
   NAME VARCHAR(255)
);

create table PRODUCTS
(
   PRODUCTID INTEGER not null
      primary key,
   PRODUCTNAME VARCHAR(255),
   UNITSONSTOCK INTEGER not null,
   CATEGORYID INTEGER
      constraint FKMGOP3YOCT41Q8YD7DPFMYI1EN
         references CATEGORY
);

create table SUPPLIERS
(
   SUPPLIERID INTEGER not null
      primary key,
   CITY VARCHAR(255),
   COMPANYNAME VARCHAR(255),
   STREET VARCHAR(255)
);

create table PRODUCTSUPPLIER
(
   SUPPLIERID INTEGER not null
      constraint FKHVXVPX6PJND3KOVD8TEG53MOS
         references SUPPLIERS,
   PRODUCTID INTEGER not null
      constraint FKSYI1HWRQIKCD1IS3N7V9F2J5A
         references PRODUCTS,
   primary key (SUPPLIERID, PRODUCTID)
);
```

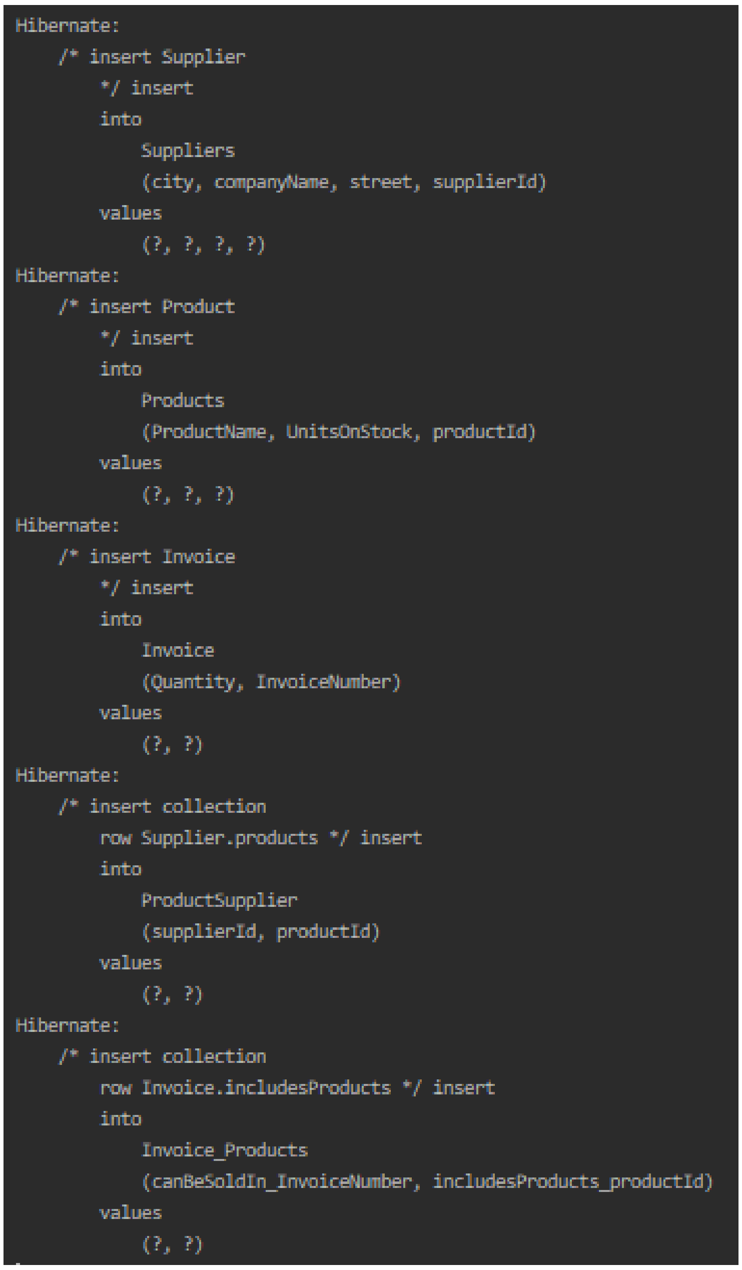
6 VII. Relacja wiele do wielu. Invoice - Product

6.1 Pobranie informacji o firmie, produkcie i fakturze



```
Enter company name:
CompanyI
Enter street:
Stret9
Enter city:
Citu09
Enter the product name:
Product7
Enter state of warehouse:
33
Enter invoice number:
7
Enter quantity:
8
```

6.2 Zapytania wykonane przez Hibernate



```
Hibernate:
    /* insert Supplier
        */ insert
        into
            Suppliers
            (city, companyName, street, supplierId)
        values
            (?, ?, ?, ?)
Hibernate:
    /* insert Product
        */ insert
        into
            Products
            (ProductName, UnitsOnStock, productId)
        values
            (?, ?, ?)
Hibernate:
    /* insert Invoice
        */ insert
        into
            Invoice
            (Quantity, InvoiceNumber)
        values
            (?, ?)
Hibernate:
    /* insert collection
        row Supplier.products */ insert
        into
            ProductSupplier
            (supplierId, productId)
        values
            (?, ?)
Hibernate:
    /* insert collection
        row Invoice.includesProducts */ insert
        into
            Invoice_Products
            (canBeSoldIn_InvoiceNumber, includesProducts_productId)
        values
            (?, ?)
```

6.3 Kod klasy Product

```java
import lombok.*;

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

@Entity(name = "Products")
@Setter
@Getter
@NoArgsConstructor
public class Product {

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

    private String ProductName;
    private int UnitsOnStock;

    @ManyToMany(mappedBy = "products")
    Set<Supplier> suppliers = new HashSet<>();

    @ManyToMany(
           mappedBy = "includesProducts",
           fetch = FetchType.EAGER,
           cascade = CascadeType.PERSIST)
    Set<Invoice> canBeSoldIn = new HashSet<>();

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

*/
```

6.4 Kod klasy Supplier

```
import lombok.NoArgsConstructor;

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

@Entity
@Table(name = "Suppliers")
@Getter
@Setter
@NoArgsConstructor
public class Supplier {

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

    private String companyName;

    private String street;

    private String city;

    @ManyToMany
    @JoinTable(
            name = "ProductSupplier",
            joinColumns = {@JoinColumn(name = "supplierId")},
            inverseJoinColumns = {@JoinColumn(name = "productId")}
    )
    Set<Product> products = new HashSet<>();

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

*/
```

6.5 Kod klasy Category

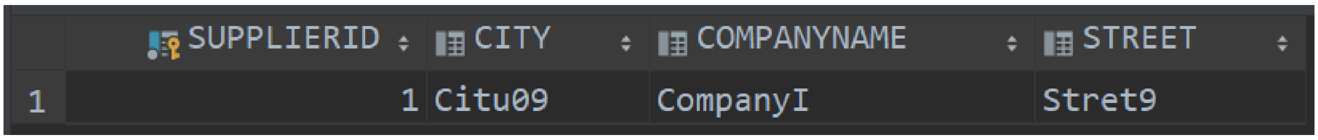
```
import lombok.Getter;
import lombok.Setter;

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

@Entity
@Setter
@Getter
public class Category {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int CategoryID;
    private String Name;

    @OneToMany
    @JoinColumn(name = "categoryId")
    List<Product> products = new ArrayList<>();

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

 */
```

6.6 Kod klasy Invoice

```
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

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

@Entity
@Getter
@Setter
@NoArgsConstructor
public class Invoice {

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

    @ManyToMany(cascade = CascadeType.PERSIST)
    Set<Product> includesProducts = new HashSet<>();

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

 */
```

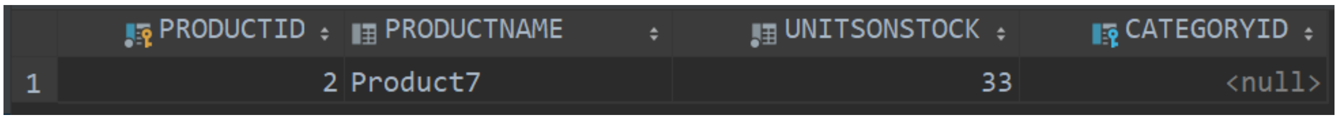
6.7 Wyniki SELECT * z poszczególnych tabel

6.7.1 Tabela Suppliers



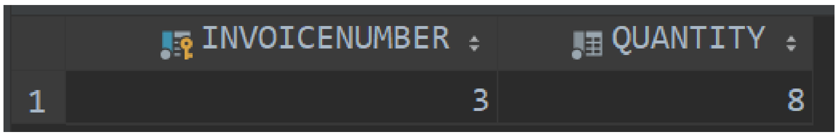
	SUPPLIERID	CITY	COMPANYNAME	STREET
1	1	Citu09	CompanyI	Stret9

6.7.2 Tabela Products



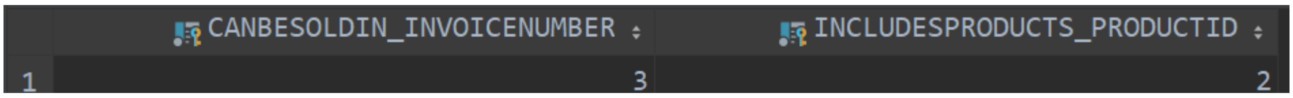
	PRODUCTID	PRODUCTNAME	UNITSONSTOCK	CATEGORYID
1	2	Product7	33	<null>

6.7.3 Tabela Invoice



	INVOICENUMBER	QUANTITY
1	3	8

6.7.4 Tabela InvoiceProducts



	CANBESOLDIN_INVOICENUMBER	INCLUDESPRODUCTS_PRODUCTID
1	3	2

6.8 Kod DDL

```
create schema APP;

create table CATEGORY
(
   CATEGORYID INTEGER not null
      primary key,
   NAME VARCHAR(255)
);

create table INVOICE
(
   INVOICENUMBER INTEGER not null
      primary key,
   QUANTITY INTEGER not null
);

create table PRODUCTS
(
   PRODUCTID INTEGER not null
      primary key,
   PRODUCTNAME VARCHAR(255),
   UNITSONSTOCK INTEGER not null,
   CATEGORYID INTEGER
      constraint FKMGOP3YOCT41Q8YD7DPFMYI1EN
         references CATEGORY (CATEGORYID)
);

create table INVOICE_PRODUCTS
(
   CANBESOLDIN_INVOICENUMBER INTEGER not null
      constraint FKRC271LI7RMOOHR5U1IK6AEXLH
         references INVOICE (INVOICENUMBER),
   INCLUDESPRODUCTS_PRODUCTID INTEGER not null
      constraint FK5QBA1N5MM1AEP8VASA057GQCP
         references PRODUCTS (PRODUCTID),
   primary key (CANBESOLDIN_INVOICENUMBER, INCLUDESPRODUCTS_PRODUCTID)
);

create table SUPPLIERS
(
   SUPPLIERID INTEGER not null
      primary key,
   CITY VARCHAR(255),
   COMPANYNAME VARCHAR(255),
   STREET VARCHAR(255)
);

create table PRODUCTSUPPLIER
(
   SUPPLIERID INTEGER not null
      constraint FKHVXVPX6PJND3K0VD8TEG53M0S
         references SUPPLIERS (SUPPLIERID),
   PRODUCTID INTEGER not null
```

```
    constraint FKSYI1HWRQIKCD1IS3N7V9F2J5A
      references PRODUCTS (PRODUCTID),
  primary key (SUPPLIERID, PRODUCTID)
);
```

7 IX. JPA. Zadanie z punktu VI z wykorzystaniem JPA

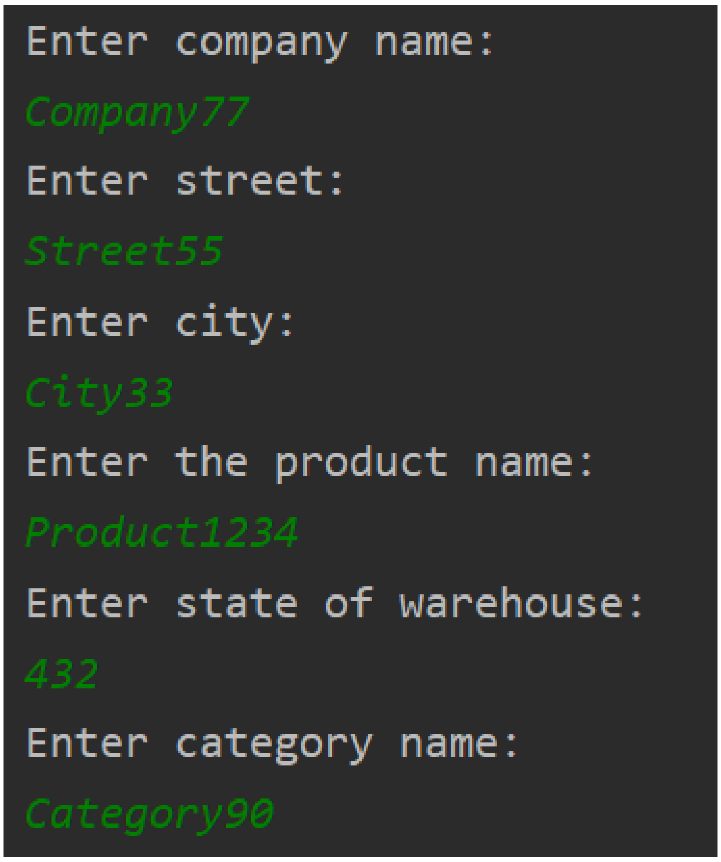
7.1 Kod klasy Main - klasa nie wykorzystująca JPA

```java
import org.hibernate.HibernateException;
import org.hibernate.Metamodel;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;
import org.hibernate.query.Query;

import javax.persistence.metamodel.EntityType;

public class Main {

    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 void main(final String[] args) throws Exception {
        try (Session session = getSession()) {
            queryAllManagedEntities(session);

            DatabasePerformer.addNewProductPoint3(session);
            DatabasePerformer.addNewSupplierPoint3(session);
            DatabasePerformer.addNewSupplierAndProducts(session);
            DatabasePerformer.addProductSupplierAndCategory(session);
            DatabasePerformer.addInvoiceAndSell(session);
        }
    }

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

    private static void queryAllManagedEntities(Session session) {
        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("from " + entityName);
            System.out.println("executing: " + query.getQueryString());
            for (Object o : query.list()) {
                System.out.println("  " + o);
            }
        }
    }
}

 */
```

7.2 Kod klasy MainJpa - klasa wykorzystująca JPA

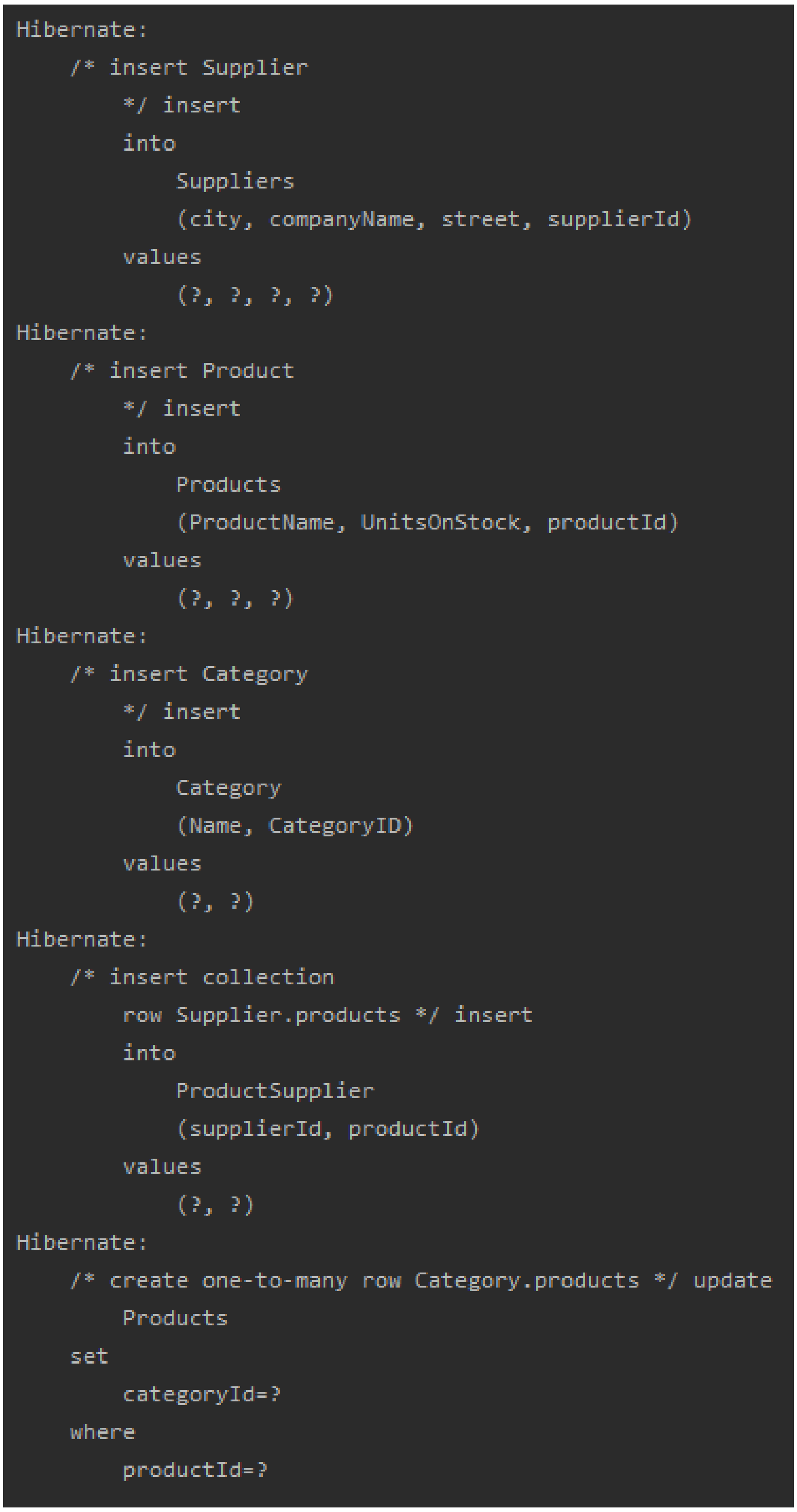
```
import org.hibernate.cfg.Configuration;

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

public class MainJpa {

    private static EntityManagerFactory entityManagerFactory;

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

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

    public static void main(final String[] args) throws Exception {
        final EntityManager entityManager = getEntityManager();

        try {
            DatabasePerformer.addProductSupplierAndCategory(entityManager);
        } finally {
            entityManager.close();
        }
    }

    private static EntityManager getEntityManager() {
        if (entityManagerFactory == null) {
            entityManagerFactory = Persistence.createEntityManagerFactory("derby");
        }

        return entityManagerFactory.createEntityManager();
    }
}


 */
```

7.3 Pobranie informacji o firmie, produkcie i kategorii



```
Enter company name:
Company77
Enter street:
Street55
Enter city:
City33
Enter the product name:
Product1234
Enter state of warehouse:
432
Enter category name:
Category90
```

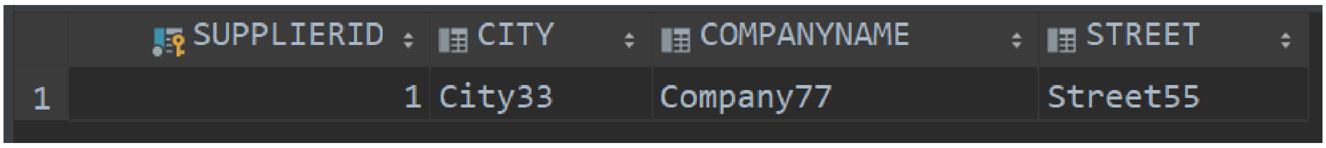
7.4 Zapytania wykonane przez Hibernate



```
Hibernate:
    /* insert Supplier
        */ insert
        into
            Suppliers
            (city, companyName, street, supplierId)
        values
            (?, ?, ?, ?)
Hibernate:
    /* insert Product
        */ insert
        into
            Products
            (ProductName, UnitsOnStock, productId)
        values
            (?, ?, ?)
Hibernate:
    /* insert Category
        */ insert
        into
            Category
            (Name, CategoryID)
        values
            (?, ?)
Hibernate:
    /* insert collection
        row Supplier.products */ insert
        into
            ProductSupplier
            (supplierId, productId)
        values
            (?, ?)
Hibernate:
    /* create one-to-many row Category.products */ update
        Products
    set
        categoryId=?
    where
        productId=?
```

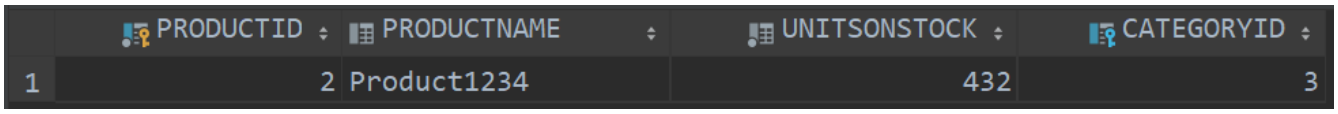
7.5 Wyniki SELECT * z poszczególnych tabel

7.5.1 Tabela Suppliers



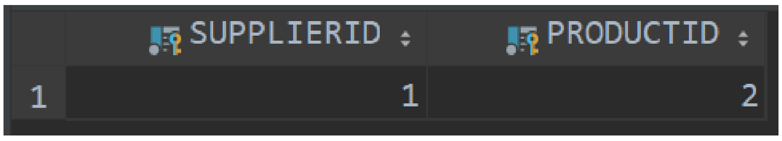
	SUPPLIERID	CITY	COMPANYNAME	STREET
1	1	City33	Company77	Street55

7.5.2 Tabela Products



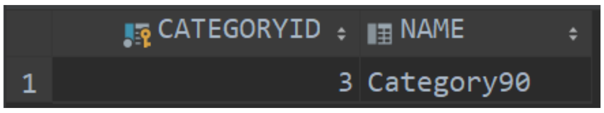
	PRODUCTID	PRODUCTNAME	UNITSONSTOCK	CATEGORYID
1	2	Product1234	432	3

7.5.3 Tabela ProductsSupplier



	SUPPLIERID	PRODUCTID
1	1	2

7.5.4 Tabela Categories



	CATEGORYID	NAME
1	3	Category90

7.6 Kod DDL

```
create schema APP;

create table CATEGORY
(
   CATEGORYID INTEGER not null
      primary key,
   NAME VARCHAR(255)
);

create table INVOICE
(
   INVOICENUMBER INTEGER not null
      primary key,
   QUANTITY INTEGER not null
);

create table PRODUCTS
(
   PRODUCTID INTEGER not null
      primary key,
   PRODUCTNAME VARCHAR(255),
   UNITSONSTOCK INTEGER not null,
   CATEGORYID INTEGER
      constraint FKMGOP3YOCT41Q8YD7DPFMYI1EN
         references CATEGORY (CATEGORYID)
);

create table INVOICE_PRODUCTS
(
   CANBESOLDIN_INVOICENUMBER INTEGER not null
      constraint FKRC271LI7RMOOHR5U1IK6AEXLH
         references INVOICE (INVOICENUMBER),
   INCLUDESPRODUCTS_PRODUCTID INTEGER not null
      constraint FK5QBA1N5MM1AEP8VASA057GQCP
         references PRODUCTS (PRODUCTID),
   primary key (CANBESOLDIN_INVOICENUMBER, INCLUDESPRODUCTS_PRODUCTID)
);

create table SUPPLIERS
(
   SUPPLIERID INTEGER not null
      primary key,
   CITY VARCHAR(255),
   COMPANYNAME VARCHAR(255),
   STREET VARCHAR(255)
);

create table PRODUCTSUPPLIER
(
   SUPPLIERID INTEGER not null
      constraint FKHVXVPX6PJND3K0VD8TEG53M0S
         references SUPPLIERS (SUPPLIERID),
   PRODUCTID INTEGER not null
```

```
    constraint FKSYI1HWRQIKCD1IS3N7V9F2J5A
      references PRODUCTS (PRODUCTID),
  primary key (SUPPLIERID, PRODUCTID)
);
```

8 X. Cascade

8.1 Kod klasy Invoice

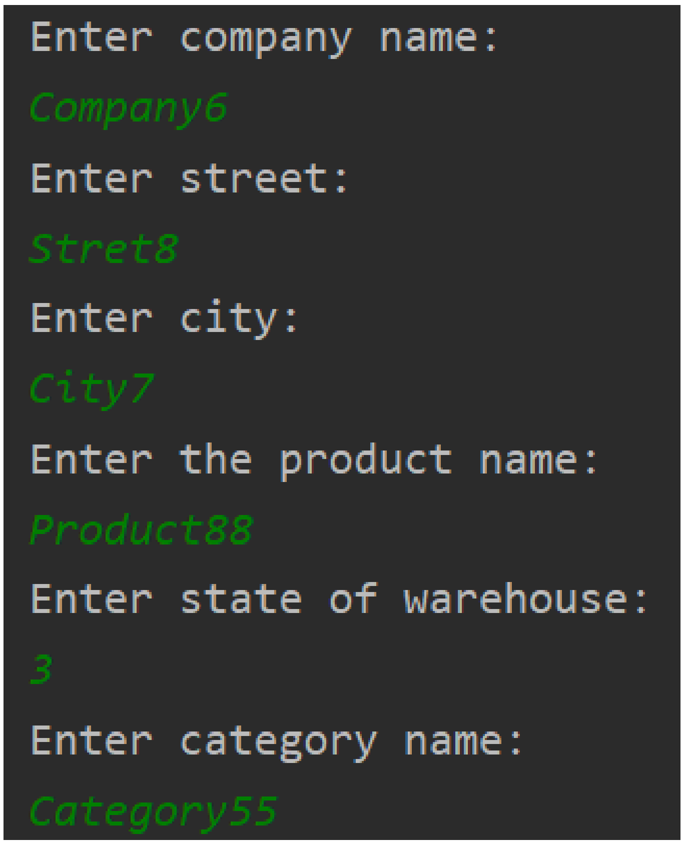
```java
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

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

@Entity
@Getter
@Setter
@NoArgsConstructor
public class Invoice {

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

    @ManyToMany(cascade = CascadeType.PERSIST)
    Set<Product> includesProducts = new HashSet<>();

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


 */
```

8.2 Kod klasy Product

```java
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

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

@Entity(name = "Products")
@Setter
@Getter
@NoArgsConstructor
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int productId;

    private String ProductName;
    private int UnitsOnStock;

    @ManyToMany(mappedBy = "products")
    Set<Supplier> suppliers = new HashSet<>();

    @ManyToMany(
            mappedBy = "includesProducts",
            fetch = FetchType.EAGER,
            cascade = CascadeType.PERSIST)
    Set<Invoice> canBeSoldIn = new HashSet<>();

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


 */
```

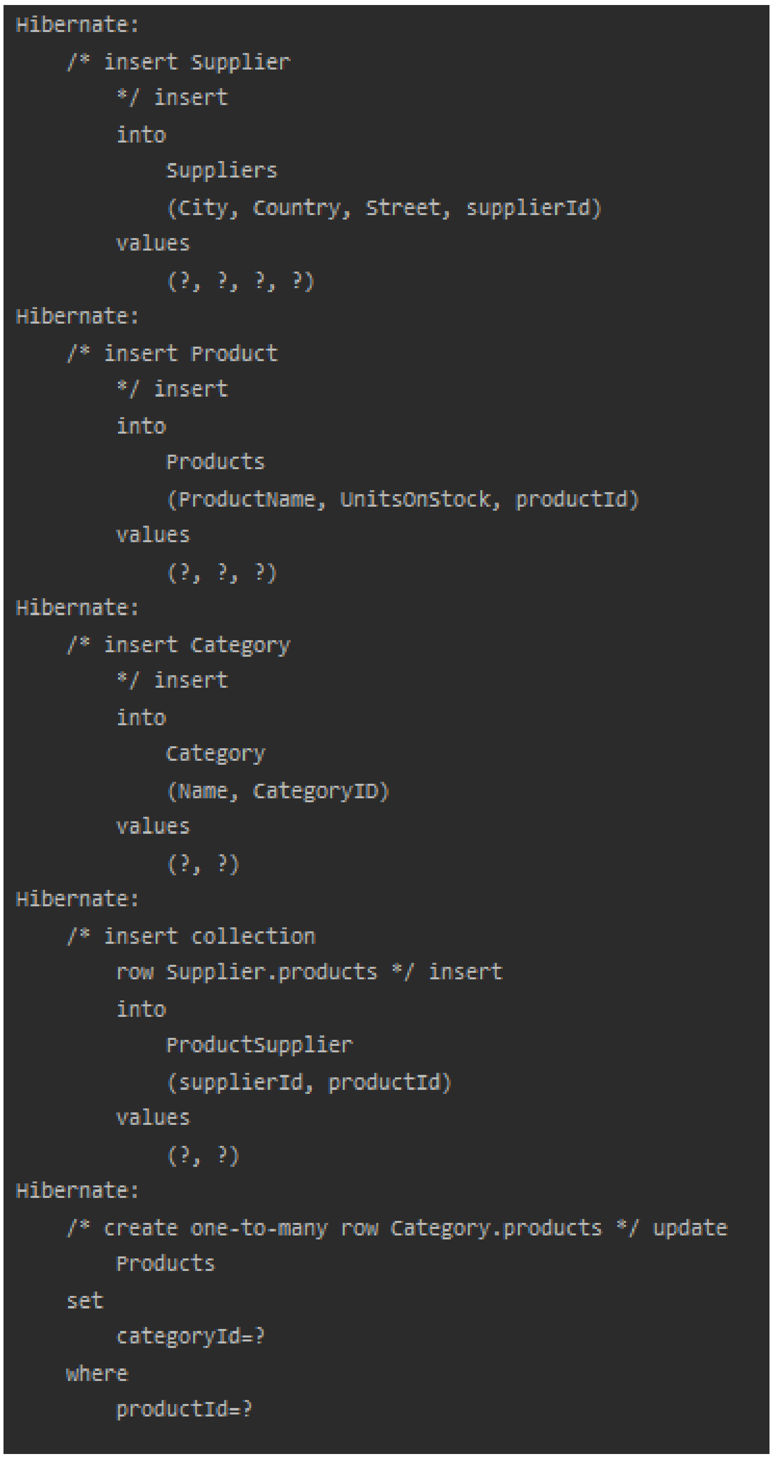
9 XI. Embedded class

9.1 Pobranie informacji o firmie, produkcie i kategorii



```
Enter company name:
Company6
Enter street:
Stret8
Enter city:
City7
Enter the product name:
Product88
Enter state of warehouse:
3
Enter category name:
Category55
```

9.2 Zapytania wykonane przez Hibernate



```
Hibernate:
    /* insert Supplier
        */ insert
        into
            Suppliers
            (City, Country, Street, supplierId)
        values
            (?, ?, ?, ?)
Hibernate:
    /* insert Product
        */ insert
        into
            Products
            (ProductName, UnitsOnStock, productId)
        values
            (?, ?, ?)
Hibernate:
    /* insert Category
        */ insert
        into
            Category
            (Name, CategoryID)
        values
            (?, ?)
Hibernate:
    /* insert collection
        row Supplier.products */ insert
        into
            ProductSupplier
            (supplierId, productId)
        values
            (?, ?)
Hibernate:
    /* create one-to-many row Category.products */ update
        Products
    set
        categoryId=?
    where
        productId=?
```

9.3 Kod klasy Address

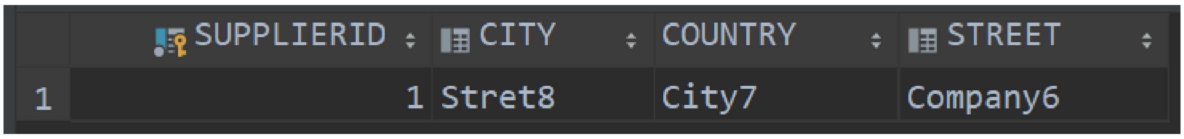
```
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import javax.persistence.Embeddable;

@Getter
@Setter
@Embeddable
@NoArgsConstructor
public class Address {

    private String Street;
    private String City;
    private String Country;

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

*/
```

9.4 Kod klasy Supplier

```
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

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

@Entity
@Table(name = "Suppliers")
@Getter
@Setter
@NoArgsConstructor
public class Supplier {

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

    @ManyToMany
    @JoinTable(
           name = "ProductSupplier",
           joinColumns = {@JoinColumn(name = "supplierId")},
           inverseJoinColumns = {@JoinColumn(name = "productId")}
    )
    Set<Product> products = new HashSet<>();

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

 */
```

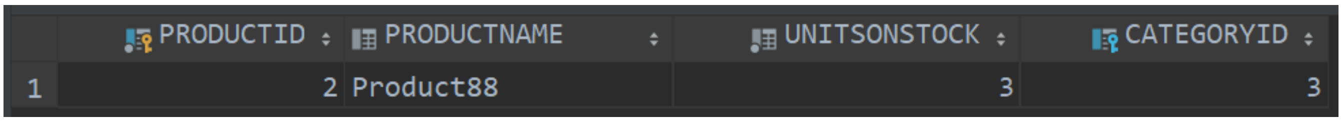
9.5 Wyniki SELECT * z poszczególnych tabel

9.5.1 Tabela Suppliers



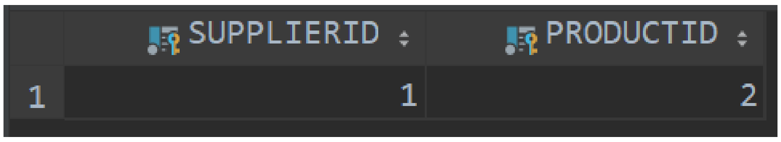
	SUPPLIERID	CITY	COUNTRY	STREET
1	1	Stret8	City7	Company6

9.5.2 Tabela Products



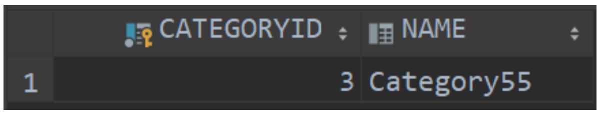
	PRODUCTID	PRODUCTNAME	UNITSONSTOCK	CATEGORYID
1	2	Product88	3	3

9.5.3 Tabela ProductsSupplier



	SUPPLIERID	PRODUCTID
1	1	2

9.5.4 Tabela Categories



	CATEGORYID	NAME
1	3	Category55

9.6 Kod DDL

```
create schema APP;

create table CATEGORY
(
   CATEGORYID INTEGER not null
      primary key,
   NAME VARCHAR(255)
);

create table INVOICE
(
   INVOICENUMBER INTEGER not null
      primary key,
   QUANTITY INTEGER not null
);

create table PRODUCTS
(
   PRODUCTID INTEGER not null
      primary key,
   PRODUCTNAME VARCHAR(255),
   UNITSONSTOCK INTEGER not null,
   CATEGORYID INTEGER
      constraint FKMGOP3YOCT41Q8YD7DPFMYI1EN
         references CATEGORY (CATEGORYID)
);

create table INVOICE_PRODUCTS
(
   CANBESOLDIN_INVOICENUMBER INTEGER not null
      constraint FKRC271LI7RMOOHR5U1IK6AEXLH
         references INVOICE (INVOICENUMBER),
   INCLUDESPRODUCTS_PRODUCTID INTEGER not null
      constraint FK5QBA1N5MM1AEP8VASA057GQCP
         references PRODUCTS (PRODUCTID),
   primary key (CANBESOLDIN_INVOICENUMBER, INCLUDESPRODUCTS_PRODUCTID)
);

create table SUPPLIERS
(
   SUPPLIERID INTEGER not null
      primary key,
   CITY VARCHAR(255),
   COMPANYNAME VARCHAR(255),
   STREET VARCHAR(255)
);

create table PRODUCTSUPPLIER
(
   SUPPLIERID INTEGER not null
      constraint FKHVXVPX6PJND3K0VD8TEG53M0S
         references SUPPLIERS (SUPPLIERID),
   PRODUCTID INTEGER not null
```

```
    constraint FKSYI1HWRQIKCD1IS3N7V9F2J5A
      references PRODUCTS (PRODUCTID),
  primary key (SUPPLIERID, PRODUCTID)
);
```

10 XII. Inheritance

10.1 Kod klasy Company - Table Per Class

```
import lombok.Getter;
import lombok.Setter;

import javax.persistence.*;

@Entity
@Getter
@Setter
@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)
abstract public class Company {

    @Id
    String CompanyName;

    String Street;
    String City;

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


 */
```

10.2 Kod klasy Company - Table Joined

```
import lombok.Getter;
import lombok.Setter;

import javax.persistence.*;

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

    @Id
    String CompanyName;

    String Street;
    String City;

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


 */
```

10.3 Kod klasy Company - Single table

```
import lombok.Getter;
import lombok.Setter;

import javax.persistence.*;

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

    @Id
    String CompanyName;

    String Street;
    String City;

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


 */
```

10.4 Kod klasy Customer

```
import javax.persistence.Entity;

@Entity
public class Customer extends Company {
    private double discount;

    public Customer() {
        super();
    }

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

    public double getDiscount() {
        return discount;
    }

    public void setDiscount(double discount) {
        this.discount = discount;
    }
}

*/
```

10.5 Kod klasy Supplier

```
import javax.persistence.Entity;
import javax.persistence.JoinColumn;
import javax.persistence.OneToMany;
import java.util.HashSet;
import java.util.Set;

@Entity
public class Supplier extends Company {

    public String bankAccountNumber;


    @OneToMany
    @JoinColumn(name = "SUPPLIED_BY")
    private Set<Product> supplies = new HashSet<>();

    public Supplier() {
        super();
    }

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

    public void addSuppliedProduct(Product p) {
        supplies.add(p);
        p.setSuppliedBy(this);
    }

    public boolean suppliesProduct(Product p) {
        return supplies.contains(p);
    }
}


 */
```

11 Kod niektórych klas wykorzystanych w projekcie

11.1 Kod klasy Creator - wykorzystywanej do pobierania informacji od użytkownika

```java
import java.util.Scanner;

public class Creator {

    public static Product createProduct() {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter the product name:");
        String productName = scanner.nextLine();

        System.out.println("Enter state of warehouse:");
        int unitsInStock = scanner.nextInt();

        return new Product(productName, unitsInStock);
    }

    public static Supplier createSupplier() {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter company name:");
        String companyName = scanner.nextLine();

        System.out.println("Enter street:");
        String street = scanner.nextLine();

        System.out.println("Enter city:");
        String city = scanner.nextLine();

        return new Supplier(companyName, street, city);

    }

    public static Category createCategory() {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter category name:");
        String categoryName = scanner.nextLine();

        return new Category(categoryName);
    }

    public static Invoice createInvoice() {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter invoice number:");
        int invoiceNumber = scanner.nextInt();

        System.out.println("Enter quantity:");
        int quantity = scanner.nextInt();
        return new Invoice(invoiceNumber, quantity);
    }
}


 */
```

11.2 Kod klasy DatabasePerformer - wykorzystywanej wykonywania różnych operacji na bazie danych

```java
import org.hibernate.Session;
import org.hibernate.Transaction;

import javax.persistence.EntityManager;
import javax.persistence.EntityTransaction;

public class DatabasePerformer {

    public static void addNewProductPoint3(Session session) {
        Product product = Creator.createProduct();

        Transaction transaction = session.beginTransaction();
        session.save(product);
        transaction.commit();
    }

    public static void addNewSupplierPoint3(Session session) {
        Supplier supplier = Creator.createSupplier();

        Transaction transaction = session.beginTransaction();
        session.save(supplier);

        Product product = session.load(Product.class, 1);
//        product.setSupplier(supplier);
        transaction.commit();
    }

    public static void addNewSupplierAndProducts(Session session) {
        Supplier supplier = Creator.createSupplier();
        Product product = Creator.createProduct();
        Product product1 = Creator.createProduct();

        session.clear();
        Transaction transaction = session.beginTransaction();

        session.save(supplier);
        session.save(product);
        session.save(product1);

        supplier.products.add(product);
        supplier.products.add(product1);

        transaction.commit();
    }

    public static void addProductSupplierAndCategory(Session session) {
        Supplier supplier = Creator.createSupplier();
        Product product = Creator.createProduct();
        Category category = Creator.createCategory();

        session.clear();
        Transaction transaction = session.beginTransaction();

        session.save(supplier);
        session.save(product);
        session.save(category);

        supplier.products.add(product);
        category.products.add(product);
        transaction.commit();
    }

    public static void addInvoiceAndSell(Session session) {
        Supplier supplier = Creator.createSupplier();
        Product product = Creator.createProduct();
        Invoice invoice = Creator.createInvoice();

        session.clear();
        Transaction transaction = session.beginTransaction();

        session.save(supplier);
        session.save(product);
        session.save(invoice);

        supplier.products.add(product);
        product.canBeSoldIn.add(invoice);
        invoice.includesProducts.add(product);

        transaction.commit();
    }

    public static void addProductSupplierAndCategory(EntityManager entityManager) {
        Supplier supplier = Creator.createSupplier();
        Product product = Creator.createProduct();
        Category category = Creator.createCategory();

        entityManager.clear();

        EntityTransaction transaction = entityManager.getTransaction();
        transaction.begin();

        entityManager.persist(supplier);
        entityManager.persist(product);
        entityManager.persist(category);

        supplier.products.add(product);
        category.products.add(product);

        transaction.commit();
    }

    public static void addSupplierAndProduct(Session session) {
        Supplier supplier = Creator.createSupplier();
        Product product = Creator.createProduct();

        session.clear();
        Transaction transaction = session.beginTransaction();

        session.save(supplier);
        session.save(product);

        supplier.products.add(product);

        transaction.commit();
    }
}


 */
```