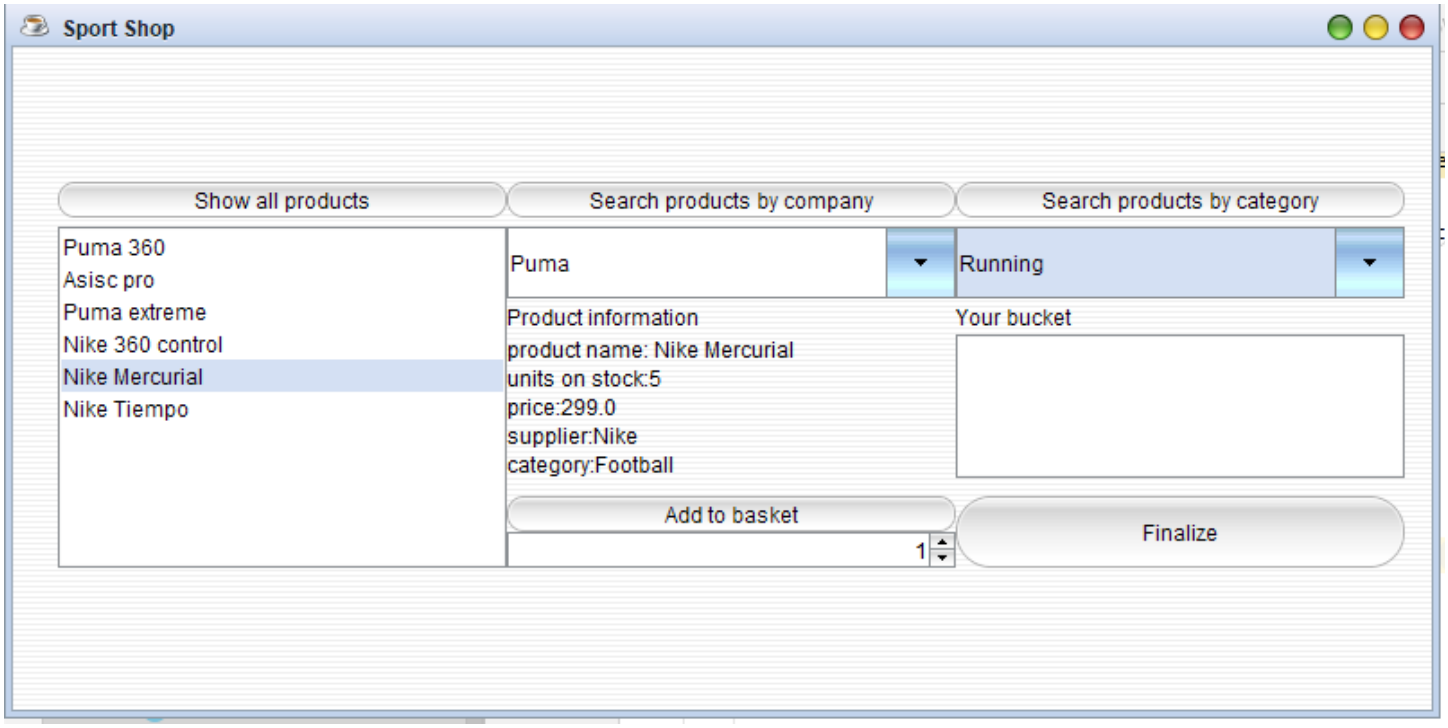
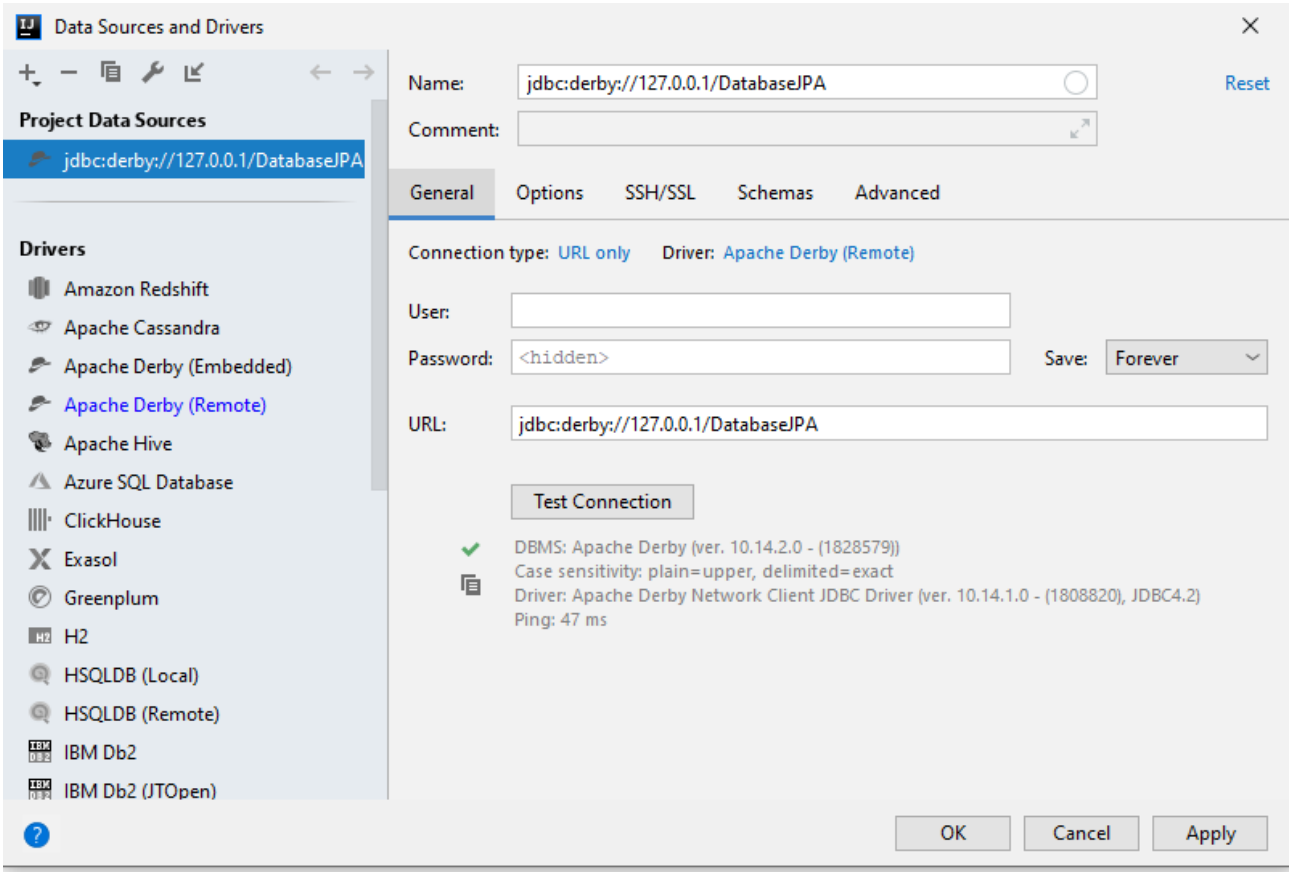
Krzysztof Bieniasz

Sprawozdanie z laboratorium nr 3 z przedmiotu Bazy Danych: Hibernate

Spis treści:



I. Basics



Udane podłączenie się z poziomu IntelliJ do uruchomionego serwera Derby do utworzonej przeze mnie bazy danych.

II Stworzenie klasy Product

Ważne jest, aby dodać mapowanie klasy w pliku konfiguracyjnym hibernate'a dodać informacje o mapowaniu klasy Product:

```
<mapping class="Product"/>
```

Kod klasy Product:

```
@Entity(name="Products")
public class Product {
    @Id
    private String productName;
    private int unitsOnStock;

    public Product() {}

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

Kod klasy dostarczającej instancję sesji Hibernate'a:

```
public class HibernateFactory {

    private static final SessionFactory sessionFactory;
    private HibernateFactory() {}
    static {
        try {
            sessionFactory = new
Configuration().configure().buildSessionFactory();
        } catch (Throwable ex) {
            System.out.println("Error in building session factory " + ex);
            throw new ExceptionInInitializerError(ex);
        }
    }
    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }
}
```

Kod klasy pokazującej działanie:

```
public class Application {
    public static void main(String[] args) {
        try (Session session =
HibernateFactory.getSessionFactory().openSession()) {
            Transaction transaction = session.beginTransaction();
            Scanner inputScanner = new Scanner(System.in);
            System.out.println("Product Name:");
            String productName = inputScanner.nextLine();
            System.out.println("Units in Stock:");
            int productQuantity = inputScanner.nextInt();
            Product product = new Product(productName, productQuantity);
            session.save(product);
            transaction.commit();
        }
    }
}
```

```
Hibernate:

    create table Products (
       productName varchar(255) not null,
        unitsOnStock integer not null,
        primary key (productName)
    )
lis 20, 2019 9:52:51 AM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator initiateService
INFO: HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
Product Name:
 Nike Revolution
Units in Stock:
10
Hibernate:
    /* insert Product
        */ insert
        into
            Products
            (unitsOnStock, productName)
        values
            (?, ?)

Process finished with exit code 0
```

Przykład selecta z DataGrip'a:

```
select * from products;
```

	PRODUCTNAME	UNITSONSTOCK
1	Nike Revolution	10

III. Wprowadzenia pojęcia dostawcy do modelu

```
@Entity(name="Suppliers")
public class Supplier {
    @Id
    private String companyName;
    private String street;
    private String city;

    public Supplier() { }

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

}
```

Należało zamodelować relację wiele-do-jednego ze strony klasy Produkt. W tym celu należało dodać pole Supplier w klasie Produkt.

```
@Entity(name="Products")
public class Product {
    @Id
    private String productName;
    private int unitsOnStock;


    @ManyToOne
    @JoinColumn
    private Supplier supplier;

    public Product() {}

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

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

Kod klasy pokazującej działanie:

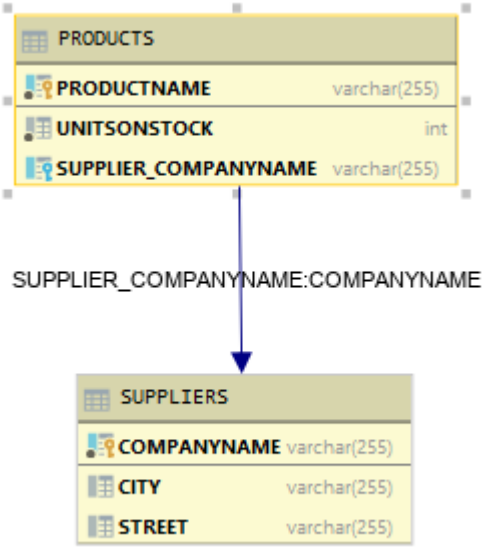
```
public class Application {
    public static void main(String[] args) {

        try (Session session =
HibernateFactory.getSessionFactory().openSession()) {
            Transaction transaction = session.beginTransaction();
            Supplier supplier = new Supplier("Adidas Corporation", "Broadway
16", "New York");
            session.save(supplier);
            Product product = new Product("Adidas 11pro", 20);
            session.save(product);
            transaction.commit();
        }


        try (Session session =
HibernateFactory.getSessionFactory().openSession()) {
            Product product = session.get(Product.class, "Adidas 11pro");
            Supplier supplier = session.get(Supplier.class, "Adidas
Corporation");
            Transaction transaction = session.beginTransaction();
            product.setSupplier(supplier);
            session.update(product);
            transaction.commit();
        }
}
```

Logi działania Hibernate’a:

```
Hibernate:
    /* insert Supplier
        */ insert
        into
            Suppliers
            (city, street, companyName)
        values
            (?, ?, ?)
Hibernate:
    /* insert Product
        */ insert
        into
            Products
            (supplier_companyName, unitsOnStock, productName)
        values
            (?, ?, ?)
Hibernate:
    select
        product0_.productName as productN1_0_0_,
        product0_.supplier_companyName as supplier3_0_0_,
        product0_.unitsOnStock as unitsOnS2_0_0_,
        supplier1_.companyName as companyN1_1_1_,
        supplier1_.city as city2_1_1_,
        supplier1_.street as street3_1_1_
    from
        Products product0_
    left outer join
        Suppliers supplier1_
            on product0_.supplier_companyName=supplier1_.companyName
    where
        product0_.productName=?
Hibernate:
    select
        supplier0_.companyName as companyN1_1_0_,
        supplier0_.city as city2_1_0_,
        supplier0_.street as street3_1_0_
```

```
    from
        Suppliers supplier0_
    where
        supplier0_.companyName=?
Hibernate:
    /* update
        Product */ update
            Products
        set
            supplier_companyName=?,
            unitsOnStock=?
        where
            productName=?
```

Diagram bazy danych:



Wywołania select'a:

	PRODUCTNAME	UNITSONSTOCK	SUPPLIER_COMPANYNAME
1	Nike Revolution	10	*<null>*
2	Adidas 11pro	20	Adidas Corporation

	COMPANYNAME	CITY	STREET
1	Nike Corporation	New York	Broadway 17
2	Adidas Corporation	New York	Broadway 16

Przypisanie dostawcy produktowi „Nike Revolution" i wywołania select'a

```java
try (Session session = HibernateUtils.getSessionFactory().openSession()) {
    Product product = session.get(Product.class, "Nike Revolution");
    Supplier supplier = session.get(Supplier.class, "Nike Corporation");
    Transaction transaction = session.beginTransaction();
    product.setSupplier(supplier);
    session.update(product);
    transaction.commit();
}
```

IV.

	PRODUCTNAME	UNITSONSTOCK	SUPPLIER_COMPANYNAME
1	Nike Revolution	10	Nike Corporation
2	Adidas 11pro	20	Adidas Corporation

Odwrócenie relacji dostawca-produkt

a) z pomocą tabeli łącznikowej

Kod klasy Supplier:

```
@Entity(name="Suppliers")
public class Supplier {
    @Id
    private String companyName;
    private String street;
    private String city;
    @OneToMany
    private Set<Product> productSet;

    public Supplier() { }

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

    public void addProductToProductSet(Product product)
    {
        productSet.add(product);
    }

}
```

Kod klasy Product:

```
@Entity(name="Products")
public class Product {
    @Id
    private String productName;
    private int unitsOnStock;

    public Product() {}

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

Przykład działania:

```
public class Application {
    public static void main(String[] args) {
        try (Session session =
HibernateFactory.getSessionFactory().openSession())
        {
```

```
            Transaction transaction = session.beginTransaction();
            Product product1 = new Product("Hexagon 2.0",5);
            Product product2 = new Product("Hexagon 3.0",5);
            Product product3 = new Product("Hexagon 4.0",5);
            session.save(product1);
            session.save(product2);
            session.save(product3);


            Supplier supplier = new Supplier("Kross","Rowerowa 10","Zabrze");
            session.save(supplier);
            transaction.commit();
        }

        System.out.println("Udane dodanie do bazy");

        try (Session session =
HibernateFactory.getSessionFactory().openSession())
        {
            Transaction transaction = session.beginTransaction();
            Supplier supplier = session.get(Supplier.class, "Kross");
            supplier.addProductToProductSet(session.get(Product.class, "Hexagon
2.0"));
            supplier.addProductToProductSet(session.get(Product.class, "Hexagon
3.0"));
            supplier.addProductToProductSet(session.get(Product.class, "Hexagon
4.0"));
            transaction.commit();

        }
}
```

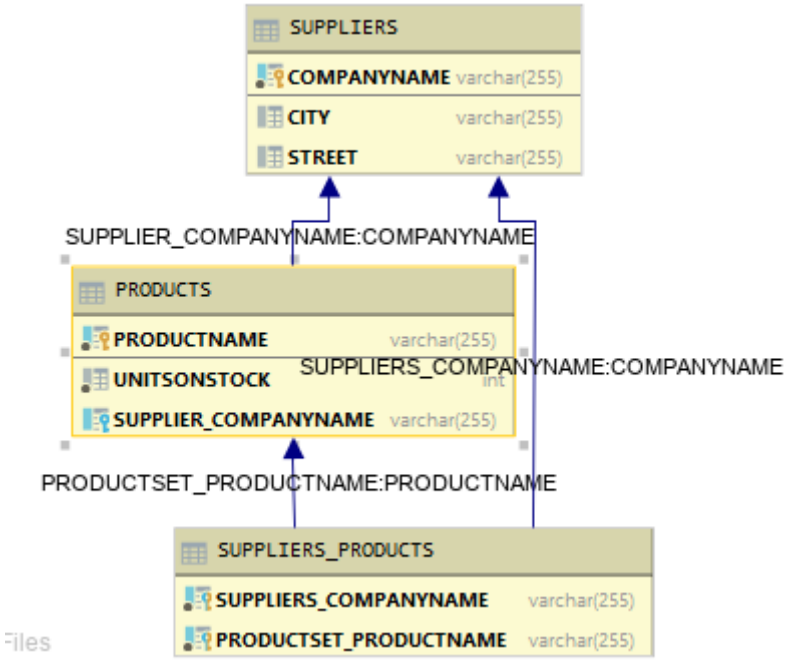
Logi Hibernate'a:

```
Hibernate:

    alter table Suppliers_Products
       drop constraint UK_4rinrs7s3svv0hiarwcsoj92c
Hibernate:

    alter table Suppliers_Products
       add constraint UK_4rinrs7s3svv0hiarwcsoj92c unique (productSet_productName)
lis 20, 2019 3:00:53 PM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator
initiateService
INFO: HHH000490: Using JtaPlatform implementation:
[org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
Hibernate:
    /* insert Product
        */ insert
        into
            Products
            (unitsOnStock, productName)
        values
            (?, ?)
Hibernate:
    /* insert Product
        */ insert
        into
            Products
            (unitsOnStock, productName)
        values
            (?, ?)
Hibernate:
    /* insert Product
        */ insert
        into
            Products
```

```
            (unitsOnStock, productName)
        values
            (?, ?)
Hibernate:
    /* insert Supplier
        */ insert
        into
            Suppliers
            (city, street, companyName)
        values
            (?, ?, ?)
Udane dodanie do bazy
Hibernate:
    select
        supplier0_.companyName as companyN1_1_0_,
        supplier0_.city as city2_1_0_,
        supplier0_.street as street3_1_0_
    from
        Suppliers supplier0_
    where
        supplier0_.companyName=?
Hibernate:
    select
        product0_.productName as productN1_0_0_,
        product0_.unitsOnStock as unitsOnS2_0_0_
    from
        Products product0_
    where
        product0_.productName=?
Hibernate:
    select
        productset0_.Suppliers_companyName as Supplier1_2_0_,
        productset0_.productSet_productName as productS2_2_0_,
        product1_.productName as productN1_0_1_,
        product1_.unitsOnStock as unitsOnS2_0_1_
    from
        Suppliers_Products productset0_
    inner join
        Products product1_
            on productset0_.productSet_productName=product1_.productName
    where
        productset0_.Suppliers_companyName=?
Hibernate:
    select
        product0_.productName as productN1_0_0_,
        product0_.unitsOnStock as unitsOnS2_0_0_
    from
        Products product0_
    where
        product0_.productName=?
Hibernate:
    select
        product0_.productName as productN1_0_0_,
        product0_.unitsOnStock as unitsOnS2_0_0_
    from
        Products product0_
    where
        product0_.productName=?
Hibernate:
    /* insert collection
        row Supplier.productSet */ insert
        into
            Suppliers_Products
            (Suppliers_companyName, productSet_productName)
        values
            (?, ?)
Hibernate:
    /* insert collection
        row Supplier.productSet */ insert
        into
            Suppliers_Products
            (Suppliers_companyName, productSet_productName)
        values
            (?, ?)
Hibernate:
    /* insert collection
        row Supplier.productSet */ insert
        into
```

```
    Suppliers_Products
    (Suppliers_companyName, productSet_productName)
values
    (?, ?)
```

Diagram bazy danych (istnieje połączenie bezpośrednie pomiędzy dostawcami a produktami, ponieważ nie usunąłem wcześniej istniejących tabel):



Przykłady select'ów:

	PRODUCTNAME	UNITSONSTOCK	SUPPLIER_COMPANYNAME
1	Nike Revolution	10	Nike Corporation
2	Adidas 11pro	20	Adidas Corporation
3	Hexagon 2.0	5	<null>
4	Hexagon 3.0	5	<null>
5	Hexagon 4.0	5	<null>

	SUPPLIERS_COMPANYNAME	PRODUCTSET_PRODUCTNAME
1	Kross	Hexagon 2.0
2	Kross	Hexagon 3.0
3	Kross	Hexagon 4.0

b) przypadek bez tabeli łącznikowej – wystarczy jedynie w klasie Supplier dodać adnotację JoinColumn nad atrybutem productSet

Kod klasy Supplier:

```
@Entity(name="Suppliers")
public class Supplier {
    @Id
    private String companyName;
    private String street;
    private String city;
    @OneToMany
    @JoinColumn
    private Set<Product> productSet;

    public Supplier() { }

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

    public void addProductToProductSet(Product product)
    {
        productSet.add(product);
    }

}
```

Logi Hibernate'a:

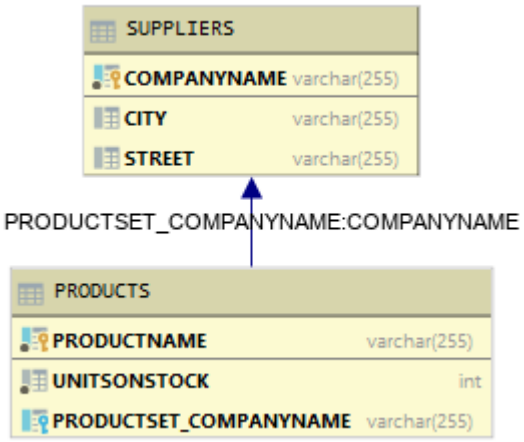
```
Hibernate:
    /* insert Product
        */ insert
        into
            Products
            (unitsOnStock, productName)
        values
            (?, ?)
Hibernate:
    /* insert Product
        */ insert
        into
            Products
            (unitsOnStock, productName)
        values
            (?, ?)
Hibernate:
    /* insert Product
        */ insert
        into
            Products
            (unitsOnStock, productName)
```

```
        values
            (?, ?)
Hibernate:
    /* insert Supplier
        */ insert
        into
            Suppliers
            (city, street, companyName)
        values
            (?, ?, ?)
Udane dodanie do bazy
Hibernate:
    select
        supplier0_.companyName as companyN1_1_0_,
        supplier0_.city as city2_1_0_,
        supplier0_.street as street3_1_0_
    from
        Suppliers supplier0_
    where
        supplier0_.companyName=?
Hibernate:
    select
        product0_.productName as productN1_0_0_,
        product0_.unitsOnStock as unitsOnS2_0_0_
    from
        Products product0_
    where
        product0_.productName=?
Hibernate:
    select
        productset0_.productSet_companyName as productS3_0_0_,
        productset0_.productName as productN1_0_0_,
        productset0_.productName as productN1_0_1_,
        productset0_.unitsOnStock as unitsOnS2_0_1_
    from
        Products productset0_
    where
        productset0_.productSet_companyName=?
Hibernate:
    select
        product0_.productName as productN1_0_0_,
        product0_.unitsOnStock as unitsOnS2_0_0_
    from
        Products product0_
    where
        product0_.productName=?
Hibernate:
    select
        product0_.productName as productN1_0_0_,
        product0_.unitsOnStock as unitsOnS2_0_0_
    from
        Products product0_
    where
        product0_.productName=?
Hibernate:
    /* create one-to-many row Supplier.productSet */ update
        Products
    set
        productSet_companyName=?
    where
        productName=?
Hibernate:
    /* create one-to-many row Supplier.productSet */ update
        Products
    set
        productSet_companyName=?
    where
        productName=?
Hibernate:
    /* create one-to-many row Supplier.productSet */ update
        Products
    set
        productSet_companyName=?
    where
        productName=?
```

Wywołanie select'a:

	PRODUCTNAME	UNITSONSTOCK	PRODUCTSET_COMPANYNAME
1	Hexagon 2.0	5	Kross
2	Hexagon 3.0	5	Kross
3	Hexagon 4.0	5	Kross

Diagram bazy danych:



V. Podwójna relacja pomiędzy produktem a dostawcą.

Istotnym elementem było nadpisanie metod equals oraz hashCode w klasie Supplier oraz Product z pominięciem atrybutów odwołujących się do obiektu relacji.

Kod klasy Product:

```
@Entity(name="Products")
public class Product {
    @Id
    private String productName;
    private int unitsOnStock;

    @ManyToOne
    @JoinColumn
    private Supplier supplier;


    public Product() {}

    public Product(String productName, int unitsOnStock) {
        this.productName = productName;
        this.unitsOnStock = unitsOnStock;
        this.supplier = null;
    }


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

    @Override
```

```
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Product product = (Product) o;
        return unitsOnStock == product.unitsOnStock &&
                productName.equals(product.productName);
    }

    @Override
    public int hashCode() {
        return Objects.hash(productName, unitsOnStock);
    }
}
```

Kod klasy Supplier:

```
@Entity(name="Suppliers")
public class Supplier {
    @Id
    private String companyName;
    private String street;
    private String city;
    @OneToMany
    @JoinColumn
    private Set<Product> productSet;

    public Supplier() { }

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

    public void addProductToProductSet(Product product)
    {
        productSet.add(product);
        product.setSupplier(this);
    }


    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Supplier supplier = (Supplier) o;
        return companyName.equals(supplier.companyName);
    }

    @Override
    public int hashCode() {
        return Objects.hash(companyName) + 17*Objects.hash(street);
    }
}
```

Kod klasy pokazującej działanie:

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

    try (Session session = HibernateFactory.getSessionFactory().openSession())
    {
        Transaction transaction = session.beginTransaction();
        Product product1 = new Product("Hexagon 2.0",5);
        Product product2 = new Product("Hexagon 3.0",5);
        Product product3 = new Product("Hexagon 4.0",5);
```

```
        session.save(product1);
        session.save(product2);
        session.save(product3);
        Supplier supplier = new Supplier("Kross","Rowerowa 10","Zabrze");
        session.save(supplier);
        transaction.commit();
    }

    System.out.println("Udane dodanie do bazy");

    try (Session session = HibernateUtils.getSessionFactory().openSession())
    {
        Transaction tx = session.beginTransaction();
        Supplier supplier = session.get(Supplier.class, "Kross");
        supplier.addProductToProductSet(session.get(Product.class, "Hexagon
2.0"));
        supplier.addProductToProductSet(session.get(Product.class, "Hexagon
3.0"));
        supplier.addProductToProductSet(session.get(Product.class, "Hexagon
4.0"));
        tx.commit();

    }
}
```

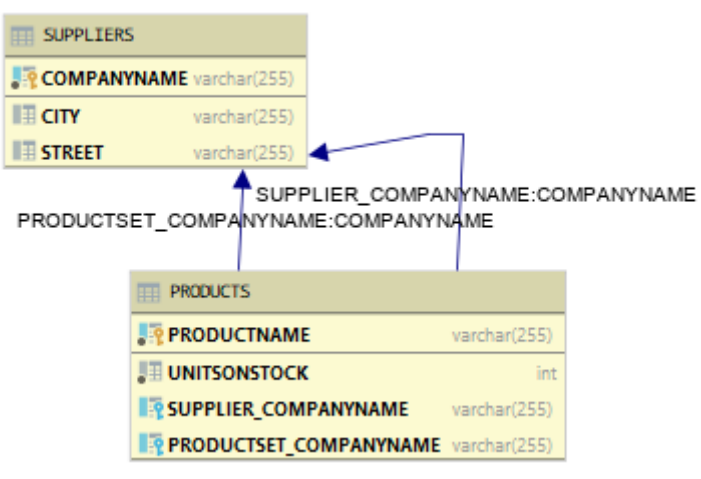
Logi Hibernate'a:

```
Hibernate:

    create table Products (
       productName varchar(255) not null,
        unitsOnStock integer not null,
        supplier_companyName varchar(255),
        productSet_companyName varchar(255),
        primary key (productName)
    )
Hibernate:

    create table Suppliers (
       companyName varchar(255) not null,
        city varchar(255),
        street varchar(255),
        primary key (companyName)
    )
Hibernate:

    alter table Products
       add constraint FKlj63cbso72i9to92f2ldetnfd
       foreign key (supplier_companyName)
       references Suppliers
Hibernate:

    alter table Products
       add constraint FKnxe0kgaule62q4byqhsjleiln
       foreign key (productSet_companyName)
       references Suppliers
lis 20, 2019 6:42:34 PM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator
initiateService
INFO: HHH000490: Using JtaPlatform implementation:
[org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
Hibernate:
    /* insert Product
        */ insert
        into
            Products
            (supplier_companyName, unitsOnStock, productName)
        values
            (?, ?, ?)
Hibernate:
    /* insert Product
        */ insert
```

```
        into
            Products
            (supplier_companyName, unitsOnStock, productName)
        values
            (?, ?, ?)
Hibernate:
    /* insert Product
        */ insert
        into
            Products
            (supplier_companyName, unitsOnStock, productName)
        values
            (?, ?, ?)
Hibernate:
    /* insert Supplier
        */ insert
        into
            Suppliers
            (city, street, companyName)
        values
            (?, ?, ?)

Hibernate:
    select
        supplier0_.companyName as companyN1_1_0_,
        supplier0_.city as city2_1_0_,
        supplier0_.street as street3_1_0_
    from
        Suppliers supplier0_
    where
        supplier0_.companyName=?
Hibernate:
    select
        product0_.productName as productN1_0_0_,
        product0_.supplier_companyName as supplier3_0_0_,
        product0_.unitsOnStock as unitsOnS2_0_0_,
        supplier1_.companyName as companyN1_1_1_,
        supplier1_.city as city2_1_1_,
        supplier1_.street as street3_1_1_
    from
        Products product0_
    left outer join
        Suppliers supplier1_
            on product0_.supplier_companyName=supplier1_.companyName
    where
        product0_.productName=?
Hibernate:
    select
        productset0_.productSet_companyName as productS4_0_0_,
        productset0_.productName as productN1_0_0_,
        productset0_.productName as productN1_0_1_,
        productset0_.supplier_companyName as supplier3_0_1_,
        productset0_.unitsOnStock as unitsOnS2_0_1_,
        supplier1_.companyName as companyN1_1_2_,
        supplier1_.city as city2_1_2_,
        supplier1_.street as street3_1_2_
    from
        Products productset0_
    left outer join
        Suppliers supplier1_
            on productset0_.supplier_companyName=supplier1_.companyName
    where
        productset0_.productSet_companyName=?
Hibernate:
    select
        product0_.productName as productN1_0_0_,
        product0_.supplier_companyName as supplier3_0_0_,
        product0_.unitsOnStock as unitsOnS2_0_0_,
        supplier1_.companyName as companyN1_1_1_,
        supplier1_.city as city2_1_1_,
        supplier1_.street as street3_1_1_
    from
        Products product0_
    left outer join
        Suppliers supplier1_
            on product0_.supplier_companyName=supplier1_.companyName
    where
        product0_.productName=?
```

```
Hibernate:
    select
        product0_.productName as productN1_0_0_,
        product0_.supplier_companyName as supplier3_0_0_,
        product0_.unitsOnStock as unitsOnS2_0_0_,
        supplier1_.companyName as companyN1_1_1_,
        supplier1_.city as city2_1_1_,
        supplier1_.street as street3_1_1_
    from
        Products product0_
    left outer join
        Suppliers supplier1_
            on product0_.supplier_companyName=supplier1_.companyName
    where
        product0_.productName=?
Hibernate:
    /* update
        Product */ update
            Products
        set
            supplier_companyName=?,
            unitsOnStock=?
        where
            productName=?
Hibernate:
    /* update
        Product */ update
            Products
        set
            supplier_companyName=?,
            unitsOnStock=?
        where
            productName=?
Hibernate:
    /* update
        Product */ update
            Products
        set
            supplier_companyName=?,
            unitsOnStock=?
        where
            productName=?
Hibernate:
    /* create one-to-many row Supplier.productSet */ update
        Products
    set
        productSet_companyName=?
    where
        productName=?
Hibernate:
    /* create one-to-many row Supplier.productSet */ update
        Products
    set
        productSet_companyName=?
    where
        productName=?
Hibernate:
    /* create one-to-many row Supplier.productSet */ update
        Products
    set
        productSet_companyName=?
    where
        productName=?
```

Diagram bazy danych (przekrzywiony, aby pokazać podpisy w DataGripie):



Przykłady selecta:

	COMPANYNAME	CITY	STREET
1	Kross	Zabrze	Rowerowa 10

	PRODUCTNAME	UNITSONSTOCK	SUPPLIER_COMPANYNAME	PRODUCTSET_COMPANYNAME
1	Hexagon 2.0	5	Kross	Kross
2	Hexagon 3.0	5	Kross	Kross
3	Hexagon 4.0	5	Kross	Kross

VI. Dodanie klasy kategoria.

Zdecydowałem się na analogiczne podwójne połączenie encji kategorii oraz produktu.

Kod klasy Category:

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

    @OneToMany
    @JoinColumn
    private Set<Product> productSet;

    public Category() {}

    public Category(String name) {
        this.name = name;
        productSet = new HashSet<>();
    }

    public void addProductToProductSet(Product product)
    {
        productSet.add(product);
        product.setCategory(this);
    }

    @Override
```

```
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Category category = (Category) o;
        return categoryID == category.categoryID &&
                Objects.equals(name, category.name);
    }
    @Override
    public int hashCode() {
        return Objects.hash(categoryID, name);
    }

    public Set<Product> getProductSet() {
        return productSet;    }
}
```

Logi Hibernate'a:

```
    create table Categories (
       categoryID integer not null,
        name varchar(255),
        primary key (categoryID)
    )
Hibernate:

    create table Products (
       productName varchar(255) not null,
        unitsOnStock integer not null,
        category_categoryID integer,
        supplier_companyName varchar(255),
        productSet_categoryID integer,
        productSet_companyName varchar(255),
        primary key (productName)
    )
Hibernate:

    create table Suppliers (
       companyName varchar(255) not null,
        city varchar(255),
        street varchar(255),
        primary key (companyName)
    )
Hibernate:

    alter table Products
       add constraint FKis8r738rffb59r366t82oth30
       foreign key (category_categoryID)
       references Categories
Hibernate:

    alter table Products
       add constraint FKlj63cbso72i9to92f2ldetnfd
       foreign key (supplier_companyName)
       references Suppliers
Hibernate:

    alter table Products
       add constraint FKek3q6b4xwuay4haavt1yxdv9a
       foreign key (productSet_categoryID)
       references Categories
Hibernate:

    alter table Products
       add constraint FKnxe0kgaule62q4byqhsjleiln
       foreign key (productSet_companyName)
       references Suppliers
lis 20, 2019 7:46:36 PM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator
initiateService
INFO: HHH000490: Using JtaPlatform implementation:
[org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
Hibernate:

values
    next value for hibernate_sequence
```

```
Hibernate:

values
    next value for hibernate_sequence
Hibernate:
    /* insert Product
        */ insert
        into
            Products
            (category_categoryID, supplier_companyName, unitsOnStock, productName)
        values
            (?, ?, ?, ?)
Hibernate:
    /* insert Product
        */ insert
        into
            Products
            (category_categoryID, supplier_companyName, unitsOnStock, productName)
        values
            (?, ?, ?, ?)
Hibernate:
    /* insert Product
        */ insert
        into
            Products
            (category_categoryID, supplier_companyName, unitsOnStock, productName)
        values
            (?, ?, ?, ?)
Hibernate:
    /* insert Supplier
        */ insert
        into
            Suppliers
            (city, street, companyName)
        values
            (?, ?, ?)
Hibernate:
    /* insert Category
        */ insert
        into
            Categories
            (name, categoryID)
        values
            (?, ?)
Hibernate:
    /* insert Category
        */ insert
        into
            Categories
            (name, categoryID)
        values
            (?, ?)
Udane dodanie do bazy
Hibernate:
    select
        supplier0_.companyName as companyN1_2_0_,
        supplier0_.city as city2_2_0_,
        supplier0_.street as street3_2_0_
    from
        Suppliers supplier0_
    where
        supplier0_.companyName=?
Hibernate:
    select
        product0_.productName as productN1_1_0_,
        product0_.category_categoryID as category3_1_0_,
        product0_.supplier_companyName as supplier4_1_0_,
        product0_.unitsOnStock as unitsOnS2_1_0_,
        category1_.categoryID as category1_0_1_,
        category1_.name as name2_0_1_,
        supplier2_.companyName as companyN1_2_2_,
        supplier2_.city as city2_2_2_,
        supplier2_.street as street3_2_2_
    from
        Products product0_
    left outer join
        Categories category1_
            on product0_.category_categoryID=category1_.categoryID
```

```
    left outer join
        Suppliers supplier2_
            on product0_.supplier_companyName=supplier2_.companyName
    where
        product0_.productName=?
Hibernate:
    select
        productset0_.productSet_companyName as productS6_1_0_,
        productset0_.productName as productN1_1_0_,
        productset0_.productName as productN1_1_1_,
        productset0_.category_categoryID as category3_1_1_,
        productset0_.supplier_companyName as supplier4_1_1_,
        productset0_.unitsOnStock as unitsOnS2_1_1_,
        category1_.categoryID as category1_0_2_,
        category1_.name as name2_0_2_,
        supplier2_.companyName as companyN1_2_3_,
        supplier2_.city as city2_2_3_,
        supplier2_.street as street3_2_3_
    from
        Products productset0_
    left outer join
        Categories category1_
            on productset0_.category_categoryID=category1_.categoryID
    left outer join
        Suppliers supplier2_
            on productset0_.supplier_companyName=supplier2_.companyName
    where
        productset0_.productSet_companyName=?
Hibernate:
    select
        product0_.productName as productN1_1_0_,
        product0_.category_categoryID as category3_1_0_,
        product0_.supplier_companyName as supplier4_1_0_,
        product0_.unitsOnStock as unitsOnS2_1_0_,
        category1_.categoryID as category1_0_1_,
        category1_.name as name2_0_1_,
        supplier2_.companyName as companyN1_2_2_,
        supplier2_.city as city2_2_2_,
        supplier2_.street as street3_2_2_
    from
        Products product0_
    left outer join
        Categories category1_
            on product0_.category_categoryID=category1_.categoryID
    left outer join
        Suppliers supplier2_
            on product0_.supplier_companyName=supplier2_.companyName
    where
        product0_.productName=?
Hibernate:
    select
        product0_.productName as productN1_1_0_,
        product0_.category_categoryID as category3_1_0_,
        product0_.supplier_companyName as supplier4_1_0_,
        product0_.unitsOnStock as unitsOnS2_1_0_,
        category1_.categoryID as category1_0_1_,
        category1_.name as name2_0_1_,
        supplier2_.companyName as companyN1_2_2_,
        supplier2_.city as city2_2_2_,
        supplier2_.street as street3_2_2_
    from
        Products product0_
    left outer join
        Categories category1_
            on product0_.category_categoryID=category1_.categoryID
    left outer join
        Suppliers supplier2_
            on product0_.supplier_companyName=supplier2_.companyName
    where
        product0_.productName=?
Hibernate:
    select
        category0_.categoryID as category1_0_0_,
        category0_.name as name2_0_0_
    from
        Categories category0_
    where
        category0_.categoryID=?
```

```
Hibernate:
    select
        productset0_.productSet_categoryID as productS5_1_0_,
        productset0_.productName as productN1_1_0_,
        productset0_.productName as productN1_1_1_,
        productset0_.category_categoryID as category3_1_1_,
        productset0_.supplier_companyName as supplier4_1_1_,
        productset0_.unitsOnStock as unitsOnS2_1_1_,
        category1_.categoryID as category1_0_2_,
        category1_.name as name2_0_2_,
        supplier2_.companyName as companyN1_2_3_,
        supplier2_.city as city2_2_3_,
        supplier2_.street as street3_2_3_
    from
        Products productset0_
    left outer join
        Categories category1_
            on productset0_.category_categoryID=category1_.categoryID
    left outer join
        Suppliers supplier2_
            on productset0_.supplier_companyName=supplier2_.companyName
    where
        productset0_.productSet_categoryID=?
Hibernate:
    select
        category0_.categoryID as category1_0_0_,
        category0_.name as name2_0_0_
    from
        Categories category0_
    where
        category0_.categoryID=?
Hibernate:
    select
        productset0_.productSet_categoryID as productS5_1_0_,
        productset0_.productName as productN1_1_0_,
        productset0_.productName as productN1_1_1_,
        productset0_.category_categoryID as category3_1_1_,
        productset0_.supplier_companyName as supplier4_1_1_,
        productset0_.unitsOnStock as unitsOnS2_1_1_,
        category1_.categoryID as category1_0_2_,
        category1_.name as name2_0_2_,
        supplier2_.companyName as companyN1_2_3_,
        supplier2_.city as city2_2_3_,
        supplier2_.street as street3_2_3_
    from
        Products productset0_
    left outer join
        Categories category1_
            on productset0_.category_categoryID=category1_.categoryID
    left outer join
        Suppliers supplier2_
            on productset0_.supplier_companyName=supplier2_.companyName
    where
        productset0_.productSet_categoryID=?
Hibernate:
    /* update
        Product */ update
            Products
        set
            category_categoryID=?,
            supplier_companyName=?,
            unitsOnStock=?
        where
            productName=?
Hibernate:
    /* update
        Product */ update
            Products
        set
            category_categoryID=?,
            supplier_companyName=?,
            unitsOnStock=?
        where
            productName=?
Hibernate:
    /* update
        Product */ update
            Products
```

```
        set
            category_categoryID=?,
            supplier_companyName=?,
            unitsOnStock=?
        where
            productName=?
Hibernate:
    /* create one-to-many row Supplier.productSet */ update
        Products
    set
        productSet_companyName=?
    where
        productName=?
Hibernate:
    /* create one-to-many row Supplier.productSet */ update
        Products
    set
        productSet_companyName=?
    where
        productName=?
Hibernate:
    /* create one-to-many row Supplier.productSet */ update
        Products
    set
        productSet_companyName=?
    where
        productName=?
Hibernate:
    /* create one-to-many row Category.productSet */ update
        Products
    set
        productSet_categoryID=?
    where
        productName=?
Hibernate:
    /* create one-to-many row Category.productSet */ update
        Products
    set
        productSet_categoryID=?
    where
        productName=?
Hibernate:
    /* create one-to-many row Category.productSet */ update
        Products
    set
        productSet_categoryID=?
    where
        productName=?
```

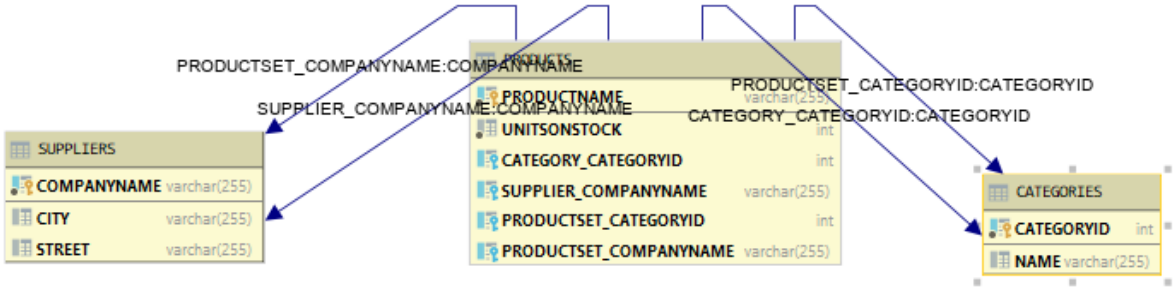
Wywołania selecta:

3 rows — Tab-se...d (TSV)

	PRODUCTNAME	UNITSONSTOCK	CATEGORY_CATEGORYID	SUPPLIER_COMPANYNAME	PRODUCTSET_CATEGORYID	PRODUCTSET_COMPANYNAME
1	Fisher 360	5	7	Fisher	7	Fisher
2	Fisher 480	5	7	Fisher	7	Fisher
3	Fisher Snowboard	5	8	Fisher	8	Fisher

	CATEGORYID	NAME
1	7	Skiing
2	8	Snowboarding

Diagram bazy danych:



Wyłuskiwanie informacji na podstawie produktu o kategorii i na odwrót

```
try (Session session = HibernateFactory.getSessionFactory().openSession())
{
    Transaction tx = session.beginTransaction();
    Product product1 =session.get(Product.class, "Fisher 360");
    Query  query1 = session.createQuery("select c from Categories c where
c.categoryID=:catID");
    query1.setParameter("catID", product1.getCategory().getCategoryID());
    // właściwie problem wyłuskiwania jest sztuczny, bo mamy odpowiednie pola
zarówno w produkcie jak i kategorii
    Category associatedCategory = (Category) query1.getResultList().get(0);

    Query query2 = session.createQuery("select p from Products p where
p.category=:cat");
    query2.setParameter("cat", associatedCategory);
    List<Product> associatedProduct = query2.getResultList();
    tx.commit();
    System.out.println(associatedCategory.getName());
    for(Product product : associatedProduct )
    {
        System.out.println(product.getProductName());
    }

}
```

Logi Hibernate’a:

```
Hibernate:
    select
        product0_.productName as productN1_1_0_,
        product0_.category_categoryID as category3_1_0_,
        product0_.supplier_companyName as supplier4_1_0_,
        product0_.unitsOnStock as unitsOnS2_1_0_,
        category1_.categoryID as category1_0_1_,
        category1_.name as name2_0_1_,
        supplier2_.companyName as companyN1_2_2_,
        supplier2_.city as city2_2_2_,
        supplier2_.street as street3_2_2_
    from
        Products product0_
    left outer join
        Categories category1_
            on product0_.category_categoryID=category1_.categoryID
    left outer join
        Suppliers supplier2_
            on product0_.supplier_companyName=supplier2_.companyName
    where
        product0_.productName=?
Hibernate:
    /* select
        c
    from
```

```
        Categories c
    where
        c.categoryID=:catID */ select
            category0_.categoryID as category1_0_,
            category0_.name as name2_0_
        from
            Categories category0_
        where
            category0_.categoryID=?
Hibernate:
    /* select
        p
    from
        Products p
    where
        p.category=:cat */ select
            product0_.productName as productN1_1_,
            product0_.category_categoryID as category3_1_,
            product0_.supplier_companyName as supplier4_1_,
            product0_.unitsOnStock as unitsOnS2_1_
        from
            Products product0_
        where
            product0_.category_categoryID=?
```

Skiing
Fisher 360
Fisher 480

VII. Dodanie klasy faktury oraz relacji wiele -do-wiele faktura-produkt.

Kod klasy Produkt:

```java
@Entity(name="Products")
public class Product {
    @Id
    private String productName;
    private int unitsOnStock;

    @ManyToOne
    @JoinColumn
    private Supplier supplier;

    @ManyToOne
    @JoinColumn
    private Category category;

    @ManyToMany
    private Set<Invoice> invoiceSet;

    public Product() {}

    public Product(String productName, int unitsOnStock) {
        this.productName = productName;
        this.unitsOnStock = unitsOnStock;
        this.supplier = null;
    }


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

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

    public void setInvoiceSet(Set<Invoice> invoiceSet) {
```

```
        this.invoiceSet = invoiceSet;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Product product = (Product) o;
        return unitsOnStock == product.unitsOnStock &&
                productName.equals(product.productName);
    }

    @Override
    public int hashCode() {
        return Objects.hash(productName, unitsOnStock);
    }

    public String getProductName() {
        return productName;
    }

    public int getUnitsOnStock() {
        return unitsOnStock;
    }

    public Supplier getSupplier() {
        return supplier;
    }

    public Category getCategory() {
        return category;
    }

    public Set<Invoice> getInvoiceSet() {
        return invoiceSet;
    }

    public void setUnitsOnStock(int unitsOnStock) {
        this.unitsOnStock = unitsOnStock;
    }
}
```

Kod klasy Invoice:

```
@Entity(name="Invoices")
public class Invoice {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int invoiceNumber;
    private int quantity;

    @ManyToMany(mappedBy = "invoiceSet")
    private Set<Product> productSet;

    public Invoice() {
        this.quantity = 0;
        productSet = new HashSet<>();
    }

    public void addProduct(Product product) {
        this.productSet.add(product);
        product.getInvoiceSet().add(this);
        this.quantity = 1;
```

```
        product.setUnitsOnStock(product.getUnitsOnStock()-1);
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Invoice invoice = (Invoice) o;
        return invoiceNumber == invoice.invoiceNumber;
    }

    @Override
    public int hashCode() {
        return Objects.hash(invoiceNumber);
    }
}
```

Kod klasy testującej działanie:

```
try (Session session = HibernateFacoty.getSessionFactory().openSession()) {
    Transaction transaction = session.beginTransaction();
    Product product1 = session.get(Product.class, "Fisher 360");
    Product product2 = session.get(Product.class, "Fisher 480");
    Product product3 = session.get(Product.class, "Fisher Snowboard");
    Invoice invoice1 = new Invoice();
    Invoice invoice2 = new Invoice();
    invoice1.addProduct(product1);
    invoice1.addProduct(product2);
    invoice2.addProduct(product2);
    invoice2.addProduct(product3);
    session.save(invoice1);
    session.save(invoice2);
    session.update(product1);
    session.update(product2);
    session.update(product3);
    transaction.commit();
}
```

Logi Hibernate’a:

```
Hibernate:
    select
        product0_.productName as productN1_2_0_,
        product0_.category_categoryID as category3_2_0_,
        product0_.supplier_companyName as supplier4_2_0_,
        product0_.unitsOnStock as unitsOnS2_2_0_,
        category1_.categoryID as category1_0_1_,
        category1_.name as name2_0_1_,
        supplier2_.companyName as companyN1_4_2_,
        supplier2_.city as city2_4_2_,
        supplier2_.street as street3_4_2_
    from
        Products product0_
    left outer join
        Categories category1_
            on product0_.category_categoryID=category1_.categoryID
    left outer join
        Suppliers supplier2_
            on product0_.supplier_companyName=supplier2_.companyName
    where
        product0_.productName=?
Hibernate:
    select
        product0_.productName as productN1_2_0_,
        product0_.category_categoryID as category3_2_0_,
        product0_.supplier_companyName as supplier4_2_0_,
        product0_.unitsOnStock as unitsOnS2_2_0_,
```

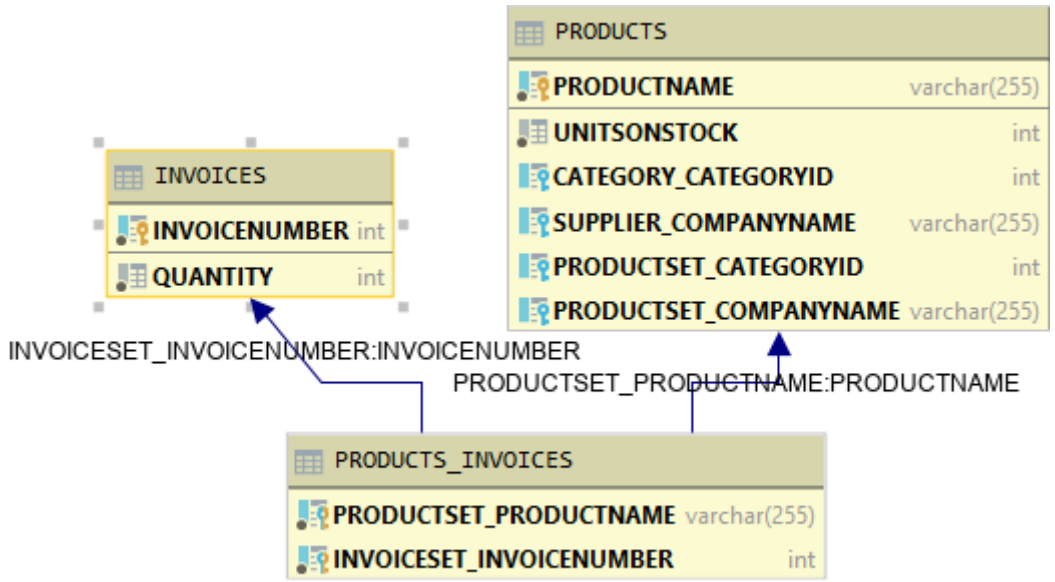
```
        category1_.categoryID as category1_0_1_,
        category1_.name as name2_0_1_,
        supplier2_.companyName as companyN1_4_2_,
        supplier2_.city as city2_4_2_,
        supplier2_.street as street3_4_2_
    from
        Products product0_
    left outer join
        Categories category1_
            on product0_.category_categoryID=category1_.categoryID
    left outer join
        Suppliers supplier2_
            on product0_.supplier_companyName=supplier2_.companyName
    where
        product0_.productName=?
Hibernate:
    select
        product0_.productName as productN1_2_0_,
        product0_.category_categoryID as category3_2_0_,
        product0_.supplier_companyName as supplier4_2_0_,
        product0_.unitsOnStock as unitsOnS2_2_0_,
        category1_.categoryID as category1_0_1_,
        category1_.name as name2_0_1_,
        supplier2_.companyName as companyN1_4_2_,
        supplier2_.city as city2_4_2_,
        supplier2_.street as street3_4_2_
    from
        Products product0_
    left outer join
        Categories category1_
            on product0_.category_categoryID=category1_.categoryID
    left outer join
        Suppliers supplier2_
            on product0_.supplier_companyName=supplier2_.companyName
    where
        product0_.productName=?
Hibernate:
    select
        invoiceset0_.productSet_productName as productS1_3_0_,
        invoiceset0_.invoiceSet_invoiceNumber as invoiceS2_3_0_,
        invoice1_.invoiceNumber as invoiceN1_1_1_,
        invoice1_.quantity as quantity2_1_1_
    from
        Products_Invoices invoiceset0_
    inner join
        Invoices invoice1_
            on invoiceset0_.invoiceSet_invoiceNumber=invoice1_.invoiceNumber
    where
        invoiceset0_.productSet_productName=?
Hibernate:
    select
        invoiceset0_.productSet_productName as productS1_3_0_,
        invoiceset0_.invoiceSet_invoiceNumber as invoiceS2_3_0_,
        invoice1_.invoiceNumber as invoiceN1_1_1_,
        invoice1_.quantity as quantity2_1_1_
    from
        Products_Invoices invoiceset0_
    inner join
        Invoices invoice1_
            on invoiceset0_.invoiceSet_invoiceNumber=invoice1_.invoiceNumber
    where
        invoiceset0_.productSet_productName=?
Hibernate:
    select
        invoiceset0_.productSet_productName as productS1_3_0_,
        invoiceset0_.invoiceSet_invoiceNumber as invoiceS2_3_0_,
        invoice1_.invoiceNumber as invoiceN1_1_1_,
        invoice1_.quantity as quantity2_1_1_
    from
        Products_Invoices invoiceset0_
    inner join
        Invoices invoice1_
            on invoiceset0_.invoiceSet_invoiceNumber=invoice1_.invoiceNumber
    where
        invoiceset0_.productSet_productName=?
Hibernate:

values
```

```
    next value for hibernate_sequence
Hibernate:

values
    next value for hibernate_sequence
Hibernate:
    /* insert Invoice
        */ insert
        into
            Invoices
            (quantity, invoiceNumber)
        values
            (?, ?)
Hibernate:
    /* insert Invoice
        */ insert
        into
            Invoices
            (quantity, invoiceNumber)
        values
            (?, ?)
Hibernate:
    /* update
        Product */ update
            Products
        set
            category_categoryID=?,
            supplier_companyName=?,
            unitsOnStock=?
        where
            productName=?
Hibernate:
    /* update
        Product */ update
            Products
        set
            category_categoryID=?,
            supplier_companyName=?,
            unitsOnStock=?
        where
            productName=?
Hibernate:
    /* update
        Product */ update
            Products
        set
            category_categoryID=?,
            supplier_companyName=?,
            unitsOnStock=?
        where
            productName=?
Hibernate:
    /* insert collection
        row Product.invoiceSet */ insert
        into
            Products_Invoices
            (productSet_productName, invoiceSet_invoiceNumber)
        values
            (?, ?)
Hibernate:
    /* insert collection
        row Product.invoiceSet */ insert
        into
            Products_Invoices
            (productSet_productName, invoiceSet_invoiceNumber)
        values
            (?, ?)
Hibernate:
    /* insert collection
        row Product.invoiceSet */ insert
        into
            Products_Invoices
            (productSet_productName, invoiceSet_invoiceNumber)
        values
            (?, ?)
```

Wywołanie selecta dla tabeli łącznikowej:

	PRODUCTSET_PRODUCTNAME	INVOICESET_INVOICENUMBER
1	Fisher 360	9
2	Fisher 480	9
3	Fisher Snowboard	10

Diagram bazy danych (oczywiście powstała tabela łącznikowa):



VIII. Wykorzystanie JPA

Najpierw stworzyłem analogiczną klasę do HibernateFactory, a mianowicie JPAFactory zwracającą entityManagerFactory. Należało też stworzyć plik konfiguracyjny persitance.xml i umieścić go w folderze META-INF.

Plik konfiguracyjny persintance.xml:

```
<?xml version="1.0"?>
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd"
             version="2.0">
    <persistence-unit name="myDatabaseConfig"
                      transaction-type="RESOURCE_LOCAL">
        <class>persistanceModel.Product</class>
        <class>persistanceModel.Supplier</class>
        <class>persistanceModel.Category</class>
        <class>persistanceModel.Invoice</class>
        <class>persistanceModel.Address</class>
        <class>persistanceModel.Company</class>
        <class>persistanceModel.Customer</class>
        <properties>
```

```
            <property name="hibernate.connection.driver_class"
                      value="org.apache.derby.jdbc.ClientDriver"/>
            <property name="hibernate.connection.url"
                      value="jdbc:derby://127.0.0.1/DatabaseJPA"/>
            <property name="hibernate.show_sql" value="true" />
            <property name="hibernate.format_sql" value="true" />
            <property name="hibernate.hbm2ddl.auto" value="update" />    ← za
pierwszym                                                                razem
create
        </properties>
    </persistence-unit>
</persistence>
```

Kod klasy JPAFactory:

```
public class JPAFactory {

    private static final EntityManagerFactory entityManagerFactory;

    private JPAFactory() {}

    static {
        try {
            entityManagerFactory =
Persistence.createEntityManagerFactory("myDatabaseConfig");
        } catch (Throwable ex) {
            System.err.println("Initial SessionFactory creation failed." + ex);
            throw new ExceptionInInitializerError(ex);
        }
    }

    public static EntityManagerFactory getEntityManagerFactory() {
        return entityManagerFactory;
    }
}
```

Kod klasy testującej działanie:

```
public class JPAAplication {
    public static void main(String[] args) {

        EntityManager entityManager =
JPAFactory.getEntityManagerFactory().createEntityManager();

        EntityTransaction entityTransaction1 = entityManager.getTransaction();
        entityTransaction1.begin();
        Product product1 = new Product("Elana ice skates 100",5);
        Product product2 = new Product("Elana ice skates 200",5);
        Product product3 = new Product("Elana ice skates 300",5);
        entityManager.persist(product1);
        entityManager.persist(product2);
        entityManager.persist(product3);
        Supplier supplier = new Supplier("Elana","Bergisel 123","Innsbruck");
        entityManager.persist(supplier);
        Category iceskating = new Category("Ice Skating");
        entityManager.persist(iceskating);
        supplier.addProductToProductSet(product1);
        supplier.addProductToProductSet(product2);
        supplier.addProductToProductSet(product3);
        iceskating.addProductToProductSet(product1);
        iceskating.addProductToProductSet(product2);
```

```
        iceskating.addProductToProductSet(product3);
        entityTransaction1.commit();
        entityManager.close();}
}
```

Logi Hibernate'a:

```
Hibernate:

    create table Categories (
       categoryID integer not null,
        name varchar(255),
        primary key (categoryID)
    )
Hibernate:

    create table Invoices (
       invoiceNumber integer not null,
        quantity integer not null,
        primary key (invoiceNumber)
    )
Hibernate:

    create table Products (
       productName varchar(255) not null,
        unitsOnStock integer not null,
        category_categoryID integer,
        supplier_companyName varchar(255),
        productSet_companyName varchar(255),
        productSet_categoryID integer,
        primary key (productName)
    )
Hibernate:

    create table Products_Invoices (
       productSet_productName varchar(255) not null,
        invoiceSet_invoiceNumber integer not null,
        primary key (productSet_productName, invoiceSet_invoiceNumber)
    )
Hibernate:

    create table Suppliers (
       companyName varchar(255) not null,
        city varchar(255),
        street varchar(255),
        primary key (companyName)
    )
Hibernate:

    alter table Products
       add constraint FKis8r738rffb59r366t82oth30
       foreign key (category_categoryID)
       references Categories
Hibernate:

    alter table Products
       add constraint FKlj63cbso72i9to92f2ldetnfd
       foreign key (supplier_companyName)
       references Suppliers
Hibernate:

    alter table Products
       add constraint FKnxe0kgaule62q4byqhsjleiln
       foreign key (productSet_companyName)
       references Suppliers
Hibernate:

    alter table Products
       add constraint FKek3q6b4xwuay4haavt1yxdv9a
       foreign key (productSet_categoryID)
       references Categories
Hibernate:

    alter table Products_Invoices
       add constraint FKbhw76x2whshtdc48d86744g6g
       foreign key (invoiceSet_invoiceNumber)
```

```
       references Invoices
Hibernate:

    alter table Products_Invoices
       add constraint FKbx21yx25oedvlnmjormx4p4ea
       foreign key (productSet_productName)
       references Products
lis 22, 2019 2:42:25 PM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator
initiateService
INFO: HHH000490: Using JtaPlatform implementation:
[org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
Hibernate:

values
    next value for hibernate_sequence
Hibernate:
    insert
    into
        Products
        (category_categoryID, supplier_companyName, unitsOnStock, productName)
    values
        (?, ?, ?, ?)
Hibernate:
    insert
    into
        Products
        (category_categoryID, supplier_companyName, unitsOnStock, productName)
    values
        (?, ?, ?, ?)
Hibernate:
    insert
    into
        Products
        (category_categoryID, supplier_companyName, unitsOnStock, productName)
    values
        (?, ?, ?, ?)
Hibernate:
    insert
    into
        Suppliers
        (city, street, companyName)
    values
        (?, ?, ?)
Hibernate:
    insert
    into
        Categories
        (name, categoryID)
    values
        (?, ?)
Hibernate:
    update
        Products
    set
        category_categoryID=?,
        supplier_companyName=?,
        unitsOnStock=?
    where
        productName=?
Hibernate:
    update
        Products
    set
        category_categoryID=?,
        supplier_companyName=?,
        unitsOnStock=?
    where
        productName=?
Hibernate:
    update
        Products
    set
        category_categoryID=?,
        supplier_companyName=?,
        unitsOnStock=?
    where
        productName=?
Hibernate:
```

```
    update
        Products
    set
        productSet_companyName=?
    where
        productName=?
Hibernate:
    update
        Products
    set
        productSet_companyName=?
    where
        productName=?
Hibernate:
    update
        Products
    set
        productSet_companyName=?
    where
        productName=?
Hibernate:
    update
        Products
    set
        productSet_categoryID=?
    where
        productName=?
Hibernate:
    update
        Products
    set
        productSet_categoryID=?
    where
        productName=?
Hibernate:
    update
        Products
    set
        productSet_categoryID=?
    where
        productName=?
```

Wywołanie selecta:

	PRODUCTNAME	UNITSONSTOCK	CATEGORY_CATEGORYID	SUPPLIER_COMPANYNAME
1	Elana ice skates…	5	1	Elana
2	Elana ice skates…	5	1	Elana
3	Elana ice skates…	5	1	Elana

IX. Wykorzystanie mechanizmu kaskad

Kod klasy Produkt:

```java
@Entity(name="Products")
public class Product {
    @Id
    private String productName;
    private int unitsOnStock;

    @ManyToOne
    @JoinColumn
    private Supplier supplier;

    @ManyToOne
    @JoinColumn
```

```
private Category category;

@ManyToMany(cascade = {CascadeType.PERSIST, CascadeType.REMOVE})
private Set<Invoice> invoiceSet = new HashSet<>();

public Product() {}

public Product(String productName, int unitsOnStock) {
    this.productName = productName;
    this.unitsOnStock = unitsOnStock;
    this.supplier = null;
}


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

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

public void setInvoiceSet(Set<Invoice> invoiceSet) {
    this.invoiceSet = invoiceSet;
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    Product product = (Product) o;
    return unitsOnStock == product.unitsOnStock &&
            productName.equals(product.productName);
}

@Override
public int hashCode() {
    return Objects.hash(productName, unitsOnStock);
}

public String getProductName() {
    return productName;
}

public int getUnitsOnStock() {
    return unitsOnStock;
}

public Supplier getSupplier() {
    return supplier;
}

public Category getCategory() {
    return category;
}

public Set<Invoice> getInvoiceSet() {
    return invoiceSet;
}

public void setUnitsOnStock(int unitsOnStock) {
    this.unitsOnStock = unitsOnStock;
```

```
    }
}
```

Kod klasy Invoice:

```
@Entity(name="Invoices")
public class Invoice {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int invoiceNumber;
    private int quantity;

    @ManyToMany(mappedBy = "invoiceSet", cascade = {CascadeType.PERSIST,
CascadeType.REMOVE})
    private Set<Product> productSet;

    public Invoice() {
        this.quantity = 0;
        productSet = new HashSet<>();
    }

    public void addProduct(Product product) {
        this.productSet.add(product);
        product.getInvoiceSet().add(this);
        this.quantity = 1;
        product.setUnitsOnStock(product.getUnitsOnStock()-1);
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Invoice invoice = (Invoice) o;
        return invoiceNumber == invoice.invoiceNumber;
    }

    @Override
    public int hashCode() {
        return Objects.hash(invoiceNumber);
    }
}
```

Przykład wykorzystania mechanizmu dodawania kaskodowego:
(Nie zapisuje explicite faktury, ale tylko powiązane z nią produkty)

```
EntityManager entityManager =
JPAFactory.getEntityManagerFactory().createEntityManager();
EntityTransaction entityTransaction1 = entityManager.getTransaction();
entityTransaction1.begin();
Product product1 = new Product("Elana ice skates 400",5);
Product product2 = new Product("Elana ice skates 500",5);
Invoice invoice = new Invoice();
invoice.addProduct(product1);
invoice.addProduct(product2);
entityManager.persist(invoice);
entityTransaction1.commit();
entityManager.close();
```

Logi Hibernate’a (wersja skrócona):

```
Hibernate:
```

```
values
    next value for hibernate_sequence
Hibernate:
    insert
    into
        Invoices
        (quantity, invoiceNumber)
    values
        (?, ?)
Hibernate:
    insert
    into
        Products
        (category_categoryID, supplier_companyName, unitsOnStock, productName)
    values
        (?, ?, ?, ?)
Hibernate:
    insert
    into
        Products
        (category_categoryID, supplier_companyName, unitsOnStock, productName)
    values
        (?, ?, ?, ?)
Hibernate:
    insert
    into
        Products_Invoices
        (productSet_productName, invoiceSet_invoiceNumber)
    values
        (?, ?)
Hibernate:
    insert
    into
        Products_Invoices
        (productSet_productName, invoiceSet_invoiceNumber)
    values
        (?, ?)
```

Faktycznie dodana jest nowa faktura do tabeli Invoices

Przykład select'ów:

	PRODUCTNAME	UNITSONSTOCK
1	Elana ice skates 500	4
2	Elana ice skates 400	4

	PRODUCTSET_PRODUCTNAME	INVOICESET_INVOICENUMBER
1	Elana ice skates 400	1
2	Elana ice skates 500	1

X. Embedded class.

Stworzyłem klasę adres, nad którą dodałem adnotację `@Embeddable.`

Kod klasy Address:

```java
@Embeddable
public class Address {
    private String street;
    private String buildingNumber;
    private String city;
    private String postalCode;
```

```
    public Address(){};

    public Address(String street, String buildingNumber, String city, String
postalCode) {
        this.street = street;
        this.buildingNumber = buildingNumber;
        this.city = city;
        this.postalCode = postalCode;
    }
}
```

Kod klasy Supplier:

```
@Entity(name="Suppliers")
public class Supplier {
    @Id
    private String companyName;
    @Embedded
    private Address address;
    @OneToMany
    @JoinColumn
    private Set<Product> productSet ;

    public Supplier() { }

    public Supplier(String companyName, String street, String buildingNumber,
String city, String postalCode) {
        this.companyName = companyName;
        this.address = new Address(street, buildingNumber, city, postalCode);
        productSet = new HashSet<>();
    }

    public void addProductToProductSet(Product product)
    {
        productSet.add(product);
        product.setSupplier(this);
    }


    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Supplier supplier = (Supplier) o;
        return companyName.equals(supplier.companyName);
    }

    @Override
    public int hashCode() {
        return Objects.hash(companyName) + 1;
    }
}
```

Logi Hiberate’a:

```
Hibernate:

    create table Categories (
       categoryID integer not null,
        name varchar(255),
        primary key (categoryID)
```

```
    )
Hibernate:

    create table Invoices (
       invoiceNumber integer not null,
        quantity integer not null,
        primary key (invoiceNumber)
    )
Hibernate:

    create table Products (
       productName varchar(255) not null,
        unitsOnStock integer not null,
        category_categoryID integer,
        supplier_companyName varchar(255),
        productSet_companyName varchar(255),
        productSet_categoryID integer,
        primary key (productName)
    )
Hibernate:

    create table Products_Invoices (
       productSet_productName varchar(255) not null,
        invoiceSet_invoiceNumber integer not null,
        primary key (productSet_productName, invoiceSet_invoiceNumber)
    )
Hibernate:

    create table Suppliers (
       companyName varchar(255) not null,
        buildingNumber varchar(255),
        city varchar(255),
        postalCode varchar(255),
        street varchar(255),
        primary key (companyName)
    )
Hibernate:

    alter table Products
       add constraint FKis8r738rffb59r366t82oth30
       foreign key (category_categoryID)
       references Categories
Hibernate:

    alter table Products
       add constraint FKlj63cbso72i9to92f2ldetnfd
       foreign key (supplier_companyName)
       references Suppliers
Hibernate:

    alter table Products
       add constraint FKnxe0kgaule62q4byqhsjleiln
       foreign key (productSet_companyName)
       references Suppliers
Hibernate:

    alter table Products
       add constraint FKek3q6b4xwuay4haavt1yxdv9a
       foreign key (productSet_categoryID)
       references Categories
Hibernate:

    alter table Products_Invoices
       add constraint FKbhw76x2whshtdc48d86744g6g
       foreign key (invoiceSet_invoiceNumber)
       references Invoices
Hibernate:

    alter table Products_Invoices
       add constraint FKbx21yx25oedvlnmjormx4p4ea
       foreign key (productSet_productName)
       references Products
lis 22, 2019 3:13:54 PM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator
initiateService
INFO: HHH000490: Using JtaPlatform implementation:
[org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
Hibernate:
    insert
```

```
    into
        Suppliers
        (buildingNumber, city, postalCode, street, companyName)
    values
        (?, ?, ?, ?, ?)
Hibernate:
    insert
    into
        Suppliers
        (buildingNumber, city, postalCode, street, companyName)
    values
        (?, ?, ?, ?, ?)
```

Przykład selecta:

	COMPANYNAME	BUILDINGNUMBER	CITY	POSTALCODE	STREET
1	Puma	12	Rzeszów	36-084	Sportowa
2	Asisc	13	Rzeszów	36-084	Sportowa

Wygląd tabeli Suppliers na diagramie bazy danych:

XI. Dziedziczenie – customers oraz suppliers dziedziczą z company.

a) jedna tabela na całą hierarchię

Kod klasy Companies:

```
@Entity(name="Companies")
@Inheritance(strategy= InheritanceType.SINGLE_TABLE)
public class Company {
    @Id
    private String companyName;
    @Embedded
    private Address address;
    public Company(String companyName, String street, String buildingNumber,
String city, String postalCode) {
        this.companyName = companyName;
        this.address = new Address(street, buildingNumber, city, postalCode);
    }
    public Company() {}
}
```

Kod klasy Customers:

```
@Entity(name="Customers")
public class Customer extends Company {
```

```
    private double discount;



    public Customer() { };

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

Kod klasy Suppliers:

```
@Entity(name="Suppliers")
public class Supplier extends  Company {

    private String bankAccount;
    @OneToMany
    @JoinColumn
    private Set<Product> productSet ;

    public Supplier(String companyName, String street, String buildingNumber,
String city, String postalCode,  String bankAccount) {
        super(companyName, street, buildingNumber, city, postalCode);
        this.bankAccount = bankAccount;
    }



    public Supplier() { }



    public void addProductToProductSet(Product product)
    {
        productSet.add(product);
        product.setSupplier(this);
    }

}
```

Kod klasy testującej działanie:

```
EntityManager entityManager =
JPAFactory.getEntityManagerFactory().createEntityManager();
EntityTransaction entityTransaction1 = entityManager.getTransaction();
entityTransaction1.begin();
Supplier supplier1 = new Supplier("Puma", "Sportowa", "12", "Rzeszów", "36-084",
"123456789");
Supplier supplier2 = new Supplier("Asisc", "Sportowa", "13", "Rzeszów", "36-
084","123456788");
Customer customer1 = new Customer("Runner Shop", "Sportowa", "14", "Rzeszów",
"36-084",0.5);
Customer customer2 = new Customer("Sport Shop", "Sportowa", "14", "Rzeszów",
"36-084",0.4);
entityManager.persist(supplier1);
entityManager.persist(supplier2);
entityManager.persist(customer1);
```

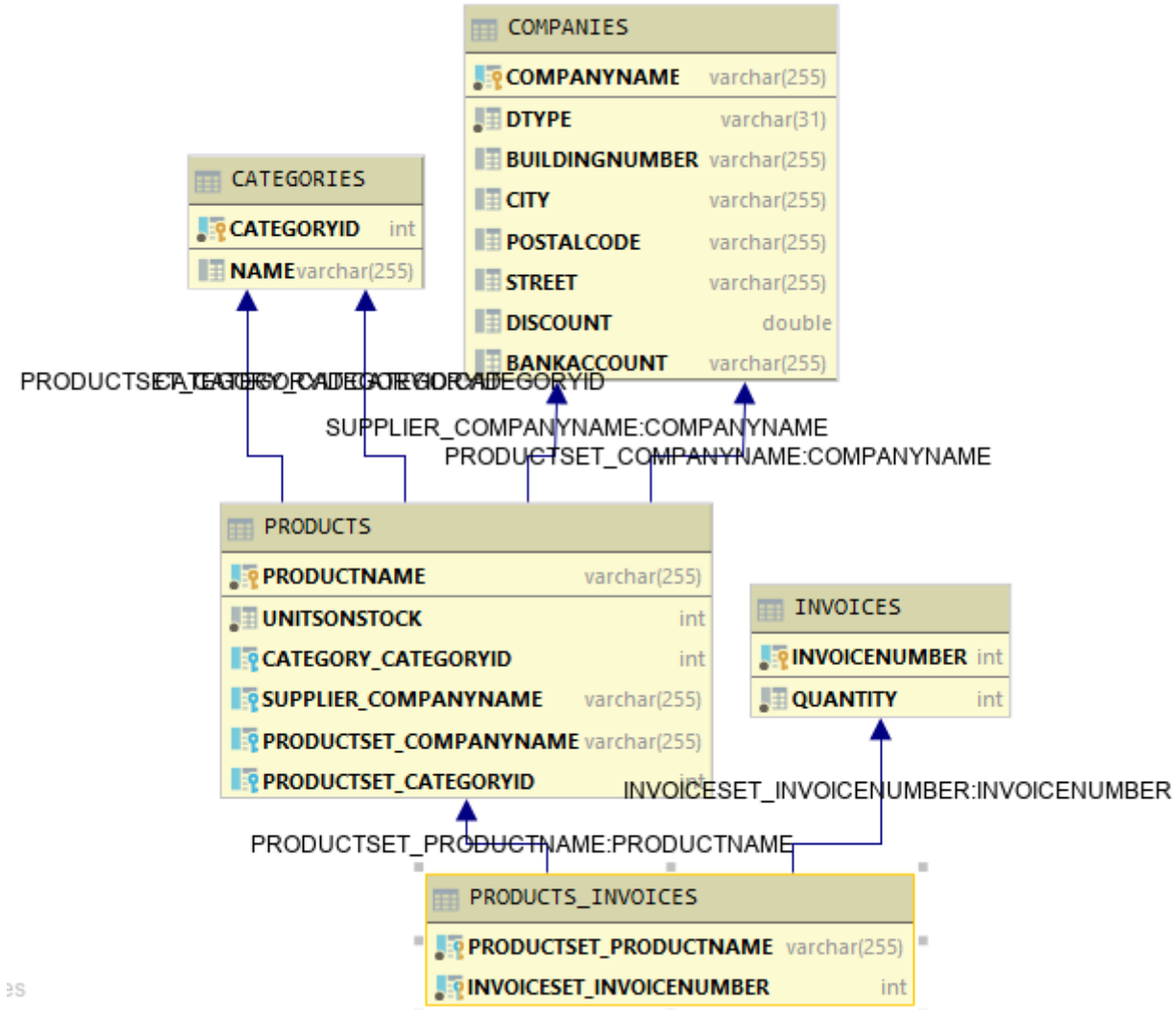
```
entityManager.persist(customer2);
entityTransaction1.commit();
entityManager.close();
```

Logi Hibernate'a:

```
Hibernate:

    create table Categories (
       categoryID integer not null,
        name varchar(255),
        primary key (categoryID)
    )
Hibernate:

    create table Companies (
       DTYPE varchar(31) not null,
        companyName varchar(255) not null,
        buildingNumber varchar(255),
        city varchar(255),
        postalCode varchar(255),
        street varchar(255),
        discount double,
        bankAccount varchar(255),
        primary key (companyName)
    )
Hibernate:

    create table Invoices (
       invoiceNumber integer not null,
        quantity integer not null,
        primary key (invoiceNumber)
    )
Hibernate:

    create table Products (
       productName varchar(255) not null,
        unitsOnStock integer not null,
        category_categoryID integer,
        supplier_companyName varchar(255),
        productSet_companyName varchar(255),
        productSet_categoryID integer,
        primary key (productName)
    )
Hibernate:

    create table Products_Invoices (
       productSet_productName varchar(255) not null,
        invoiceSet_invoiceNumber integer not null,
        primary key (productSet_productName, invoiceSet_invoiceNumber)
    )
Hibernate:

    alter table Products
       add constraint FKis8r738rffb59r366t82oth30
       foreign key (category_categoryID)
       references Categories
Hibernate:

    alter table Products
       add constraint FKmb8ukwymde2r3nj9uoupx4kmn
       foreign key (supplier_companyName)
       references Companies
Hibernate:

    alter table Products
       add constraint FK6qk6a4nlra5u9boiw3gnuq4va
       foreign key (productSet_companyName)
       references Companies
Hibernate:

    alter table Products
       add constraint FKek3q6b4xwuay4haavt1yxdv9a
       foreign key (productSet_categoryID)
       references Categories
Hibernate:
```

```
    alter table Products_Invoices
       add constraint FKbhw76x2whshtdc48d86744g6g
       foreign key (invoiceSet_invoiceNumber)
       references Invoices
Hibernate:

    alter table Products_Invoices
       add constraint FKbx21yx25oedvlnmjormx4p4ea
       foreign key (productSet_productName)
       references Products
lis 22, 2019 4:12:36 PM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator
initiateService
INFO: HHH000490: Using JtaPlatform implementation:
[org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
Hibernate:
    insert
    into
        Companies
        (buildingNumber, city, postalCode, street, bankAccount, DTYPE, companyName)
    values
        (?, ?, ?, ?, ?, 'Suppliers', ?)
Hibernate:
    insert
    into
        Companies
        (buildingNumber, city, postalCode, street, bankAccount, DTYPE, companyName)
    values
        (?, ?, ?, ?, ?, 'Suppliers', ?)
Hibernate:
    insert
    into
        Companies
        (buildingNumber, city, postalCode, street, discount, DTYPE, companyName)
    values
        (?, ?, ?, ?, ?, 'Customers', ?)
Hibernate:
    insert
    into
        Companies
        (buildingNumber, city, postalCode, street, discount, DTYPE, companyName)
    values
        (?, ?, ?, ?, ?, 'Customers', ?)
```

Diagram bazy danych:



Przykład selecta z tabeli Companies:

	DTYPE	COMPANYNAME	BUILDINGNUMBER	CITY	POSTALCODE	STREET	DISCOUNT	BANKACCOUNT
1	Suppliers	Puma	12	Rzeszów	36-084	Sportowa	<null>	123456789
2	Suppliers	Asisc	13	Rzeszów	36-084	Sportowa	<null>	123456788
3	Customers	Runner Shop	14	Rzeszów	36-084	Sportowa	0.5	<null>
4	Customers	Sport Shop	14	Rzeszów	36-084	Sportowa	0.4	<null>

b) tabela łączona

Kod klasy Companies (pozostałe klasy bez zmian):

```
@Entity(name="Companies")
@Inheritance(strategy= InheritanceType.JOINED)
public class Company {
    @Id
    private String companyName;

    @Embedded
    private Address address;


    public Company(String companyName, String street, String buildingNumber,
String city, String postalCode) {
        this.companyName = companyName;
```

```
        this.address = new Address(street, buildingNumber, city, postalCode);

    }


    public Company() {

    }
}
```

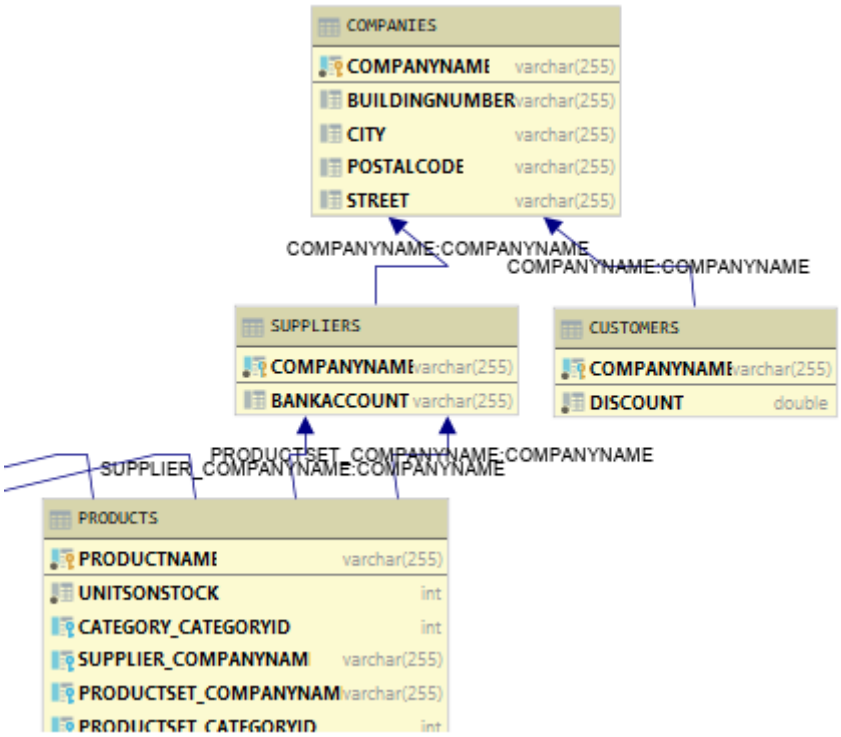
Logi Hibernate'a:

```
Hibernate:

    create table Categories (
       categoryID integer not null,
        name varchar(255),
        primary key (categoryID)
    )
Hibernate:

    create table Companies (
       companyName varchar(255) not null,
        buildingNumber varchar(255),
        city varchar(255),
        postalCode varchar(255),
        street varchar(255),
        primary key (companyName)
    )
Hibernate:

    create table Customers (
       discount double not null,
        companyName varchar(255) not null,
        primary key (companyName)
    )
Hibernate:

    create table Invoices (
       invoiceNumber integer not null,
        quantity integer not null,
        primary key (invoiceNumber)
    )
Hibernate:

    create table Products (
       productName varchar(255) not null,
        unitsOnStock integer not null,
        category_categoryID integer,
        supplier_companyName varchar(255),
        productSet_companyName varchar(255),
        productSet_categoryID integer,
        primary key (productName)
    )
Hibernate:

    create table Products_Invoices (
       productSet_productName varchar(255) not null,
        invoiceSet_invoiceNumber integer not null,
        primary key (productSet_productName, invoiceSet_invoiceNumber)
    )
Hibernate:

    create table Suppliers (
       bankAccount varchar(255),
        companyName varchar(255) not null,
        primary key (companyName)
    )
Hibernate:

    alter table Customers
       add constraint FK2i5poknhb3cqcap542vwb075k
       foreign key (companyName)
       references Companies
Hibernate:
```

```
    alter table Products
       add constraint FKis8r738rffb59r366t82oth30
       foreign key (category_categoryID)
       references Categories
Hibernate:

    alter table Products
       add constraint FKlj63cbso72i9to92f2ldetnfd
       foreign key (supplier_companyName)
       references Suppliers
Hibernate:

    alter table Products
       add constraint FKnxe0kgaule62q4byqhsjleiln
       foreign key (productSet_companyName)
       references Suppliers
Hibernate:

    alter table Products
       add constraint FKek3q6b4xwuay4haavt1yxdv9a
       foreign key (productSet_categoryID)
       references Categories
Hibernate:

    alter table Products_Invoices
       add constraint FKbhw76x2whshtdc48d86744g6g
       foreign key (invoiceSet_invoiceNumber)
       references Invoices
Hibernate:

    alter table Products_Invoices
       add constraint FKbx21yx25oedvlnmjormx4p4ea
       foreign key (productSet_productName)
       references Products
Hibernate:

    alter table Suppliers
       add constraint FKqifubftlo8k5xevp2a3u49w05
       foreign key (companyName)
       references Companies
lis 22, 2019 4:18:45 PM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator
initiateService
INFO: HHH000490: Using JtaPlatform implementation:
[org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
Hibernate:
    insert
    into
        Companies
        (buildingNumber, city, postalCode, street, companyName)
    values
        (?, ?, ?, ?, ?)
Hibernate:
    insert
    into
        Suppliers
        (bankAccount, companyName)
    values
        (?, ?)
Hibernate:
    insert
    into
        Companies
        (buildingNumber, city, postalCode, street, companyName)
    values
        (?, ?, ?, ?, ?)
Hibernate:
    insert
    into
        Suppliers
        (bankAccount, companyName)
    values
        (?, ?)
Hibernate:
    insert
    into
        Companies
        (buildingNumber, city, postalCode, street, companyName)
    values
```

```
        (?, ?, ?, ?, ?)
Hibernate:
    insert
    into
        Customers
        (discount, companyName)
    values
        (?, ?)
Hibernate:
    insert
    into
        Companies
        (buildingNumber, city, postalCode, street, companyName)
    values
        (?, ?, ?, ?, ?)
Hibernate:
    insert
    into
        Customers
        (discount, companyName)
    values
        (?, ?)
```

Diagram bazy danych:



Przykłady selecta z tabeli Companies oraz z tabeli Suppliers:

	COMPANYNAME	BUILDINGNUMBER	CITY	POSTALCODE	STREET
1	Puma	12	Rzeszów	36-084	Sportowa
2	Asisc	13	Rzeszów	36-084	Sportowa
3	Runner Shop	14	Rzeszów	36-084	Sportowa
4	Sport Shop	14	Rzeszów	36-084	Sportowa

	BANKACCOUNT	COMPANYNAME
1	123456789	Puma
2	123456788	Asisc

c) jedna tabela na klasę:

Kod klasy Companies (pozostałe klasy bez zmian):

```
@Entity(name="Companies")
@Inheritance(strategy= InheritanceType.TABLE_PER_CLASS)
public class Company {
    @Id
    private String companyName;

    @Embedded
    private Address address;


    public Company(String companyName, String street, String buildingNumber,
String city, String postalCode) {
        this.companyName = companyName;
        this.address = new Address(street, buildingNumber, city, postalCode);

    }


    public Company() {

    }
}
```

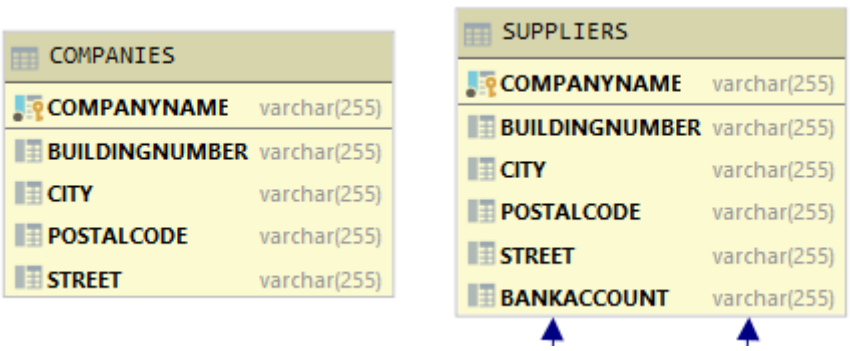
Logi Hibernate'a:

```
Hibernate:

    create table Customers (
       companyName varchar(255) not null,
        buildingNumber varchar(255),
        city varchar(255),
        postalCode varchar(255),
        street varchar(255),
        discount double not null,
        primary key (companyName)
    )
Hibernate:

    create table Invoices (
```

```
       invoiceNumber integer not null,
        quantity integer not null,
        primary key (invoiceNumber)
    )
Hibernate:

    create table Products (
       productName varchar(255) not null,
        unitsOnStock integer not null,
        category_categoryID integer,
        supplier_companyName varchar(255),
        productSet_companyName varchar(255),
        productSet_categoryID integer,
        primary key (productName)
    )
Hibernate:

    create table Products_Invoices (
       productSet_productName varchar(255) not null,
        invoiceSet_invoiceNumber integer not null,
        primary key (productSet_productName, invoiceSet_invoiceNumber)
    )
Hibernate:

    create table Suppliers (
       companyName varchar(255) not null,
        buildingNumber varchar(255),
        city varchar(255),
        postalCode varchar(255),
        street varchar(255),
        bankAccount varchar(255),
        primary key (companyName)
    )
Hibernate:

    alter table Products
       add constraint FKis8r738rffb59r366t82oth30
       foreign key (category_categoryID)
       references Categories
Hibernate:

    alter table Products
       add constraint FKlj63cbso72i9to92f2ldetnfd
       foreign key (supplier_companyName)
       references Suppliers
Hibernate:

    alter table Products
       add constraint FKnxe0kgaule62q4byqhsjleiln
       foreign key (productSet_companyName)
       references Suppliers
Hibernate:

    alter table Products
       add constraint FKek3q6b4xwuay4haavt1yxdv9a
       foreign key (productSet_categoryID)
       references Categories
Hibernate:

    alter table Products_Invoices
       add constraint FKbhw76x2whshtdc48d86744g6g
       foreign key (invoiceSet_invoiceNumber)
```

```
       references Invoices
Hibernate:

    alter table Products_Invoices
       add constraint FKbx21yx25oedvlnmjormx4p4ea
       foreign key (productSet_productName)
       references Products
lis 22, 2019 4:29:03 PM
org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator
initiateService
INFO: HHH000490: Using JtaPlatform implementation:
[org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
Hibernate:
    insert
    into
        Suppliers
        (buildingNumber, city, postalCode, street, bankAccount, companyName)
    values
        (?, ?, ?, ?, ?, ?)
Hibernate:
    insert
    into
        Suppliers
        (buildingNumber, city, postalCode, street, bankAccount, companyName)
    values
        (?, ?, ?, ?, ?, ?)
Hibernate:
    insert
    into
        Customers
        (buildingNumber, city, postalCode, street, discount, companyName)
    values
        (?, ?, ?, ?, ?, ?)
Hibernate:
    insert
    into
        Customers
        (buildingNumber, city, postalCode, street, discount, companyName)
    values
        (?, ?, ?, ?, ?, ?)
```

Diagram bazy danych (widzimy, że companies jest nie powiązane na diagramie z suppliers):



Przykłady selectów z tabeli Customers, Suppliers oraz Companies:

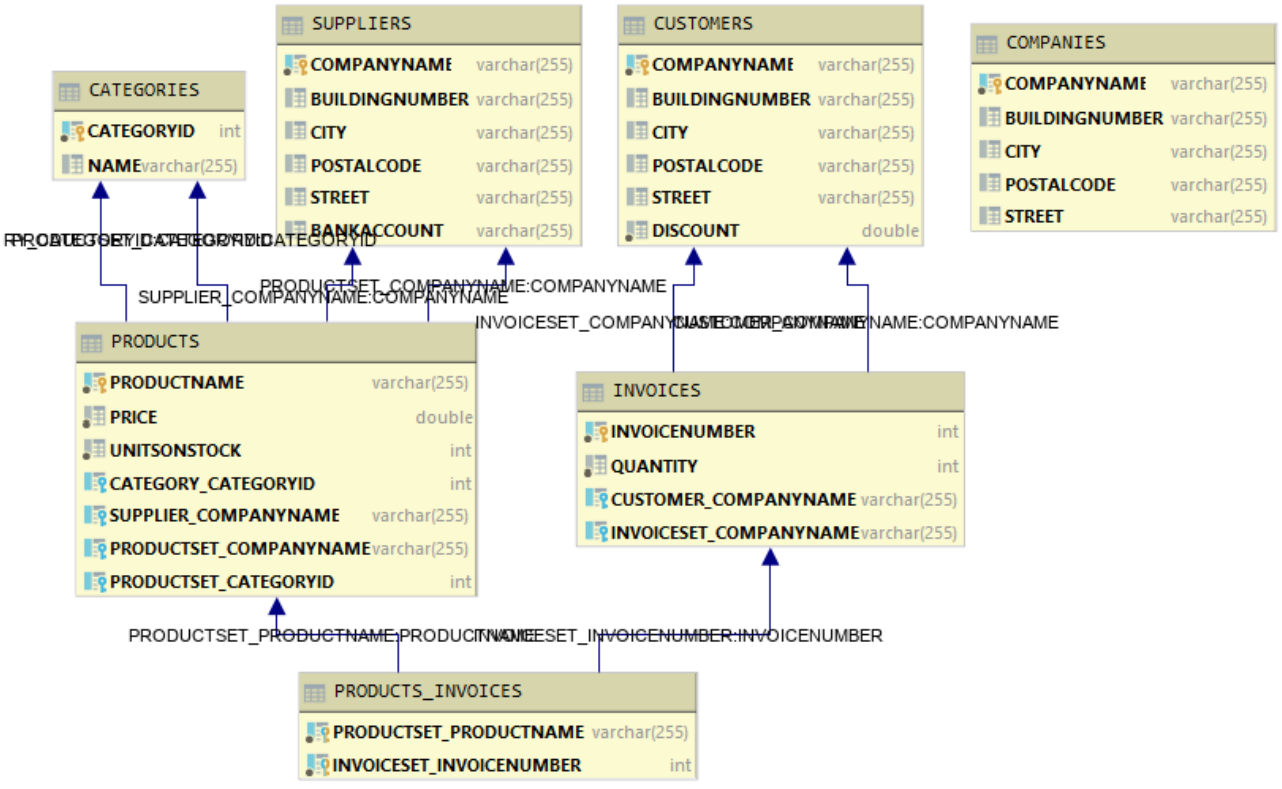
	COMPANYNAME	BUILDINGNUMBER	CITY	POSTALCODE	STREET	DISCOUNT
1	Runner Shop	14	Rzeszów	36-084	Sportowa	0.5
2	Sport Shop	14	Rzeszów	36-084	Sportowa	0.4

	COMPANYNAME	BUILDINGNUMBER	CITY	POSTALCODE	STREET	BANKACCOUNT
1	Puma	12	Rzeszów	36-084	Sportowa	123456789
2	Asisc	13	Rzeszów	36-084	Sportowa	123456788

	COMPANYNAME	BUILDINGNUMBER	CITY	POSTALCODE	STREET
1	Puma	12	Rzeszów	36-084	Sportowa
2	Asisc	13	Rzeszów	36-084	Sportowa
3	Runner Shop	14	Rzeszów	36-084	Sportowa
4	Sport Shop	14	Rzeszów	36-084	Sportowa

XII. Własna aplikacja

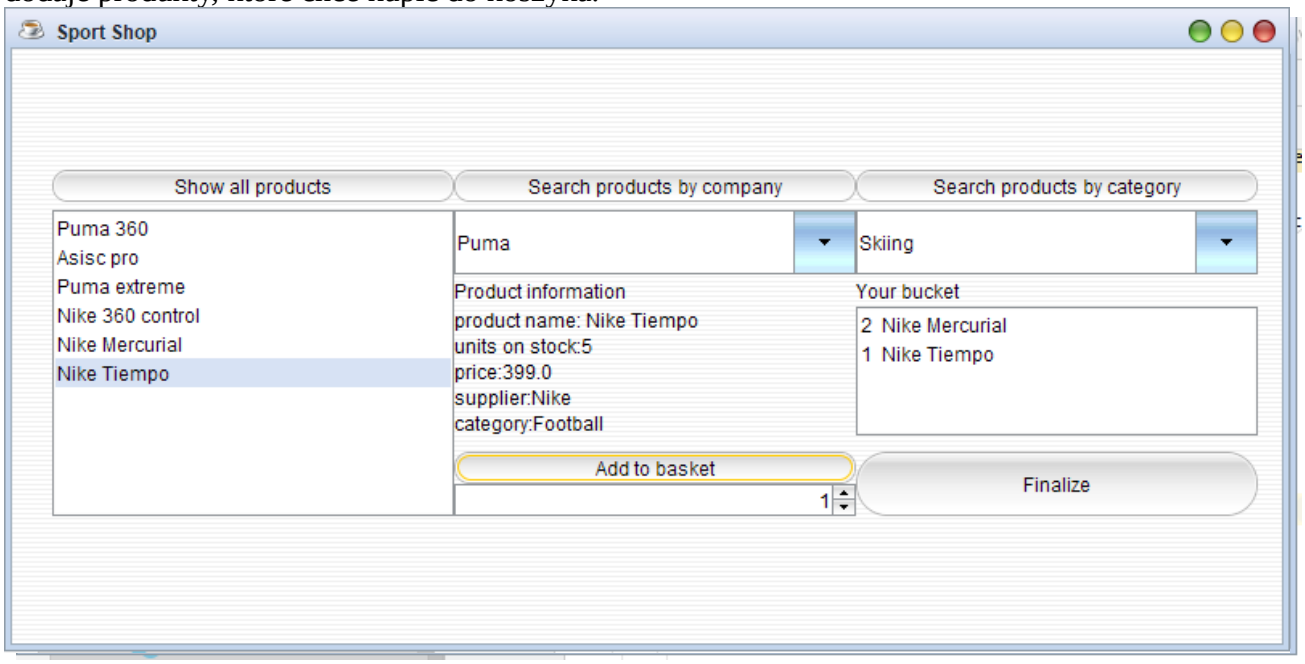
a) schemat bazy danych:



Zdecydowałem się dodać relację pomiędzy wiele-do-jeden invoices – customers oraz jeden-do-wiele customers – invoices. Aby takie podwójne relacje się nie rozjeżdżały dbam o odpowiednie ustawienia referencji na obiekty w klasach powiązanych relacją.

b) aplikacja

Okno startowe – użytkownik może wybrać i wyświetlić informację o interesujących go produktach, ponadto istnieje możliwość szukania produktów według dostawców oraz kategorii. Użytkownik dodaje produkty, które chce kupić do koszyka.



Kod klasy StartWindow:

```
public class StartWindow extends JFrame implements ActionListener {
    private List<Order> productsInBasket = new ArrayList<>();


    private JButton bShowAllProducts;
    private JButton bFilterCompany;
    private JButton bFilterCategory;
    private JButton bAddToBasket;
    private JButton bFinalize;

    private JSpinner spinnerAmount;

    private List<Product> productList; //= new ArrayList<>();
    private List<Category> categoryList;// = new ArrayList<>();
    private List<Supplier> supplierList;// = new ArrayList<>();
    private JComboBox<Category> categoryJComboBox;
    private JComboBox<Supplier> supplierJComboBox;
    private JList<Product> productJList;
    private JList<Order> orderJList;
    JScrollPane scrollpane;
    JScrollPane scrollPaneBucket;

    private JLabel jLabelProductDescription;
    private JLabel jLabelProductDescriptionInfo;
    private JLabel jLabelCurrentBucketState;



    public StartWindow()
    {
        //addExampleData2();
        downloadData();
        initializeComponents();


    }

    private void downloadData()
    {
        EntityManager entityManager =
JPAFactory.getEntityManagerFactory().createEntityManager();
        EntityTransaction entityTransaction1 = entityManager.getTransaction();
        entityTransaction1.begin();
        categoryList = entityManager.createQuery("select c from Categories
c").getResultList();
        supplierList = entityManager.createQuery("select s from Suppliers
s").getResultList();
        productList = entityManager.createQuery("select p from Products
p").getResultList();
        entityTransaction1.commit();
        entityManager.close();
    }

    private void initializeComponents(){
        setSize (800,400);
        setTitle("Sport Shop");
        setLayout(null);
```

```
bShowAllProducts = new JButton("Show all products");
bShowAllProducts.setBounds(25, 75, 250, 20);
add(bShowAllProducts);
bShowAllProducts.addActionListener(this);

bFilterCompany = new JButton("Search products by company");
bFilterCompany.setBounds(275, 75, 250, 20);
add(bFilterCompany);
bFilterCompany.addActionListener(this);

bFilterCategory = new JButton("Search products by category");
bFilterCategory.setBounds(525, 75, 250, 20);
add(bFilterCategory);
bFilterCategory.addActionListener(this);



categoryJComboBox = new JComboBox<Category>();
for(Category category : categoryList) {
    categoryJComboBox.addItem(category);
}
categoryJComboBox.setBounds(525, 100, 250, 40);
this.add(categoryJComboBox);

supplierJComboBox = new JComboBox<Supplier>();
for(Supplier supplier : supplierList) {
    supplierJComboBox.addItem(supplier);
}

supplierJComboBox.setBounds(275, 100, 250, 40);
this.add(supplierJComboBox);

DefaultListModel<Product> listModel = new DefaultListModel<>();
for(Product product : productList)
{
    listModel.addElement(product);
}

productJList = new JList<Product>(listModel);
addProductListSelectionListener();
scrollpane = new JScrollPane(productJList);
scrollpane.setBounds(25, 100, 250, 190);
this.add(scrollpane);

jLabelProductDescription = new JLabel();

jLabelProductDescription.setBounds(275, 160, 250, 80);
//jLabelProductDescription.setForeground(Color.lightGray);
this.add(jLabelProductDescription);

bAddToBasket = new JButton("Add to basket");
bAddToBasket.setBounds(275, 250, 250, 20);
add(bAddToBasket);
bAddToBasket.addActionListener(this);

spinnerAmount = new JSpinner();
spinnerAmount.setBounds(275, 270, 250,20);
add(spinnerAmount);
```

```
        orderJList = new JList<Order>();
        scrollPaneBucket = new JScrollPane(orderJList);
        scrollPaneBucket.setBounds(525, 160, 250, 80);
        this.add(scrollPaneBucket);


        bFinalize = new JButton("Finalize");
        bFinalize.setBounds(525, 250, 250, 40);
        add(bFinalize);
        bFinalize.addActionListener(this);

        jLabelProductDescriptionInfo = new JLabel("Product information");
        jLabelProductDescriptionInfo.setBounds(275, 140, 250, 20);
        add(jLabelProductDescriptionInfo);

        jLabelCurrentBucketState = new JLabel("Your bucket");
        jLabelCurrentBucketState.setBounds(525, 140, 250, 20);
        add(jLabelCurrentBucketState);


    }

    public static void main(String[] args) throws IOException,
ClassNotFoundException, UnsupportedLookAndFeelException, InstantiationException,
IllegalAccessException {


        UIManager.setLookAndFeel("com.jtattoo.plaf.mcwin.McWinLookAndFeel");

        StartWindow startWindow = new StartWindow();
        startWindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        startWindow.setVisible(true);



    }


    public void addExampleData2()
    {
        EntityManager entityManager =
JPAFactory.getEntityManagerFactory().createEntityManager();
        EntityTransaction entityTransaction1 = entityManager.getTransaction();
        entityTransaction1.begin();
        Supplier supplier1 = new Supplier("Nike", "Sportowa", "17", "Rzeszów",
"36-084", "123456777");
        Customer customer1 = new Customer("Jan Kowalski", "Sportowa", "20",
"Rzeszów", "36-084",0.5);
        Customer customer2 = new Customer("Piotr Nowak", "Sportowa", "18",
"Rzeszów", "36-084",0.4);

        Category category1 = new Category("Football");
        Product product1 = new Product("Nike 360 control",5, 199);
        Product product2 = new Product("Nike Mercurial",5, 299);
        Product product3 = new Product("Nike Tiempo",5, 399);

        supplier1.addProductToProductSet(product1);
        supplier1.addProductToProductSet(product3);
```

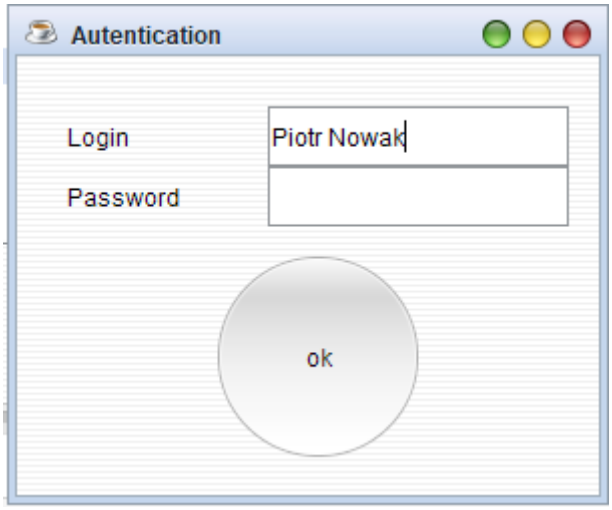
```
        supplier1.addProductToProductSet(product2);

        category1.addProductToProductSet(product1);
        category1.addProductToProductSet(product2);
        category1.addProductToProductSet(product3);

        entityManager.persist(supplier1);

        entityManager.persist(category1);


        entityManager.persist(customer1);
        entityManager.persist(customer2);

        entityManager.persist(product1);
        entityManager.persist(product2);
        entityManager.persist(product3);

        entityTransaction1.commit();
        entityManager.close();
    }



    public void addExampleData()
    {
        EntityManager entityManager =
JPAFactory.getEntityManagerFactory().createEntityManager();
        EntityTransaction entityTransaction1 = entityManager.getTransaction();
        entityTransaction1.begin();
        Supplier supplier1 = new Supplier("Puma", "Sportowa", "12", "Rzeszów",
"36-084", "123456789");
        Supplier supplier2 = new Supplier("Asisc", "Sportowa", "13", "Rzeszów",
"36-084","123456788");
        Customer customer1 = new Customer("Runner Shop", "Sportowa", "14",
"Rzeszów", "36-084",0.5);
        Customer customer2 = new Customer("Sport Shop", "Sportowa", "14",
"Rzeszów", "36-084",0.4);

        Category category1 = new Category("Skiing");
        Category category2 = new Category("Running");
        Category category3 = new Category("Football");

        Product product1 = new Product("Puma 360",5, 199);
        Product product2 = new Product("Asisc pro",5, 299);
        Product product3 = new Product("Puma extreme",5, 399);

        supplier1.addProductToProductSet(product1);
        supplier1.addProductToProductSet(product3);
        supplier2.addProductToProductSet(product2);

        category2.addProductToProductSet(product1);
        category2.addProductToProductSet(product2);
        category2.addProductToProductSet(product3);

        entityManager.persist(supplier1);

        entityManager.persist(category1);
        entityManager.persist(category2);
```

```
        entityManager.persist(category3);

        entityManager.persist(supplier2);
        entityManager.persist(customer1);
        entityManager.persist(customer2);

        entityManager.persist(product1);
        entityManager.persist(product2);
        entityManager.persist(product3);

        entityTransaction1.commit();
        entityManager.close();
    }


    @Override
    public void actionPerformed(ActionEvent e) {
        Object eventSource = e.getSource();
        if(eventSource == bFilterCategory)
        {
            if(categoryJComboBox.getSelectedItem() != null) // bez sensu
            {
                EntityManager entityManager =
JPAFactory.getEntityManagerFactory().createEntityManager();
                String categoryName
=categoryJComboBox.getSelectedItem().toString();
                List<Category> categories = entityManager.createQuery("select c
from Categories c" +
                        " where c.name =: nameX")
                        .setParameter("nameX",categoryName)
                        .getResultList();


                List<Product> results = entityManager.createQuery("select p from
Products p" +
                        " where p.category =: categoryX")
                        .setParameter("categoryX",categories.get(0))
                        .getResultList();
                DefaultListModel<Product> listModel = new DefaultListModel<>();
                for(Product product : results)
                {
                    listModel.addElement(product);
                }

                productJList = new JList<Product>(listModel);
                addProductListSelectionListener();
                scrollpane = new JScrollPane(productJList);
                scrollpane.setBounds(25, 100, 250, 190);
                this.add(scrollpane);
                scrollpane.updateUI();
                entityManager.close();
                return;
            }
        }

        if(eventSource == bFilterCompany)
        {
            if(supplierJComboBox.getSelectedItem() != null)
            {
                EntityManager entityManager =
```

```
JPAFactory.getEntityManagerFactory().createEntityManager();
                String companyName
=supplierJComboBox.getSelectedItem().toString();
                List<Supplier> suppliers = entityManager.createQuery("select s
from Suppliers s" +
                        " where s.companyName =: nameX")
                        .setParameter("nameX",companyName)
                        .getResultList();


                List<Product> results = entityManager.createQuery("select p from
Products p" +
                        " where p.supplier =: supplierX")
                        .setParameter("supplierX",suppliers.get(0))
                        .getResultList();
                DefaultListModel<Product> listModel = new DefaultListModel<>();
                for(Product product : results)
                {
                    listModel.addElement(product);
                }

                productJList = new JList<Product>(listModel);
                addProductListSelectionListener();
                scrollpane = new JScrollPane(productJList);
                scrollpane.setBounds(25, 100, 250, 190);
                this.add(scrollpane);
                scrollpane.updateUI();
                entityManager.close();
                return;
            }
        }

        if(eventSource == bShowAllProducts)
        {

                EntityManager entityManager =
JPAFactory.getEntityManagerFactory().createEntityManager();
                List<Product> results = entityManager.createQuery("select p from
Products p")
                        .getResultList();
                DefaultListModel<Product> listModel = new DefaultListModel<>();
                for(Product product : results)
                {
                    listModel.addElement(product);
                }

                productJList = new JList<Product>(listModel);
                scrollpane = new JScrollPane(productJList);
                scrollpane.setBounds(25, 100, 250, 100);
                this.add(scrollpane);
                scrollpane.updateUI();
                entityManager.close();

                return;

        }

        if(eventSource == bAddToBasket)
        {
```

```
            EntityManager entityManager =
JPAFactory.getEntityManagerFactory().createEntityManager();
            String productName = productJList.getSelectedValue().toString();
            List<Product> results = entityManager.createQuery("select p from
Products p" +
                    " where p.productName =: productX")
                    .setParameter("productX",productName)
                    .getResultList();
            Product product = results.get(0);
            int amount = (Integer) spinnerAmount.getValue();
            productsInBasket.add(new Order(product, amount));

            DefaultListModel<Order> listModel = new DefaultListModel<>();
            for(Order order : productsInBasket)
            {
                listModel.addElement(order);
            }
             orderJList= new JList<Order>(listModel);
            scrollPaneBucket = new JScrollPane(orderJList);
            scrollPaneBucket.setBounds(525, 160, 250, 80);
            this.add(scrollPaneBucket);
            scrollPaneBucket.updateUI();
            entityManager.close();
            return;

        }
        if(eventSource == bFinalize)

        {
            LoginWindow loginWindow = new LoginWindow(productsInBasket);
            loginWindow.setVisible(true);

        }



    }

    public void addProductListSelectionListener()
    {
        productJList.addListSelectionListener(new ListSelectionListener() {
            @Override
            public void valueChanged(ListSelectionEvent e) {
                EntityManager entityManager =
JPAFactory.getEntityManagerFactory().createEntityManager();
                String productName = productJList.getSelectedValue().toString();
                List<Product> results = entityManager.createQuery("select p from
Products p" +
                        " where p.productName =: productX")
                        .setParameter("productX",productName)
                        .getResultList();
                Product product = results.get(0);
                jLabelProductDescription.setText(product.displayDescription());
                entityManager.close();
            }
        });
    }

}
```

Okno logowania – po kliknięciu przycisku Finalize pojawia się okienko logowania. Na potrzeby zadania weryfikowana jest tylko nazwa użytkownika – CompanyName.

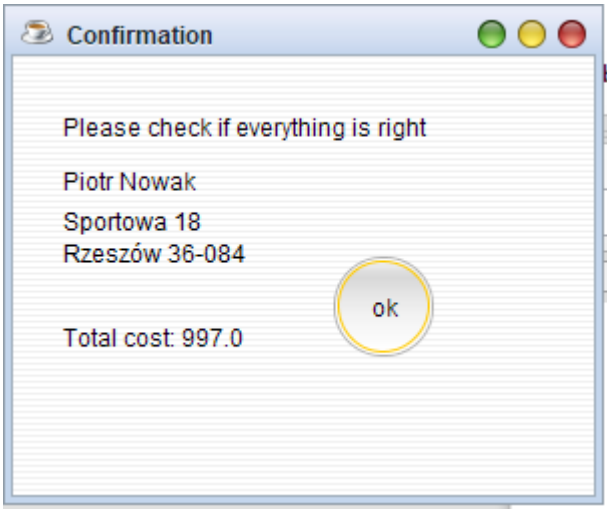


Kod klasy LoginWindow:

```
public class LoginWindow  extends JFrame implements ActionListener {

    JButton bConfirm;
    JLabel labelLogin;
    JLabel labelPassword;
    JTextField textFieldLogin;
    JTextField textFieldPassword;
    List<Order> bucket;


    public LoginWindow(List<Order> bucket)
    {
        this.bucket = bucket;
        setSize(300, 250);
        setTitle("Autentication");
        setLayout(null);

        labelLogin = new JLabel("Login");
        labelLogin.setBounds(25, 25, 100, 30);
        add(labelLogin);

        labelPassword = new JLabel("Password");
        labelPassword.setBounds(25, 55, 100, 30);
        add(labelPassword);

        textFieldLogin = new JTextField();
        textFieldLogin.setBounds(125, 25, 150, 30);
        add(textFieldLogin);

        textFieldPassword = new JTextField();
        textFieldPassword.setBounds(125, 55, 150, 30);
        add(textFieldPassword);
```

```
        bConfirm = new JButton("ok");
        bConfirm.setBounds(100,100,100,100);
        bConfirm.addActionListener(this);
        add(bConfirm);


    }

    @Override
    public void actionPerformed(ActionEvent e) {
        Object eventSource = e.getSource();
        if(eventSource == bConfirm)
        {
            EntityManager entityManager =
JPAFactory.getEntityManagerFactory().createEntityManager();
            String companyName  = textFieldLogin.getText();
            List<Customer> customers = entityManager.createQuery("select c from
Customers c" +
                    " where c.companyName =: nameX")
                    .setParameter("nameX",companyName)
                    .getResultList();
            if(customers.size() == 0)
            {

                System.out.println("No such customer");
                return;
                //TODO
            }
            Customer customer = customers.get(0);
            entityManager.close();

            ConfirmationWindow confirmationWindow = new
ConfirmationWindow(bucket, customer);
            confirmationWindow.setVisible(true);
            this.dispose();

        }

    }
}
```

Okienko potwierdzenia zakupu i adresu dostawy. Wyświetlane są dane adresowe kupującego oraz całkowita cena.



Kod klasy ConfirmationWindow:

```
public class ConfirmationWindow  extends JFrame implements ActionListener {

    JButton bConfirm;
    JLabel labeldeliveryInfo;
    JLabel label1, label2;
    List<Order> bucket;
    Customer customer;
    JLabel labelTotalCost;




    public ConfirmationWindow(List<Order> bucket, Customer customer)
    {
        this.customer = customer;
        this.bucket = bucket;
        setSize(300, 250);
        setTitle("Confirmation");
        setLayout(null);

        label1 = new JLabel("Please check if everything is right");
        label1.setBounds(25, 25, 200, 20);
        add(label1);

        label2 = new JLabel(customer.getCompanyName());
        label2.setBounds(25, 55, 200, 15);
        add(label2);

        labeldeliveryInfo = new JLabel(customer.getAddress().deliverInfo());
        labeldeliveryInfo.setBounds(25, 65, 100, 50);
        add(labeldeliveryInfo);

        float totalValue = 0;
        for(Order order : bucket)
        {
            totalValue+= order.getAmount()*order.getProduct().getPrice();
        }
        labelTotalCost = new JLabel("Total cost: " + totalValue);
        labelTotalCost.setBounds(25, 110, 100, 60);
        add(labelTotalCost);

        bConfirm = new JButton("ok");
        bConfirm.setBounds(160,100,50,50);
        bConfirm.addActionListener(this);
        add(bConfirm);


    }

    @Override
    public void actionPerformed(ActionEvent e) {
```

```
        Object eventSource = e.getSource();
        if(eventSource == bConfirm)
        {
            EntityManager entityManager =
JPAFactory.getEntityManagerFactory().createEntityManager();
            EntityTransaction entityTransaction =
entityManager.getTransaction();
            entityTransaction.begin();
            Invoice invoice = new Invoice();

            List<Customer> customers = entityManager.createQuery("select c from
Customers c" +
                    " where companyName =: nameX")
                    .setParameter("nameX",customer.getCompanyName())
                    .getResultList();
            invoice.setCustomer(customers.get(0));
            for(Order order : bucket)
            {
                List<Product> results = entityManager.createQuery("select p from
Products p" +
                        " where p.productName =: nameX")
                        .setParameter("nameX",order.getProduct().getProductName(
))
                        .getResultList();


                invoice.addProduct(results.get(0));
                entityManager.persist(results.get(0));
            }

            entityManager.persist(invoice);
            entityManager.persist(customers.get(0));
            entityTransaction.commit();
            entityManager.close();
            this.dispose();


        }    }
}
```