Po zainstalowaniu Derby i odpaleniu w konsoli wszystko działa poprawnie:

```
PS C:\Derby\db-derby-10.14.2.0-bin\bin> ./startNetworkServer
Wed Apr 19 16:48:31 CEST 2023 : Security manager installed using the Basic server security policy.
Wed Apr 19 16:48:31 CEST 2023 : Serwer sieciowy Apache Derby - 10.14.2.0 - (1828579) uruchomiony i gotowy do zaakceptowa
nia po|łączenia" na porcie 1527 w {3}
```

W Project Structure dodałem wszystkie odpowiednie pliki .jar

Export	Scope
19 (Oracle OpenJDK version 19)	
<Module source>	
Hibernate 5.4.11-5.4.11	Compile
derby.jar (C:\Derby\db-derby-10.14.2.0-bin\lib)	Compile
derbyclient.jar (C:\Derby\db-derby-10.14.2.0-bin\lib)	Compile
derbynet.jar (C:\Derby\db-derby-10.14.2.0-bin\lib)	Compile
derbytools.jar (C:\Derby\db-derby-10.14.2.0-bin\lib)	Compile

Do pliku hibernate.cfg.xml:

```
    "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <property name="connection.url">jdbc:derby://127.0.0.1/WWoznyJPA</property>
    <property name="connection.driver_class">org.apache.derby.jdbc.ClientDriver</property>
    <property name="hibernate.hbm2ddl.auto">update</property>
    <property name="show_sql">true</property>
    <property name="format_sql">true</property>
    <!— <property name="connection.username"/> —>
    <!— <property name="connection.password"/> —>
```

Po dodaniu odpowiedniego kodu do klasy Main wszystko działa poprawnie:

```
kwi 19, 2023 4:49:14 PM org.hibernate.Version logVersion
INFO: HHH000412: Hibernate Core {5.4.11.Final}
kwi 19, 2023 4:49:14 PM org.hibernate.annotations.common.reflection.java.JavaReflectionManager <clinit>
INFO: HCANN000001: Hibernate Commons Annotations {5.1.0.Final}
kwi 19, 2023 4:49:15 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl configure
WARN: HHH10001002: Using Hibernate built-in connection pool (not for production use!)
kwi 19, 2023 4:49:15 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001005: using driver [org.apache.derby.jdbc.ClientDriver] at URL [jdbc:derby://127.0.0.1/WWoznyJPA]
kwi 19, 2023 4:49:15 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001001: Connection properties: {}
kwi 19, 2023 4:49:15 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001003: Autocommit mode: false
kwi 19, 2023 4:49:15 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl$PooledConnections <init>
INFO: HHH000115: Hibernate connection pool size: 20 (min=1)
kwi 19, 2023 4:49:15 PM org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.DerbyTenSevenDialect
kwi 19, 2023 4:49:16 PM org.hibernate.resource.transaction.backend.jdbc.internal.DdlTransactionIsolatorNonJtaImpl getIsolatedConnection
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator$ConnectionProviderJdbcConnectionAccess@6622a690] for (non-JTA) DDL e
kwi 19, 2023 4:49:16 PM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator initiateService
INFO: HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]

Process finished with exit code 0
```

Pomyślnie dodałem bazę danych DerbyApache:

Database

jdbc:derby://127.0.0.1/WWoznyJPA 1 of 11

APP

Teraz zająłem się dodawaniem nowego produktu do bazy danych. Na początku stworzyłem klasę Product:

```
package org.example;

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

4 usages
@Entity
public class Product {

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

    1 usage
    private String ProductName;
    1 usage
    private int UnitsOnStock;

    public Product() {}

    1 usage
    public Product(String productName, int unitsOnStock) {
        this.ProductName = productName;
        this.UnitsOnStock = unitsOnStock;
    }

}
```

Dodałem odpowiednią adnotację @Entity, nominowane pole ID oraz pusty konstruktor. Następnym krokiem było zmodyfikowanie maina w następujący sposób:

```
    public static void main(final String[] args) throws Exception {
        Product product = new Product( productName: "Flamaster",  unitsOnStock: 12);
        final Session session = getSession();  // opened session already

        try {
            Transaction tx = session.beginTransaction();
            session.save(product);
            tx.commit();
        } finally {
            session.close();
        }
    }
}
```

Stworzyłem nowy obiekt klasy Product o nazwie „flamaster” i o ilości na magazynie 12 i dodałem go do bazy danych. Poniższe screeny potwierdzają, że dodawanie powiodło się:

```
Hibernate:

    create table Product (
       dbID integer not null,
        ProductName varchar(255),
        UnitsOnStock integer not null,
        primary key (dbID)
    )
Hibernate: create sequence hibernate_sequence start with 1 increment by 1
kwi 19, 2023 5:33:04 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
        Product
        (ProductName, UnitsOnStock, dbID)
    values
        (?, ?, ?)

Process finished with exit code 0
```

```
select * from PRODUCT;
```

	DBID	PRODUCTNAME	UNITSONSTOCK
1	1	Flamaster	12

Zadanie II

Zadanie zacząłem od dodania klasy Supplier:

```
6 usages
@Entity
public class Supplier {

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

    1 usage
    private String CompanyName;
    1 usage
    private String Street;
    1 usage
    private String City;


    public Supplier() {
    }

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

Potem zmodyfikowałem klasę Product dodając nowe pole Supplier, oznaczyłem je adnotacją @ManyToOne oraz dodałem seter dla Suppliera

```
4 usages
@Entity
public class Product {

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

    1 usage
    private String ProductName;
    1 usage
    private int UnitsOnStock;
    1 usage
    @ManyToOne
    private Supplier Supplier;

    public Product() {}

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

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

Dodałem do istniejącego już produktu nowego dostawcę:

```
public static void main(final String[] args) throws Exception {
    final Session session = getSession();  // opened session already
    Supplier supplier = new Supplier( companyName: "DHL",  street: "mazowiecka",  city: "Krakow");

    try {
        Transaction tx = session.beginTransaction();
        Product product = session.get(Product.class,  serializable: 2);
        product.setSupplier(supplier);
        session.save(supplier);
        tx.commit();
    } finally {
        session.close();
    }
}
```

Wszystko zostało dodane do bazy danych:

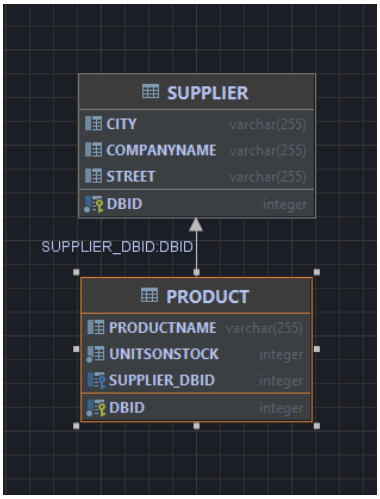
```
select * from PRODUCT;
```

	DBID	PRODUCTNAME	UNITSONSTOCK	SUPPLIER_DBID
1	2	Flamaster	12	102

```
select * from SUPPLIER;
```

	DBID	CITY	COMPANYNAME	STREET
1	102	Krakow	DHL	mazowiecka

Diagram bazy danych po wykonanych czynnościach wygląda tak:



Zadanie III

a) Wersja z tabelą łącznikową

Zmodyfikowałem klasę Supplier dodając set Products z produktami oraz dodając metodę, która dodaje produkty do pola Products:

```
@Entity
public class Supplier {

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

    1 usage
    private String CompanyName;
    1 usage
    private String Street;
    1 usage
    private String City;
    1 usage
    @OneToMany
    private Set<Product> Products;


    public Supplier() {
    }

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

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

Dodałem nowe produkty do bazy:

```java
public static void main(final String[] args) throws Exception {
    final Session session = getSession();  // opened session already
    Product product1 = new Product( productName: "product1",  unitsOnStock: 10);
    Product product2 = new Product( productName: "product1",  unitsOnStock: 5);
    Product product3 = new Product( productName: "product1",  unitsOnStock: 2);
    Product product4 = new Product( productName: "product1",  unitsOnStock: 111);

    try {
        Transaction tx = session.beginTransaction();
        session.save(product1);
        session.save(product2);
        session.save(product3);
        session.save(product4);
        tx.commit();
    } finally {
        session.close();
    }
}
}
```

Output | APP.PRODUCT

5 rows | Tx: Auto | DDL

	DBID	PRODUCTNAME	UNITSONSTOCK	SUPPLIER_DBID
1	2	Flamaster	12	102
2	103	product1	10	<null>
3	104	product1	5	<null>
4	105	product1	2	<null>
5	106	product1	111	<null>

Oraz nowego dostawcę:

```java
public static void main(final String[] args) throws Exception {
    final Session session = getSession();  // opened session already
    Supplier supplier = new Supplier( companyName: "SUP",  street: "kolorowa",  city: "Warszawa");

    try {
        Transaction tx = session.beginTransaction();
        session.save(supplier);
        tx.commit();
    } finally {
        session.close();
    }
}
}
```

Output | APP.SUPPLIER

2 rows | Tx: Auto | DDL

	DBID	CITY	COMPANYNAME	STREET
1	102	Krakow	DHL	mazowiecka
2	107	Warszawa	SUP	kolorowa

Po dodaniu produktów do nowo dodanego dostawcy tabela łącznikowa wygląda tak:

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

    try {
        Transaction tx = session.beginTransaction();
        Supplier supplier = session.get(Supplier.class,  serializable: 107);
        Product product1 = session.get(Product.class,  serializable: 103);
        Product product2 = session.get(Product.class,  serializable: 104);
        Product product3 = session.get(Product.class,  serializable: 105);
        Product product4 = session.get(Product.class,  serializable: 106);
        supplier.addProduct(product1);
        supplier.addProduct(product2);
        supplier.addProduct(product3);
        supplier.addProduct(product4);
        tx.commit();
```

	SUPPLIER_DBID	PRODUCTS_DBID
1	107	103
2	107	104
3	107	105
4	107	106

b) Wersja bez tabeli łącznikowej

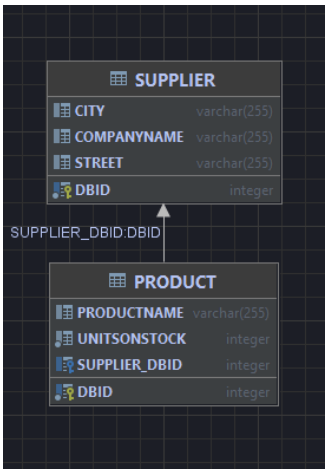
Aby uzyskać wersję bez tabeli łącznikowej dodałem adnotację do pola Products @JoinColumn:

```
1 usage
@OneToMany
@JoinColumn(name = "SUPPLIER_DBID")
private Set<Product> Products;
```

Wtedy select * from PRODUCT wygląda tak:

	DBID	PRODUCTNAME	UNITSONSTOCK	SUPPLIER_DBID
1	2	Flamaster	12	102
2	103	product1	10	107
3	104	product1	5	107
4	105	product1	2	107
5	106	product1	111	107

A schemat bazy danych tak:



Zadanie IV

Aby stworzyć relację dwukierunkową, w klasie Supplier dodałem następującą adnotację do pola Products:

```
2 usages
@OneToMany
@JoinColumn(name = "SUPPLIER_DBID")
private Set<Product> Products = new HashSet<Product>();
```

I zmodyfikowałem metodę addProduct:

```
5 usages
public void addProduct(Product product) {
    this.Products.add(product);
    product.setSupplier(this);
}
```

W klasie Product podobnie:

```
2 usages
@ManyToOne
@JoinColumn(name = "SUPPLIER_DBID")
private Supplier Supplier;
```

```
1 usage
public void setSupplier(Supplier supplier) {
    this.Supplier = supplier;
    this.Supplier.getProducts().add(this);
}
```

Po dodaniu do bazy danych nowych produktów i dostawców wygląda to tak:

```
public static void main(final String[] args) throws Exception {
    final Session session = getSession();  // opened session already
    Supplier supplier1 = new Supplier( companyName: "DHL",  street: "mazowiecka",  city: "Krakow");
    Supplier supplier2 = new Supplier( companyName: "SUPP",  street: "kolorowa",  city: "Warszawa");

    Product product1 = new Product( productName: "product1",  unitsOnStock: 10);
    Product product2 = new Product( productName: "product2",  unitsOnStock: 111);
    Product product3 = new Product( productName: "product3",  unitsOnStock: 20);
    Product product4 = new Product( productName: "product4",  unitsOnStock: 15);
    Product product5 = new Product( productName: "product5",  unitsOnStock: 4);

    try {
        Transaction tx = session.beginTransaction();
        session.save(product1);
        session.save(product2);
        session.save(product3);
        session.save(product4);
        session.save(product5);
        session.save(supplier1);
        session.save(supplier2);
        supplier1.addProduct(product1);
        supplier1.addProduct(product2);
        supplier1.addProduct(product3);
        supplier2.addProduct(product4);
        supplier2.addProduct(product5);
        tx.commit();
    } finally {
        session.close();
    }
}
```

	DBID	CITY	COMPANYNAME	STREET
1	127	Krakow	DHL	mazowiecka
2	128	Warszawa	SUPP	kolorowa

	DBID ▲ 1	PRODUCTNAME	UNITSONSTOCK	SUPPLIER_DBID
1	122	product1	10	127
2	123	product2	111	127
3	124	product3	20	127
4	125	product4	15	128
5	126	product5	4	128

Zadanie V

Nowa klasa Category dodana:

```java
@Entity
public class Category {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int CategoryID;
    1 usage
    private String Name;
    1 usage
    @OneToMany
    private List<Product> Products = new ArrayList<Product>();

    public Category() {}

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

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

Dodałem nową kategorię, a istniejące produkty przypisałem do tych nowych kategorii:

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

    Category category1 = new Category( name: "cat1");
    Category category2 = new Category( name: "cat2");

    try {
        Transaction tx = session.beginTransaction();
        Product product1 = session.get(Product.class,  serializable: 122);
        Product product2 = session.get(Product.class,  serializable: 123);
        Product product3 = session.get(Product.class,  serializable: 124);
        Product product4 = session.get(Product.class,  serializable: 125);
        Product product5 = session.get(Product.class,  serializable: 126);
        category1.addProduct(product1);
        category1.addProduct(product2);
        category2.addProduct(product3);
        category2.addProduct(product4);
        category2.addProduct(product5);
        session.save(category1);
        session.save(category2);
        tx.commit();
    } finally {
        session.close();
    }
}
```

Po dodaniu do bazy tabela Category:

	CATEGORYID	NAME
1	129	cat1
2	130	cat2

I tabela łącznikowa:

	CATEGORY_CATEGORYID	PRODUCTS_DBID
1	129	122
2	129	123
3	130	124
4	130	125
5	130	126

Wypisanie kategorii, w której jest produkt ‘product1’:

```
try {
    Transaction tx = session.beginTransaction();
    Query q = session.createQuery( s: "from Category as category join category.Products as " +
            "product where product.ProductName = 'product1'");
    List<Object[]> categories = q.getResultList();
    for (Object[] category : categories) {
        System.out.println(category[0] + "-------" + category[1]);
    }
    tx.commit();
} finally {
    session.close();
}
```

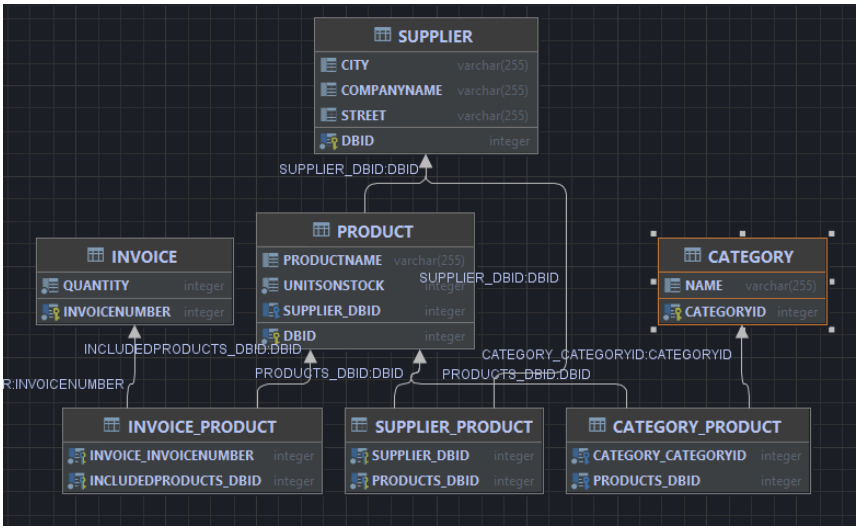
```
        supplier0_.dbID=?
org.example.Category@1fd9893c-------org.example.Product@5d1e0fbb
```

Zadanie VI

Dodana nowa klasa Invoice:

```
2 usages
@Entity
public class Invoice {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int InvoiceNumber;
    1 usage
    private int Quantity;
    1 usage
    @ManyToMany
    Set<Product> IncludedProducts = new HashSet<Product>();

    public Invoice() {}

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

    public void sellProduct(Product product) {
        this.IncludedProducts.add(product);
    }
```

Schemat bazy danych:



Sprzedawanie produktów:

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

    Invoice invoice1 = new Invoice( quantity: 2);
    Invoice invoice2 = new Invoice( quantity: 4);

    try {
        Transaction tx = session.beginTransaction();
        Product prod1 = session.get(Product.class,  serializable: 122);
        Product prod2 = session.get(Product.class,  serializable: 124);
        Product prod3 = session.get(Product.class,  serializable: 126);

        invoice1.sellProduct(prod1);
        invoice1.sellProduct(prod2);
        invoice2.sellProduct(prod3);

        session.save(invoice1);
        session.save(invoice2);

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

	INVOICENUMBER	QUANTITY
1	131	2
2	132	4

	INVOICE_INVOICENUMBER	INCLUDEDPRODUCTS_DBID
1	131	122
2	131	124
3	132	126

Zadanie VII

persistence.xml:

```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="WWoznyJPAConfig"
                    transaction-type="RESOURCE_LOCAL">
    <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/WWoznyJPA"/>
      <property name="hibernate.show_sql" value="true" />
      <property name="hibernate.format_sql" value="true" />
      <property name="hibernate.hbm2ddl.auto" value="create" />
    </properties>
  </persistence-unit>
</persistence>
```

Nowy main:

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

        Product product = em.find(Product.class,  o: 122);
        System.out.println(product.getProductName());

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

```
    where
        product0_.dbID=?
product1

Process finished with exit code 0
```

Zadanie VIII

Następująco zmodyfikowałem klasy Invoice oraz Product:

```
2 usages
@ManyToMany (cascade = {CascadeType.PERSIST})
Set<Product> IncludedProducts = new HashSet<>();
```

```
1 usage
@ManyToMany (mappedBy = "IncludedProducts", cascade = {CascadeType.PERSIST})
private Set<Invoice> Invoices = new HashSet<Invoice>();
```

Następnie dodałem nowe produkty, od razu z fakturą:

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

        Product product1 = new Product( productName: "product7",  unitsOnStock: 11);
        Product product2 = new Product( productName: "product8",  unitsOnStock: 11);

        Invoice invoice1 = new Invoice( quantity: 3);

        em.persist(product1);
        em.persist(product2);

        invoice1.sellProduct(product1);
        invoice1.sellProduct(product2);

        product1.addInvoice(invoice1);
        product2.addInvoice(invoice1);

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

Font size: 14pt Reset to 12pt

	INVOICENUMBER	QUANTITY
1	3	3

	INVOICES_INVOICENUMBER	INCLUDEDPRODUCTS_DBID
1	3	1
2	3	2

	DBID	PRODUCTNAME	UNITSONSTOCK	SUPPLIER_DBID
1	1	product7	11	<null>
2	2	product8	11	<null>
3	122	product1	10	127

Zadanie IX

Zacząłem od zrobienia nowej klasy Address:

```java
package org.example;

import javax.persistence.Embeddable;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

5 usages
@Embeddable
public class Address {

    1 usage
    private String Street;
    1 usage
    private String City;

    public Address() {}

    2 usages
    public Address(String street, String city) {
        this.Street = street;
        this.City = city;
    }

}
```

I zmodyfikowałem klasę Supplier:

```java
2 usages
@Embedded
private Address Address;
```

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

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

Po dodaniu wszystko działa jak należy:

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

        Address address = new Address( street: "mazowiecka",  city: "Rzeszow");
        Supplier supplier = new Supplier( companyName: "suppppp", address);
        em.persist(supplier);

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

	DBID	CITY	COMPANYNAME	STREET
1	127	Krakow	DHL	mazowiecka
2	128	Warszawa	SUPP	kolorowa
3	1	Rzeszow	suppppp	mazowiecka

Zadanie X

Strategia MappedSuperclass:

```
2 usages   2 inheritors
@MappedSuperclass
public abstract class Company {

    @Id
    @GeneratedValue
    private int CompanyID;

    2 usages
    private String CompanyName;
    1 usage
    private String Street;
    1 usage
    private String City;
    1 usage
    private String ZipCode;

    2 usages
    public Company() {}

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

    public String getCompanyName() {
        return this.CompanyName;
    }

}
```

```
3 usages
@Entity
public class Supplier extends Company {

    1 usage
    private String bankAccountNumber;

    public Supplier() {}

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

}
```

```
public class Customer extends Company {

    1 usage
    private double Discount;

    public Customer() {}

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

Strategia SingleTable:

```
2 usages   2 inheritors
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
public abstract class Company {

    @Id
```

Strategia JoinedTable:

```
2 usages   2 inheritors
@Inheritance(strategy = InheritanceType.JOINED)
public abstract class Company {

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