Bazy danych – Hibernate

Imię i nazwisko: Błażej Kustra

Tydzień B, czwartek 12:50

1. Podstawowa konfiguracja projektu w IntelliJ oraz wstawienie pierwszego produktu.

```
(base) blazejkustra@bk-3 ~ % /Users/blazejkustra/Desktop/db-derby-10.14.2.0-bin/bin/ij
wersja ij 10.14
ij> connect 'jdbc:derby://127.0.0.1/BKustraJPA;create=true';
ij> show tables;
TABLE_SCHEM         |TABLE_NAME                    |REMARKS
------------------------------------------------------------------------------
SYS                 |SYSALIASES                    |
SYS                 |SYSCHECKS                     |
SYS                 |SYSCOLPERMS                   |
SYS                 |SYSCOLUMNS                    |
SYS                 |SYSCONGLOMERATES              |
SYS                 |SYSCONSTRAINTS                |
SYS                 |SYSDEPENDS                    |
SYS                 |SYSFILES                      |
SYS                 |SYSFOREIGNKEYS                |
SYS                 |SYSKEYS                       |
SYS                 |SYSPERMS                      |
SYS                 |SYSROLES                      |
SYS                 |SYSROUTINEPERMS               |
SYS                 |SYSSCHEMAS                    |
SYS                 |SYSSEQUENCES                  |
SYS                 |SYSSTATEMENTS                 |
SYS                 |SYSSTATISTICS                 |
SYS                 |SYSTABLEPERMS                 |
SYS                 |SYSTABLES                     |
SYS                 |SYSTRIGGERS                   |
SYS                 |SYSUSERS                      |
SYS                 |SYSVIEWS                      |
SYSIBM              |SYSDUMMY1                     |

23 wierszy wybranych
ij>
```

Skonfigurowałem config Hibernate.

```xml
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="connection.url">jdbc:derby://127.0.0.1/BKustraJPA</property>
        <property name="connection.driver_class">org.apache.derby.jdbc.ClientDriver</property>
        <property name="dialect">org.hibernate.dialect.DerbyTenSevenDialect</property>
        <property name="format_sql">true</property>
        <property name="show_sql">true</property>
        <property name="use_sql_comments">true</property>
        <property name="hibernate.hbm2ddl.auto">update</property>
        <mapping class="Product"></mapping>
    </session-factory>
</hibernate-configuration>
```

Stworzyłem klasę Product z odpowiednimi polami.

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

@Entity
public class Product {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int productID;
    private String productName;
    private int unitsOnStock;

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

    @Override
    public String toString() {
        return "Product{" +
                "productID=" + productID +
                ", productName='" + productName + '\'' +
                ", unitsOnStock=" + unitsOnStock +
                '}';
    }
}
```

Dodałem produkt “Mleko” do bazy.

```java
import ...

public class Main {
    private static final SessionFactory ourSessionFactory;

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

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

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

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

            Product product = new Product( productName: "Mleko",  unitsOnStock: 6);
            Transaction transaction = session.beginTransaction();
            session.save(product);
            transaction.commit();

            System.out.println("querying all the managed entities...");
            final Metamodel metamodel = session.getSessionFactory().getMetamodel();
            for (EntityType<?> entityType : metamodel.getEntities()) {
                final String entityName = entityType.getName();
                final Query query = session.createQuery( s: "from " + entityName);
                System.out.println("executing: " + query.getQueryString());
                for (Object o : query.list()) {
                    System.out.println("  " + o);
                }
            }
        } finally {
            session.close();
        }
    }
```

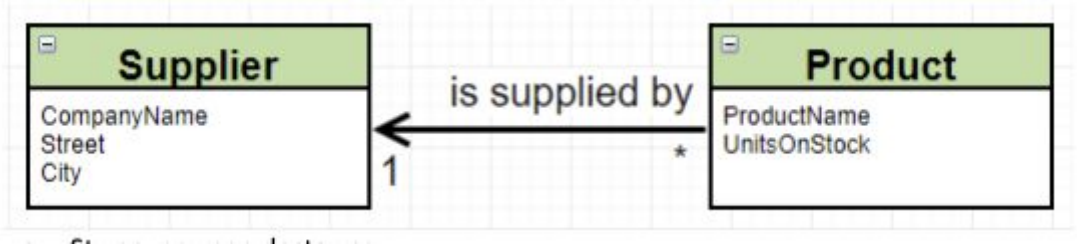
Wynik wykonania Maina.

```
Hibernate:

values
    next value for hibernate_sequence
Hibernate:
    /* insert Product
        */ insert
        into
            Product
            (productName, unitsOnStock, productID)
        values
            (?, ?, ?)
querying all the managed entities...
executing: from Product
Hibernate:
    /*
from
    Product */ select
        product0_.productID as producti1_0_,
        product0_.productName as productn2_0_,
        product0_.unitsOnStock as unitsons3_0_
    from
        Product product0_
  Product{productID=1, productName='Mleko', unitsOnStock=6}

Process finished with exit code 0
```

Wynik w Datagrip.

	PRODUCTID	PRODUCTNAME	UNITSONSTOCK
1	1	Mleko	6

2. Zmodyfikuj model wprowadzając pojęcie Dostawcy jak poniżej:



a) Stwórz nowego dostawcę.
b) Znajdź poprzednio wprowadzony produkt i ustaw jego dostawcę na właśnie dodanego.

Klasa supplier:

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

@Entity
public class Supplier {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int supplierID;
    private String companyName;
    private String street;
    private String city;

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

    public Supplier() {
    }

    @Override
    public String toString() {
        return "Supplier{" +
                "supplierID=" + supplierID +
                ", companyName='" + companyName + '\'' +
                ", street='" + street + '\'' +
                ", city='" + city + '\'' +
                '}';
    }
}
```

Dodałem pole supplier w produkcie oraz wygenerowałem setter dla suppliera. Następnie wykonałem zmieny w mainie i stworzyłem suppliera.

Zmiany w mainie:

```java
Product product = session.find(Product.class, o: 1);
Supplier supplier = new Supplier( companyName: "Żabka", street: "Westerplatte", city: "Kraków");
product.setSupplier(supplier);
session.save(product);
session.save(supplier);
transaction.commit();
```

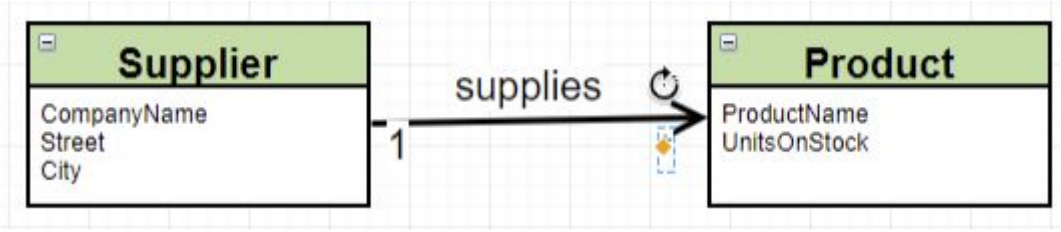
tabela produktów:

	PRODUCTID	PRODUCTNAME	UNITSONSTOCK	SUPPLIER_SUPPLIERID
1	1	Mleko	6	2

tabela dostawców:

	SUPPLIERID	CITY	COMPANYNAME	STREET
1	2	Kraków	Żabka	Westerplatte

3. Odwróć relacje zgodnie z poniższym schematem



a) Zamodeluj powyższe w dwóch wariantach „z” i „bez” tabeli łącznikowej b. c.
b) Stwórz kilka produktów
c) Dodaj je do produktów dostarczanych przez nowo stworzonego dostawcę

Klasa Product:

```
import javax.persistence.*;

@Entity
public class Product {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int productID;
    private String productName;
    private int unitsOnStock;

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

    public Product() {}

    @Override
    public String toString() {
        return "Product{" +
                "productID=" + productID +
                ", productName='" + productName + '\'' +
                ", unitsOnStock=" + unitsOnStock +
                '}';
    }
}
```

Klasa Dostawcy:

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

@Entity
public class Supplier {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int supplierID;
    private String companyName;
    private String street;
    private String city;

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

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

    public Supplier() {
    }

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

    @Override
    public String toString() {
        return "Supplier{" +
                "supplierID=" + supplierID +
                ", companyName='" + companyName + '\'' +
                ", street='" + street + '\'' +
                ", city='" + city + '\'' +
                '}';
    }
}
```

Klasa main:

```java
Transaction transaction = session.beginTransaction();

Product product1 = new Product( productName: "Woda",  unitsOnStock: 100);
Product product2 = new Product( productName: "Maseczki",  unitsOnStock: 10);

Supplier supplier = new Supplier( companyName: "Groszek",  street: "Wrocławska",  city: "Kraków");

supplier.addProduct(product1);
supplier.addProduct(product2);
session.save(supplier);
session.save(product1);
session.save(product2);
transaction.commit();
```

Tabela produktów:

	PRODUCTID	PRODUCTNAME	UNITSONSTOCK	SUPPLIER_SUPPLIERID
1	1	Mleko	6	2
2	4	Woda	100	<null>
3	5	Maseczki	10	<null>

Tabela dostawców:

	SUPPLIERID	CITY	COMPANYNAME	STREET
1	2	Kraków	Żabka	Westerplatte
2	3	Kraków	Groszek	Wrocławska

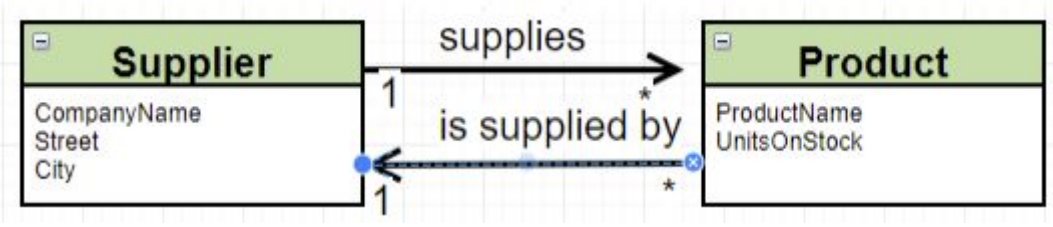
Automatycznie wygenerowana tabela:

	SUPPLIER_SUPPLIERID	PRODUCTS_PRODUCTID
1	3	4
2	3	5

Uruchomienie maina:

```
        (productName, unitsOnStock, productID)
        values
            (?, ?, ?)
Hibernate:
    /* insert collection
        row Supplier.products */ insert
        into
            Supplier_Product
            (Supplier_supplierID, products_productID)
        values
            (?, ?)
Hibernate:
    /* insert collection
        row Supplier.products */ insert
        into
            Supplier_Product
            (Supplier_supplierID, products_productID)
        values
            (?, ?)
querying all the managed entities...
executing: from Product
Hibernate:
    /*
from
    Product */ select
        product0_.productID as producti1_0_,
        product0_.productName as productn2_0_,
        product0_.unitsOnStock as unitsons3_0_
    from
        Product product0_
  Product{productID=1, productName='Mleko', unitsOnStock=6}
  Product{productID=4, productName='Woda', unitsOnStock=100}
  Product{productID=5, productName='Maseczki', unitsOnStock=10}
executing: from Supplier
Hibernate:
    /*
from
    Supplier */ select
        supplier0_.supplierID as supplier1_1_,
        supplier0_.city as city2_1_,
        supplier0_.companyName as companyn3_1_,
        supplier0_.street as street4_1_
    from
        Supplier supplier0_
  Supplier{supplierID=2, companyName='Żabka', street='Westerplatte', city='Kraków'}
  Supplier{supplierID=3, companyName='Groszek', street='Wrocławska', city='Kraków'}

Process finished with exit code 0
```

4. Zamodeluj relacje dwustronną jak poniżej:



a) Tradycyjnie: Stworz kilka produktow
b) Dodaj je do produktow dostarczanych przez nowo stworzonego dostawcę (pamiętaj o poprawnej obsłudze dwustronności relacji)
c) Udokumentuj wykonane kroki oraz uzyskane rezultaty (logi wywołań sqlowych, describe table/diagram z datagrip, select * from….)

Klasa Product:

```java
import javax.persistence.*;

@Entity
public class Product {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int productID;
    private String productName;
    private int unitsOnStock;

    @ManyToOne
    private Supplier supplier;


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

    public Product() {}

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

    @Override
    public String toString() {
        return "Product{" +
                "productID=" + productID +
                ", productName='" + productName + '\'' +
                ", unitsOnStock=" + unitsOnStock +
                '}';
    }
}
```

Klasa Dostawcy:

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

@Entity
public class Supplier {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int supplierID;
    private String companyName;
    private String street;
    private String city;

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

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

    public Supplier() {
    }

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

    @Override
    public String toString() {
        return "Supplier{" +
                "supplierID=" + supplierID +
                ", companyName='" + companyName + '\'' +
                ", street='" + street + '\'' +
                ", city='" + city + '\'' +
                '}';
    }
}
```

Zmiany w mainie:

```
30            Transaction transaction = session.beginTransaction();
31            Supplier supplier = new Supplier( companyName: "Małpka",  street: "Długa",  city: "Kraków");
32
33            Product product1 = new Product( productName: "Czekolada",  unitsOnStock: 1, supplier);
34            Product product2 = new Product( productName: "Sok",  unitsOnStock: 5, supplier);
35            supplier.addProduct(product1);
36            supplier.addProduct(product2);
37
38            session.save(supplier);
39            session.save(product1);
40            session.save(product2);
41            transaction.commit();
```

Wykonanie maina:

```
querying all the managed entities...
executing: from Product
Hibernate:
    /*
from
    Product */ select
        product0_.productID as producti1_0_,
        product0_.productName as productn2_0_,
        product0_.supplier_supplierID as supplier4_0_,
        product0_.unitsOnStock as unitsons3_0_
    from
        Product product0_
Hibernate:
    select
        supplier0_.supplierID as supplier1_1_0_,
        supplier0_.city as city2_1_0_,
        supplier0_.companyName as companyn3_1_0_,
        supplier0_.street as street4_1_0_
    from
        Supplier supplier0_
    where
        supplier0_.supplierID=?
  Product{productID=1, productName='Mleko', unitsOnStock=6}
  Product{productID=4, productName='Woda', unitsOnStock=100}
  Product{productID=5, productName='Maseczki', unitsOnStock=10}
  Product{productID=7, productName='Czekolada', unitsOnStock=1}
  Product{productID=8, productName='Sok', unitsOnStock=5}
executing: from Supplier
Hibernate:
    /*
from
    Supplier */ select
        supplier0_.supplierID as supplier1_1_,
        supplier0_.city as city2_1_,
        supplier0_.companyName as companyn3_1_,
        supplier0_.street as street4_1_
    from
        Supplier supplier0_
  Supplier{supplierID=2, companyName='Żabka', street='Westerplatte', city='Kraków'}
  Supplier{supplierID=3, companyName='Groszek', street='Wrocławska', city='Kraków'}
  Supplier{supplierID=6, companyName='Małpka', street='Długa', city='Kraków'}
```

Tabela Produktów:

	PRODUCTID	PRODUCTNAME	UNITSONSTOCK	SUPPLIER_SUPPLIERID
1	1	Mleko	6	2
2	4	Woda	100	<null>
3	5	Maseczki	10	<null>
4	7	Czekolada	1	6
5	8	Sok	5	6

Tabela Dostawców:

	SUPPLIERID	CITY	COMPANYNAME	STREET
1	2	Kraków	Żabka	Westerplatte
2	3	Kraków	Groszek	Wrocławska
3	6	Kraków	Małpka	Długa

Automatycznie wygenerowana tabela:

	SUPPLIER_SUPPLIERID	PRODUCTS_PRODUCTID
1	3	4
2	3	5
3	6	7
4	6	8

5. Dodaj klase Category z property int CategoryID, String Name oraz listą produktów List Products

a) Zmodyfikuj produkty dodając wskazanie na kategorie do której należy.
b) Stworz kilka produktow i kilka kategorii
c) Dodaj kilka produktów do wybranej kategorii

Klasa Category (dodany mapping w configu):

```java
import javax.persistence.*;
import java.lang.reflect.Array;
import java.util.*;

@Entity
public class Category {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int categoryID;
    private String name;

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

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

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

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

    public Category() {}

}
```

Klasa Product:

```java
import javax.persistence.*;

@Entity
public class Product {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int productID;
    private String productName;
    private int unitsOnStock;

    @ManyToOne
    private Supplier supplier;

    @ManyToOne
    private Category category;

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

    public Product() {}

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

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

    @Override
    public String toString() {
        return "Product{" +
                "productID=" + productID +
                ", productName='" + productName + '\'' +
                ", unitsOnStock=" + unitsOnStock +
                '}';
    }
}
```

Zmiany maina:

```java
        Transaction transaction = session.beginTransaction();
        Supplier supplier = new Supplier( companyName: "Drewit",  street: "Prądnik",  city: "Kraków");
        Category category1 = new Category( name: "Meble");
        Category category2 = new Category( name: "Ubrania");

        Product product1 = new Product( productName: "Biurko",  unitsOnStock: 1, supplier, category1);
        Product product2 = new Product( productName: "Buty adidas",  unitsOnStock: 5, supplier, category2);

        supplier.addProduct(product1);
        supplier.addProduct(product2);

        category1.addProduct(product1);
        category2.addProduct(product2);

        session.save(supplier);
        session.save(category1);
        session.save(category2);
        session.save(product1);
        session.save(product2);
        transaction.commit();
```

Wykonanie maina:

```
    Product */ select
        product0_.productID as producti1_2_,
        product0_.category_categoryID as category4_2_,
        product0_.productName as productn2_2_,
        product0_.supplier_supplierID as supplier5_2_,
        product0_.unitsOnStock as unitsons3_2_
    from
        Product product0_
Hibernate:
    select
        supplier0_.supplierID as supplier1_3_0_,
        supplier0_.city as city2_3_0_,
        supplier0_.companyName as companyn3_3_0_,
        supplier0_.street as street4_3_0_
    from
        Supplier supplier0_
    where
        supplier0_.supplierID=?
Hibernate:
    select
        supplier0_.supplierID as supplier1_3_0_,
        supplier0_.city as city2_3_0_,
        supplier0_.companyName as companyn3_3_0_,
        supplier0_.street as street4_3_0_
    from
        Supplier supplier0_
    where
        supplier0_.supplierID=?
  Product{productID=1, productName='Mleko', unitsOnStock=6}
  Product{productID=4, productName='Woda', unitsOnStock=100}
  Product{productID=5, productName='Maseczki', unitsOnStock=10}
  Product{productID=7, productName='Czekolada', unitsOnStock=1}
  Product{productID=8, productName='Sok', unitsOnStock=5}
  Product{productID=12, productName='Biurko', unitsOnStock=1}
  Product{productID=13, productName='Buty adidas', unitsOnStock=5}
executing: from Category
Hibernate:
    /*
from
    Category */ select
        category0_.categoryID as category1_0_,
        category0_.name as name2_0_
    from
        Category category0_
  Category@749f539e
  Category@6075b2d3
executing: from Supplier
Hibernate:
    /*
from
    Supplier */ select
        supplier0_.supplierID as supplier1_3_,
        supplier0_.city as city2_3_,
        supplier0_.companyName as companyn3_3_,
        supplier0_.street as street4_3_
    from
        Supplier supplier0_
  Supplier{supplierID=2, companyName='Żabka', street='Westerplatte', city='Kraków'}
  Supplier{supplierID=3, companyName='Groszek', street='Wrocławska', city='Kraków'}
  Supplier{supplierID=6, companyName='Małpka', street='Długa', city='Kraków'}
  Supplier{supplierID=9, companyName='Drewit', street='Prądnik', city='Kraków'}
```

Tabela produktów:

	PRODUCTID	PRODUCTNAME	UNITSONSTOCK	SUPPLIER_SUPPLIERID	CATEGORY_CATEGORYID
1	1	Mleko	6	2	<null>
2	4	Woda	100	<null>	<null>
3	5	Maseczki	10	<null>	<null>
4	7	Czekolada	1	6	<null>
5	8	Sok	5	6	<null>
6	12	Biurko	1	9	10
7	13	Buty adidas	5	9	11

Tabela kategorii:

	CATEGORYID	NAME
1	10	Meble
2	11	Ubrania

Tabela dostawców:

	SUPPLIERID	CITY	COMPANYNAME	STREET
1	2	Kraków	Żabka	Westerplatte
2	3	Kraków	Groszek	Wrocławska
3	6	Kraków	Małpka	Długa
4	9	Kraków	Drewit	Prądnik

Automatycznie wygenerowane tabele:

	SUPPLIER_SUPPLIERID	PRODUCTS_PRODUCTID
1	3	4
2	3	5
3	6	7
4	6	8
5	9	12
6	9	13

	CATEGORY_CATEGORYID	PRODUCTS_PRODUCTID
1	10	12
2	11	13

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

Produkty należące do kategorii 10:

```
Category category = session.find(Category.class, o: 10);

Query q = session.createQuery( s: "FROM Product P WHERE P.category = " + category);

System.out.println(category);

for (Object o : q.list()) {
    System.out.println("  " + o);
}
```

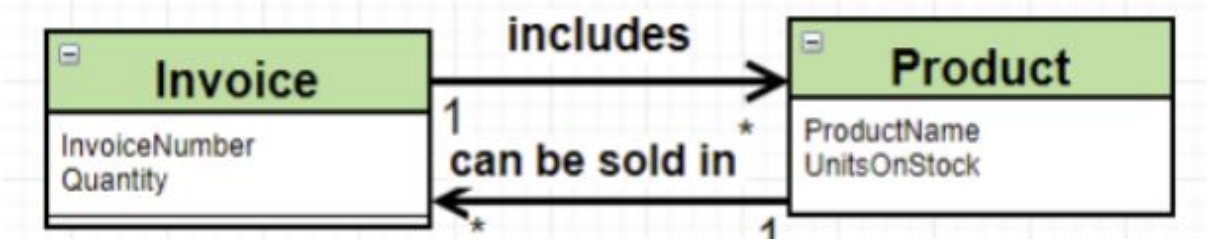
```
Product{productID=12, productName='Biurko', unitsOnStock=1, supplier=Supplier{supplierID=9, companyName='Drewit', street='Prądnik', city='Kraków'},
```

Kategoria produktu 12:

```
Product product = session.find(Product.class, o: 12);
System.out.println("\n\n\n"+product.getCategory() + "\n\n\n");
```

```
Category{categoryID=10, name='Meble'}
```

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



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

Klasa Invoice:

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

@Entity
public class Invoice {

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

    @ManyToMany
    private Set<Product> products = new LinkedHashSet<>();

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

    public Invoice() {}

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

    @Override
    public String toString() {
        return "Invoice{" +
                "invoiceID=" + invoiceID +
                ", quantity=" + quantity +
                '}';
    }
}
```

Zmiany w klasie Product:

```java
    @ManyToMany
    private Set<Invoice> invoices = new LinkedHashSet<>();

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

Zmiany w mainie:

```java
public static void main(final String[] args) throws Exception {
    final Session session = getSession();
    try {
        Transaction transaction = session.beginTransaction();
        Category category = session.find(Category.class,  o: 11);
        Supplier supplier = session.find(Supplier.class,  o: 6);

        Product product1 = new Product( productName: "buty puma",  unitsOnStock: 5, supplier, category);
        Product product2 = new Product( productName: "Buty adidas",  unitsOnStock: 5, supplier, category);
        Product product3 = new Product( productName: "Buty nike",  unitsOnStock: 5, supplier, category);

        Invoice invoice1 = new Invoice( quantity: 1);
        Invoice invoice2 = new Invoice( quantity: 2);

        supplier.addProduct(product1);
        supplier.addProduct(product2);
        supplier.addProduct(product3);

        category.addProduct(product1);
        category.addProduct(product2);
        category.addProduct(product3);

        invoice1.addProduct(product1);
        invoice1.addProduct(product2);
        invoice2.addProduct(product3);

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

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

        session.save(product1);
        session.save(product2);
        session.save(product3);

        transaction.commit();
```

Wywołanie maina:

```
    Product */ select
        product0_.productID as producti1_4_,
        product0_.category_categoryID as category4_4_,
        product0_.productName as productn2_4_,
        product0_.supplier_supplierID as supplier5_4_,
        product0_.unitsOnStock as unitsons3_4_
    from
        Product product0_
Hibernate:
    select
        supplier0_.supplierID as supplier1_6_0_,
        supplier0_.city as city2_6_0_,
        supplier0_.companyName as companyn3_6_0_,
        supplier0_.street as street4_6_0_
    from
        Supplier supplier0_
    where
        supplier0_.supplierID=?
  Product{productID=1, productName='Mleko', unitsOnStock=6, supplier=Supplier{supplierID=2, companyName='Żabka', street='Westerplatte', city='Kraków'}, category=nu
  Product{productID=4, productName='Woda', unitsOnStock=100, supplier=null, category=null}
  Product{productID=5, productName='Maseczki', unitsOnStock=10, supplier=null, category=null}
  Product{productID=7, productName='Czekolada', unitsOnStock=1, supplier=Supplier{supplierID=6, companyName='Małpka', street='Długa', city='Kraków'}, category=null
  Product{productID=8, productName='Sok', unitsOnStock=5, supplier=Supplier{supplierID=6, companyName='Małpka', street='Długa', city='Kraków'}, category=null}
  Product{productID=12, productName='Biurko', unitsOnStock=1, supplier=Supplier{supplierID=9, companyName='Drewit', street='Prądnik', city='Kraków'}, category=Cate
  Product{productID=13, productName='Buty adidas', unitsOnStock=5, supplier=Supplier{supplierID=9, companyName='Drewit', street='Prądnik', city='Kraków'}, category
  Product{productID=16, productName='buty puma', unitsOnStock=5, supplier=Supplier{supplierID=6, companyName='Małpka', street='Długa', city='Kraków'}, category=Cat
  Product{productID=17, productName='Buty adidas', unitsOnStock=5, supplier=Supplier{supplierID=6, companyName='Małpka', street='Długa', city='Kraków'}, category=C
  Product{productID=18, productName='Buty nike', unitsOnStock=5, supplier=Supplier{supplierID=6, companyName='Małpka', street='Długa', city='Kraków'}, category=Cat
executing: from Invoice
Hibernate:
    /*
from
    Invoice */ select
        invoice0_.invoiceID as invoicei1_2_,
        invoice0_.quantity as quantity2_2_
    from
        Invoice invoice0_
  Invoice{invoiceID=14, quantity=1}
  Invoice{invoiceID=15, quantity=2}
executing: from Supplier
Hibernate:
    /*
from
    Supplier */ select
        supplier0_.supplierID as supplier1_6_,
        supplier0_.city as city2_6_,
        supplier0_.companyName as companyn3_6_,
        supplier0_.street as street4_6_
    from
        Supplier supplier0_
  Supplier{supplierID=2, companyName='Żabka', street='Westerplatte', city='Kraków'}
  Supplier{supplierID=3, companyName='Groszek', street='Wrocławska', city='Kraków'}
  Supplier{supplierID=6, companyName='Małpka', street='Długa', city='Kraków'}
  Supplier{supplierID=9, companyName='Drewit', street='Prądnik', city='Kraków'}
```

b) Pokaż produkty sprzedane w ramach wybranej faktury/transakcji

Kod:

```
Invoice invoice = session.find(Invoice.class, o: 14);
invoice.getProducts().forEach(System.out::println);
```

Wywołanie:

```
Product{productID=16, productName='buty puma', unitsOnStock=5, supplier=Supplier{supplierID=6, companyName='Małpka', street='Długa', city='Kraków'}, category=Categ
Product{productID=17, productName='Buty adidas', unitsOnStock=5, supplier=Supplier{supplierID=6, companyName='Małpka', street='Długa', city='Kraków'}, category=Cat
```

c) Pokaż faktury w ramach których był sprzedany wybrany produkt

Kod:

```
Product product = session.find(Product.class, o: 16);
product.getInvoices().forEach(System.out::println);
```

Wywołanie:

```
Invoice{invoiceID=14, quantity=1}
```

7. Stwórz nowego maina w którym zrobisz to samo co w punkcie 4 ale z wykorzystaniem JPA.

Klasa nowego maina:

```
public class newMain {

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

        Category category = new Category( name: "Picie");
        Supplier supplier = new Supplier( companyName: "Ikea",  street: "Opolska",  city: "Krakow");

        Product product1 = new Product( productName: "Woda",  unitsOnStock: 2, supplier, category);
        Product product2 = new Product( productName: "Cydr",  unitsOnStock: 2, supplier, category);

        supplier.addProduct(product1);
        supplier.addProduct(product2);

        category.addProduct(product2);
        category.addProduct(product2);

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

Dokument 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="myDatabaseConfig"
                      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/BKustraJPA"/>
            <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>
```

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

Klasa Dostawcy:

```java
@Entity
public class Supplier {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int supplierID;
    private String companyName;
    private String street;
    private String city;

    @OneToMany(cascade = CascadeType.PERSIST)
    private Set<Product> products = new LinkedHashSet<>();

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

    public Supplier() {
    }

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

Klasa Product:

```java
public class Product {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int productID;
    private String productName;
    private int unitsOnStock;

    @ManyToOne(cascade = CascadeType.PERSIST)
    private Supplier supplier;

    @ManyToOne(cascade = CascadeType.PERSIST)
    private Category category;

    @ManyToMany(cascade = CascadeType.PERSIST)
    private Set<Invoice> invoices = new LinkedHashSet<>();

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

    public Product() {}
```

Klasa Faktury:

```java
@Entity
public class Invoice {

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

    @ManyToMany(cascade = CascadeType.PERSIST)
    private Set<Product> products = new LinkedHashSet<>();

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

    public Invoice() {}

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

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

Klasa kategorii:

```java
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int categoryID;
    private String name;

    @OneToMany(cascade = CascadeType.PERSIST)
    private Set<Product> products = new LinkedHashSet<>();

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

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

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

    @Override
    public String toString() {
        return "Category{" +
                "categoryID=" + categoryID +
                ", name='" + name + '\'' +
```

Main:

```java
        Transaction transaction = session.beginTransaction();
        Category category = session.find(Category.class,  o: 11);
        Supplier supplier = session.find(Supplier.class,  o: 6);

        Product product1 = new Product( productName: "buty puma",  unitsOnStock: 5, supplier, category);
        Product product2 = new Product( productName: "Buty adidas",  unitsOnStock: 5, supplier, category);
        Product product3 = new Product( productName: "Buty nike",  unitsOnStock: 5, supplier, category);

        Invoice invoice1 = new Invoice( quantity: 1);
        Invoice invoice2 = new Invoice( quantity: 2);

        supplier.addProduct(product1);
        supplier.addProduct(product2);
        supplier.addProduct(product3);

        category.addProduct(product1);
        category.addProduct(product2);
        category.addProduct(product3);

        invoice1.addProduct(product1);
        invoice1.addProduct(product2);
        invoice2.addProduct(product3);

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

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

Tabela produkty:

	PRODUCTID	PRODUCTNAME	UNITSONSTOCK	CATEGORY_CATEGORYID	SUPPLIER_SUPPLIERID
1	19	Buty puma	5	11	6
2	20	Buty adidas	5	11	6
3	21	Buty nike	5	11	6

9. Embedded class - Dodaj do modelu klase adres. „Wbuduj” ją do tabeli Dostawców. Zmodyfikuj model w taki sposób, że dane adresowe znajdują się w klasie dostawców. Zmapuj to do dwóch osobnych tabel.

Klasa Address:

```java
import javax.persistence.Embeddable;

@Embeddable
public class Address {

    private String street;
    private String city;

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

    public Address() {}
}
```

Klasa Dostawcy:

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

@Entity
public class Supplier {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int supplierID;
    private String companyName;

    @Embedded
    private Address address;

    @OneToMany(cascade = CascadeType.PERSIST)
    private Set<Product> products = new LinkedHashSet<>();

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

    public Supplier() {
    }

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

    @Override
```

Kod:

```
32          Address address = new Address( street: "Myczkowskiego",  city: "Krakow");
33          Supplier supplier = new Supplier( companyName: "Dekanex", address)
34          session.save(supplier);
```

Rekord w tabeli:

2	2	Krakow	Dekanex	Myczkowskiego