

# 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           | SYSDDUMMY1        |

23 wierszy wybranych
ij>
```

Skonfigurowałem config Hibernate.

```
1  <?xml version='1.0' encoding='utf-8'?>
2  <!DOCTYPE hibernate-configuration PUBLIC
3      "-//Hibernate/Hibernate Configuration DTD//EN"
4      "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
5  <hibernate-configuration>
6      <session-factory>
7          <property name="connection.url">jdbc:derby://127.0.0.1/BKustraJPA</property>
8          <property name="connection.driver_class">org.apache.derby.jdbc.ClientDriver</property>
9          <property name="dialect">org.hibernate.dialect.DerbyTenSevenDialect</property>
10         <property name="format_sql">>true</property>
11         <property name="show_sql">>true</property>
12         <property name="use_sql_comments">>true</property>
13         <property name="hibernate.hbm2ddl.auto">update</property>
14         <mapping class="Product"></mapping>
15     </session-factory>
16 </hibernate-configuration>
```

Stworzyłem klasę Product z odpowiednimi polami.

```
1  import javax.persistence.Entity;
2  import javax.persistence.GeneratedValue;
3  import javax.persistence.GenerationType;
4  import javax.persistence.Id;
5
6  @Entity
7  public class Product {
8
9      @Id
10     @GeneratedValue(strategy = GenerationType.AUTO)
11     private int productID;
12     private String productName;
13     private int unitsOnStock;
14
15     public Product(String productName, int unitsOnStock) {
16         this.productName = productName;
17         this.unitsOnStock = unitsOnStock;
18     }
19     public Product() {}
20
21     @Override
22     public String toString() {
23         return "Product{" +
24             "productID=" + productID +
25             ", productName='" + productName + '\'' +
26             ", unitsOnStock=" + unitsOnStock +
27             '}';
28     }
29 }
```

Dodałem produkt "Mleko" do bazy.

```
1  import ...
2
3  public class Main {
4      private static final SessionFactory ourSessionFactory;
5
6      static {
7          try {
8              Configuration configuration = new Configuration();
9              configuration.configure();
10
11              ourSessionFactory = configuration.buildSessionFactory();
12          } catch (Throwable ex) {
13              throw new ExceptionInInitializerError(ex);
14          }
15      }
16
17      public static Session getSession() throws HibernateException {
18          return ourSessionFactory.openSession();
19      }
20
21      public static void main(final String[] args) throws Exception {
22          final Session session = getSession();
23          try {
24              Product product = new Product("Mleko", 6);
25              Transaction transaction = session.beginTransaction();
26              session.save(product);
27              transaction.commit();
28
29              System.out.println("querying all the managed entities...");
30              final Metamodel metamodel = session.getSessionFactory().getMetamodel();
31              for (EntityType<?> entityType : metamodel.getEntities()) {
32                  final String entityName = entityType.getName();
33                  final Query query = session.createQuery("from " + entityName);
34                  System.out.println("executing: " + query.getQueryString());
35                  for (Object o : query.list()) {
36                      System.out.println("  " + o);
37                  }
38              }
39          } finally {
40              session.close();
41          }
42      }
43  }
```

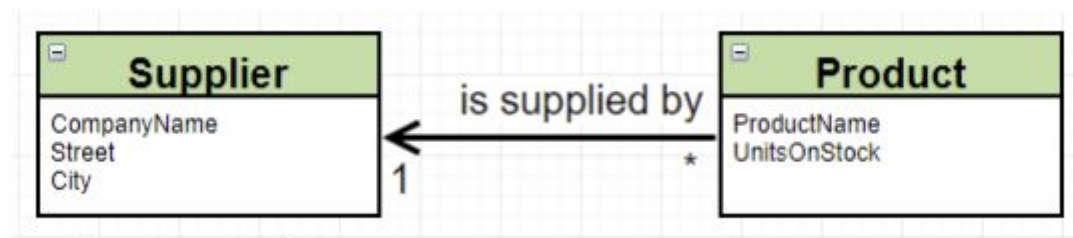
Wynik wykonania Maina.

```
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 product1_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:



- Stwórz nowego dostawcę.
- Znajdź poprzednio wprowadzony produkt i ustaw jego dostawcę na właśnie dodanego.

Klasa supplier:

```
1 import javax.persistence.Entity;
2 import javax.persistence.GeneratedValue;
3 import javax.persistence.GenerationType;
4 import javax.persistence.Id;
5
6 @Entity
7 public class Supplier {
8
9     @Id
10    @GeneratedValue(strategy = GenerationType.AUTO)
11    private int supplierID;
12    private String companyName;
13    private String street;
14    private String city;
15
16    public Supplier(String companyName, String street, String city) {
17        this.companyName = companyName;
18        this.street = street;
19        this.city = city;
20    }
21
22    public Supplier() {
23    }
24
25    @Override
26    public String toString() {
27        return "Supplier{" +
28            "supplierID=" + supplierID +
29            ", companyName='" + companyName + '\'' +
30            ", street='" + street + '\'' +
31            ", city='" + city + '\'' +
32            '}';
33    }
34 }
35
```

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

Zmiany w mainie:

```
32 Product product = session.find(Product.class, 1);
33 Supplier supplier = new Supplier( companyName: "Żabka", street: "Westerplatte", city: "Kraków");
34 product.setSupplier(supplier);
35 session.save(product);
36 session.save(supplier);
37 transaction.commit();
```

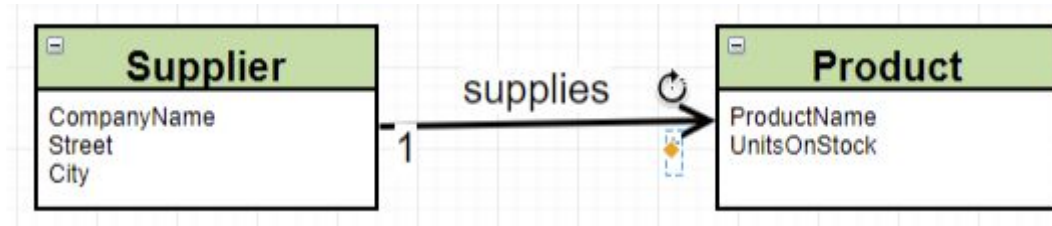
tabela produktów:

	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



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

Klasa Product:

```
1  import javax.persistence.*;
2
3  @Entity
4  public class Product {
5
6      @Id
7      @GeneratedValue(strategy = GenerationType.AUTO)
8      private int productID;
9      private String productName;
10     private int unitsOnStock;
11
12     public Product(String productName, int unitsOnStock) {
13         this.productName = productName;
14         this.unitsOnStock = unitsOnStock;
15     }
16
17     public Product() {}
18
19     @Override
20     public String toString() {
21         return "Product{" +
22             "productID=" + productID +
23             ", productName='" + productName + '\'' +
24             ", unitsOnStock=" + unitsOnStock +
25             '}' ;
26     }
27 }
```

## Klasa Dostawcy:

```

1  import javax.persistence.*;
2  import java.util.LinkedHashMap;
3  import java.util.*;
4
5  @Entity
6  public class Supplier {
7
8      @Id
9      @GeneratedValue(strategy = GenerationType.AUTO)
10     private int supplierID;
11     private String companyName;
12     private String street;
13     private String city;
14
15     @OneToMany
16     private Set<Product> products = new LinkedHashSet<>();
17
18     public Supplier(String companyName, String street, String city) {
19         this.companyName = companyName;
20         this.street = street;
21         this.city = city;
22     }
23
24     public Supplier() {
25     }
26
27     public void addProduct(Product product) {
28         this.products.add(product);
29     }
30
31     @Override
32     public String toString() {
33         return "Supplier{" +
34             "supplierID=" + supplierID +
35             ", companyName='" + companyName + '\'' +
36             ", street='" + street + '\'' +
37             ", city='" + city + '\'' +
38             '}';
39     }
40 }

```

## Klasa main:

```

30     Transaction transaction = session.beginTransaction();
31
32     Product product1 = new Product( productName: "Woda", unitsOnStock: 100);
33     Product product2 = new Product( productName: "Maseczki", unitsOnStock: 10);
34
35     Supplier supplier = new Supplier( companyName: "Groszek", street: "Wrocławska", city: "Kraków");
36
37     supplier.addProduct(product1);
38     supplier.addProduct(product2);
39     session.save(supplier);
40     session.save(product1);
41     session.save(product2);
42     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

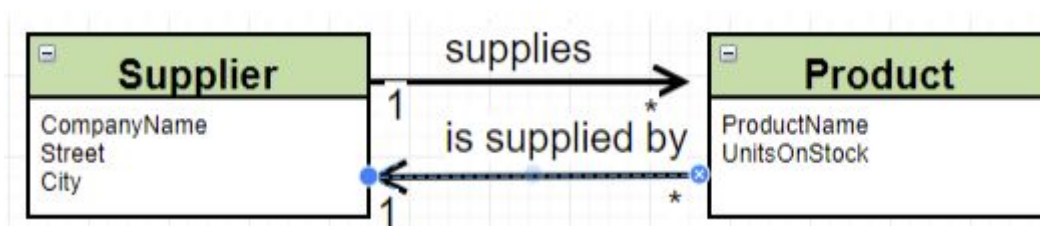
Uruchomienie maina:

```

(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 product1_0_,
product0_.productName as product2_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 relację dwustronną jak poniżej:



- Tradycyjnie: Stwórz kilka produktów
- Dodaj je do produktów dostarczanych przez nowo stworzonego dostawcę (pamiętaj o poprawnej obsłudze dwustronności relacji)
- Udokumentuj wykonane kroki oraz uzyskane rezultaty (logi wywołań sqlowych, describe table/diagram z datagrip, select \* from....)

## Klasa Product:

```
1  import javax.persistence.*;
2
3  @Entity
4  public class Product {
5
6      @Id
7      @GeneratedValue(strategy = GenerationType.AUTO)
8      private int productID;
9      private String productName;
10     private int unitsOnStock;
11
12     @ManyToOne
13     private Supplier supplier;
14
15
16     public Product(String productName, int unitsOnStock, Supplier supplier) {
17         this.productName = productName;
18         this.unitsOnStock = unitsOnStock;
19         this.supplier = supplier;
20     }
21
22     public Product() {}
23
24     public void setSupplier(Supplier supplier) {
25         this.supplier = supplier;
26     }
27
28     @Override
29     public String toString() {
30         return "Product{" +
31             "productID=" + productID +
32             ", productName=" + productName + '\n' +
33             ", unitsOnStock=" + unitsOnStock +
34             '}';
35     }
36 }
```

## Klasa Dostawcy:

```
1  import javax.persistence.*;
2  import java.util.LinkedHashMap;
3  import java.util.*;
4
5  @Entity
6  public class Supplier {
7
8      @Id
9      @GeneratedValue(strategy = GenerationType.AUTO)
10     private int supplierID;
11     private String companyName;
12     private String street;
13     private String city;
14
15     @OneToMany
16     private Set<Product> products = new LinkedHashSet<>();
17
18     public Supplier(String companyName, String street, String city) {
19         this.companyName = companyName;
20         this.street = street;
21         this.city = city;
22     }
23
24     public Supplier() {}
25
26
27     public void addProduct(Product product) {
28         this.products.add(product);
29         product.setSupplier(this);
30     }
31
32     @Override
33     public String toString() {
34         return "Supplier{" +
35             "supplierID=" + supplierID +
36             ", companyName=" + companyName + '\n' +
37             ", street=" + street + '\n' +
38             ", city=" + city + '\n' +
39             '}';
40     }
41 }
```



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 product1_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 klasę Category z property int CategoryID, String Name oraz listą produktów List Products

- Zmodyfikuj produkty dodając wskazanie na kategorie do której należy.
- Stwórz kilka produktów i kilka kategorii
- Dodaj kilka produktów do wybranej kategorii

Klasa Category (dodany mapping w configu):

```
1  import javax.persistence.*;
2  import java.lang.reflect.Array;
3  import java.util.*;
4
5  @Entity
6  public class Category {
7
8      @Id
9      @GeneratedValue(strategy = GenerationType.AUTO)
10     private int categoryID;
11     private String name;
12
13     @OneToMany
14     private Set<Product> products = new LinkedHashSet<>();
15
16     public Category(String name) {
17         this.name = name;
18     }
19
20     public Set<Product> getProducts() {
21         return products;
22     }
23
24     public void addProduct(Product product) {
25         this.products.add(product);
26         product.setCategory(this);
27     }
28
29     public Category() {}
30
31 }
```

## Klasa Product:

```
1  import javax.persistence.*;
2
3  @Entity
4  public class Product {
5
6      @Id
7      @GeneratedValue(strategy = GenerationType.AUTO)
8      private int productID;
9      private String productName;
10     private int unitsOnStock;
11
12     @ManyToOne
13     private Supplier supplier;
14
15     @ManyToOne
16     private Category category;
17
18     public Product(String productName, int unitsOnStock, Supplier supplier, Category category) {
19         this.productName = productName;
20         this.unitsOnStock = unitsOnStock;
21         this.supplier = supplier;
22         this.category = category;
23     }
24
25     public Product() {}
26
27     public void setSupplier(Supplier supplier) {
28         this.supplier = supplier;
29     }
30
31     public void setCategory(Category category) {
32         this.category = category;
33     }
34
35     @Override
36     public String toString() {
37         return "Product{" +
38             "productID=" + productID +
39             ", productName='" + productName + '\'' +
40             ", unitsOnStock=" + unitsOnStock +
41             '}';
42     }
43 }
```

## Zmiany maina:

```
30 Transaction transaction = session.beginTransaction();
31 Supplier supplier = new Supplier( companyName: "Drewit", street: "Pradnik", city: "Kraków");
32 Category category1 = new Category( name: "Meble");
33 Category category2 = new Category( name: "Ubrania");
34
35 Product product1 = new Product( productName: "Bjurko", unitsOnStock: 1, supplier, category1);
36 Product product2 = new Product( productName: "Buty adidas", unitsOnStock: 5, supplier, category2);
37
38 supplier.addProduct(product1);
39 supplier.addProduct(product2);
40
41 category1.addProduct(product1);
42 category2.addProduct(product2);
43
44 session.save(supplier);
45 session.save(category1);
46 session.save(category2);
47 session.save(product1);
48 session.save(product2);
49 transaction.commit();
```

Wykonanie maina:

```
Product */ select
    product0_.productID as product1_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='Matpka', 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	Mebles
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:

```

52 Category category = session.find(Category.class, o: 10);
53
54 Query q = session.createQuery( s: "FROM Product P WHERE P.category = " + category);
55
56 System.out.println(category);
57
58 for (Object o : q.list()) {
59     System.out.println(" " + o);
60 }

```

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

Kategoria produktu 12:

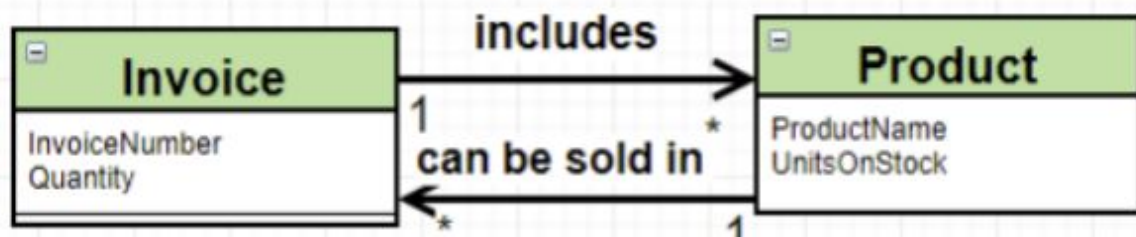
```

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

```
1  import javax.persistence.*;
2  import java.util.*;
3
4  @Entity
5  public class Invoice {
6
7      @Id
8      @GeneratedValue(strategy = GenerationType.AUTO)
9      private int invoiceID;
10     private int quantity;
11
12     @ManyToMany
13     private Set<Product> products = new LinkedHashSet<>();
14
15     public Invoice(int quantity) {
16         this.quantity = quantity;
17     }
18
19     public Invoice() {}
20
21     public void addProduct(Product product) {
22         this.products.add(product);
23         product.addInvoice(this);
24     }
25
26     @Override
27     public String toString() {
28         return "Invoice{" +
29             "invoiceID=" + invoiceID +
30             ", quantity=" + quantity +
31             '}';
32     }
33 }
```

Zmiany w klasie Product:

```
41     @ManyToMany
42     private Set<Invoice> invoices = new LinkedHashSet<>();
43
44     public void addInvoice(Invoice invoice) {
45         this.invoices.add(invoice);
46     }
47 }
```



Zmiany w mainie:

```
28 public static void main(final String[] args) throws Exception {
29     final Session session = getSession();
30     try {
31         Transaction transaction = session.beginTransaction();
32         Category category = session.find(Category.class, 10);
33         Supplier supplier = session.find(Supplier.class, 6);
34
35         Product product1 = new Product( productName: "buty puma", unitsOnStock: 5, supplier, category);
36         Product product2 = new Product( productName: "Buty adidas", unitsOnStock: 5, supplier, category);
37         Product product3 = new Product( productName: "Buty nike", unitsOnStock: 5, supplier, category);
38
39         Invoice invoice1 = new Invoice( quantity: 1);
40         Invoice invoice2 = new Invoice( quantity: 2);
41
42         supplier.addProduct(product1);
43         supplier.addProduct(product2);
44         supplier.addProduct(product3);
45
46         category.addProduct(product1);
47         category.addProduct(product2);
48         category.addProduct(product3);
49
50         invoice1.addProduct(product1);
51         invoice1.addProduct(product2);
52         invoice2.addProduct(product3);
53
54         session.save(supplier);
55         session.save(category);
56
57         session.save(invoice1);
58         session.save(invoice2);
59
60         session.save(product1);
61         session.save(product2);
62         session.save(product3);
63
64         transaction.commit();
```

Wywołanie maina:

```
Product #/ select
product0_.productId as product1_4_,
product0_.categoryId as category4_4_,
product0_.productName as product2_4_,
product0_.supplierId as supplier5_4_,
product0_.unitsOnStock as units3_4_
from
    Product product0_
Hibernate:
select
    supplier0_.supplierId as supplier1_6_0_,
    supplier0_.city as city2_6_0_,
    supplier0_.companyName as company3_6_0_,
    supplier0_.street as street4_6_0_
from
    Supplier supplier0_
where
    supplier0_.supplierId=7
Product(productID=1, productName='Mleko', unitsOnStock=6, supplier=Supplier{supplierID=2, companyName='Żabka', street='Westerplatte', city='Kraków'}, category=null)
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='Biorzo', unitsOnStock=1, supplier=Supplier{supplierID=9, companyName='Drewit', street='Prądnik', city='Kraków'}, category=Cat)
Product(productID=13, productName='Buty adidas', unitsOnStock=5, supplier=Supplier{supplierID=9, companyName='Drewit', street='Prądnik', city='Kraków'}, category=Cat)
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=Cat)
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 invoice1_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 company3_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:

```
77 Invoice invoice = session.find(Invoice.class, o: 14);  
78 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:

```
77 Product product = session.find(Product.class, o: 16);  
78 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("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:

```
1  <?xml version="1.0"?>
2  <persistence xmlns="http://java.sun.com/xml/ns/persistence"
3             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4             xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
5             http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd"
6             version="2.0">
7      <persistence-unit name="myDatabaseConfig"
8                      transaction-type="RESOURCE_LOCAL">
9
10         <properties>
11             <property name="hibernate.connection.driver_class"
12                     value="org.apache.derby.jdbc.ClientDriver"/>
13             <property name="hibernate.connection.url"
14                     value="jdbc:derby://127.0.0.1/BKustrajPA"/>
15             <property name="hibernate.show_sql" value="true"/>
16             <property name="hibernate.format_sql" value="true"/>
17             <property name="hibernate.hbm2ddl.auto" value="create"/>
18         </properties>
19     </persistence-unit>
20 </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:

```
5  @Entity
6  public class Supplier {
7
8      @Id
9      @GeneratedValue(strategy = GenerationType.AUTO)
10     private int supplierID;
11     private String companyName;
12     private String street;
13     private String city;
14
15     @OneToMany(cascade = CascadeType.PERSIST)
16     private Set<Product> products = new LinkedHashSet<>();
17
18     public Supplier(String companyName, String street, String city) {
19         this.companyName = companyName;
20         this.street = street;
21         this.city = city;
22     }
23
24     public Supplier() {
25     }
26
27     public void addProduct(Product product) {
28         this.products.add(product);
29         product.setSupplier(this);
30     }
31 }
```

### Klasa Product:

```
6 public class Product {
7
8     @Id
9     @GeneratedValue(strategy = GenerationType.AUTO)
10    private int productId;
11    private String productName;
12    private int unitsOnStock;
13
14    @ManyToOne(cascade = CascadeType.PERSIST)
15    private Supplier supplier;
16
17    @ManyToOne(cascade = CascadeType.PERSIST)
18    private Category category;
19
20    @ManyToMany(cascade = CascadeType.PERSIST)
21    private Set<Invoice> invoices = new LinkedHashSet<>();
22
23    public Product(String productName, int unitsOnStock, Supplier supplier, Category category) {
24        this.productName = productName;
25        this.unitsOnStock = unitsOnStock;
26        this.supplier = supplier;
27        this.category = category;
28    }
29
30    public Product() {}
31}
```

### Klasa Faktury:

```
4 @Entity
5 public class Invoice {
6
7     @Id
8     @GeneratedValue(strategy = GenerationType.AUTO)
9     private int invoiceID;
10    private int quantity;
11
12    @ManyToMany(cascade = CascadeType.PERSIST)
13    private Set<Product> products = new LinkedHashSet<>();
14
15    public Invoice(int quantity) {
16        this.quantity = quantity;
17    }
18
19    public Invoice() {}
20
21    public void addProduct(Product product) {
22        this.products.add(product);
23        product.addInvoice(this);
24    }
25
26    public Set<Product> getProducts() {
27        return products;
28    }
29}
```

Klasa kategorii:

```

9      @Id
10     @GeneratedValue(strategy = GenerationType.AUTO)
11     private int categoryID;
12     private String name;
13
14     @OneToMany(cascade = CascadeType.PERSIST)
15     private Set<Product> products = new LinkedHashSet<>();
16
17     public Category(String name) {
18         this.name = name;
19     }
20
21     public Set<Product> getProducts() {
22         return products;
23     }
24
25     public void addProduct(Product product) {
26         this.products.add(product);
27         product.setCategory(this);
28     }
29
30     @Override
31     public String toString() {
32         return "Category{" +
33             "categoryID=" + categoryID +
34             ", name='" + name + '\'' +

```

Main:

```

3      Transaction transaction = session.beginTransaction();
4      Category category = session.find(Category.class, 11);
5      Supplier supplier = session.find(Supplier.class, 6);
6
7      Product product1 = new Product("Buty puma", 5, supplier, category);
8      Product product2 = new Product("Buty adidas", 5, supplier, category);
9      Product product3 = new Product("Buty nike", 5, supplier, category);
10
11      Invoice invoice1 = new Invoice(1);
12      Invoice invoice2 = new Invoice(2);
13
14      supplier.addProduct(product1);
15      supplier.addProduct(product2);
16      supplier.addProduct(product3);
17
18      category.addProduct(product1);
19      category.addProduct(product2);
20      category.addProduct(product3);
21
22      invoice1.addProduct(product1);
23      invoice1.addProduct(product2);
24      invoice2.addProduct(product3);
25
26      session.save(supplier);
27      session.save(category);
28
29      session.save(invoice1);
30      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 klasę 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:

```
1  import javax.persistence.Embeddable;
2
3  @Embeddable
4  public class Address {
5
6      private String street;
7      private String city;
8
9      public Address(String street, String city) {
10         this.street = street;
11         this.city = city;
12     }
13
14     public Address() {}
15 }
16
```

Klasa Dostawcy:

```
1  import javax.persistence.*;
2  import java.util.LinkedHashMap;
3  import java.util.*;
4
5  @Entity
6  public class Supplier {
7
8      @Id
9      @GeneratedValue(strategy = GenerationType.AUTO)
10     private int supplierID;
11     private String companyName;
12
13     @Embedded
14     private Address address;
15
16     @OneToMany(cascade = CascadeType.PERSIST)
17     private Set<Product> products = new LinkedHashSet<>();
18
19     public Supplier(String companyName, Address address) {
20         this.companyName = companyName;
21         this.address = address;
22     }
23
24     public Supplier() {
25     }
26
27     public void addProduct(Product product) {
28         this.products.add(product);
29         product.setSupplier(this);
30     }
31
32     @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
---	---	--------	---------	---------------