Andrii Trishch
JPA LAB

Zadanie 1-3

1. Udało się uruchomić server i do niego podłaczyć się

```
ij> connect 'jdbc:derby://127.0.0.1/ATrishchJPA;';
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                     |
APP                 |PRODUCT                       |

24 rows selected
ij>
```

2. Stworzona klasa Product z polami ProductID,ProductName oraz UnitsOnStock

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

//Required by Hibernate
@Entity
public class Product {
    @Id
    @GeneratedValue(
            strategy = GenerationType.AUTO)
    private int dbID;
    int productID;
    String productsName;
    int unitsOnStock;

    public Product() {

    }

    public Product(String productsName) {
        this.productsName = productsName;
        this.unitsOnStock = 0;
    }

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

3. Został zmieniony config

```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/ATrishchJPA</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>
        <!-- DB schema will be updated if needed -->
        <property name="hibernate.hbm2ddl.auto">update</property>
        <mapping class="Product"></mapping>
    </session-factory>
</hibernate-configuration>
```

4. Został zmieniony Main dla tworzenia nowego produktu, i obserwowania skutku wprowadzonych modyfikacij

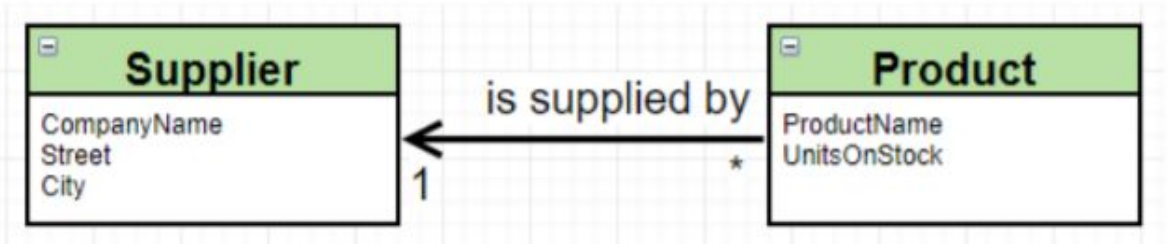
```java
public static void main(final String[] args) throws Exception {
    Product product=new Product( productsName: "Chips", unitsOnStock: 24);
    Session session=ourSessionFactory.openSession();
    Transaction tx=session.beginTransaction();
    session.save(product);
    tx.commit();
    try {
        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();
    }
```

5. Wygląd z poziomu DataGrip

	DBID	PRODUCTID	PRODUCTSNAME	UNITSONSTOCK
1	1	0	Chips	0
2	4	0	Chips	24

Zadanie 4

Zmodyfikuj model wprowadzając pojęcie Dostawcy jak poniżej



1. Stworzona klasa Suppliers

```
import javax.persistence.*;



//Required by Hibernate
@Entity
public class Supplier {
    @Id
    @GeneratedValue(
            strategy = GenerationType.AUTO)
    private int dbID;
    String companyName;
    String street;
    String city;
    Supplier(){}
    Supplier(String companyName,String street,String city){
        this.companyName=companyName;
        this.street=street;
        this.city=city;
    }
}
```

2. Została zmodyfikowana klasa Product

```
//Required by Hibernate
@Entity
public class Product {
    @Id
    @GeneratedValue(
            strategy = GenerationType.AUTO)
    private int dbID;
    int productID;
    String productsName;
    int unitsOnStock;
    @ManyToOne
    public Supplier supplier;

    public Product() {

    }

    public Product(String productsName) {
        this.productsName = productsName;
        this.unitsOnStock = 0;
    }

    public Product(String productsName, int unitsOnStock) {
        this.productsName = productsName;
        this.unitsOnStock = unitsOnStock;
    }
    public void setSupplier(Supplier supplier){
        this.supplier=supplier;
    }
}
```

3. Z configu dodana klasa Supplier

```
<hibernate-configuration>
    <session-factory>
        <property name="connection.url">jdbc:derby://127.0.0.1/ATrishchJPA</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>
        <!-- DB schema will be updated if needed -->
        <property name="hibernate.hbm2ddl.auto">update</property>
        <mapping class="Product"></mapping>
        <mapping class="Supplier"></mapping>
    </session-factory>
</hibernate-configuration>
```

4. W mainie dodałem nowego Supplier i Update Customera

```
public static void main(final String[] args) throws Exception {
    Session session=ourSessionFactory.openSession();
    Supplier supplier=new Supplier( companyName: "Google", street: "Johns Street", city: "London");
    Product productToUpdate=session.get(Product.class, serializable: 1);
    productToUpdate.setSupplier(supplier);
    Transaction tx=session.beginTransaction();
    session.save(supplier);
    session.save(productToUpdate);
    tx.commit();
    try {
        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();
    }
}
```

5. Wygląd z poziomu DataGrip

	DBID	CITY	COMPANYNAME	STREET	SUPPLIERID
1	101	London	Google	Johns Street	0

	DBID	PRODUCTID	PRODUCTSNAME	UNITSONSTOCK	SUPPLIER_DBID
1	1	0	Chips	0	101

Zadanie 5

Odwróć relacje zgodnie z poniższym schematem

1. Po obserwowaniu zmian, zauważyłem że tworzenie productID i supplierID przy obecności dbID jest bezużyteczne przez to w następnej implementacji zostało usunięte.

2. Do klasy Supplier dodano związek @OneToMany, który pozwała wykonać powyżej zadanie. Też dodano metodę addProduct() dla dodawania produktów.

```
@Entity
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int dbID;
    String companyName;
    String street;
    String city;
    @OneToMany
    private Set<Product> products = new HashSet<>();

    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); }
}
```

3. Został zmieniony Main dla obserwowania wprowadzonych zmian.

```java
public static void main(final String[] args) throws Exception {
    Session session = ourSessionFactory.openSession();
    try {
        Transaction tx = session.beginTransaction();
        Supplier supplier = new Supplier( companyName: "Amazon",  street: "Gandry",  city: "London");
        Product product = new Product( productsName: "Apple",  unitsOnStock: 100);
        Product product1 = new Product( productsName: "Water",  unitsOnStock: 10);
        supplier.addProduct(product);
        supplier.addProduct(product1);
        session.save(product);
        session.save(product1);
        session.save(supplier);
        tx.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();
    }
}
```

4. Wygląd z poziomu DataGrip

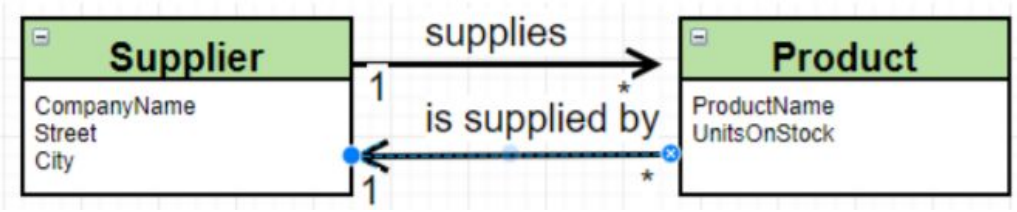
	DBID	CITY	COMPANYNAME	STREET
1	9	London	Amazon	Gandry
2	112	Krakow	Amazon	Gandry
3	115	Lviv	Amazon	Gandry

	DBID	PRODUCTSNAME	UNITSONSTOCK
1	110	Monitor	100
2	111	Bacon	10
3	113	M4A1	100
4	114	Water	10
5	7	Apple	100
6	8	Connector	10

Tabela łącznikowa.

	SUPPLIER_DBID	PRODUCTS_DBID
1	9	7
2	9	8
3	112	110
4	112	111
5	115	113
6	115	114

Zadanie 6



1. Dodano poszczególne związki przy pomocy @OneToMany i @ManyToOne
2. Klasa Supplier

```
@Entity
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int dbID;
    String companyName;
    String street;
    String city;
    @OneToMany
    @JoinColumn(name = "Product_FK")
    private Set<Product> products = new HashSet<>();

    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); }
}
```

3. Klasa Product

```java
@Entity
public class Product {
    @Id
    @GeneratedValue(
            strategy = GenerationType.AUTO)
    private int dbID;
    String productsName;
    int unitsOnStock;

    @ManyToOne
    @JoinColumn(name = "Supplier_FK")
    private Supplier supplier;

    public Product() {

    }

    public Product(String productsName) {
        this.productsName = productsName;
        this.unitsOnStock = 0;
    }

    public Product(String productsName, int unitsOnStock) {
        this.productsName = productsName;
        this.unitsOnStock = unitsOnStock;
    }
    public void setSupplier(Supplier supplier) { this.supplier=supplier; }
}
```

4. Wygład z poziomu DataGrip

	DBID	PRODUCTSNAME	UNITSONSTOCK	SUPPLIER_FK	PRODUCT_FK 1
1	214	Water	15	215	215
2	213	Apple	20	215	215

	DBID	CITY	COMPANYNAME	STREET
1	215	City1	Facebook	Mazepy
2	9	London	Amazon	Gandry
3	112	Krakow	Amazon	Gandry
4	115	Lviv	Amazon	Gandry

Zadanie 7

VII. Dodaj klase Category z property int CategoryID, String Name oraz listą produktow List<Product> Products

1. Została dodana klasa Category

```
@Entity
public class Category {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int dbID;
    private String name;
    @OneToMany
    @JoinColumn(name = "Product_FK")
    Set<Product> products = new HashSet<>();

    public Category() {
    }

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

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

2. Dodanie produktów z Main`a

```java
public static void main(final String[] args) throws Exception {
    Session session = ourSessionFactory.openSession();
    try {
        Transaction tx = session.beginTransaction();
        Category category = new Category( name: "Electronic");
        Product product = new Product( productsName: "iPhone",  unitsOnStock: 12);
        Product product1 = new Product( productsName: "AppleWatch",  unitsOnStock: 10);
        category.addProduct(product);
        category.addProduct(product1);
        session.save(product);
        session.save(product1);
        session.save(category);
        tx.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();
    }
}
```

3. Widok z poziomu DataGrip.

Product

	DBID	PRODUCTSNAME	UNITSONSTOCK	SUPPLIER_FK	PRODUCT_FK
1	413	iPhone	12	<null>	415
2	414	AppleWatch	10	<null>	415

Category

	DBID	NAME
1	415	Electronic

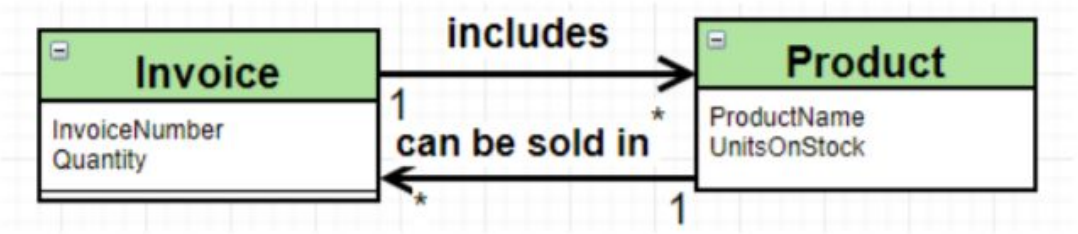
4. Modyfikacja Main dla wywietlenia wszystkich Produktów

```
public static void main(final String[] args) throws Exception {
    Session session = ourSessionFactory.openSession();
    try {
        Category category = session.find(Category.class,  o: 415);
        for (Product p : category.getProducts()) {
            System.out.println(p);
        }
        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();
    }
}
```

5. Wynik:

```
Name --> iPhone Category -->Electronic Units --> 12
Name --> AppleWatch Category -->Electronic Units --> 10
```

Zadanie 8



1. Została utworzona klasa Invoice z polami invoiceNumber oraz quantity

```java
@Entity
public class Invoice {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int dbID;
    private int quantity;

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

    public Invoice() {
    }

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

    public int getInvoiceNumber() { return dbID; }

    public int getQuantity() { return quantity; }

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

    public void addProduct(Product product) {
        products.add(product);
        product.getInvoices().add(this);
        this.quantity++;
    }
}
```

2. Została zmieniona klasa Product

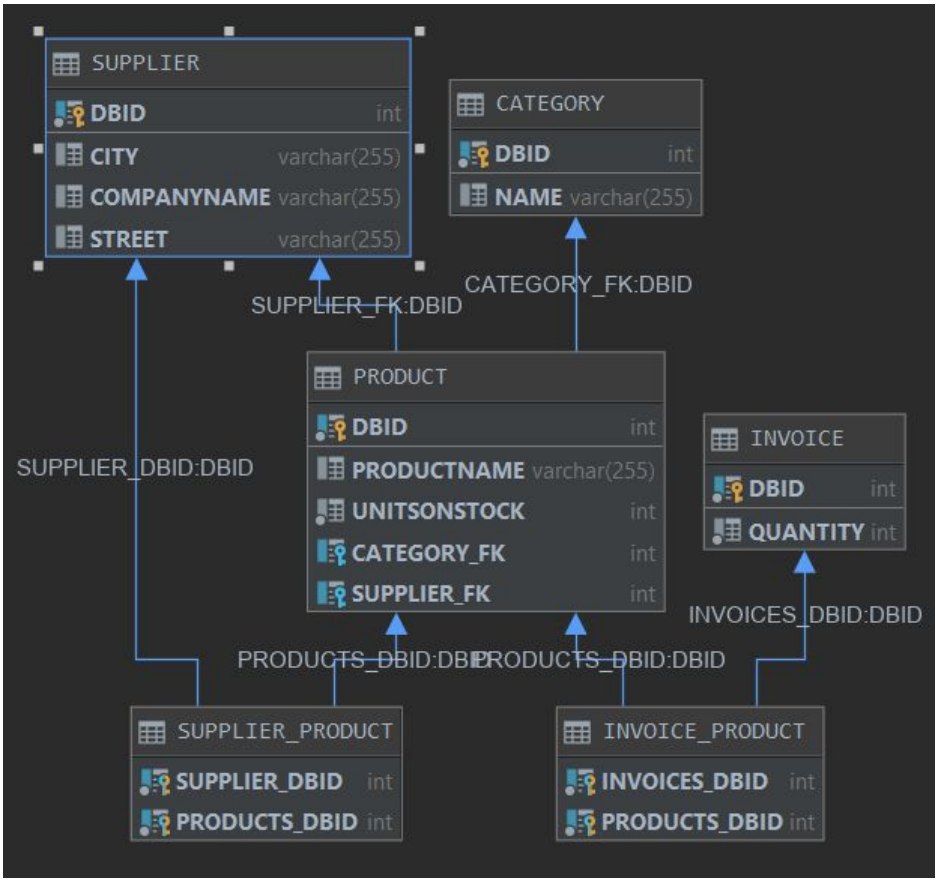
```java
@Entity
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int dbID;
    private String productName;
    private int unitsOnStock;

    @ManyToOne
```

```java
    @JoinColumn(name = "Supplier_FK")
    private Supplier supplier;

    @ManyToOne
    @JoinColumn(name = "Category_FK")
    private Category category;

    @ManyToMany(mappedBy = "products")
    private Set<Invoice> invoices = new HashSet<>();

    public Product() {
    }

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

    public String getProductName() {
        return productName;
    }

    public void setSupplier(Supplier supplier) {
        this.supplier = supplier;
        if (!supplier.getProducts().contains(this)) {
            supplier.addProduct(this);
        }
    }

    public Category getCategory() {
        return category;
    }

    public void setCategory(Category category) {
        this.category = category;
        if (!category.getProducts().contains(this)) {
            category.addProduct(this);
        }
    }

    public Set<Invoice> getInvoices() {
        return invoices;
    }
}
```

3. Została zmodyfikowana klasa Main dla wprowadzenia nowych danych

```java
public static void main(final String[] args) throws Exception {
    Session session = getSession();
    Transaction transaction = session.beginTransaction();
    Product product = new Product( productName: "Milk",  unitsOnStock: 45);
    Product product1 = new Product( productName: "Yogurt",  unitsOnStock: 76);
    Product product2 = new Product( productName: "Crisps",  unitsOnStock: 34);
    Supplier supplier = new Supplier( companyName: "ProductsEco",  street: "Reymonta",  city: "Kraków");
    Category category = new Category( name: "Food");
    Invoice invoice = new Invoice( quantity: 0);
    supplier.addProduct(product);
    supplier.addProduct(product1);
    supplier.addProduct(product2);
    category.addProduct(product);
    category.addProduct(product1);
    category.addProduct(product2);
    invoice.addProduct(product);
    invoice.addProduct(product1);
    invoice.addProduct(product2);

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

    session.save(supplier);
    session.save(category);
    session.save(invoice);
    transaction.commit();
```

4. Wizualizacja wprowadzonych zmian



5. Pokaż produkty sprzedane w ramach wybranej faktury/transakcji

- Dodano methodę to String do klasy Invoice

```
@Override
public String toString() {
    String result = "Products from invoice nr." + this.dbID + "\n";
    for (Product p : products) {
        result += p.getProductName() + "\n";
    }
    return result;
}
```

- Main

```
public static void main(final String[] args) throws Exception {
    Session session = getSession();
    Invoice invoice = session.find(Invoice.class, o: 829);
    invoice.toString();
    try {
```

- Wynik

```
Products from invoice nr.829
Milk
Yogurt
Crisps
```

6.Pokaż faktury w ramach których był sprzedany wybrany produkt

- Main

```
Session session = getSession();
Product product=session.find(Product.class, o: 824);
product.getInvoices().toString();
try {
```

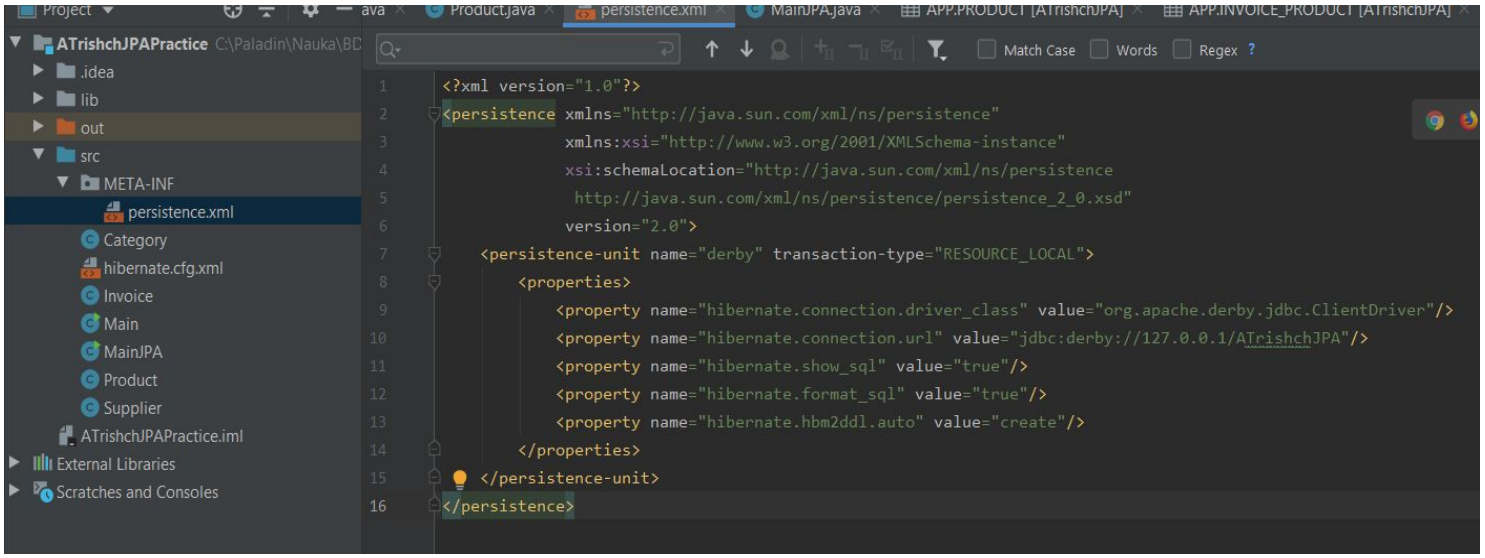
- Wynik

```
Products from invoice nr.829
Yogurt
Milk
Crisps
```

JPA

Zadanie 10

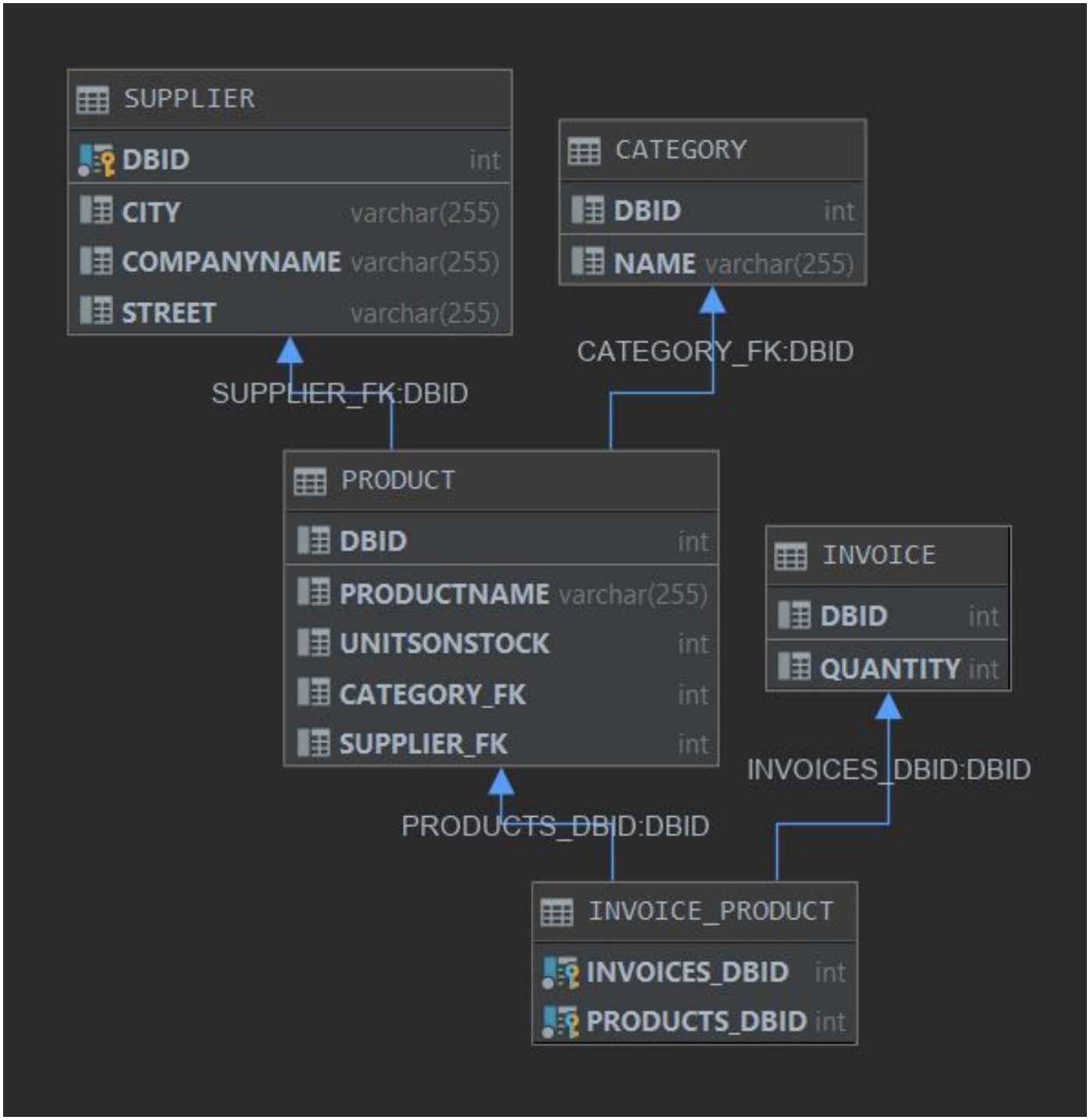
1. W folderze *META-INF* został stworzony plik persistence.xml



2. I stworzona klasa MainJPA

```
public class MainJPA {
    private static EntityManagerFactory entityManagerFactory;

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

    public static void main(String[] argv) {
        EntityManager entityManager = getEntityManager();
        EntityTransaction entityTransaction = entityManager.getTransaction();
        entityTransaction.begin();

        Product product1 = new Product( productName: "Water",  unitsOnStock: 23);
        Supplier supplier = new Supplier( companyName: "Supplier",  street: "Mazepy",  city: "Krakow");

        supplier.addProduct(product1);

        entityManager.persist(product1);
        entityManager.persist(supplier);

        entityTransaction.commit();
        entityManager.close();
    }
}
```

3.Pozostałe klasy nie zostały modyfikowane. Wynik:



Kaskady

Zadanie 11

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

1. Klasa Product

```
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int dbID;
    private String productName;
    private int unitsOnStock;

    @ManyToOne(cascade = CascadeType.PERSIST)
    @JoinColumn(name = "Supplier_FK")
    private Supplier supplier;

    @ManyToOne(cascade = CascadeType.PERSIST)
    @JoinColumn(name = "Category_FK")
    private Category category;

    @ManyToMany(mappedBy = "products",cascade = CascadeType.PERSIST)
    private Set<Invoice> invoices = new HashSet<>();

    public Product() {
    }

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

    public String getProductName() {
        return productName;
    }

    public void setSupplier(Supplier supplier) {
        this.supplier = supplier;
        if (!supplier.getProducts().contains(this)) {
            supplier.addProduct(this);
        }
    }
```

```java
    public Category getCategory() {
        return category;
    }

    public void setCategory(Category category) {
        this.category = category;
        if (!category.getProducts().contains(this)) {
            category.addProduct(this);
        }
    }

    public Set<Invoice> getInvoices() {
        return invoices;
    }
}
```

2. Klasa Supplier

```java
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int dbID;
    private String companyName;
    private String street;
    private String city;
    @OneToMany(mappedBy = "supplier")
    private Set<Product> products = new HashSet<>();

    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); }

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

3. Klasa Category

```java
@Entity
public class Category {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int dbID;
    private String name;
    @OneToMany(cascade=CascadeType.PERSIST)
    @JoinColumn(name = "Category_FK")
    private Set<Product> products = new HashSet<>();

    public Category() {
    }

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

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

    public String getName() {
        return this.name;
    }

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

4. Klasa Invoice

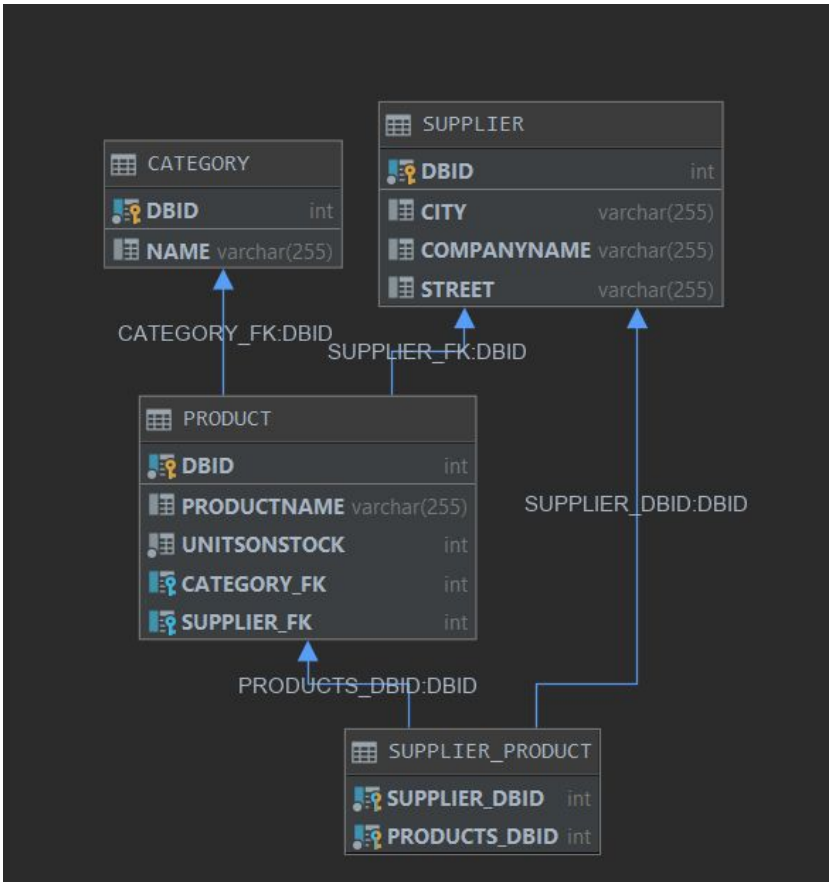
```java
@Entity
public class Invoice {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int dbID;
    private int quantity;

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

    public Invoice() {
    }

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

    public int getInvoiceNumber() { return dbID; }

    public int getQuantity() { return quantity; }

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

    public void addProduct(Product product) {
        products.add(product);
        product.getInvoices().add(this);
        this.quantity++;
    }
```

5. Modykacja Main dla obserwacji wprowadzonych zmian

```java
public static void main(String[] argv) {
    EntityManager entityManager = getEntityManager();
    EntityTransaction entityTransaction = entityManager.getTransaction();
    entityTransaction.begin();

    Product newProduct1 = new Product( productName: "Milk",  unitsOnStock: 45);
    Product newProduct2 = new Product( productName: "Yogurt",  unitsOnStock: 76);

    Supplier newSupplier = new Supplier( companyName: "Grocery store",  street: "Reymonta",  city: "Kraków");
    Category newCategory = new Category( name: "Food");
    Invoice newInvoice1 = new Invoice( quantity: 0);

    newSupplier.addProduct(newProduct1);
    newSupplier.addProduct(newProduct2);

    newCategory.addProduct(newProduct1);
    newCategory.addProduct(newProduct2);

    newInvoice1.addProduct(newProduct1);
    newInvoice1.addProduct(newProduct2);

    entityManager.persist(newSupplier);
    entityManager.persist(newCategory);
    entityManager.persist(newInvoice1);

    entityTransaction.commit();
    entityManager.close();
}
```

6. Wyniki:

- Diagrama



- Supplier

	DBID	CITY	COMPANYNAME	STREET
1	2	Anywhere	Amazon	Mazepy
2	1	Kraków	Grocery store	Reymonta

- Product

	DBID	PRODUCTNAME	UNITSONSTOCK	CATEGORY_FK	SUPPLIER_FK
1	1	Water	23	2	1
2	3	Milk	45	2	1
3	5	Yogurt	76	2	2

- Invoices

	DBID	QUANTITY
1	4	2

- Category

	DBID	NAME
1	2	Food

Embedded class

Zadanie 12

Część 1

1. Została stworzona klasa Address

```java
@Embeddable
class Address {
    private String country;
    private String city;
    private String street;
    private String zipCode;

    public Address() {
    }

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

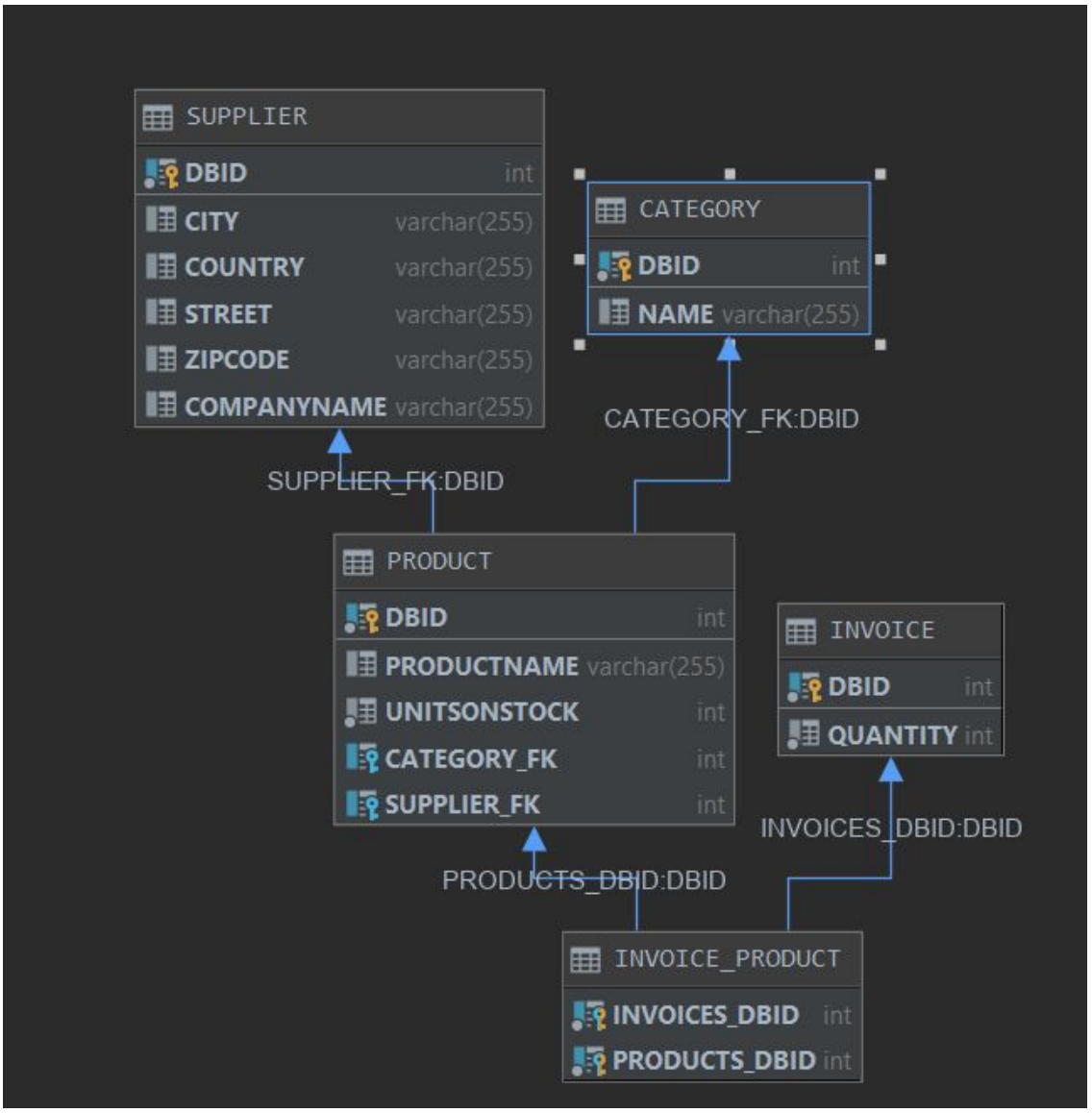
2. Klasa Supplier

```java
@Entity
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int dbID;
    private String companyName;

    @Embedded
    private Address address;

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

    public Supplier() {
    }

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

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

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

3. Dla obserwacji wprowadzonych zmian

```java
    public static void main(String[] argv) {
        EntityManager entityManager = getEntityManager();
        EntityTransaction entityTransaction = entityManager.getTransaction();
        entityTransaction.begin();

        entityManager.persist(new Supplier( companyName: "Amazon",
                new Address( country: "Poland",  city: "Krakow",  street: "Budryka",  zipCode: "30-022")));
        entityManager.persist(new Supplier( companyName: "Food Supplier",
                new Address( country: "Ukraine", city: "Lviv", street: "Mazepy", zipCode: "78-018")));

        entityTransaction.commit();
        entityManager.close();
    }
}
```

4. Wynik:



Klasa Supplier

	DBID	CITY	COUNTRY	STREET	ZIPCODE	COMPANYNAME
1	1	Krakow	Poland	Budryka	30-022	Amazon
2	2	Lviv	Ukraine	Mazepy	78-018	Food Supplier

Cześć 2

1.Została zmodyfikowana klasa Supplier

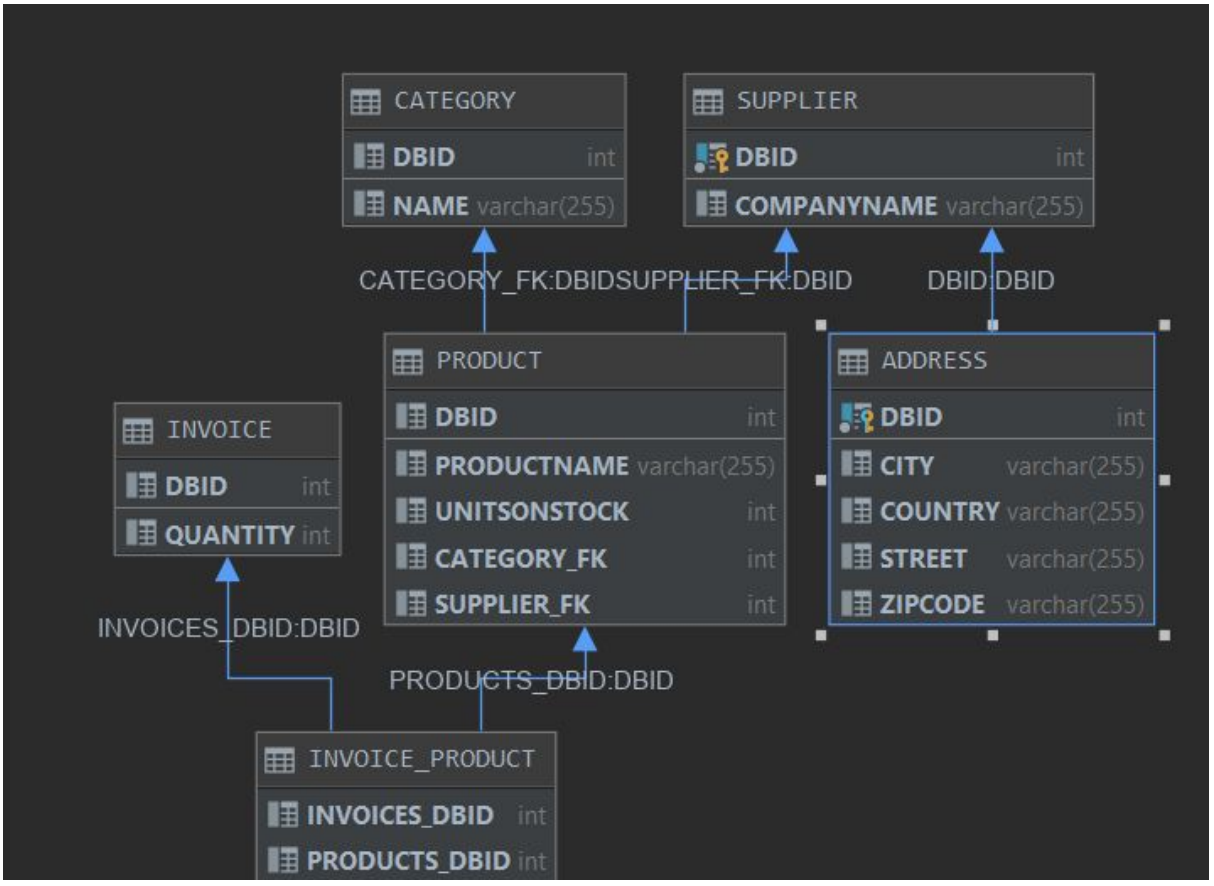
```
@Entity
@SecondaryTable(name="Address")
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int dbID;
    private String companyName;

    @Column(table="Address")
    private String country;
    @Column(table="Address")
    private String city;
    @Column(table="Address")
    private String street;
    @Column(table="Address")
    private String zipCode;

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

    public Supplier() {
    }

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

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

    public void addProduct(Product product) {
```

```
        this.products.add(product);
        product.setSupplier(this);
    }
}
```

2.Dla obserwacji wprowadzonych zmian modyfikacja Main

```
public static void main(String[] argv) {
    EntityManager entityManager = getEntityManager();
    EntityTransaction entityTransaction = entityManager.getTransaction();
    entityTransaction.begin();

    entityManager.persist(new Supplier( companyName: "Grocery store",  country: "Poland",  city: "Krakow",
            street: "Budryka",  zipCode: "30-072"));
    entityManager.persist(new Supplier( companyName: "Food Supplier",  country: "Belarus",  city: "Minsk",
            street: "Kammennogorskaya",  zipCode: "220017"));

    entityTransaction.commit();
    entityManager.close();
}
```

3.Wynik:



	CITY	COUNTRY	STREET	ZIPCODE	DBID
1	Krakow	Poland	Budryka	30-072	1
2	Minsk	Belarus	Kammennogorskaya	220017	2