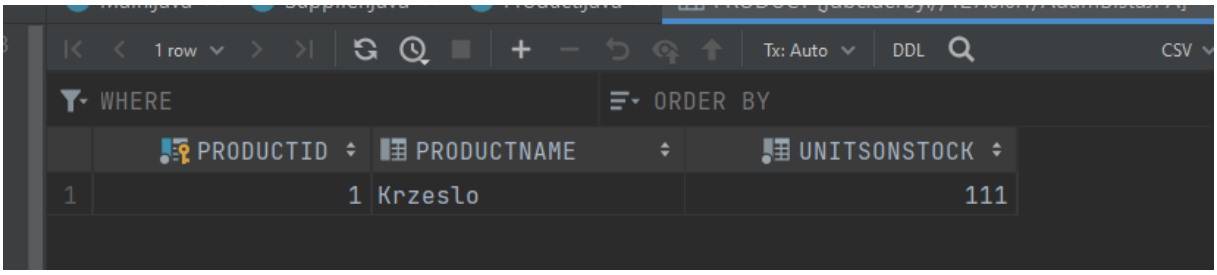
Adam Biśta, 22.04.2023

Akutalnie w bazie danych posiadamy tylko tabelę Product:



	PRODUCTID	PRODUCTNAME	UNITSONSTOCK
1	1	Krzeslo	111

I. Podpunkt już został zrobiony w calosci i wcześniej wyslany.

II. Tworzymy klasę Supplier:

```
package org.example;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import java.util.Set;

@Entity
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;

    private String companyName;
    private String street;
    private String city;

    public Supplier() {
    }

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

```
}
```

Robimy mapping w hibernate.cfg.xml:

```
<mapping class="org.example.Supplier"/>
```

Wprowadzając relację ManyToOne, wystarczy zmodyfikować klasę Product dodając podany kod:

```
@ManyToOne
private Supplier supplier;
```

Klasa Product w pełni wygląda następująco:

```
package org.example;
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) {
        ProductName = productName;
        UnitsOnStock = unitsOnStock;
        this.supplier = supplier;
    }
    public Product(String productName, int unitsOnStock) {
        ProductName = productName;
        UnitsOnStock = unitsOnStock;
    }
    public Product(){

    }

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

    public int getUnitsOnStock() {
        return UnitsOnStock;
```

```
    }



    public void setUnitsOnStock(int unitsOnStock) {
        UnitsOnStock = unitsOnStock;
    }

    @Override
    public String toString() {
        return "Product{" +
                "ProductName='" + ProductName + '\'' +
                ", UnitsOnStock=" + UnitsOnStock +
                '}';
    }

}
```

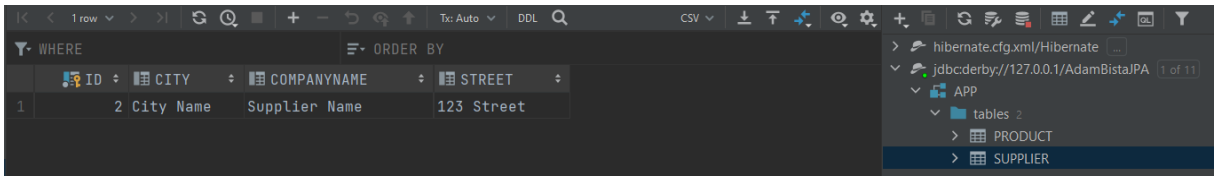
Oba podpunkty zostały zrealizowane w podanym kodzie w public static void main:

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

    // a) Create a new supplier
    Supplier supplier = new Supplier("Supplier Name", "123
Street", "City Name");

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

    // b) Find the previously added product and set its
supplier
    Product product;
    session = getSession();
    try {
        product = session.find(Product.class, 1);
        if (product != null) {
            Transaction tx = session.beginTransaction();
```

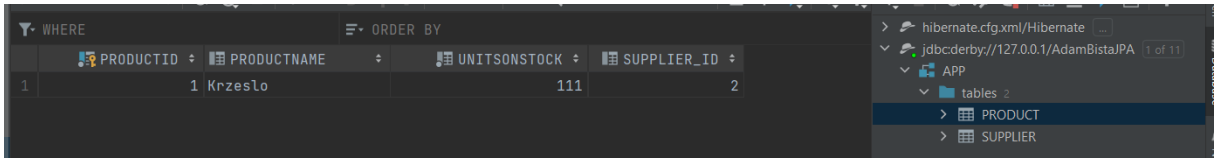
```
            product.setSupplier(supplier);
            session.update(product);
            tx.commit();
        }
    } finally {
        session.close();
    }
}
```

Wygląd naszych tabel:

Supplier:



Product:



III. Modyfikujemy struktury tabel. W klasie Product usuwamy ostatnie zapiski i to co pozostaje to:

```
package org.example;
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) {
```

```
        ProductName = productName;
        UnitsOnStock = unitsOnStock;
    }
    public Product(){

    }

    public int getUnitsOnStock() {
        return UnitsOnStock;
    }

    public void setUnitsOnStock(int unitsOnStock) {
        UnitsOnStock = unitsOnStock;
    }

    @Override
    public String toString() {
        return "Product{" +
                "ProductName='" + ProductName + '\'' +
                ", UnitsOnStock=" + UnitsOnStock +
                '}';
    }

}
```

Zastosujemy najpierw wariant dla tabeli łącznikowej. Wobec tego klasa Supplier wygląda tak:

```
package org.example;

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

@Entity
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;

    private String companyName;
    private String street;
    private String city;

    @OneToMany
    private Set<Product> products;

    public Supplier() {
```

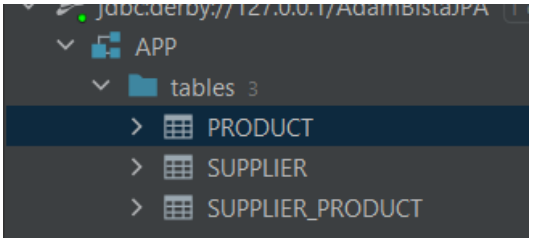
```
    }

    public Supplier(String companyName, String street, String
city) {
        this.companyName = companyName;
        this.street = street;
        this.city = city;
    }
    // Getter for products
    public Set<Product> getProducts() {
        return products;
    }
    // Setter for products
    public void setProducts(Set<Product> products) {
        this.products = products;
    }


}
```

podpunkty b i c wykonujemy w main:

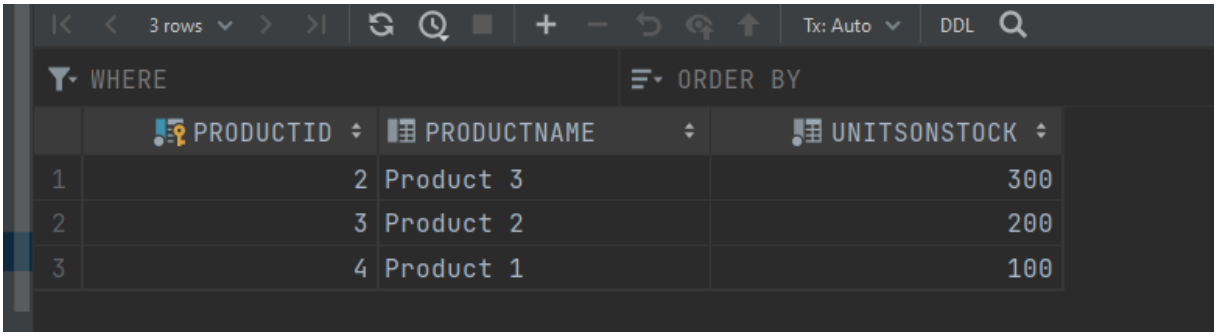
```
public static void main(final String[] args) throws Exception
{
    Session session = getSession();
    // Create a new supplier
    Supplier supplier = new Supplier("Supplier Name", "123
Street", "City Name");
    try {
        Transaction tx = session.beginTransaction();
        session.save(supplier);
        tx.commit();
    } finally {
        session.close();
    }
    // Create several products
    Product product1 = new Product("Product 1", 100);
    Product product2 = new Product("Product 2", 200);
    Product product3 = new Product("Product 3", 300);
    Set<Product> products = new HashSet<>();
    products.add(product1);
    products.add(product2);
    products.add(product3);
    supplier.setProducts(products);

    session = getSession();
    try {
        Transaction tx = session.beginTransaction();
        for (Product product : products) {
            session.save(product);
```

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

Wynik wywolania jest następujący:

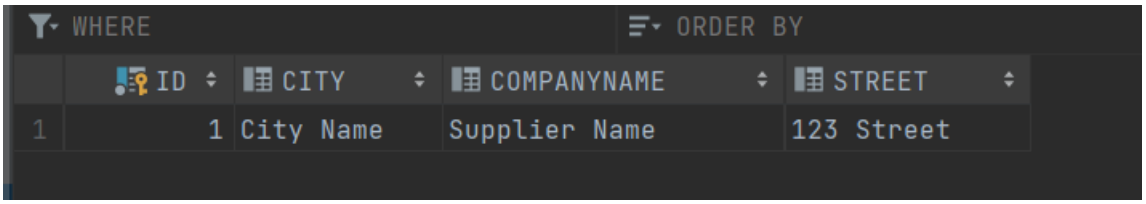


Product:



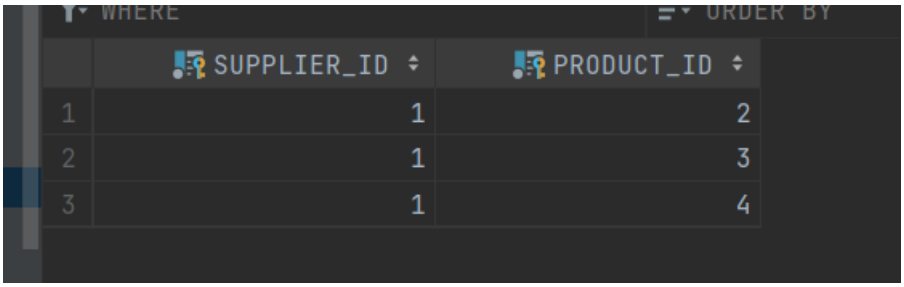
	PRODUCTID	PRODUCTNAME	UNITSONSTOCK
1	2	Product 3	300
2	3	Product 2	200
3	4	Product 1	100

Supplier:



	ID	CITY	COMPANYNAME	STREET
1	1	City Name	Supplier Name	123 Street

Product_supplier:



	SUPPLIER_ID	PRODUCT_ID
1	1	2
2	1	3
3	1	4

W wariancie bez tabeli łącznikowej:

Product

```java
package org.example;

import javax.persistence.*;

@Entity
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int productID;

    private String ProductName;
    private int UnitsOnStock;

    @Column(name="SUPPLIER_FK")
   private Integer supplier_fk;


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

    }

    public int getUnitsOnStock() {
        return UnitsOnStock;
    }

    public void setUnitsOnStock(int unitsOnStock) {
        UnitsOnStock = unitsOnStock;
    }

    @Override
    public String toString() {
        return "Product{" +
                "ProductName='" + ProductName + '\'' +
                ", UnitsOnStock=" + UnitsOnStock +
                '}';
    }

}
```

Supplier:

```java
package org.example;

import javax.persistence.*;
```

```java
import java.util.HashSet;
import java.util.Set;

@Entity
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String companyName;

    private String street;

    private String city;

    @OneToMany
    @JoinColumn(name="SUPPLIER_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;
    }
    // Getter for products
    public Set<Product> getProducts() {
        return products;
    }
    // Setter for products
    public void setProducts(Set<Product> products) {
        this.products = products;
    }


}
```

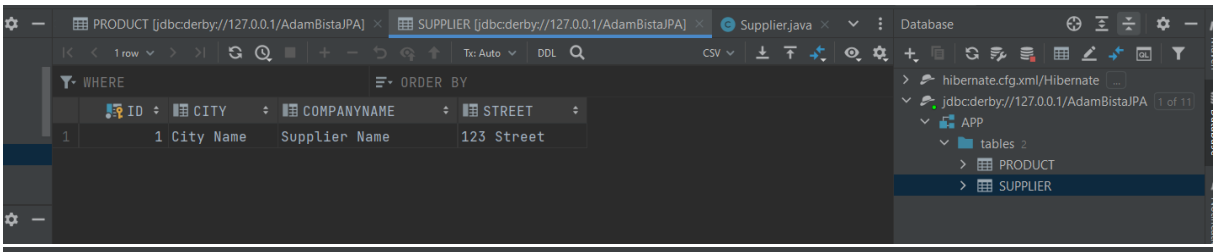
Main:

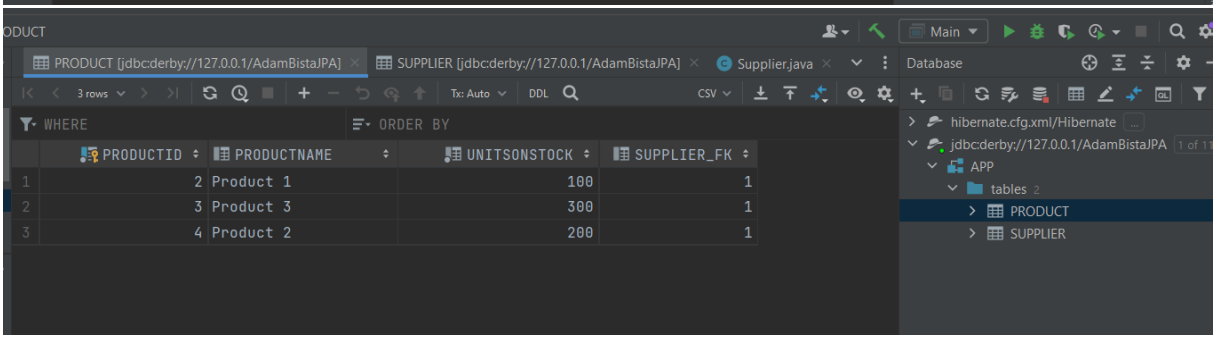
```java
package org.example;

import org.hibernate.HibernateException;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;
```

```
import java.util.HashSet;
import java.util.Set;

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 {
        Session session = getSession();
        // Create a new supplier
        Supplier supplier = new Supplier("Supplier Name", "123
Street", "City Name");
        try {
            Transaction tx = session.beginTransaction();
            session.save(supplier);
            tx.commit();
        } finally {
            session.close();
        }
        // Create several products
        Product product1 = new Product("Product 1", 100);
        Product product2 = new Product("Product 2", 200);
        Product product3 = new Product("Product 3", 300);
        Set<Product> products = new HashSet<>();
        products.add(product1);

        products.add(product2);
        products.add(product3);
        supplier.setProducts(products);

        session = getSession();
        try {
            Transaction tx = session.beginTransaction();
```

```
            for (Product product : products) {
                session.save(product);
            }
            session.update(supplier);
            tx.commit();
        } finally {
            session.close();
        }
    }

}
```

Wywolanie main:





IV. Struktura tabel:

Klasa Supplier – tutaj pozostaje @OneToMany

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

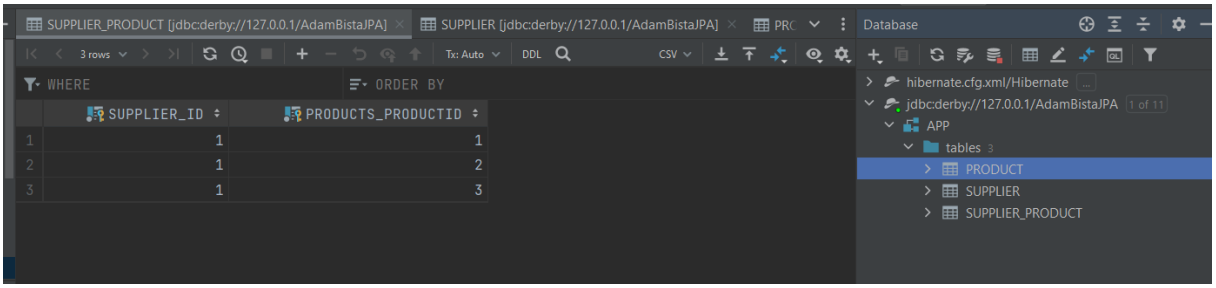
Klasa Product – tutaj pozostaje @ManyToOne

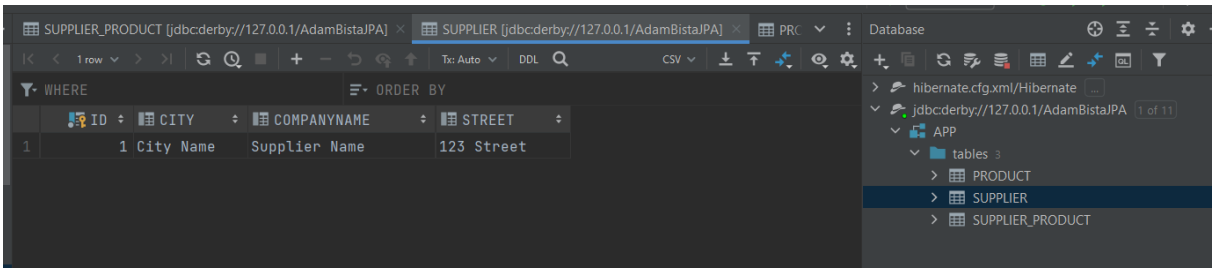
```
@ManyToOne
private Supplier supplier;

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

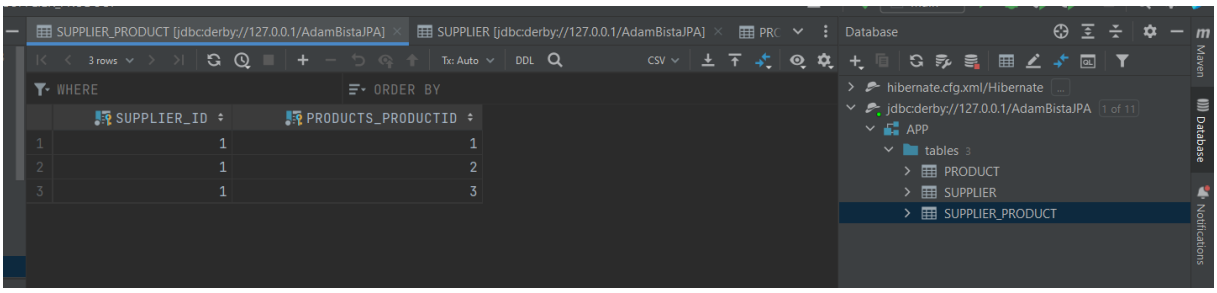
metoda main:

```java
public static void main(final String[] args) throws Exception
{
    Session session = getSession();
    // Create a new supplier
    Supplier supplier = new Supplier("Supplier Name", "123
Street", "City Name");
    try {
        Transaction tx = session.beginTransaction();
        session.save(supplier);
        tx.commit();
    } finally {
        session.close();
    }
    // Create several products
    Product product1 = new Product("Product 1", 100);
    Product product2 = new Product("Product 2", 200);
    Product product3 = new Product("Product 3", 300);
    Set<Product> products = new HashSet<>();
    products.add(product1);

    products.add(product2);
    products.add(product3);
    supplier.setProducts(products);

    session = getSession();
    try {
        Transaction tx = session.beginTransaction();
        for (Product product : products) {
            product.setSupplier(supplier);
            session.save(product);
        }
        session.update(supplier);
        tx.commit();
    } finally {
        session.close();
    }
}
```

Wynik wywolania:







V. Struktura tabel:

Wpisujemy w hibernate.cfg.xml:

```
<mapping class="org.example.Category"/>
```

Klasa Category:

```
package org.example;

import javax.persistence.*;
import java.util.ArrayList;
import java.util.List;

@Entity
public class Category {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int CategoryID;

    private String Name;

    @OneToMany
    private List<Product> Products = new ArrayList<>();

    public Category() {
    }

    public Category(String name) {
        Name = name;
    }

    public int getCategoryID() {
        return CategoryID;
    }
```

```
    public String getName() {
        return Name;
    }

    public void setName(String name) {
        Name = name;
    }

    public List<Product> getProducts() {
        return Products;
    }

    public void setProducts(List<Product> products) {
        Products = products;
    }
    public void addProduct(Product product){
        this.Products.add(product);
        product.setCategory(this);
    }
}
```

Klasa Product: (dodajemy podany kod)

```
@ManyToOne
private Category category;

public Category getCategory() {
    return category;
}

public void setCategoryInBothPlaces(Category category) {
    this.category = category;
    category.getProducts().add(this);
}

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

a) W main wpisujemy:

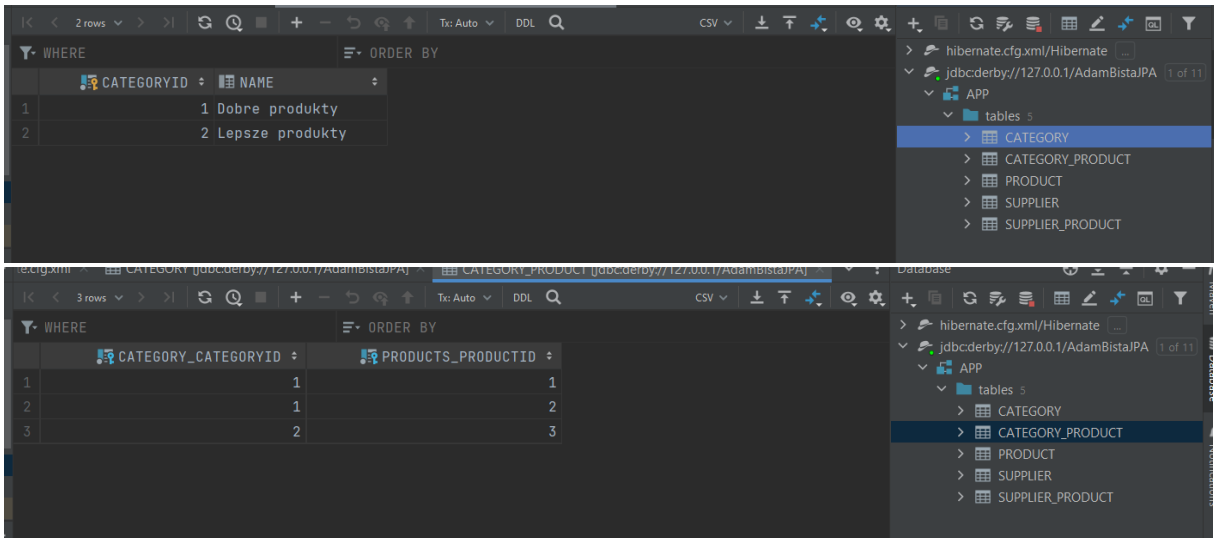
```
b) try (Session session = getSession()) {
        Transaction tx = session.beginTransaction();

        // Assuming you have the product IDs of the existing
   products
        int existingProduct1ID = 1;
        int existingProduct2ID = 2;
        int existingProduct3ID = 3;

        // Load existing products
```

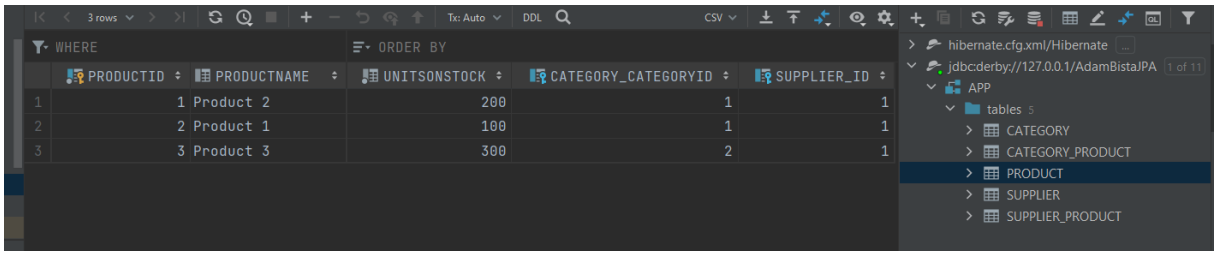
```
    Product existingProduct1 = session.get(Product.class,
existingProduct1ID);
    Product existingProduct2 = session.get(Product.class,
existingProduct2ID);
    Product existingProduct3 = session.get(Product.class,
existingProduct3ID);

    //create categories
    Category existingProductsCategory1 = new
Category("Dobre produkty");
    Category existingProductsCategory2 = new
Category("Lepsze produkty");
    // Add products to the categories

existingProductsCategory1.addProduct(existingProduct1);

existingProductsCategory1.addProduct(existingProduct2);

existingProductsCategory2.addProduct(existingProduct3);

    // Save the category and update the products
    session.save(existingProductsCategory1);
    session.save(existingProductsCategory2);
    session.update(existingProduct1);
    session.update(existingProduct2);
    session.update(existingProduct3);
    tx.commit();
}
```

Nie zapomnijmy też, że musimy mieć odpowiednie property w hibernate.cfg.xml:

```
<property name="hibernate.hbm2ddl.auto">update</property>
```

Wynik wywołania:





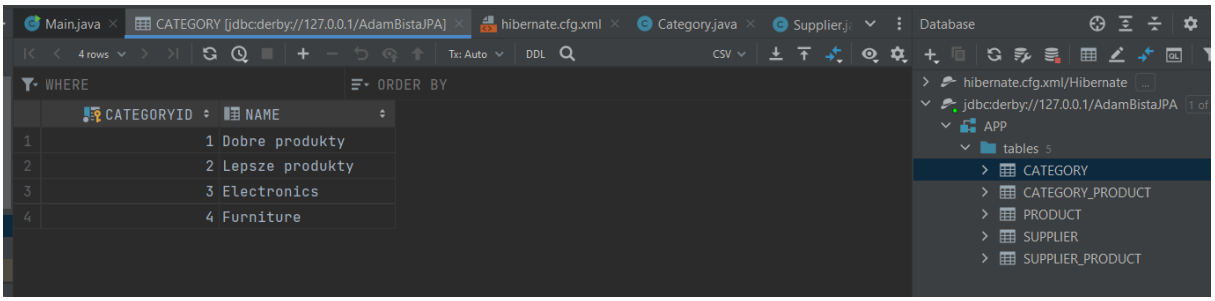
	PRODUCTID	PRODUCTNAME	UNITSONSTOCK	CATEGORY_CATEGORYID	SUPPLIER_ID
1	1	Product 2	200	1	1
2	2	Product 1	100	1	1
3	3	Product 3	300	2	1

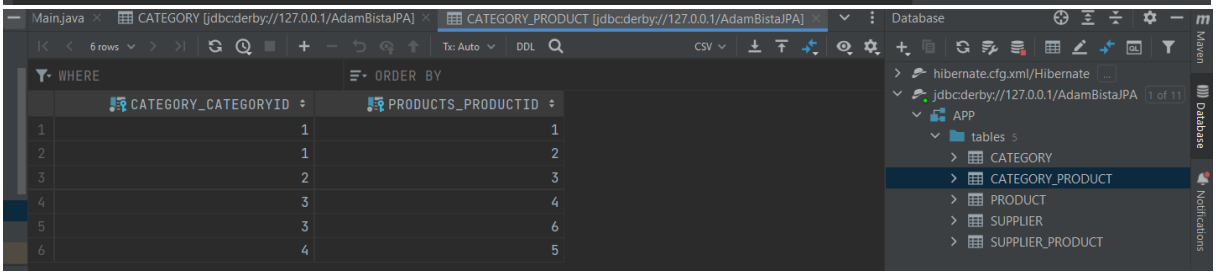
c) i c) W main wpisujemy:

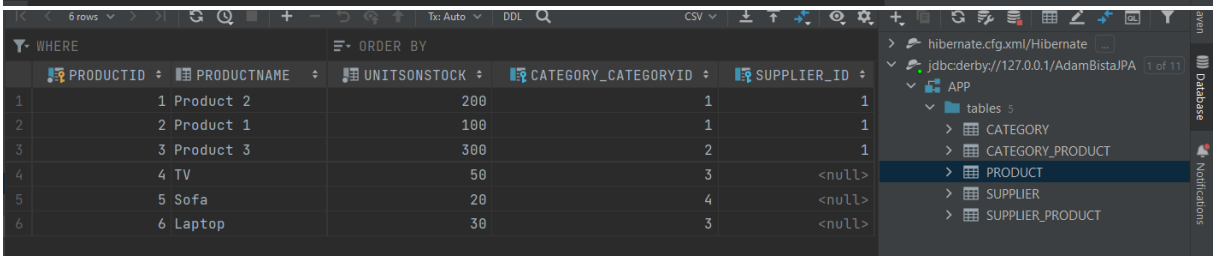
Wywolanie:

```
public static void main(final String[] args) throws Exception
{
    // Create a few categories
    Category category1 = new Category("Electronics");
    Category category2 = new Category("Furniture");

        // Create a few products
    Product product1 = new Product("TV", 50);
    Product product2 = new Product("Sofa", 20);
    Product product3 = new Product("Laptop", 30);

        // Add products to the selected category and proper
category to product (in one function)
    category1.addProduct(product1);
    category1.addProduct(product3);
    category2.addProduct(product2);

    //saving categories and products to database
    try (Session session = getSession()) {
        Transaction tx = session.beginTransaction();
        session.save(category1);
        session.save(category2);
        session.save(product1);
        session.save(product2);
        session.save(product3);
        tx.commit();
    }

}
```

Wyniki wywolania:







d) W main wpisujemy:

```
e) try (Session session = getSession()) {
        Transaction tx = session.beginTransaction();

        // Get products from the selected category
        Category selectedCategory =
    session.get(Category.class, 1);
        List<Product> productsInCategory =
    selectedCategory.getProducts();
        System.out.println("Products in category " +
    selectedCategory.getName() + ": " + productsInCategory);

        // Get the category a selected product belongs to
        Product selectedProduct = session.get(Product.class,
    1);
        Category categoryOfProduct =
    selectedProduct.getCategory();
        System.out.println("Category of product " +
    selectedProduct.getProductName() + ": " +
    categoryOfProduct.getName());

        tx.commit();
    }
```

Wynik wywolania widoczny na konsolce:

```
        Supplier supplier3_
            on product1_.supplier_id=supplier3_.id
    where
        products0_.Category_CategoryID=?
Products in category Dobre produkty: [Product{ProductName='Product 2', UnitsOnStock=200}, Product{ProductName='Product 1', UnitsOnStock=100}]
Category of product Product 2: Dobre produkty

Process finished with exit code 0
```

Dodatkowo jeszcze podpatrzę wszystkie produkty i wszystkie kategorie

```
try (Session session = getSession()) {
    Transaction tx = session.beginTransaction();

    // Retrieve categories by name
    List<Category> categories = session.createQuery("FROM
Category", Category.class).getResultList();
    for (Category category : categories) {
        System.out.println("Category: " + category.getName());
    }

    // Retrieve products by name
    List<Product> products = session.createQuery("FROM
Product", Product.class).getResultList();
    for (Product product : products) {
        System.out.println("Product: " +
product.getProductName() + ", Units on stock: " +
product.getUnitsOnStock() + ", Category: " +
product.getCategory().getName());
    }

    tx.commit();
}
```

```
Category: Dobre produkty
Category: Lepsze produkty
Category: Electronics
Category: Furniture
Hibernate:
```

```
Product: Product 3, Units on stock: 300, Category: Dobre produkty
Product: Product 1, Units on stock: 100, Category: Dobre produkty
Product: Product 2, Units on stock: 200, Category: Lepsze produkty
Product: TV, Units on stock: 50, Category: Electronics
Product: Sofa, Units on stock: 20, Category: Furniture
Product: Laptop, Units on stock: 30, Category: Electronics
```

VI. Nowa struktura tabel:

Dodajemy do hibernate.cfg.xml:

```
<mapping class="org.example.Invoice"/>
```

Klasa Invoice:

```
package org.example;

import javax.persistence.*;
import java.util.ArrayList;
import java.util.List;

@Entity
public class Invoice {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int InvoiceNumber;

    private int Quantity;

    @ManyToMany
    private List<Product> products = new ArrayList<>();

    public Invoice() {
    }

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

    public int getInvoiceNumber() {
        return InvoiceNumber;
    }

    public int getQuantity() {
        return Quantity;
    }

    public void setQuantity(int quantity) {
        Quantity = quantity;
    }

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

    public void setProducts(List<Product> products) {
```

```
        this.products = products;
    }

    public void AddProductDoubleRelation(Product product){
        this.products.add(product);
        product.getInvoices().add(this);
    }
}
```

W klasie Product dodajemy:

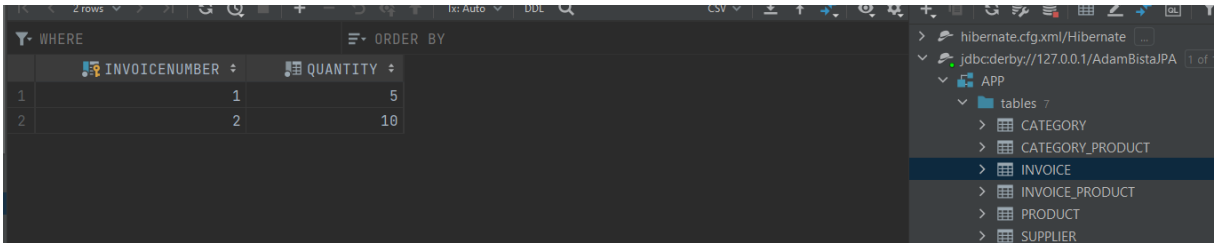
```
@ManyToMany(mappedBy = "products")
private List<Invoice> invoices = new ArrayList<>();

// Add getter and setter for the 'invoices' field
public List<Invoice> getInvoices() {
    return invoices;
}

public void setInvoices(List<Invoice> invoices) {
    this.invoices = invoices;
}
public void AddInvoiceDoubleRelation(Invoice invoice){
    this.invoices.add(invoice);
    invoice.getProducts().add(this);
}
```

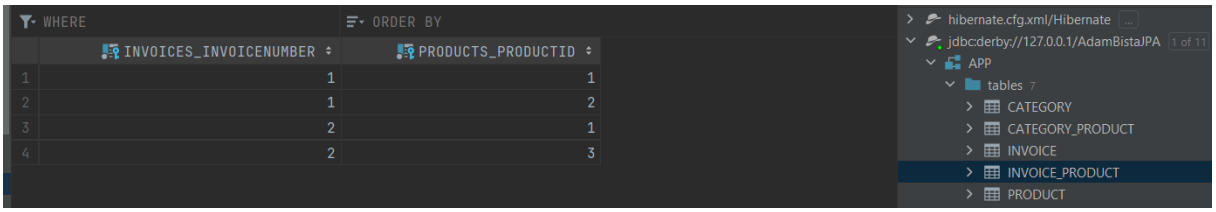
Funkcja main:

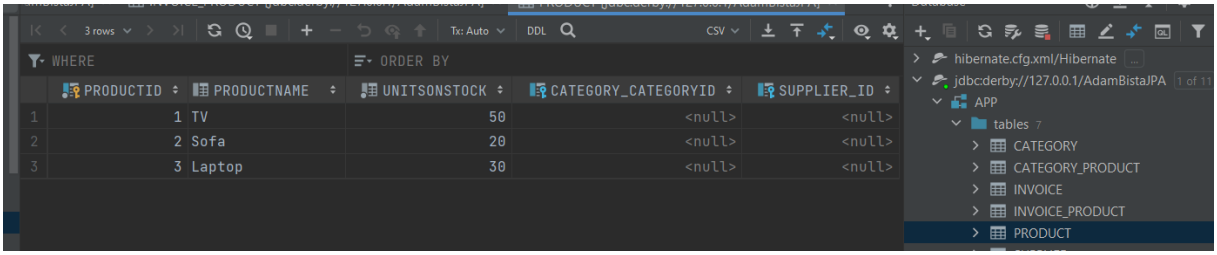
```
public static void main(final String[] args) throws Exception
{
        // Create a few products
        Product product1 = new Product("TV", 50);
        Product product2 = new Product("Sofa", 20);
        Product product3 = new Product("Laptop", 30);

// Create invoices and sell products
        Invoice invoice1 = new Invoice(5);
        Invoice invoice2 = new Invoice(10);

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

        try (Session session = getSession()) {
            Transaction tx = session.beginTransaction();
            session.save(product1);
            session.save(product2);
            session.save(product3);
            session.save(invoice1);
            session.save(invoice2);
```

```
            tx.commit();
        }
    }
```

Wywolanie:







b) i c):

```
public static void main(final String[] args) throws Exception
{
    int invoiceId = 1;
    int productId = 1;
    try (Session session = getSession()) {
        // Wypisz produkty sprzedane w ramach faktury o
identyfikatorze 1
        Invoice invoice = session.get(Invoice.class,
invoiceId);
        if (invoice != null) {
            System.out.println("Produkty sprzedane w ramach
faktury o identyfikatorze 1:");
            List<Product> products = invoice.getProducts();
            for (Product product : products) {
                System.out.println(product.getProductName());
            }
        } else {
            System.out.println("Nie znaleziono faktury o
identyfikatorze " + invoiceId);
        }
```

```
        // Wypisz faktury, w ramach których był sprzedany
produkt o identyfikatorze 1
        Product product = session.get(Product.class,
productId);
        if (product != null) {
            System.out.println("Faktury, w ramach których był
sprzedany produkt o identyfikatorze 1:");
            List<Invoice> invoices = product.getInvoices();
            for (Invoice inv : invoices) {
                System.out.println("Faktura " +
inv.getInvoiceNumber());
            }
        } else {
            System.out.println("Nie znaleziono produktu o
identyfikatorze " + productId);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

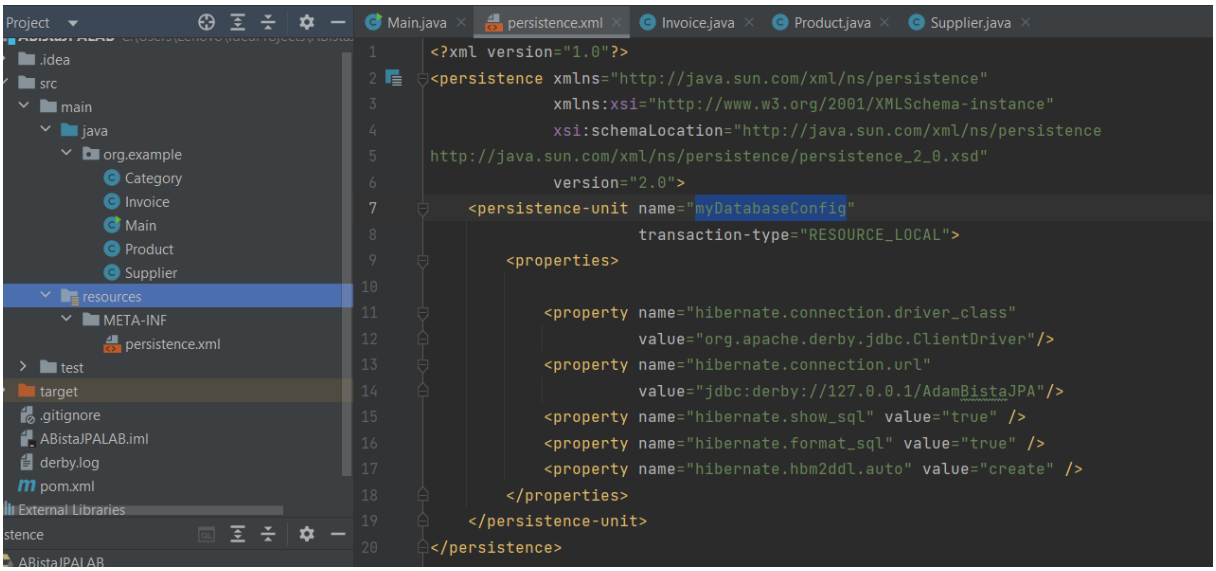
wynik wywolania w konsoli:

```
Produkty sprzedane w ramach faktury o identyfikatorze 1:
Hibernate:
    select
        products0_.invoices_InvoiceNumber as invoices1_3_0_,
        products0_.products_productID as products2_3_0_,
        product1_.productID as producti1_4_1_,
        product1_.ProductName as productn2_4_1_,
        product1_.UnitsOnStock as unitsons3_4_1_,
        product1_.category_CategoryID as category4_4_1_,
        product1_.supplier_id as supplier5_4_1_,
        category2_.CategoryID as category1_0_2_,
        category2_.Name as name2_0_2_,
        supplier3_.id as id1_5_3_,
        supplier3_.city as city2_5_3_,
        supplier3_.companyName as companyn3_5_3_,
        supplier3_.street as street4_5_3_
    from
        Invoice_Product products0_
    inner join
        Product product1_
            on products0_.products_productID=product1_.productID
    left outer join
        Category category2_
            on product1_.category_CategoryID=category2_.CategoryID
    left outer join
        Supplier supplier3_
            on product1_.supplier_id=supplier3_.id
    where
        products0_.invoices_InvoiceNumber=?
```

```
            on products0_.products_productID=product1_.productID
    left outer join
        Category category2_
            on product1_.category_CategoryID=category2_.CategoryID
    left outer join
        Supplier supplier3_
            on product1_.supplier_id=supplier3_.id
    where
        products0_.invoices_InvoiceNumber=?
TV
Sofa
Faktury, w ramach których był sprzedany produkt o identyfikatorze 1:
Hibernate:
    select
        invoices0_.products_productID as products2_3_0_,
        invoices0_.invoices_InvoiceNumber as invoices1_3_0_,
        invoice1_.InvoiceNumber as invoicen1_2_1_,
        invoice1_.Quantity as quantity2_2_1_
    from
        Invoice_Product invoices0_
    inner join
        Invoice invoice1_
            on invoices0_.invoices_InvoiceNumber=invoice1_.InvoiceNumber
    where
        invoices0_.products_productID=?
Faktura 1
Faktura 2
```

VII. Dodawanie JPA:

Tworzymy nowy plik persistence.xml:



pom.xml:

```
<!-- Java Persistence API -->
<dependency>
    <groupId>javax.persistence</groupId>
    <artifactId>javax.persistence-api</artifactId>
    <version>2.2</version>
</dependency>
```

W klasie glownej Main: (dodajemy też kod analogiczny do podpunktu poprzedniego)

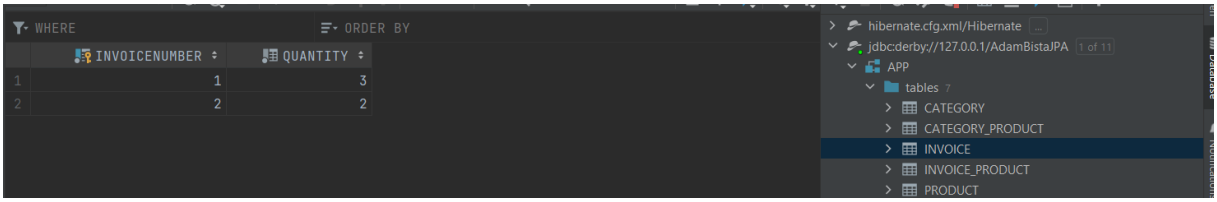
```
package org.example;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

public class Main {

    public static void main(final String[] args) {
        EntityManagerFactory entityManagerFactory =
Persistence.createEntityManagerFactory("myDatabaseConfig");
        EntityManager entityManager =
entityManagerFactory.createEntityManager();

        // Create a few products
        Product product1 = new Product("TV", 50);
        Product product2 = new Product("Sofa", 20);
        Product product3 = new Product("Laptop", 30);

        // Create invoices and sell products
        Invoice invoice1 = new Invoice(3);
        Invoice invoice2 = new Invoice(2);

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

        // Save invoices and products to the database
        entityManager.getTransaction().begin();
        entityManager.persist(product1);
        entityManager.persist(product2);
        entityManager.persist(product3);
        entityManager.persist(invoice1);
        entityManager.persist(invoice2);
        entityManager.getTransaction().commit();

        // Get products from the selected invoice
        int selectedInvoiceNumber = 1;
        Invoice selectedInvoice =
entityManager.find(Invoice.class, selectedInvoiceNumber);
        System.out.println("Products in invoice " +
```

```
selectedInvoiceNumber + ": " + selectedInvoice.getProducts());

        // Get invoices for the selected product

        Product selectedProduct =
entityManager.find(Product.class, 1);
        System.out.println("Invoices with product " +
selectedProduct.getProductName() + ": ");
        selectedProduct.getInvoices().forEach(invoice ->
System.out.println("Invoice number: " +
invoice.getInvoiceNumber()));

        entityManager.close();
        entityManagerFactory.close();
    }
}
```

Wynik wywołania:

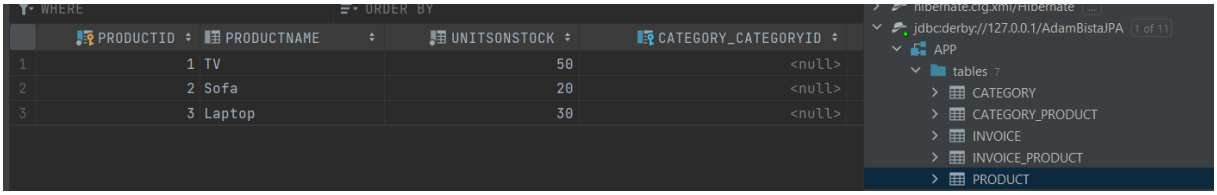
```
        (?, ?)
Products in invoice 1: [Product{ProductName='TV', UnitsOnStock=50}, Product{ProductName='Sofa', UnitsOnStock=20}]
Invoices with product TV:
Invoice number: 1
Invoice number: 2
```

Tabele:



	INVOICENUMBER	QUANTITY
1	1	3
2	2	2

	INVOICES_INVOICENUMBER	PRODUCTS_PRODUCTID
1	1	1
2	1	2
3	2	1
4	2	3



	PRODUCTID	PRODUCTNAME	UNITSONSTOCK	CATEGORY_CATEGORYID
1	1	TV	50	<null>
2	2	Sofa	20	<null>
3	3	Laptop	30	<null>

VIII. Kaskady:

W tabeli Invoice modyfikujemy:

```
@ManyToMany(cascade = CascadeType.PERSIST)
private List<Product> products = new ArrayList<>();
```

W tabeli Product analogicznie:

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

W main podmieniamy kod:

Przed:

entityManager.getTransaction().begin();

entityManager.persist(product1);

entityManager.persist(product2);

entityManager.persist(product3);

entityManager.persist(invoice1);

entityManager.persist(invoice2);

entityManager.getTransaction().commit();

Po:

```
// Save invoices and products to the database
entityManager.getTransaction().begin();
entityManager.persist(invoice1); // This will also persist
products associated with this invoice
entityManager.persist(invoice2); // This will also persist
products associated with this invoice
entityManager.getTransaction().commit();
```

Wynik wywolania:

```
        (?, ?)
Products in invoice 1: [Product{ProductName='TV', UnitsOnStock=50}, Product{ProductName='Sofa', UnitsOnStock=20}]
Invoices with product TV:
Invoice number: 1
Invoice number: 2
kwi 23, 2023 1:28:58 AM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl$PoolState stop
```

Tabele wyglądają identycznie jak poprzednio i wywołanie daje te same rezultaty.

IX. Embedded Class:

a) Tworzymy klasę Address:

```
package org.example;

import javax.persistence.Embeddable;

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

    public String getStreet() {
        return street;
    }

    public void setStreet(String street) {
        this.street = street;
    }

    public String getCity() {
        return city;
    }

    public void setCity(String city) {
        this.city = city;
    }

    public String getZipCode() {
        return zipCode;
    }

    public void setZipCode(String zipCode) {
        this.zipCode = zipCode;
    }

    public String getCountry() {
        return country;
    }

    public void setCountry(String country) {
        this.country = country;
    }

    public Address() {
    }

    public Address(String street, String city, String zipCode,
```

```
String country) {
        this.street = street;
        this.city = city;
        this.zipCode = zipCode;
        this.country = country;
    }


}
```

Klasa Supplier teraz wygląda tak:

```
package org.example;

import javax.persistence.*;
import java.util.HashSet;
import java.util.Set;

@Entity
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String companyName;

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

    @Embedded
    private Address address;

    public Supplier() {
    }

    public String getCompanyName() {
        return companyName;
    }

    public void setCompanyName(String companyName) {
        this.companyName = companyName;
    }

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

    public Address getAddress() {
        return address;
```

```
    }

    public void setAddress(Address address) {
        this.address = address;
    }

    // Getter for products
    public Set<Product> getProducts() {
        return products;
    }
    // Setter for products
    public void setProducts(Set<Product> products) {
        this.products = products;
    }


}
```

Przykładowe wywołanie:

```
package org.example;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

public class Main {

    public static void main(final String[] args) {
        EntityManagerFactory entityManagerFactory =
Persistence.createEntityManagerFactory("myDatabaseConfig");
        EntityManager entityManager =
entityManagerFactory.createEntityManager();

        // Create a few addresses
        Address address1 = new Address("Main St.", "New York",
"10001", "USA");
        Address address2 = new Address("Market St.", "San
Francisco", "94103", "USA");

        // Create suppliers
        Supplier supplier1 = new Supplier("Electronics Plus",
address1);
        Supplier supplier2 = new Supplier("Furniture World",
address2);

        // Save suppliers to the database
        entityManager.getTransaction().begin();
        entityManager.persist(supplier1);
        entityManager.persist(supplier2);
        entityManager.getTransaction().commit();
```

```
        // Find a supplier by ID
        int supplierId = 1;
        Supplier foundSupplier =
entityManager.find(Supplier.class, supplierId);
        System.out.println("Found supplier: " +
foundSupplier.getCompanyName());
        System.out.println("Address: " +
foundSupplier.getAddress().getStreet() + ", " +
                foundSupplier.getAddress().getCity() + ", " +
                foundSupplier.getAddress().getZipCode() + ", "
+
                foundSupplier.getAddress().getCountry());

        entityManager.close();
        entityManagerFactory.close();
    }


}
```

Efekt wywołania w konsoli:

```
        (id, city, country, street, zipCode, companyName)
    values
        (default, ?, ?, ?, ?, ?)
Hibernate:

values
    identity_val_local()
Found supplier: Electronics Plus
Address: Main St., New York, 10001, USA
kwi 23, 2023 12:01:46 PM org.hibernate.engine.jdbc.connection
INFO: HHH10001008: Cleaning up connection pool [jdbc:derby://

Process finished with exit code 0
```

Tabela:

WHERE ORDER BY

	ID	CITY	COUNTRY	STREET	ZIPCODE	COMPANYNAME
1	1	New York	USA	Main St.	10001	Electronics Plus
2	2	San Francisco	USA	Market St.	94103	Furniture World

hibernate.cfg.xml/Hibernate
jdbc:derby://127.0.0.1/AdamBistaJPA
APP
tables
CATEGORY
CATEGORY_PRODUCT
INVOICE
INVOICE_PRODUCT
PRODUCT
SUPPLIER

b)

Usuwamy klasę Address zdefiniowaną w a).

Nasza struktura klasy Supplier teraz przybiera taką postać:

```java
package org.example;

import javax.persistence.*;
import java.util.HashSet;
import java.util.Set;

@Entity
//@SecondaryTable(name = "supplier_addresses", pkJoinColumns =
@PrimaryKeyJoinColumn(name = "supplier_id"))
@SecondaryTable(name = "supplier_addresses")
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String companyName;

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

    @Column(table = "supplier_addresses")
    private String street;

    @Column(table = "supplier_addresses")
    private String city;

    @Column(table = "supplier_addresses")
    private String postalCode;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getStreet() {
        return street;
    }

    public void setStreet(String street) {
        this.street = street;
    }

    public String getCity() {
        return city;
    }

    public void setCity(String city) {
        this.city = city;
```

```
    }

    public String getPostalCode() {
        return postalCode;
    }

    public void setPostalCode(String postalCode) {
        this.postalCode = postalCode;
    }

    public String getCountry() {
        return country;
    }

    public void setCountry(String country) {
        this.country = country;
    }

    @Column(table = "supplier_addresses")
    private String country;

    public Supplier() {
    }

    public String getCompanyName() {
        return companyName;
    }

    public void setCompanyName(String companyName) {
        this.companyName = companyName;
    }


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

    // Getter for products
    public Set<Product> getProducts() {
        return products;
    }
    // Setter for products
    public void setProducts(Set<Product> products) {
        this.products = products;
    }
```

```
}
```

W Main tworzymy przykładowe wywołanie:

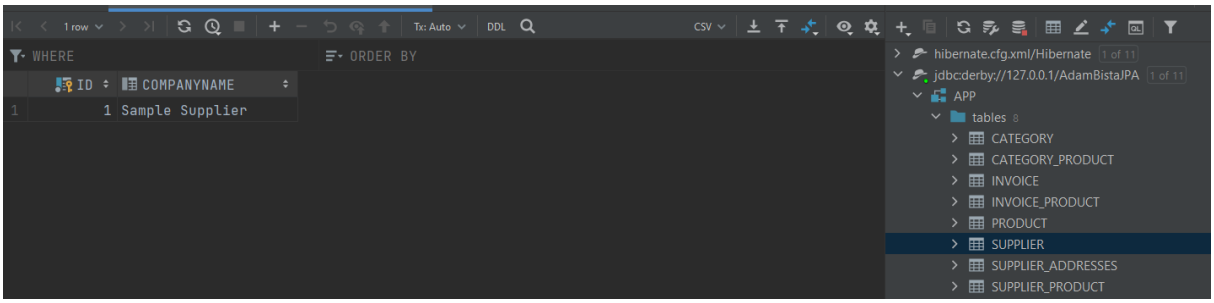
```
package org.example;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

public class Main {

    public static void main(String[] args) {
        EntityManagerFactory entityManagerFactory =
Persistence.createEntityManagerFactory("myDatabaseConfig");
        EntityManager entityManager =
entityManagerFactory.createEntityManager();

        entityManager.getTransaction().begin();

        // Create a new Supplier
        Supplier supplier = new Supplier();
        supplier.setCompanyName("Sample Supplier");
        supplier.setStreet("123 Main St");
        supplier.setCity("New York");
        supplier.setPostalCode("10001");
        supplier.setCountry("USA");

        // Persist the new Supplier
        entityManager.persist(supplier);

        // Commit the transaction
        entityManager.getTransaction().commit();

        // Retrieve the Supplier by ID
        Supplier retrievedSupplier =
entityManager.find(Supplier.class, supplier.getId());

        // Print the retrieved Supplier details
        System.out.println("Retrieved Supplier:");
        System.out.println("Name: " +
retrievedSupplier.getCompanyName());
        System.out.println("Street: " +
retrievedSupplier.getStreet());
        System.out.println("City: " +
retrievedSupplier.getCity());
        System.out.println("Postal Code: " +
retrievedSupplier.getPostalCode());
        System.out.println("Country: " +
```

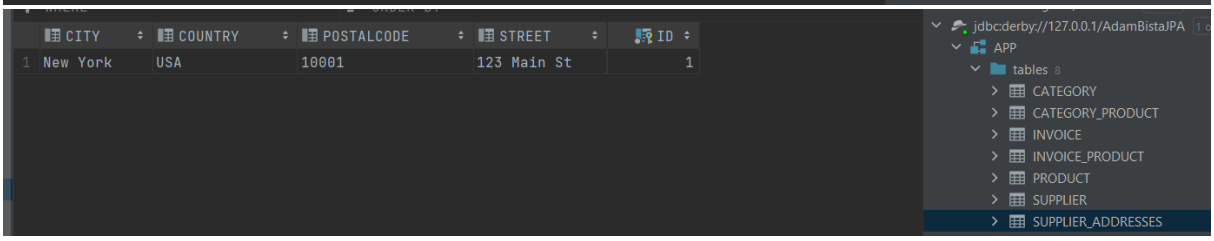
```
retrievedSupplier.getCountry());

        entityManager.close();
        entityManagerFactory.close();
    }


}
```

Wynik wywołania:

```
    values
        (?, ?, ?, ?, ?)
Retrieved Supplier:
Name: Sample Supplier
Street: 123 Main St
City: New York
Postal Code: 10001
Country: USA
kwi 23, 2023 12:28:04 PM org.hibernate.en
INFO: HHH10001008: Cleaning up connection

Process finished with exit code 0
```

Tabele:





X. Hierarchie dziedziczenia:

1) SINGLE TABLE:

Company:

```
package org.example;

import javax.persistence.*;
@Entity(name="company")
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name = "company_type")
public abstract class Company {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String companyName;
    private String street;

    private String city;
    private String zipCode;
    public Company(String name, String street, String city,
String zipCode) {
        this.companyName = name;
        this.street = street;
        this.city = city;
        this.zipCode = zipCode;

    }
    public Company(){

    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }




    public String getCompanyName() {
        return companyName;
    }

    public void setCompanyName(String companyName) {
        this.companyName = companyName;
```

```
    }

    public String getStreet() {
        return street;
    }

    public void setStreet(String street) {
        this.street = street;
    }

    public String getCity() {
        return city;
    }

    public void setCity(String city) {
        this.city = city;
    }

    public String getZipCode() {
        return zipCode;
    }

    public void setZipCode(String zipCode) {
        this.zipCode = zipCode;
    }


}
```

Supplier:

```
package org.example;

import javax.persistence.*;
import java.util.HashSet;
import java.util.Set;

@Entity
@DiscriminatorValue("SUPPLIER")
public class Supplier extends Company{
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;


    private String bankAccountNumber;

    public Supplier(String supplierName, String
supplierStreet, String supplierCity, String supplierZipCode,
String number) {

super(supplierName,supplierStreet,supplierCity,supplierZipCode
```

```
);
        this.bankAccountNumber = number;
    }

    public String getBankAccountNumber() {
        return bankAccountNumber;
    }

    public void setBankAccountNumber(String bankAccountNumber)
{
        this.bankAccountNumber = bankAccountNumber;
    }

    public Supplier() {
        super("default","default","default","default");
    }

    public Supplier(String bankAccountNumber) {
        super("default","default","default","default");
        this.bankAccountNumber = bankAccountNumber;
    }

    @Override
    public int getId() {
        return id;
    }

    @Override
    public void setId(int id) {
        this.id = id;
    }
}
```

Customer:

```
package org.example;
import javax.persistence.*;
@Entity
@DiscriminatorValue("CUSTOMER")
public class Customer extends Company {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    private double discount;

    public Customer(String customerName, String
customerStreet, String customerCity, String customerZipCode,
double i) {

super(customerName,customerStreet,customerCity,customerZipCode
);
        this.discount = i;
```

```
    }

    public Customer() {
        super("default","default","default","default");
    }

    public double getDiscount() {
        return discount;
    }

    public void setDiscount(double discount) {
        this.discount = discount;
    }
// Getters and setters
}
```

Wywolanie w Main:

```
package org.example;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.persistence.*;
import java.util.*;

public class Main {

    public static void main(String[] args) {
        EntityManagerFactory entityManagerFactory =
Persistence.createEntityManagerFactory("myDatabaseConfig");
        EntityManager entityManager =
entityManagerFactory.createEntityManager();

        // Tworzenie przykładowych obiektów
        Supplier supplier = new Supplier("ExampleSupplier",
"123 Supplier Street", "SupplierCity", "12345", "1234567890");
        Customer customer = new Customer("ExampleCustomer",
"456 Customer Street", "CustomerCity", "67890", 10.0);

        // Zapisywanie obiektów do bazy danych
        entityManager.getTransaction().begin();
        entityManager.persist(supplier);
        entityManager.persist(customer);
        entityManager.getTransaction().commit();

        // Pobieranie wszystkich danych z tabeli Company
        TypedQuery<Company> companyQuery =
entityManager.createQuery("SELECT c FROM company c",
Company.class);
```

```
        List<Company> companyList =
companyQuery.getResultList();

        // Wypisywanie wyników
        for (Company company : companyList) {
            if (company instanceof Supplier) {
                System.out.println("Supplier: " +
company.getCompanyName());
            } else if (company instanceof Customer) {
                System.out.println("Customer: " +
company.getCompanyName());
            }
        }

        entityManager.close();
        entityManagerFactory.close();
    }


}
```

Wynik:

```
        company0_.company_type as company_1_2_
    from
        company company0_
Supplier: ExampleSupplier
Customer: ExampleCustomer
kwi 23, 2023 1:24:30 PM org.hibernate.engine.jdbc.c
INFO: HHH10001008: Cleaning up connection pool [jdb

Process finished with exit code 0
```

Tabela Company:

Main.java | COMPANY [jdbc:derby://127.0.0.1/AdamBistaJPA] | persistence.xml | Supplier.java | Company.java | Customer.java

WHERE | ORDER BY

	COMPANY_TYPE	ID	CITY	COMPANYNAME	STREET	ZIPCODE	BANKACCOUNTNUMBER	DISCOUNT
1	SUPPLIER	1	SupplierCity	ExampleSupplier	123 Supplier Street	12345	1234567890	<null>
2	CUSTOMER	2	CustomerCity	ExampleCustomer	456 Customer Street	67890	<null>	10

2) TABLE PER CLASS:

Pokażę tylko zmiany wprowadzone w klasach:

Company:

```
@Entity(name="company")
@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)
public abstract class Company {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;
```

Supplier:

```
@Entity
public class Supplier extends Company{
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;
```

Customer:

```
@Entity
public class Customer extends Company {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;
```

Wywolanie w Main:

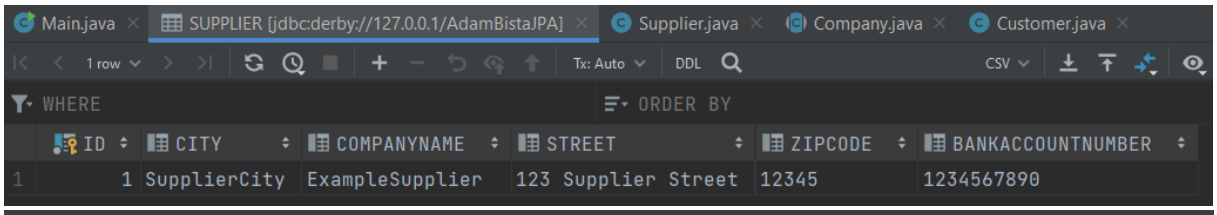
```
package org.example;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.persistence.*;
import java.util.*;

public class Main {

    public static void main(String[] args) {
        EntityManagerFactory entityManagerFactory =
Persistence.createEntityManagerFactory("myDatabaseConfig");
        EntityManager entityManager =
entityManagerFactory.createEntityManager();

        // Tworzenie przykładowych obiektów
        // Tworzenie przykładowych obiektów
        Supplier supplier = new Supplier("ExampleSupplier",
"123 Supplier Street", "SupplierCity", "12345", "1234567890");
        Customer customer = new Customer("ExampleCustomer",
"456 Customer Street", "CustomerCity", "67890", 10.0);

        // Zapisywanie obiektów do bazy danych
        entityManager.getTransaction().begin();
        entityManager.persist(supplier);
```

```
        entityManager.persist(customer);
        entityManager.getTransaction().commit();

        // Pobieranie wszystkich danych z tabeli Supplier
        TypedQuery<Supplier> supplierQuery =
entityManager.createQuery("SELECT s FROM Supplier s",
Supplier.class);
        List<Supplier> supplierList =
supplierQuery.getResultList();

        // Pobieranie wszystkich danych z tabeli Customer
        TypedQuery<Customer> customerQuery =
entityManager.createQuery("SELECT c FROM Customer c",
Customer.class);
        List<Customer> customerList =
customerQuery.getResultList();

        // Wypisywanie wyników
        System.out.println("Suppliers:");
        for (Supplier s : supplierList) {
            System.out.println(s.getCompanyName());
        }

        System.out.println("Customers:");
        for (Customer c : customerList) {
            System.out.println(c.getCompanyName());
        }

        entityManager.close();
        entityManagerFactory.close();
    }


}
```

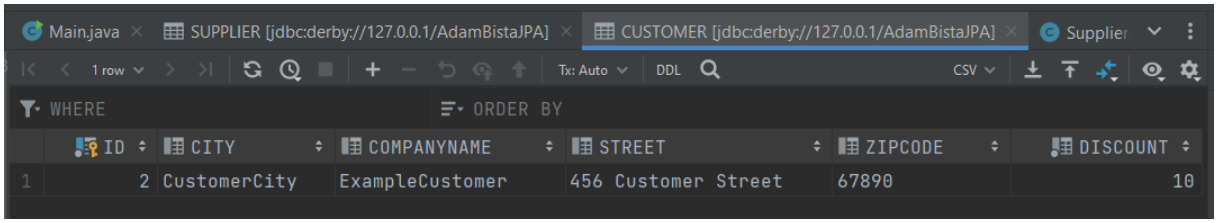
Wynik wywolania:

```
Suppliers:
ExampleSupplier
Customers:
ExampleCustomer
kwi 23, 2023 1:29:36 PM org.hibernate.en
INFO: HHH10001008: Cleaning up connectio

Process finished with exit code 0
```

Tabele



	ID	CITY	COMPANYNAME	STREET	ZIPCODE	BANKACCOUNTNUMBER
1	1	SupplierCity	ExampleSupplier	123 Supplier Street	12345	1234567890



	ID	CITY	COMPANYNAME	STREET	ZIPCODE	DISCOUNT
1	2	CustomerCity	ExampleCustomer	456 Customer Street	67890	10

Tabela Company nie powstaje.

3) JOINED

Wprowadzone zmiany w klasach:

Company:

```
@Entity(name="company")
@Inheritance(strategy = InheritanceType.JOINED)
public abstract class Company {
```

Klasa Supplier i Customer wyglądają tak samo jak w 2)

Wywolanie w Main:

```
package org.example;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.persistence.*;
import java.util.*;

public class Main {

    public static void main(String[] args) {
        EntityManagerFactory entityManagerFactory =
Persistence.createEntityManagerFactory("myDatabaseConfig");
        EntityManager entityManager =
entityManagerFactory.createEntityManager();

        // Tworzenie przykładowych obiektów
        Supplier supplier = new Supplier("ExampleSupplier",
"123 Supplier Street", "SupplierCity", "12345", "1234567890");
        Customer customer = new Customer("ExampleCustomer",
"456 Customer Street", "CustomerCity", "67890", 10.0);
```

```
        // Zapisywanie obiektów do bazy danych
        entityManager.getTransaction().begin();
        entityManager.persist(supplier);
        entityManager.persist(customer);
        entityManager.getTransaction().commit();

        // Pobieranie wszystkich danych z tabeli Supplier
        TypedQuery<Supplier> supplierQuery =
entityManager.createQuery("SELECT s FROM Supplier s",
Supplier.class);
        List<Supplier> supplierList =
supplierQuery.getResultList();

        // Pobieranie wszystkich danych z tabeli Customer
        TypedQuery<Customer> customerQuery =
entityManager.createQuery("SELECT c FROM Customer c",
Customer.class);
        List<Customer> customerList =
customerQuery.getResultList();

        // Wypisywanie wyników
        System.out.println("Suppliers:");
        for (Supplier s : supplierList) {
            System.out.println(s.getCompanyName());
        }

        System.out.println("Customers:");
        for (Customer c : customerList) {
            System.out.println(c.getCompanyName());
        }

        entityManager.close();
        entityManagerFactory.close();
    }


}
```

Wyniki wywołań:

```
            on customer0_.id=customer0_1_.id
Suppliers:
ExampleSupplier
Customers:
ExampleCustomer
kwi 23, 2023 1:36:57 PM org.hibernate.engine.jdbc.
INFO: HHH10001008: Cleaning up connection pool [jd

Process finished with exit code 0
```

Tabele:

Main.java | CUSTOMER [jdbc:derby://127.0.0.1/AdamBistaJPA] | COMPANY [jdbc:derby://127.0.0.1/AdamBistaJPA]

2 rows | Tx: Auto | DDL | CSV

WHERE | ORDER BY

	ID	CITY	COMPANYNAME	STREET	ZIPCODE
1	1	SupplierCity	ExampleSupplier	123 Supplier Street	12345
2	2	CustomerCity	ExampleCustomer	456 Customer Street	67890

Main.java | CUSTOMER [jdbc:derby://127.0.0.1/AdamBistaJPA] | COMPANY [j

1 row | Tx: Auto | DDL

WHERE | ORDER BY

	DISCOUNT	ID
1	10	2

Main.java | CUSTOMER [jdbc:derby://127.0.0.1/AdamBistaJPA] | SUPPLIER [jdbc:derby://127.0.0.1/AdamBistaJPA]

1 row | Tx: Auto | DDL | CSV

WHERE | ORDER BY

	BANKACCOUNTNUMBER	ID
1	1234567890	1