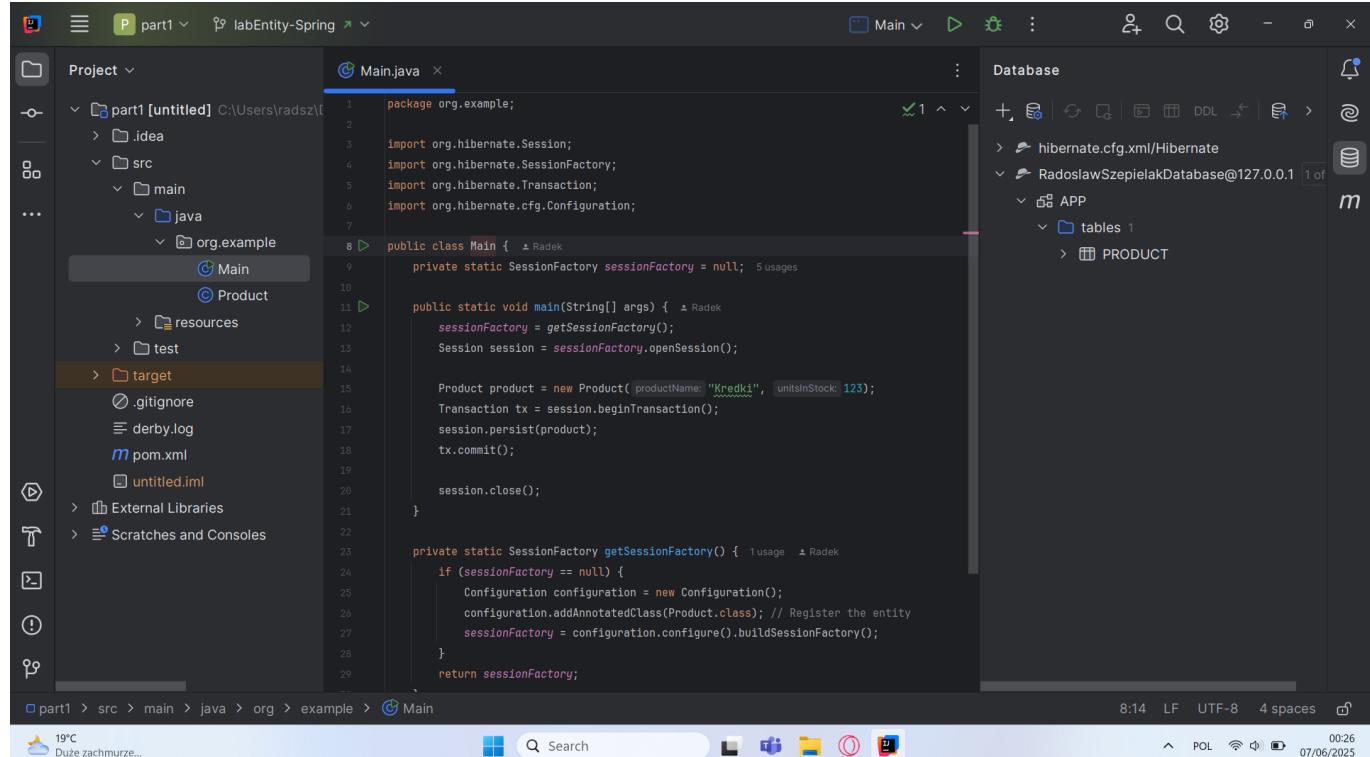


# Sprawozdanie Hibernate

Autorzy: Radosław Szepielak, Kacper Wdowiak

## Część 1:

Poniżej przedstawiamy zrzuty ekranu wykonania pracy podczas zajęć:



```
package org.example;

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

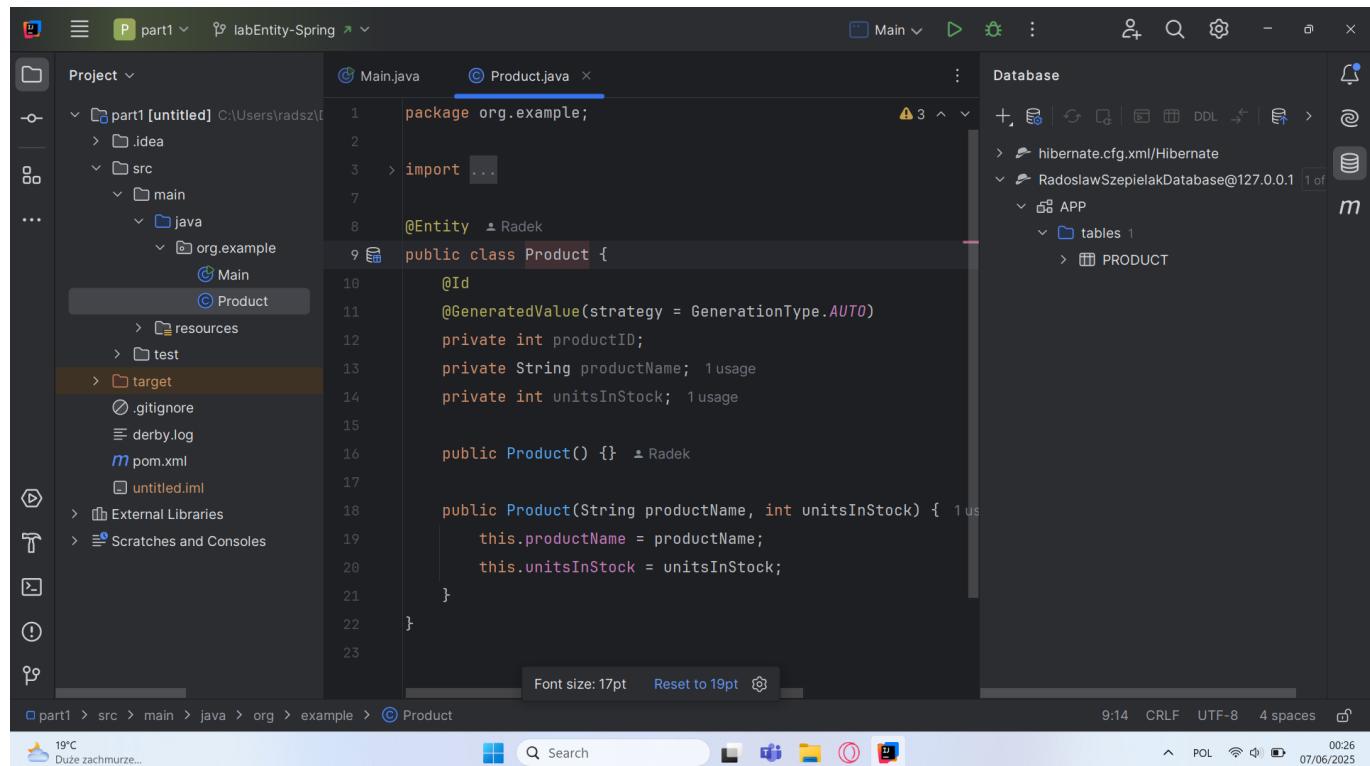
public class Main {
    private static SessionFactory sessionFactory = null; 5 usages

    public static void main(String[] args) { ▲ Radek
        sessionFactory = getSessionFactory();
        Session session = sessionFactory.openSession();

        Product product = new Product( productName: "Kredki", unitsInStock: 123);
        Transaction tx = session.beginTransaction();
        session.persist(product);
        tx.commit();

        session.close();
    }

    private static SessionFactory getSessionFactory() { 1 usage ▲ Radek
        if (sessionFactory == null) {
            Configuration configuration = new Configuration();
            configuration.addAnnotatedClass(Product.class); // Register the entity
            sessionFactory = configuration.configure().buildSessionFactory();
        }
        return sessionFactory;
    }
}
```



```
package org.example;

import ...;

@Entity ▲ Radek
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int productID;
    private String productName; 1 usage
    private int unitsInStock; 1 usage

    public Product() {} ▲ Radek

    public Product(String productName, int unitsInStock) { 1 us
        this.productName = productName;
        this.unitsInStock = unitsInStock;
    }
}
```

## Część 2:

Zadanie I: Modyfikacja modelu i wprowadzenie Dostawcy (is supplied by)

Realizację zadania I) przedstawia kod poniżej:

```
package org.example;

import jakarta.persistence.*;

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

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

    public Product() {}

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

    @Override
    public String toString() {
        return productName + " (liczba sztuk" + unitsInStock+ ")";
    }

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

    public Supplier getSupplier() {
        return supplier;
    }
}
```

```
package org.example;

import jakarta.persistence.*;

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

    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;
    }

    @Override
    public String toString() {
        return companyName;
    }
}
```

```
package org.example;

import jakarta.persistence.Query;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;

import java.util.List;

public class Main {
    private static SessionFactory sessionFactory = null;

    public static void main(String[] args) {
        sessionFactory = getSessionFactory();
        Session session = sessionFactory.openSession();
        Transaction tx = session.beginTransaction();

        Supplier supplier = new Supplier("Extra dostawca", "Tokarskiego",
"Kraków");
        Product productTest = new Product("kredka", 123);

        session.persist(productTest);
    }
}
```

```
Product product = session.get(Product.class, 1);
product.setSupplier(supplier);
session.persist(supplier);
tx.commit();

List<Product> products = session
    .createQuery("from Product", Product.class).getResultList();

for (Product p : products) {
    System.out.println("Produkt '" + p + "' jest dostarczany przez
dostawcę ''"
    + p.getSupplier() + "'");
}
session.close();
}

public static SessionFactory getSessionFactory() {
    if (sessionFactory == null) {
        try {
            Configuration configuration = new Configuration();
            configuration.configure();
            sessionFactory = configuration.buildSessionFactory();
        } catch (Throwable ex) {
            throw new ExceptionInInitializerError(ex);
        }
    }
    return sessionFactory;
}
}
```

Poniżej zamieszczamy zrzuty ekranu realizacji przez nas powyższego kodu wraz z uzyskanym rezultatem:

The screenshot shows the IntelliJ IDEA interface with the project structure on the left and two code editors on the right. The top editor contains `Main.java` and the bottom editor contains `Product.java`. The `Main.java` code runs a Hibernate session to fetch a product from the database and prints its supplier information. The `Product.java` code defines a simple Product entity with a Supplier relationship. The execution output in the bottom panel shows the printed message: "Produkt 'kredka (liczba sztuk123)' jest dostarczany przez dostawcę 'Extra dostawca'". The status bar at the bottom indicates the run configuration is `main`.

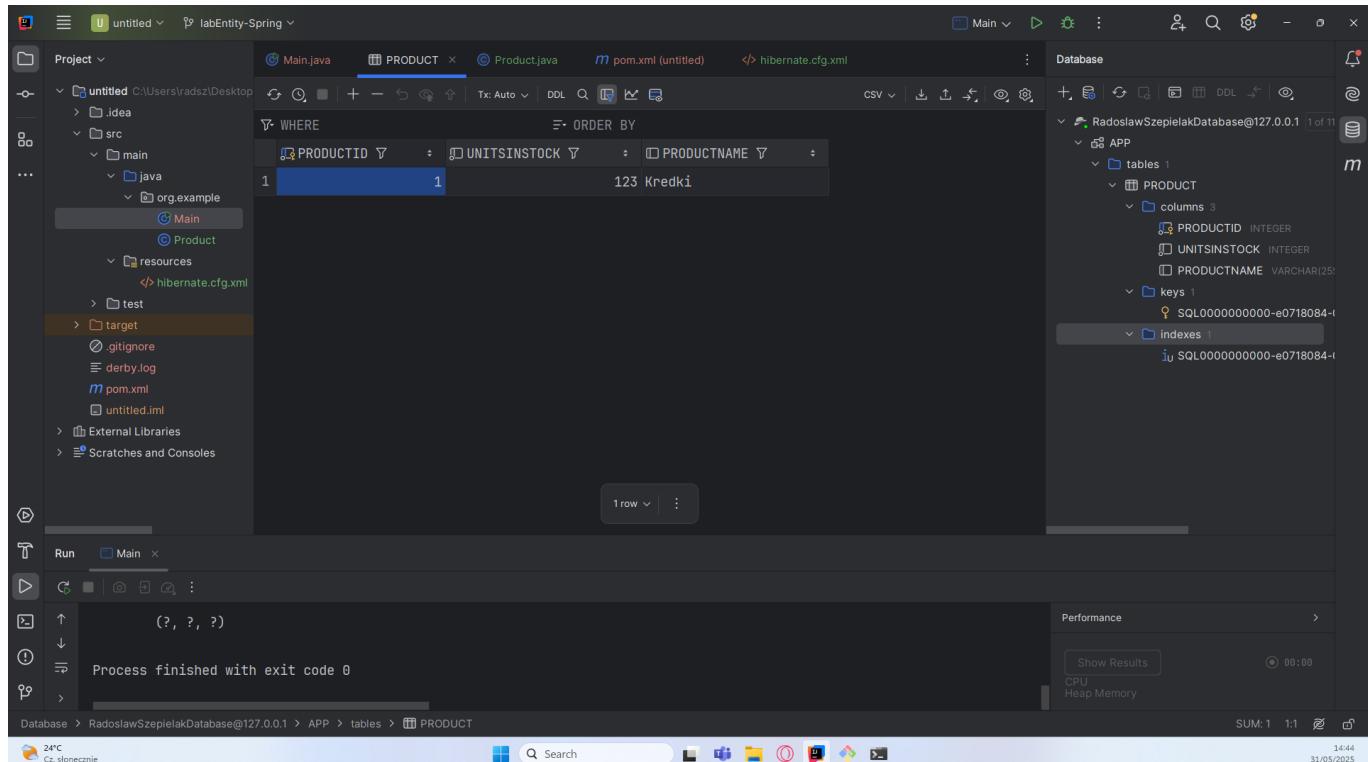
```
public class Main {    public static void main(String[] args) {        Session session = null;        Transaction tx = null;        try {            session = SessionFactoryUtil.getSessionFactory().openSession();            tx = session.beginTransaction();            Product product = session.get(Product.class, 1);            product.setSupplier(supplier);            session.persist(supplier);            tx.commit();        } catch (Exception e) {            e.printStackTrace();            tx.rollback();        } finally {            session.close();        }    }    public static SessionFactory getSessionFactory() {        if (sessionFactory == null) {            try {                sessionFactory = new Configuration().configure("hibernate.cfg.xml").buildSessionFactory();            } catch (Exception e) {                e.printStackTrace();            }        }        return sessionFactory;    } }
```

```
p1_0.unitsInStock
from
Product p1_0
Produkt 'kredka (liczba sztuk123)' jest dostarczany przez dostawcę 'Extra dostawca'
Process finished with exit code 0
```

This screenshot is similar to the one above, but it includes a `Database` tool window on the right side. The `Database` window displays the schema of the `RadoslawSzepielakDatabase`, specifically the `APP` schema which contains the `PRODUCT` and `SUPPLIER` tables. The code in `Main.java` and the execution output remain the same.

```
public class Main {    public static void main(String[] args) {        Session session = null;        Transaction tx = null;        try {            session = SessionFactoryUtil.getSessionFactory().openSession();            tx = session.beginTransaction();            Product product = session.get(Product.class, 1);            product.setSupplier(supplier);            session.persist(supplier);            tx.commit();        } catch (Exception e) {            e.printStackTrace();            tx.rollback();        } finally {            session.close();        }    }    public static SessionFactory getSessionFactory() {        if (sessionFactory == null) {            try {                sessionFactory = new Configuration().configure("hibernate.cfg.xml").buildSessionFactory();            } catch (Exception e) {                e.printStackTrace();            }        }        return sessionFactory;    } }
```

```
p1_0.unitsInStock
from
Product p1_0
Produkt 'kredka (liczba sztuk123)' jest dostarczany przez dostawcę 'Extra dostawca'
```



## Zadanie II: Odwróć relacje zgodnie z poniższym schematem

Realizację zadania II) przedstawia kod poniżej:

Wariant One To Many:

```
package org.example.model.onetomany;

import jakarta.persistence.*;

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

    private String productName;
    private int unitsOnStock;

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

    public Product() {}

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

```
}

public Long getId() { return id; }
public String getProductName() { return productName; }
public int getUnitsOnStock() { return unitsOnStock; }
public Supplier getSupplier() { return supplier; }

public void setProductName(String productName) { this.productName =
productName; }
public void setUnitsOnStock(int unitsOnStock) { this.unitsOnStock =
unitsOnStock; }
public void setSupplier(Supplier supplier) { this.supplier = supplier; }
}
```

```
package org.example.model.onetomany;

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

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

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

    @OneToMany(mappedBy = "supplier", cascade = CascadeType.ALL)
    private List<Product> products = new ArrayList<>();

    public Supplier() {}

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

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

    public Long getId() { return id; }
    public String getCompanyName() { return companyName; }
    public String getStreet() { return street; }
    public String getCity() { return city; }
    public List<Product> getProducts() { return products; }
}
```

```
public void setCompanyName(String companyName) { this.companyName =  
companyName; }  
public void setStreet(String street) { this.street = street; }  
public void setCity(String city) { this.city = city; }  
}
```

```
package org.example.model.onetomany;  
  
import org.hibernate.Session;  
import org.hibernate.SessionFactory;  
import org.hibernate.cfg.Configuration;  
  
public class MainOneToMany {  
    public static void main(String[] args) {  
        Configuration config = new  
Configuration().configure("hibernate_onetomany.cfg.xml");  
        SessionFactory sessionFactory = config.buildSessionFactory();  
  
        try (Session session = sessionFactory.openSession()) {  
            session.beginTransaction();  
  
            System.out.println("==> [OneToMany] Tworzenie dostawcy i 3 produktów  
==>");  
  
            Supplier supplier = new Supplier("Acme Corp", "Main Street",  
"Krakow");  
            Product p1 = new Product("Wiertarka", 15);  
            Product p2 = new Product("Młotek", 30);  
            Product p3 = new Product("Śrubokręt", 50);  
  
            supplier.addProduct(p1);  
            supplier.addProduct(p2);  
            supplier.addProduct(p3);  
  
            session.persist(supplier);  
  
            session.getTransaction().commit();  
  
            System.out.println("Dostawca został zapisany: " +  
supplier.getCompanyName());  
            System.out.println("Produkty:");  
            for (Product p : supplier.getProducts()) {  
                System.out.println(" - " + p.getProductName() + ", "  
                    + p.getUnitsOnStock() + " szt.");  
            }  
        }  
  
        sessionFactory.close();
```

```
}
```

Poniżej zamieszczamy zrzuty ekranu realizacji przez nas powyższego kodu wraz z uzyskanym rezultatem:

The screenshot shows the IntelliJ IDEA interface. The left sidebar displays the project structure under 'src/main/java'. The 'MainOneToMany.java' file is open in the editor, containing Java code for creating a supplier and three products. The code uses Hibernate annotations (@Entity, @ManyToOne, @JoinColumn) and configuration files (hibernate\_manytomany.cfg.xml, hibernate\_onetomany.cfg.xml). The bottom pane shows the run output, which includes the printed supplier details and a list of products.

```

public class MainOneToMany {
    public static void main(String[] args) {
        System.out.println("===[ [OneToMany] Tworzenie dostawcy i 3 produktów ===");
        Supplier supplier = new Supplier( companyName: "Acme Corp", street: "Main Street", city: "Krakow");
        Product p1 = new Product( productName: "Wiertarka", unitsOnStock: 15);
        Product p2 = new Product( productName: "Młotek", unitsOnStock: 30);
        Product p3 = new Product( productName: "Śrubokręt", unitsOnStock: 50);

        supplier.addProduct(p1);
        supplier.addProduct(p2);
        supplier.addProduct(p3);

        session.persist(supplier);

        session.getTransaction().commit();

        System.out.println("Dostawca został zapisany: " + supplier.getCompanyName());
        System.out.println("Produkty:");
    }
}

```

Dostawca został zapisany: Acme Corp  
Produkty:  
- Wiertarka, 15 szt.  
- Młotek, 30 szt.  
- Śrubokręt, 50 szt.

The screenshot shows the DB Manager interface with the 'PRODUCT' table selected. The table contains three rows of data: Wiertarka, Młotek, and Śrubokręt, each associated with a supplier ID of 1.

ID	PRODUCTNAME	UNITSONSTOCK	SUPPLIER_ID
1	Wiertarka	15	1
2	Młotek	30	1
3	Śrubokręt	50	1

The screenshot shows the DB Manager interface with the 'SUPPLIER' table selected. The table contains one row of data: Acme Corp located in Krakow.

ID	CITY	COMPANYNAME	STREET
1	Krakow	Acme Corp	Main Street

Variant Many To Many:

```
package org.example.model.manytomany;

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

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

    private String productName;
    private int unitsOnStock;

    @ManyToMany(mappedBy = "products")
    private List<Supplier> suppliers = new ArrayList<>();

    public Product() {}

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

    public Long getId() { return id; }
    public String getProductName() { return productName; }
    public int getUnitsOnStock() { return unitsOnStock; }
    public List<Supplier> getSuppliers() { return suppliers; }

    public void setProductName(String productName) { this.productName =
productName; }
    public void setUnitsOnStock(int unitsOnStock) { this.unitsOnStock =
unitsOnStock; }
}
```

```
package org.example.model.manytomany;

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

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

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

@ManyToMany(cascade = CascadeType.ALL)
@JoinTable(
    name = "supplier_product",
    joinColumns = @JoinColumn(name = "supplier_id"),
    inverseJoinColumns = @JoinColumn(name = "product_id")
)
private List<Product> products = new ArrayList<>();

public Supplier() {}

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

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

public Long getId() { return id; }
public String getCompanyName() { return companyName; }
public String getStreet() { return street; }
public String getCity() { return city; }
public List<Product> getProducts() { return products; }

public void setCompanyName(String companyName) { this.companyName =
companyName; }
public void setStreet(String street) { this.street = street; }
public void setCity(String city) { this.city = city; }
}
```

```
package org.example.model.manytomany;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class MainManyToMany {
    public static void main(String[] args) {
        Configuration config = new
Configuration().configure("hibernate_manytomany.cfg.xml");
        SessionFactory sessionFactory = config.buildSessionFactory();

        try (Session session = sessionFactory.openSession()) {
            session.beginTransaction();
```

```
System.out.println("===[ManyToMany] Tworzenie dostawcy i 3 produktów");
===");

Supplier supplier = new Supplier("Beta Tools", "Industrial Rd",
"Warszawa");
Product p1 = new Product("Piła", 20);
Product p2 = new Product("Kombinerki", 40);
Product p3 = new Product("Klucz francuski", 25);

supplier.addProduct(p1);
supplier.addProduct(p2);
supplier.addProduct(p3);

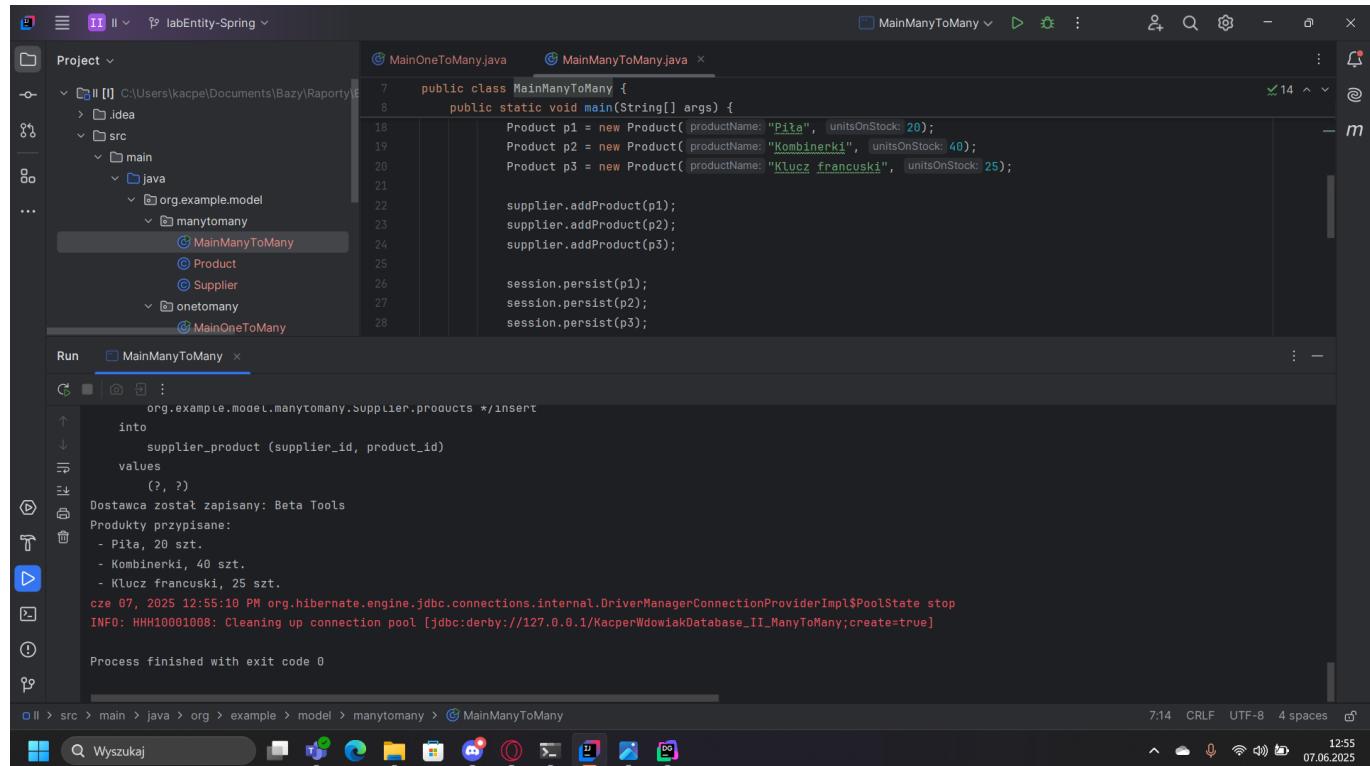
session.persist(p1);
session.persist(p2);
session.persist(p3);
session.persist(supplier);

session.getTransaction().commit();

System.out.println("Dostawca został zapisany: " +
supplier.getCompanyName());
System.out.println("Produkty przypisane:");
for (Product p : supplier.getProducts()) {
    System.out.println(" - " + p.getProductName() + ", " +
p.getUnitsOnStock() + " szt.");
}
}

sessionFactory.close();
}
}
```

Poniżej zamieszczamy zrzuty ekranu realizacji przez nas powyższego kodu wraz z uzyskanym rezultatem:



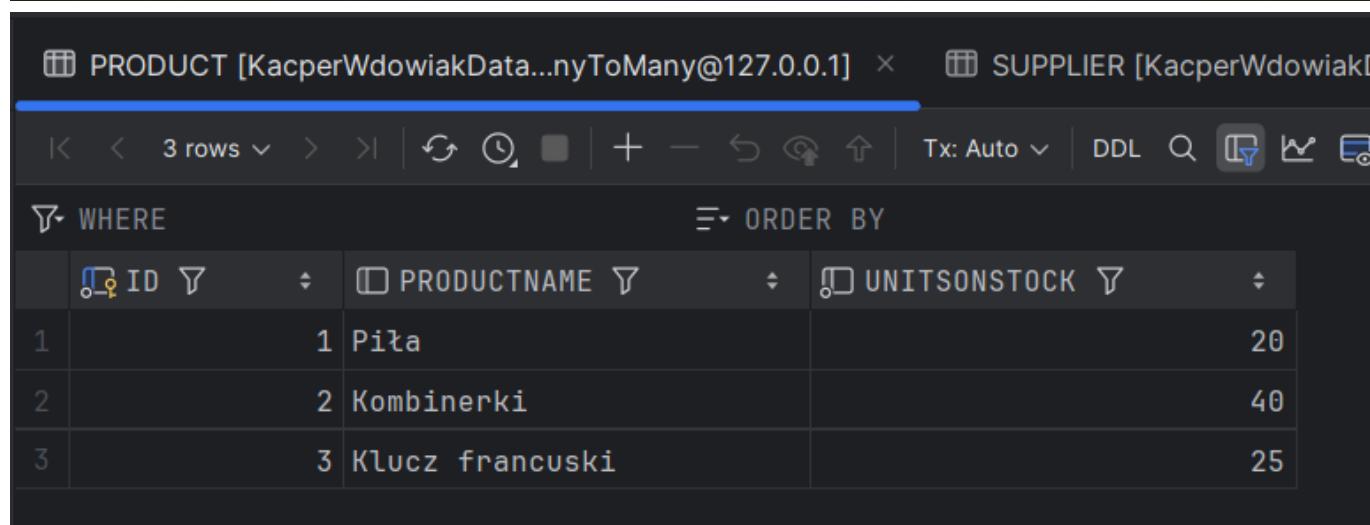
The screenshot shows the IntelliJ IDEA interface with the following details:

- Project View:** Shows a file structure for a project named "labEntity-Spring". Inside "src/main/java/org.example.model", there are three packages: "manytomany", "onetomany", and "MainOneToMany". The "manytomany" package contains classes "MainManyToMany", "Product", and "Supplier".
- MainManyToMany.java:**

```
public class MainManyToMany {
    public static void main(String[] args) {
        Product p1 = new Product( productName: "Piła", unitsOnStock: 20 );
        Product p2 = new Product( productName: "Kombinerki", unitsOnStock: 40 );
        Product p3 = new Product( productName: "Klucz francuski", unitsOnStock: 25 );

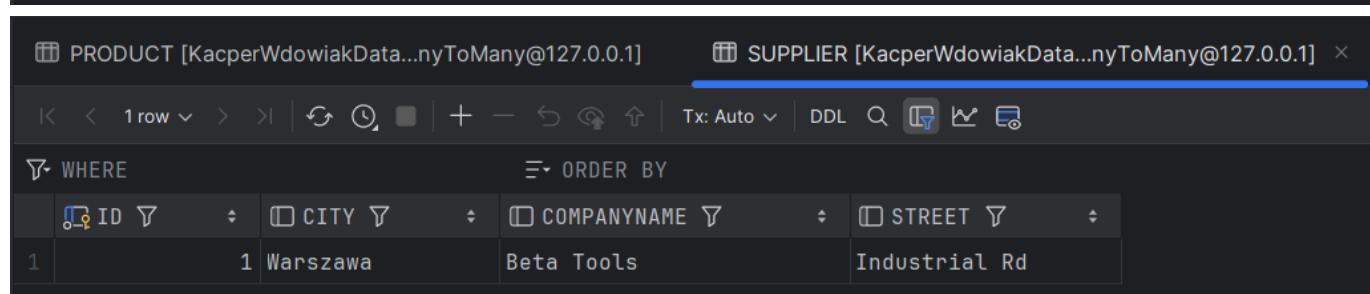
        supplier.addProduct(p1);
        supplier.addProduct(p2);
        supplier.addProduct(p3);

        session.persist(p1);
        session.persist(p2);
        session.persist(p3);
    }
}
```
- Run Tab:** Shows the output of running the application. It includes SQL insert statements and the resulting data in the database.
- Output Log:** Shows Hibernate logs indicating successful connection and session persist operations.
- System Bar:** Shows the system tray with icons for battery, signal, and date/time (07.06.2025).



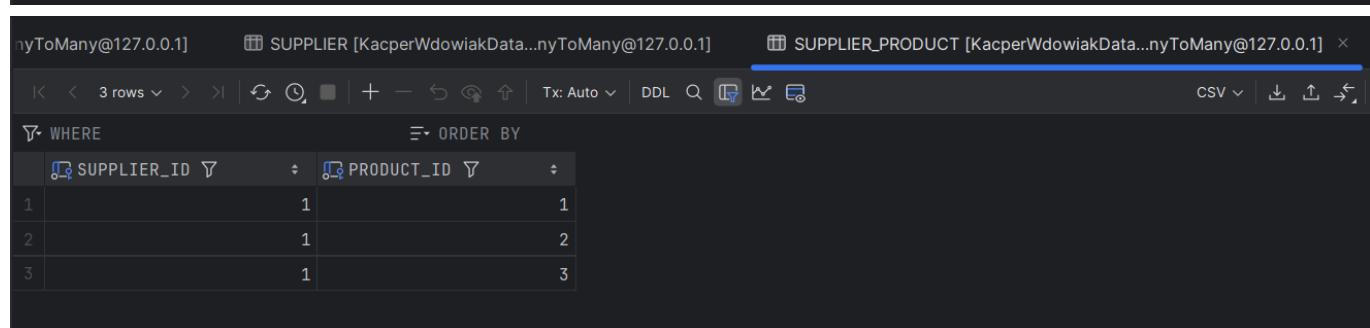
The screenshot shows the H2 Database Browser with the "PRODUCT" table open. The table has three columns: ID, PRODUCTNAME, and UNITSONSTOCK. The data is as follows:

ID	PRODUCTNAME	UNITSONSTOCK
1	Piła	20
2	Kombinerki	40
3	Klucz francuski	25



The screenshot shows the H2 Database Browser with the "SUPPLIER" table open. The table has four columns: ID, CITY, COMPANYNAME, and STREET. The data is as follows:

ID	CITY	COMPANYNAME	STREET
1	Warszawa	Beta Tools	Industrial Rd



The screenshot shows the H2 Database Browser with the "SUPPLIER\_PRODUCT" table open. The table has three columns: SUPPLIER\_ID, PRODUCT\_ID, and QUANTITY. The data is as follows:

SUPPLIER_ID	PRODUCT_ID	QUANTITY
1	1	1
1	2	2
1	3	3

### Zadanie III: Zamodelowanie relacji dwustronnej (supplies & is supplied by)

Realizację zadania III) przedstawia kod poniżej:

```
package org.example;

import jakarta.persistence.*;

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

    private String productName;
    private int unitsOnStock;

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

    public Product() {}

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

    public Long getId() { return id; }
    public String getProductName() { return productName; }
    public int getUnitsOnStock() { return unitsOnStock; }
    public Supplier getSupplier() { return supplier; }

    public void setProductName(String productName) { this.productName =
productName; }
    public void setUnitsOnStock(int unitsOnStock) { this.unitsOnStock =
unitsOnStock; }
    public void setSupplier(Supplier supplier) { this.supplier = supplier; }
}
```

```
package org.example;

import jakarta.persistence.*;
import java.util.ArrayList;
import java.util.List;
```

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

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

    @OneToMany(mappedBy = "supplier", cascade = CascadeType.ALL)
    private List<Product> products = new ArrayList<>();

    public Supplier() {}

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

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

    public Long getId() { return id; }
    public String getCompanyName() { return companyName; }
    public String getStreet() { return street; }
    public String getCity() { return city; }
    public List<Product> getProducts() { return products; }

    public void setCompanyName(String companyName) { this.companyName =
companyName; }
    public void setStreet(String street) { this.street = street; }
    public void setCity(String city) { this.city = city; }
}
```

```
package org.example;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class Main {
    public static void main(String[] args) {
        Configuration config = new Configuration().configure();
        SessionFactory sessionFactory = config.buildSessionFactory();

        try (Session session = sessionFactory.openSession()) {
```

```
        session.beginTransaction();

        System.out.println("==[III] Tworzenie dostawcy i produktów (relacja
dwustronna)==");

        Supplier supplier = new Supplier("Gamma Sp. z o.o.", "Kwiatowa",
"Wrocław");
        Product p1 = new Product("Młot pneumatyczny", 7);
        Product p2 = new Product("Szlifierka taśmowa", 14);
        Product p3 = new Product("Piła tarczowa", 11);

        supplier.addProduct(p1);
        supplier.addProduct(p2);
        supplier.addProduct(p3);

        session.persist(supplier);

        session.getTransaction().commit();

        System.out.println("Dostawca zapisany: " + supplier.getCompanyName());
        System.out.println("Produkty:");
        for (Product p : supplier.getProducts()) {
            System.out.println(" - " + p.getProductName() + ", "
+ p.getUnitsOnStock() + " szt.");
        }
    }

    sessionFactory.close();
}
}
```

Poniżej zamieszamy zrzuty ekranu realizacji przez nas powyższego kodu wraz z uzyskanym rezultatem:

The screenshot shows the IntelliJ IDEA interface with the project 'labEntity-Spring' open. The code editor displays the Main.java file:

```
public class Main {
    public static void main(String[] args) {
        System.out.println("===[ III ] Tworzenie dostawcy i produktów (relacja dwustronna) ===");
        Supplier supplier = new Supplier( companyName: "Gamma Sp. z o.o.", street: "Kwiatowa", city: "Wrocław");
        Product p1 = new Product( productName: "Młot pneumatyczny", unitsOnStock: 7);
        Product p2 = new Product( productName: "Szlifierka taśmowa", unitsOnStock: 14);
        Product p3 = new Product( productName: "Piła tarczowa", unitsOnStock: 11);

        supplier.addProduct(p1);
        supplier.addProduct(p2);
        supplier.addProduct(p3);

        session.persist(supplier); // kaskadowo zapisze produkty
    }
}
```

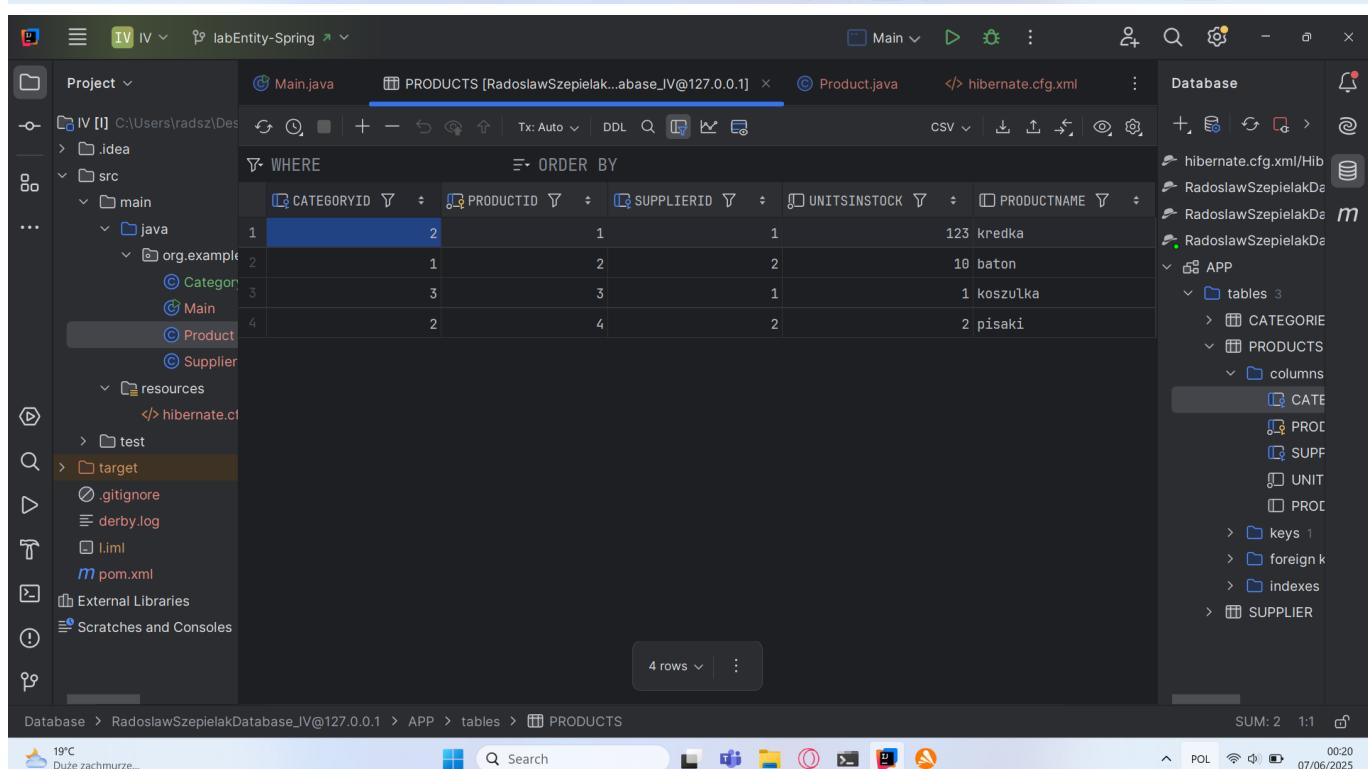
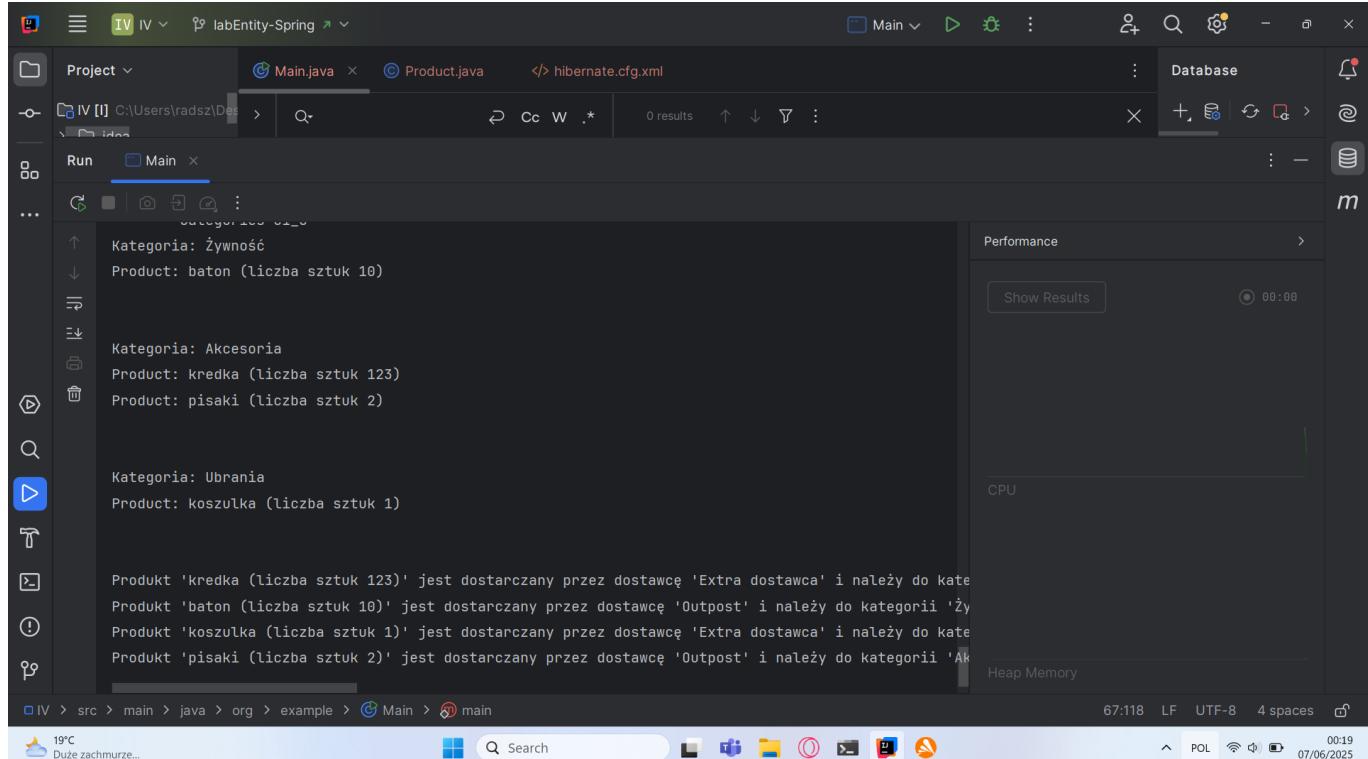
The run configuration 'Main' is selected. The terminal shows the output of the application:

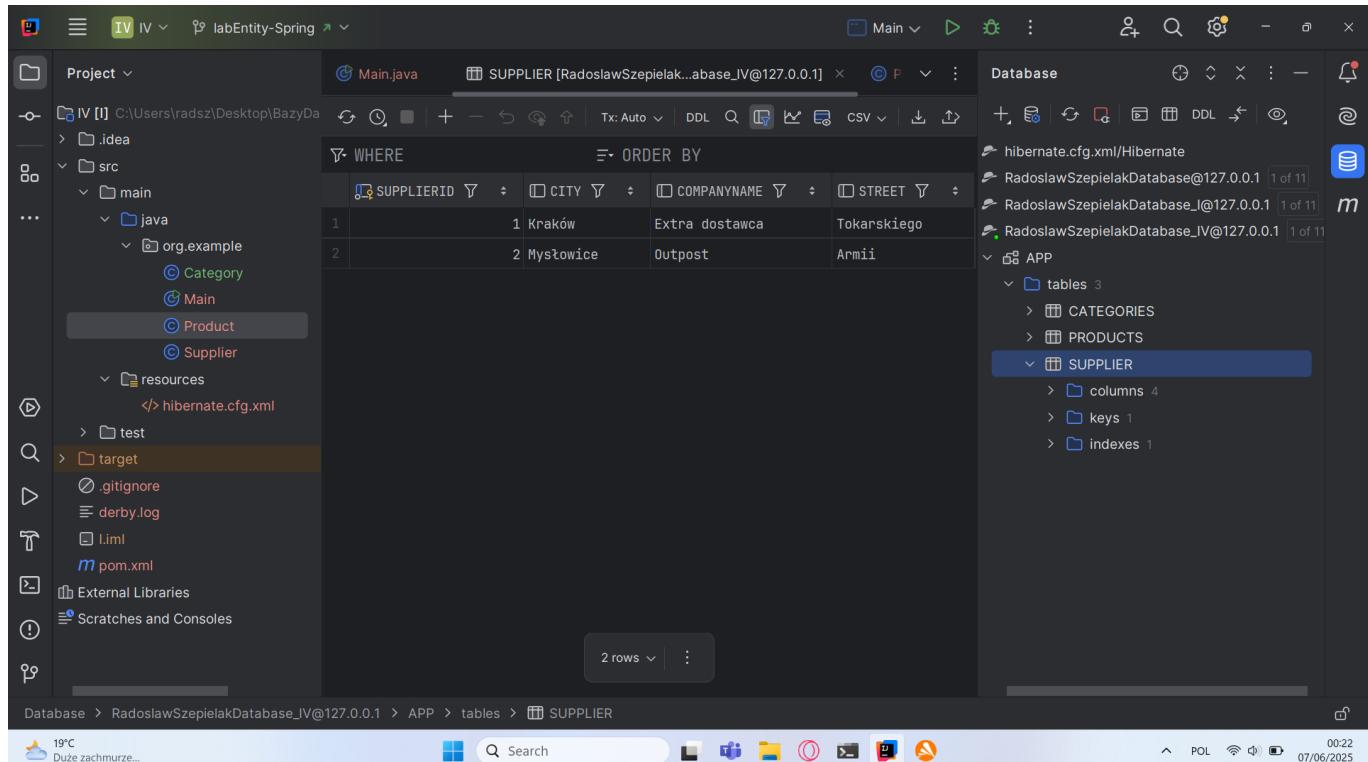
```
Dostawca zapisany: Gamma Sp. z o.o.
Produkty:
- Młot pneumatyczny, 7 szt.
- Szlifierka taśmowa, 14 szt.
- Piła tarczowa, 11 szt.
```

The database browser shows two tables:

PRODUCT [KacperWdowiakDatabase_III@127.0.0.1]			
WHERE	ORDER BY		
ID	PRODUCTNAME	UNITSONSTOCK	SUPPLIER_ID
1	Młot pneumatyczny	7	1
2	Szlifierka taśmowa	14	1
3	Piła tarczowa	11	1

SUPPLIER [KacperWdowiakDatabase_III@127.0.0.1]			
WHERE	ORDER BY		
ID	CITY	COMPANYNAME	STREET
1	Wrocław	Gamma Sp. z o.o.	Kwiatowa





Zadanie IV: Dodaj klase Category z property int CategoryID, String Name oraz listą produktów

Realizację zadania IV) przedstawia kod poniżej:

```
package org.example;

import jakarta.persistence.*;

@Entity
@Table(name = "Products")
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int productID;
    private String productName;
    private int unitsInStock;

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

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

```
public Product() {}

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

@Override
public String toString() {
    return productName + " (liczba sztuk " + unitsInStock+ ")";
}

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

public Supplier getSupplier() {
    return supplier;
}

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

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

```
package org.example;

import jakarta.persistence.*;

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

    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;
    }
}
```

```
    @Override
    public String toString() {
        return companyName;
    }
}
```

```
package org.example;

import jakarta.persistence.*;
import java.util.ArrayList;
import java.util.Collection;
import java.util.List;

@Entity
@Table(name = "Categories")
public class Category {
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator =
"Category_GEN")
    @SequenceGenerator(name = "Category_GEN", sequenceName = "Category_SEQ")
    private int categoryID;

    private String name;

    @OneToMany(mappedBy = "category")
    private List<Product> products = new ArrayList<Product>();

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

    public Category() {
    }

    @Override
    public String toString() {
        return name;
    }

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

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

```
package org.example;

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

import java.util.List;

public class Main {
    private static SessionFactory sessionFactory = null;

    public static void main(String[] args) {
        sessionFactory = getSessionFactory();
        Session session = sessionFactory.openSession();
        Transaction tx = session.beginTransaction();

        Supplier supplier = new Supplier("Extra dostawca", "Tokarskiego",
"Kraków");
        Supplier otherSupp = new Supplier("Outpost", "Armii", "Mysłowice");
        Product productTest1 = new Product("kredka", 123);
        Product productTest2 = new Product("baton", 10);
        Product productTest3 = new Product("koszulka", 1);
        Product productTest4 = new Product("pisaki", 2);
        Category category1 = new Category("Żywność");
        Category category2 = new Category("Akcesoria");
        Category category3 = new Category("Ubrania");

        session.persist(category1);
        session.persist(category2);
        session.persist(category3);
        session.persist(supplier);
        session.persist(otherSupp);

        productTest1.setCategory(category2);
        productTest2.setCategory(category1);
        productTest3.setCategory(category3);
        productTest4.setCategory(category2);

        productTest1.setSupplier(supplier);
        productTest2.setSupplier(otherSupp);
        productTest3.setSupplier(supplier);
        productTest4.setSupplier(otherSupp);

        category1.addProduct(productTest2);
        category2.addProduct(productTest1);
        category2.addProduct(productTest4);
        category3.addProduct(productTest3);

        session.persist(productTest1);
        session.persist(productTest2);
        session.persist(productTest3);
```

```
session.persist(productTest4);

List<Product> products = session.createQuery("from Product",
Product.class)
    .getResultSet();

session.createQuery("from Category", Category.class).getResultSet()
    .forEach(cat -> {
        System.out.println("Kategoria: " + cat);
        cat.getProducts().forEach(p -> {
            System.out.println("Produkt: " + p);
        });
        System.out.println("\n");
    });

tx.commit();

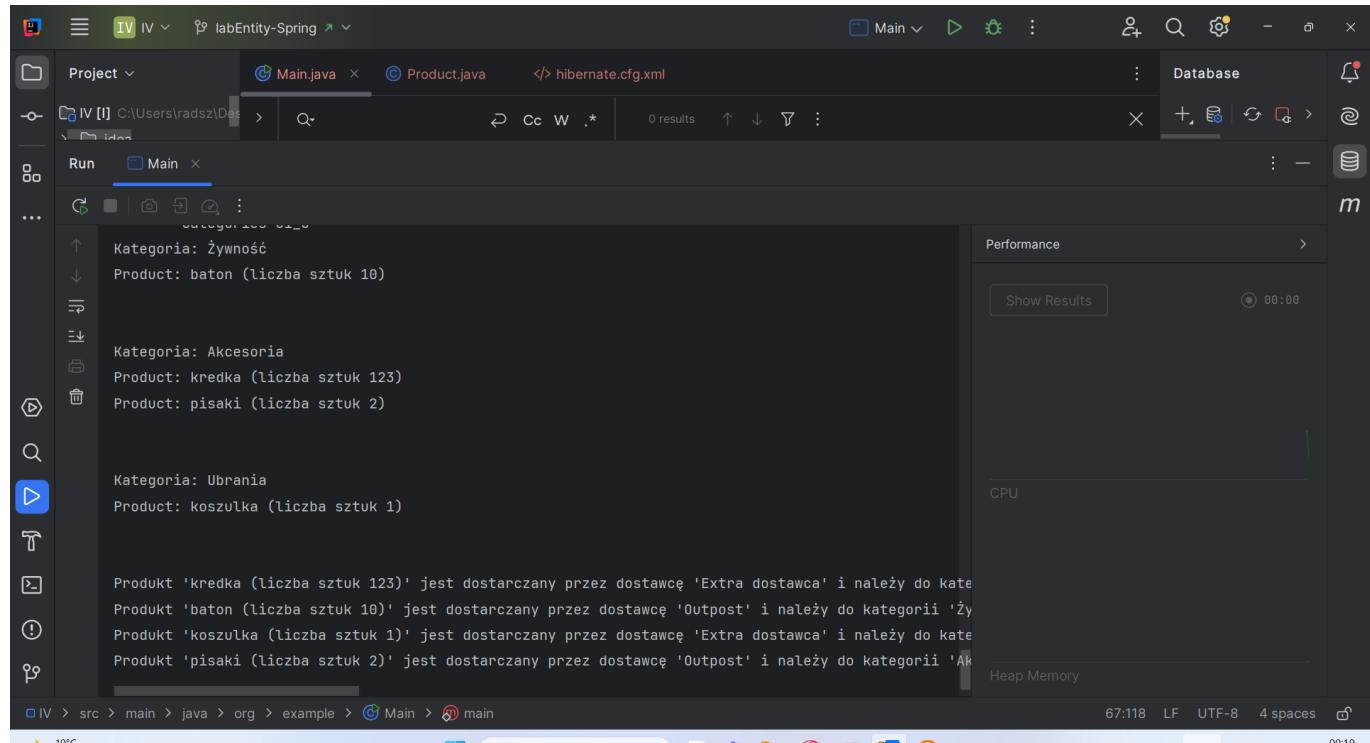
//      for (Category category : categories) { //kategorie i ich produkty jeśli
zrobiliśmy
//  jak dla products wyżej
//      System.out.println("Kategoria: " + category + " ");
//      for (Product product : category.getProducts()) {
//          System.out.println(product);
//      }
//  }

for (Product p : products) { //produkty i ich kategorie
    System.out.println("Produkt '" + p + "' jest dostarczany przez
dostawcę ''"
        + p.getSupplier() + """
        + " i należy do kategorii '" + p.getCategory() + "'");
}

session.close();
}

public static SessionFactory getSessionFactory() {
    if (sessionFactory == null) {
        try {
            Configuration configuration = new Configuration();
            configuration.configure();
            sessionFactory = configuration.buildSessionFactory();
        } catch (Throwable ex) {
            throw new ExceptionInInitializerError(ex);
        }
    }
    return sessionFactory;
}
}
```

Poniżej zamieszamy zrzuty ekranu realizacji przez nas powyższego kodu wraz z uzyskanym rezultatem:



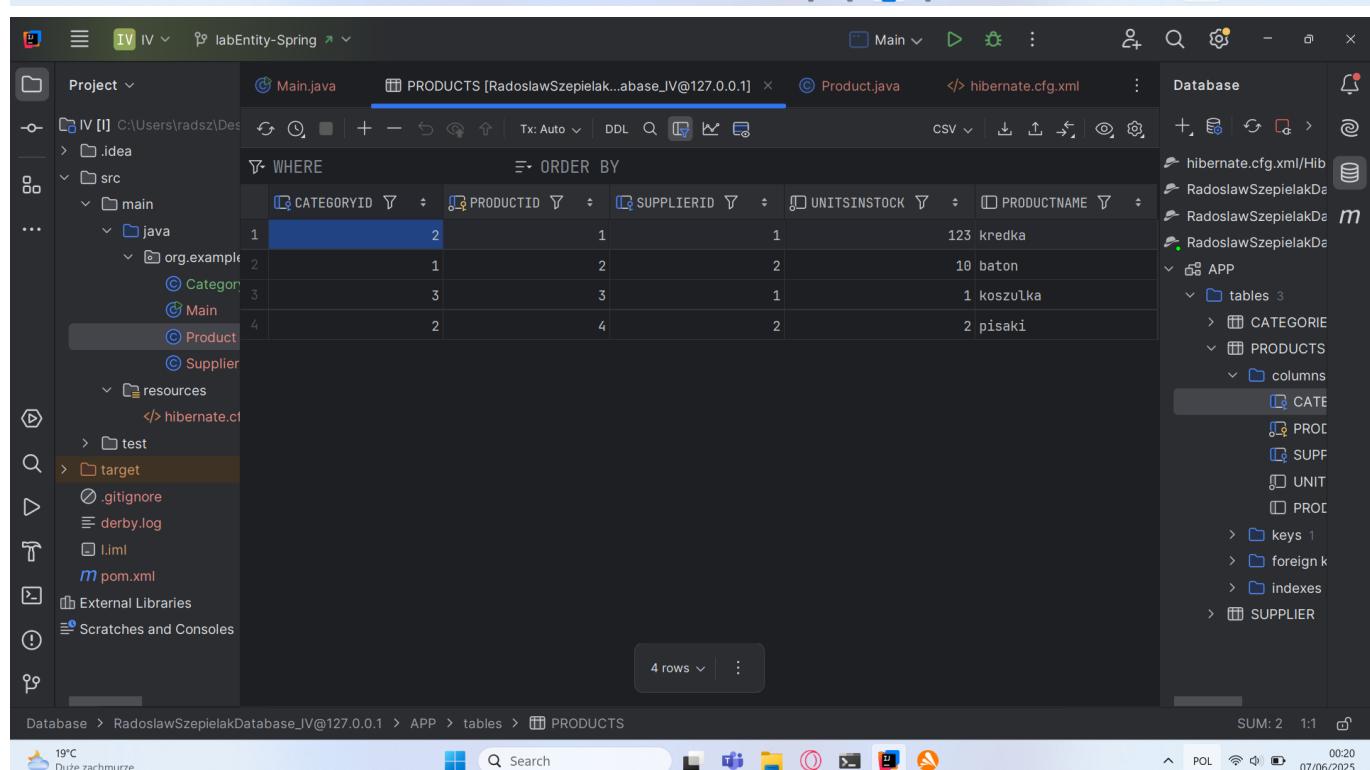
```

Kategoria: Żywność
Product: baton (liczba sztuk 10)

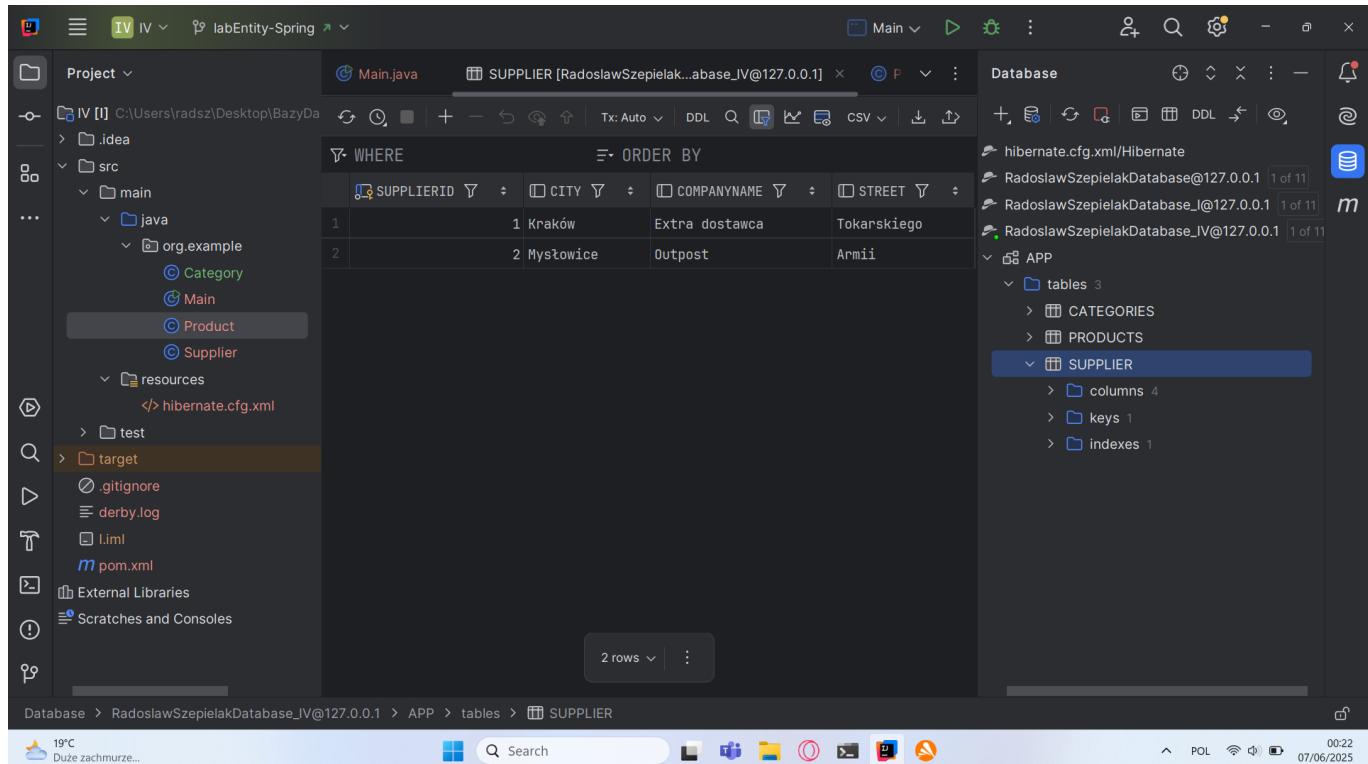
Kategoria: Akcesoria
Product: kredka (liczba sztuk 123)
Product: pisaki (liczba sztuk 2)

Kategoria: Ubrania
Product: koszulka (liczba sztuk 1)

Produkt 'kredka (liczba sztuk 123)' jest dostarczany przez dostawcę 'Extra dostawca' i należy do kategorii 'Akcesoria'.
Produkt 'baton (liczba sztuk 10)' jest dostarczany przez dostawcę 'Outpost' i należy do kategorii 'Żywność'.
Produkt 'koszulka (liczba sztuk 1)' jest dostarczany przez dostawcę 'Extra dostawca' i należy do kategorii 'Ubrania'.
Produkt 'pisaki (liczba sztuk 2)' jest dostarczany przez dostawcę 'Outpost' i należy do kategorii 'Akcesoria'.
  
```



CATEGORYID	PRODUCTID	SUPPLIERID	UNITSINSTOCK	PRODUCTNAME
1	2	1	1	123 kredka
2	1	2	2	10 baton
3	3	3	1	1 koszulka
4	2	4	2	2 pisaki



## Zadanie V: (includes & can be sold in)

Realizację zadania V) przedstawia kod poniżej:

```
package org.example;

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

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

    private String productName;
    private int unitsOnStock;

    @OneToMany(mappedBy = "product", cascade = CascadeType.ALL)
    private Set<InvoiceProduct> invoiceProducts = new HashSet<>();

    public Product() {}

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

```
        this.unitsOnStock = unitsOnStock;
    }

    public Long getId() {
        return id;
    }

    public String getProductName() {
        return productName;
    }

    public int getUnitsOnStock() {
        return unitsOnStock;
    }

    public Set<InvoiceProduct> getInvoiceProducts() {
        return invoiceProducts;
    }

    public void addInvoiceProduct(InvoiceProduct invoiceProduct) {
        invoiceProducts.add(invoiceProduct);
    }
}
```

```
package org.example;

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

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

    private String invoiceNumber;

    @OneToMany(mappedBy = "invoice", cascade = CascadeType.ALL)
    private Set<InvoiceProduct> invoiceProducts = new HashSet<>();

    public Invoice() {}

    public Invoice(String invoiceNumber) {
        this.invoiceNumber = invoiceNumber;
    }

    public Long getId() {
        return id;
    }
```

```
public String getInvoiceNumber() {
    return invoiceNumber;
}

public Set<InvoiceProduct> getInvoiceProducts() {
    return invoiceProducts;
}

public void addInvoiceProduct(InvoiceProduct invoiceProduct) {
    invoiceProducts.add(invoiceProduct);
}

}
```

```
package org.example;

import jakarta.persistence.*;

@Entity
@IdClass(InvoiceProductId.class)
public class InvoiceProduct {

    @Id
    @ManyToOne
    private Invoice invoice;

    @Id
    @ManyToOne
    private Product product;

    private int quantity;

    public InvoiceProduct() {}

    public InvoiceProduct(Invoice invoice, Product product, int quantity) {
        this.invoice = invoice;
        this.product = product;
        this.quantity = quantity;
        invoice.addInvoiceProduct(this);
        product.addInvoiceProduct(this);
    }

    public Invoice getInvoice() {
        return invoice;
    }

    public Product getProduct() {
        return product;
    }
}
```

```
    public int getQuantity() {
        return quantity;
    }
}
```

```
package org.example;

import java.io.Serializable;
import java.util.Objects;

public class InvoiceProductId implements Serializable {
    private Long invoice;
    private Long product;

    public InvoiceProductId() {}

    public InvoiceProductId(Long invoice, Long product) {
        this.invoice = invoice;
        this.product = product;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof InvoiceProductId)) return false;
        InvoiceProductId that = (InvoiceProductId) o;
        return Objects.equals(invoice, that.invoice) &&
               Objects.equals(product, that.product);
    }

    @Override
    public int hashCode() {
        return Objects.hash(invoice, product);
    }
}
```

```
package org.example;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class Main {
    public static void main(String[] args) {
        Configuration config = new Configuration().configure();
        SessionFactory sessionFactory = config.buildSessionFactory();

        try (Session session = sessionFactory.openSession()) {
            session.beginTransaction();
```

```
Product p1 = new Product("Młotek", 30);
Product p2 = new Product("Wiertarka", 20);

Invoice i1 = new Invoice("FV/001");
Invoice i2 = new Invoice("FV/002");

session.persist(p1);
session.persist(p2);
session.persist(i1);
session.persist(i2);

InvoiceProduct ip1 = new InvoiceProduct(i1, p1, 5); // młotek w FV/001
InvoiceProduct ip2 = new InvoiceProduct(i1, p2, 2); // wiertarka w
FV/001
InvoiceProduct ip3 = new InvoiceProduct(i2, p1, 1); // młotek w FV/002

session.persist(ip1);
session.persist(ip2);
session.persist(ip3);

session.getTransaction().commit();

session.beginTransaction();

Invoice loadedInvoice = session.createQuery(
    "FROM Invoice i WHERE i.invoiceNumber = :nr",
Invoice.class)
    .setParameter("nr", "FV/001")
    .getSingleResult();

System.out.println("Produkty z faktury " +
loadedInvoice.getInvoiceNumber() + ":");
for (InvoiceProduct ip : loadedInvoice.getInvoiceProducts()) {
    System.out.println(" - " + ip.getProduct().getProductName() +
        " x " + ip.getQuantity());
}

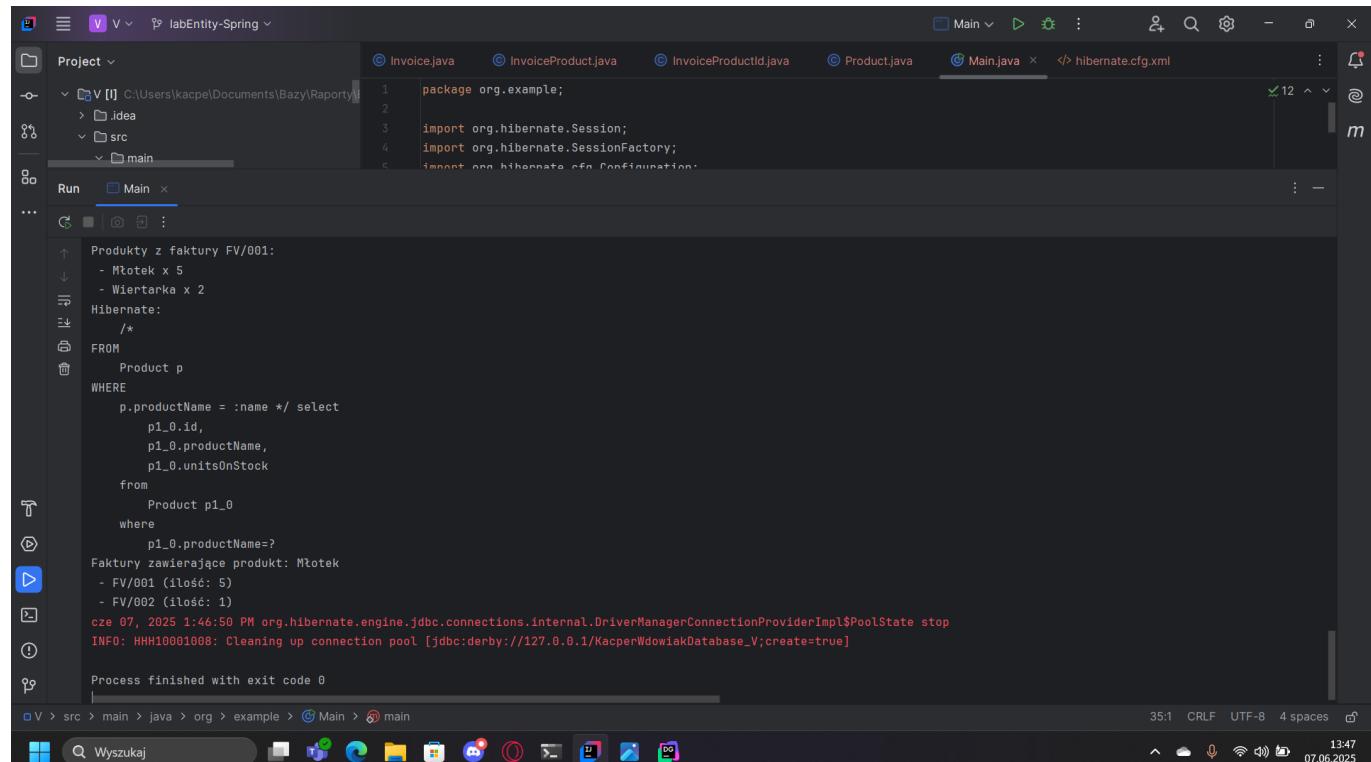
Product loadedProduct = session.createQuery(
    "FROM Product p WHERE p.productName = :name",
Product.class)
    .setParameter("name", "Młotek")
    .getSingleResult();

System.out.println("Faktury zawierające produkt: " +
loadedProduct.getProductName());
for (InvoiceProduct ip : loadedProduct.getInvoiceProducts()) {
    System.out.println(" - " + ip.getInvoice().getInvoiceNumber() +
        " (ilość: " + ip.getQuantity() + ")");
}

session.getTransaction().commit();
}
```

```
        sessionFactory.close();
    }
}
```

Poniżej zamieszamy zrzuty ekranu realizacji przez nas powyższego kodu wraz z uzyskanym rezultatem:

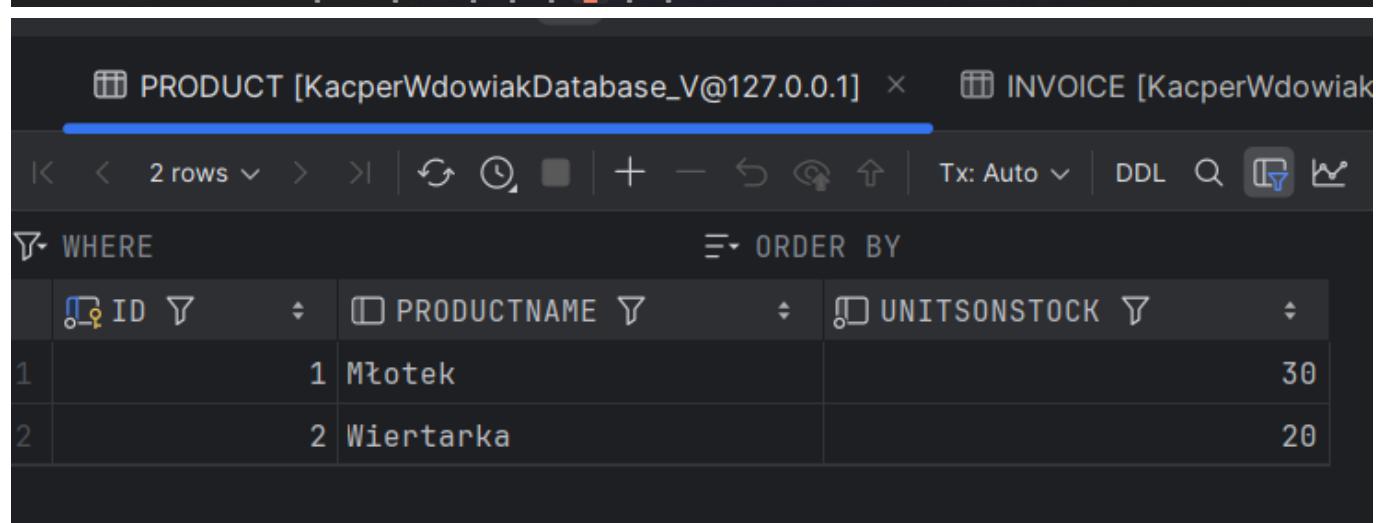


The screenshot shows the IntelliJ IDEA interface with the project 'labEntity-Spring' open. The code editor displays the Main.java file, which contains the following code:

```
package org.example;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;
```

The code uses HQL to query a database for products named 'Młotek' or 'Wierte' (likely a typo for 'Wiertarka'). It retrieves the product ID, name, and units on stock. The results show two products: 'Młotek' with ID 1 and 'Wierte' with ID 2.

Below the code editor, the terminal window shows the command-line output of the application running. It includes logs from Hibernate and the JDBC connection provider, indicating the connection pool was cleaned up.



The screenshot shows the DBeaver interface with two tabs: 'PRODUCT [KacperWdowiakDatabase\_V@127.0.0.1]' and 'INVOICE [KacperWdowiakDatabase\_V@127.0.0.1]'. The 'PRODUCT' tab is active, displaying a table with three columns: ID, PRODUCTNAME, and UNITSONSTOCK. The data is as follows:

	ID	PRODUCTNAME	UNITSONSTOCK
1	1	Młotek	30
2	2	Wierte	20

The screenshot shows a database interface with two tabs: 'PRODUCT' and 'INVOICE'. The 'INVOICE' tab is active, displaying the following data:

ID	INVOICENUMBER
1	FV/001
2	FV/002

The 'INVOICE' tab is also active, displaying the following data:

QUANTITY	PRODUCT_ID	INVOICE_ID
2	2	1
1	1	2
5		1

## Zadanie VI: JPA

Realizację zadania VI przedstawia kod poniżej:

```
package org.example;

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

@Entity
public class Product {
    @Id
    @GeneratedValue
    private Long id;

    private String productName;
    private int unitsOnStock;

    @OneToMany(mappedBy = "product", cascade = CascadeType.ALL)
    private Set<InvoiceProduct> invoiceProducts = new HashSet<>();

    public Product() {}
```

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

public String getProductName() {
    return productName;
}

public Set<InvoiceProduct> getInvoiceProducts() {
    return invoiceProducts;
}

}
```

```
package org.example;

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

@Entity
public class Invoice {
    @Id
    @GeneratedValue
    private Long id;

    private String invoiceNumber;

    @OneToMany(mappedBy = "invoice", cascade = CascadeType.ALL)
    private Set<InvoiceProduct> invoiceProducts = new HashSet<>();

    public Invoice() {}
    public Invoice(String invoiceNumber) {
        this.invoiceNumber = invoiceNumber;
    }

    public String getInvoiceNumber() {
        return invoiceNumber;
    }

    public Set<InvoiceProduct> getInvoiceProducts() {
        return invoiceProducts;
    }
}
```

```
package org.example;

import jakarta.persistence.*;
```

```
@Entity
@IdClass(InvoiceProductId.class)
public class InvoiceProduct {
    @Id
    @ManyToOne
    private Invoice invoice;

    @Id
    @ManyToOne
    private Product product;

    private int quantity;

    public InvoiceProduct() {}
    public InvoiceProduct(Invoice invoice, Product product, int quantity) {
        this.invoice = invoice;
        this.product = product;
        this.quantity = quantity;
        invoice.getInvoiceProducts().add(this);
        product.getInvoiceProducts().add(this);
    }

    public int getQuantity() {
        return quantity;
    }
    public Product getProduct() {
        return product;
    }
    public Invoice getInvoice() {
        return invoice;
    }
}
```

```
package org.example;

import java.io.Serializable;
import java.util.Objects;

public class InvoiceProductId implements Serializable {
    private Long invoice;
    private Long product;

    public InvoiceProductId() {}

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof InvoiceProductId)) return false;
        InvoiceProductId that = (InvoiceProductId) o;
        return Objects.equals(invoice, that.invoice) &&
            Objects.equals(product, that.product);
```

```
    }

    @Override
    public int hashCode() {
        return Objects.hash(invoice, product);
    }
}
```

```
package org.example;

import jakarta.persistence.*;

public class Main {
    public static void main(String[] args) {
        EntityManagerFactory emf = Persistence.createEntityManagerFactory("my-jpa-unit");
        EntityManager em = emf.createEntityManager();

        em.getTransaction().begin();

        Product p1 = new Product("Młotek", 30);
        Product p2 = new Product("Wiertarka", 20);

        Invoice i1 = new Invoice("FV/001");
        Invoice i2 = new Invoice("FV/002");

        em.persist(p1);
        em.persist(p2);
        em.persist(i1);
        em.persist(i2);

        InvoiceProduct ip1 = new InvoiceProduct(i1, p1, 5);
        InvoiceProduct ip2 = new InvoiceProduct(i1, p2, 2);
        InvoiceProduct ip3 = new InvoiceProduct(i2, p1, 1);

        em.persist(ip1);
        em.persist(ip2);
        em.persist(ip3);

        em.getTransaction().commit();

        System.out.println("Produkty z faktury FV/001:");
        for (InvoiceProduct ip : i1.getInvoiceProducts()) {
            System.out.println(" - " + ip.getProduct().getProductName() + " x "
                    + ip.getQuantity());
        }

        System.out.println("Faktury zawierające produkt: " + p1.getProductName());
        for (InvoiceProduct ip : p1.getInvoiceProducts()) {
            System.out.println(" - " + ip.getInvoice().getInvoiceNumber() +
                    " (ilość: " + ip.getQuantity() + ")");
        }
    }
}
```

```
    }

    em.close();
    emf.close();
}

}
```

Poniżej zamieszamy zrzuty ekranu realizacji przez nas powyższego kodu wraz z uzyskanym rezultatem:

The screenshot shows the IntelliJ IDEA interface. The left sidebar displays the project structure with a file named 'VI [I] C:\Users\kacpe\Documents\Bazy\Raporty'. The main editor window shows the 'Main.java' file with the following code:

```
package org.example;
import jakarta.persistence.*;
public class Main {
```

The 'Run' tab is selected, showing the output of the application's execution. The output window displays the following log messages:

```
Hibernate: alter table InvoiceProduct add constraint FK628edrh6ejmgscurpdcpmq16 foreign key (product_id) references Product
Hibernate: values next value for Product_SEQ
Hibernate: values next value for Product_SEQ
Hibernate: values next value for Invoice_SEQ
Hibernate: values next value for Invoice_SEQ
Hibernate: insert into Product (productName,unitsOnStock,id) values (?, ?, ?)
Hibernate: insert into Product (productName,unitsOnStock,id) values (?, ?, ?)
Hibernate: insert into Invoice (invoiceNumber,id) values (?, ?)
Hibernate: insert into Invoice (invoiceNumber,id) values (?, ?)
Hibernate: insert into InvoiceProduct (quantity,invoice_id,product_id) values (?, ?, ?)
Hibernate: insert into InvoiceProduct (quantity,invoice_id,product_id) values (?, ?, ?)
Hibernate: insert into InvoiceProduct (quantity,invoice_id,product_id) values (?, ?, ?)
Produkty z faktury FV/001:
- Wiertarka x 2
- Młotek x 5
Faktury zawierające produkt: Młotek
- FV/002 (ilość: 1)
- FV/001 (ilość: 5)
cze 07, 2025 2:11:18 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl$PoolState stop
INFO: HHH10001008: Cleaning up connection pool [jdbcd:derby://127.0.0.1/KacperWdowiakDatabase_VI;create=true]
Process finished with exit code 0
```

The bottom status bar indicates the current time as 14:11 and the date as 07.06.2025.

The screenshot shows the DBeaver interface connected to the database 'KacperWdowiakDatabase\_VI'. The top navigation bar shows the connection details: '[@dbmanage.lab.ii.agh.edu.pl [3]]' and 'PRODUCT [KacperWdowiakDatabase\_VI@127.0.0.1]'. The main area displays the 'PRODUCT' table with the following data:

UNITSONSTOCK	ID	PRODUCTNAME
1	30	1 Młotek
2	20	2 Wiertarka

The screenshot shows a database interface with two tables displayed side-by-side.

**Top Table (PRODUCT):**

- Table name: PRODUCT [KacperWdowiakDatabase\_VI@127.0.0.1]
- Columns: ID (Primary Key), INVOICENUMBER
- Data:
 

ID	INVOICENUMBER
1	FV/001
2	FV/002

**Bottom Table (INVOICE):**

- Table name: INVOICE [KacperWdowiakDatabase\_VI@127.0.0.1]
- Columns: QUANTITY, INVOICE\_ID, PRODUCT\_ID
- Data:
 

QUANTITY	INVOICE_ID	PRODUCT_ID
2	1	1
1	2	2
5	1	1

## Zadanie VII: Kaskady

Realizację zadania VII) przedstawia kod poniżej:

```
package org.example;

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

@Entity
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "product_gen")
    @SequenceGenerator(name = "product_gen", sequenceName = "product_seq",
allocationSize = 1)
    private Long id;

    private String name;
    private int unitsInStock;

    @OneToMany(mappedBy = "product", cascade = CascadeType.ALL, orphanRemoval =
true)
    private Set<InvoiceProduct> invoiceProducts = new HashSet<>();
```

```
public Product() {}

public Product(String name, int unitsInStock) {
    this.name = name;
    this.unitsInStock = unitsInStock;
}

public Long getId() { return id; }
public String getName() { return name; }
public int getUnitsInStock() { return unitsInStock; }
public Set<InvoiceProduct> getInvoiceProducts() { return invoiceProducts; }

public void setName(String name) { this.name = name; }
public void setUnitsInStock(int unitsInStock) { this.unitsInStock =
unitsInStock; }
}
```

```
package org.example;

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

@Entity
public class Invoice {
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "invoice_gen")
    @SequenceGenerator(name = "invoice_gen", sequenceName = "invoice_seq",
allocationSize = 1)
    private Long id;

    private String invoiceNumber;

    @OneToMany(mappedBy = "invoice", cascade = CascadeType.ALL, orphanRemoval =
true)
    private Set<InvoiceProduct> invoiceProducts = new HashSet<>();

    public Invoice() {}

    public Invoice(String invoiceNumber) {
        this.invoiceNumber = invoiceNumber;
    }

    public Long getId() { return id; }
    public String getInvoiceNumber() { return invoiceNumber; }
    public Set<InvoiceProduct> getInvoiceProducts() { return invoiceProducts; }

    public void setInvoiceNumber(String invoiceNumber) { this.invoiceNumber =
invoiceNumber; }
}
```

```
package org.example;

import jakarta.persistence.*;

@Entity
public class InvoiceProduct {
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "invprod_gen")
    @SequenceGenerator(name = "invprod_gen", sequenceName = "invprod_seq",
allocationSize = 1)
    private Long id;

    @ManyToOne
    private Invoice invoice;

    @ManyToOne
    private Product product;

    private int quantity;

    public InvoiceProduct() {}

    public InvoiceProduct(Invoice invoice, Product product, int quantity) {
        this.invoice = invoice;
        this.product = product;
        this.quantity = quantity;
    }

    public Invoice getInvoice() { return invoice; }
    public Product getProduct() { return product; }
    public int getQuantity() { return quantity; }
}
```

```
package org.example;

import jakarta.persistence.*;

public class Main {
    public static void main(String[] args) {
        EntityManagerFactory emf = Persistence.createEntityManagerFactory("my-jpa-unit");
        EntityManager em = emf.createEntityManager();

        em.getTransaction().begin();

        Product p1 = new Product("Młotek", 100);
        Product p2 = new Product("Wkrętarka", 50);
        Product p3 = new Product("Piła", 25);
```

```
Invoice invoice = new Invoice("FV/2025/001");

invoice.getInvoiceProducts().add(new InvoiceProduct(invoice, p1, 10));
invoice.getInvoiceProducts().add(new InvoiceProduct(invoice, p2, 5));
invoice.getInvoiceProducts().add(new InvoiceProduct(invoice, p3, 3));

em.persist(p1);
em.persist(p2);
em.persist(p3);
em.persist(invoice);

em.getTransaction().commit();

System.out.println("== Zapisano fakturę: " + invoice.getInvoiceNumber());
for (InvoiceProduct ip : invoice.getInvoiceProducts()) {
    System.out.println(" → " + ip.getQuantity() + " x " +
ip.getProduct().getName());
}

em.getTransaction().begin();
System.out.println("\n== Odczyt z bazy:");
Invoice found = em.find(Invoice.class, invoice.getId());
System.out.println("Faktura: " + found.getInvoiceNumber());
for (InvoiceProduct ip : found.getInvoiceProducts()) {
    System.out.println(" • " + ip.getQuantity() + " x " +
ip.getProduct().getName());
}
em.getTransaction().commit();

em.close();
emf.close();
}

}
```

Poniżej zamieszamy zrzuty ekranu realizacji przez nas powyższego kodu wraz z uzyskanym rezultatem:

The screenshot shows an IDE interface with a project named "VII" containing "src" and "main" packages. The "Main.java" file is open, showing code for persisting entities to a database:

```

public class Main {
    public static void main(String[] args) {
        em.getTransaction().commit();
        System.out.println("== Zapisano fakturę: " + invoice.getInvoiceNumber());
        for (InvoiceProduct ip : invoice.getInvoiceProducts()) {
            ...
        }
    }
}

```

The output window displays the following log:

```

Hibernate: insert into Product (name,unitsInStock,id) values (?, ?, ?)
Hibernate: insert into Product (name,unitsInStock,id) values (?, ?, ?)
Hibernate: insert into Product (name,unitsInStock,id) values (?, ?, ?)
Hibernate: insert into Invoice (invoiceNumber,id) values (?, ?)
Hibernate: insert into InvoiceProduct (invoice_id,product_id,quantity,id) values (?, ?, ?, ?)
Hibernate: insert into InvoiceProduct (invoice_id,product_id,quantity,id) values (?, ?, ?, ?)
Hibernate: insert into InvoiceProduct (invoice_id,product_id,quantity,id) values (?, ?, ?, ?)
== Zapisano fakturę: FV/2025/001
→ 3 x Piła
→ 5 x Wkrętarka
→ 10 x Młotek

== Odczyt z bazy:
Faktura: FV/2025/001
• 3 x Piła
• 5 x Wkrętarka
• 10 x Młotek
cze 07, 2025 2:48:01 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl$PoolState stop
INFO: HHH10001008: Cleaning up connection pool [jdbc:derby://localhost:1527/KacperWdowiakDatabase_VII;create=true]

Process finished with exit code 0

```

The status bar at the bottom indicates: 26:1 CRLF UTF-8 4 spaces.

The screenshot shows a database management tool interface with three tabs: PRODUCT, INVOICE, and INVOICEPRODUCT. The PRODUCT tab is active, displaying the following data:

	UNITSINSTOCK	ID	NAME
1	100	1	Młotek
2	50	2	Wkrętarka
3	25	3	Piła

The INVOICE tab is active, displaying the following data:

	ID	INVOICENUMBER
1	1	FV/2025/001