JPA Sprawozdanie

1. Basics

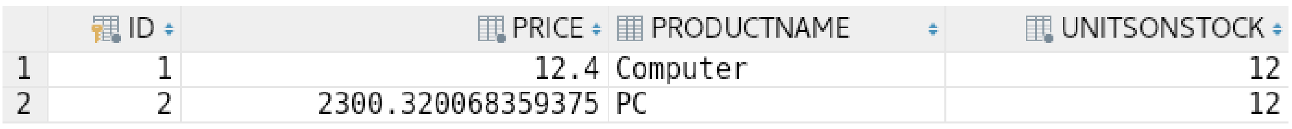
Kod mapowanej klasy:

```java
@Entity(name = "Products")
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;
    private String productName;
    private int unitsOnStock;
    private double price;
    public Product() {
    }
    public Product(String productName, int unitsOnStock, double price) {
        this.productName = productName;
        this.unitsOnStock = unitsOnStock;
        this.price = price;
    }
    @Override
    public String toString() {
        return String.format("ID: %d, Name: %s, Units: %d, Price: %.2f",
                id, productName, unitsOnStock, price);
    }
}
```

Przykładowe polecenie dodanie do bazy:

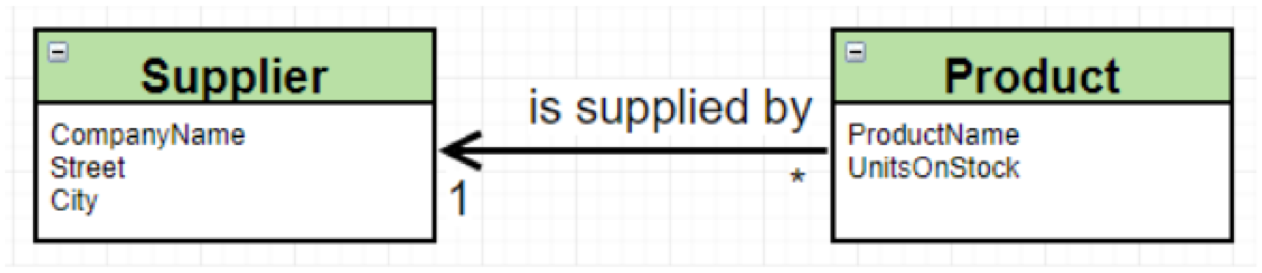
```java
public class AddProduct implements Command {

    private Session session;
    public AddProduct(Session session) {
        this.session = session;
    }
    @Override
    public void execute() {
        System.out.print("Name: ");
        String productName = scanner.nextLine();
        System.out.print("Price: ");
        double price = Float.parseFloat(scanner.nextLine());
        System.out.print("UnitsOnStock: ");
        int onStock = Integer.parseInt(scanner.nextLine());
        Transaction tx = session.beginTransaction();
        session.save(new Product(productName, onStock, price));
        tx.commit();
    }
}
```

```sql
SELECT * FROM PRODUCTS
```



	ID	PRICE	PRODUCTNAME	UNITSONSTOCK
1	1	12.4	Computer	12
2	2	2300.320068359375	PC	12

2. Wprowadzenie modelu dostawcy.



```
@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;
    }
    @Override
    public String toString() {
        return String.format("ID: %d, CompanyName: %s, Street: %s, City: %s",
                id, companyName, street, city);
    }
}
```

Zmiana w modelu Produktu:

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

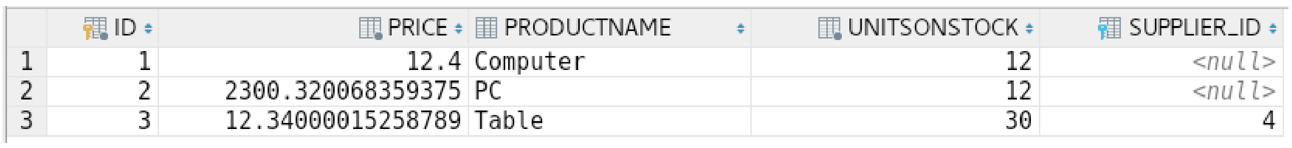
Dodanie dostawcy do produktu:

```
public void execute() {
    System.out.print("Product id: ");
    int id = Integer.parseInt(scanner.nextLine());
    System.out.print("Supplier id: ");
    int c_id = Integer.parseInt(scanner.nextLine());
    Transaction tx = session.beginTransaction();
    Product product = session.get(Product.class, id);
    Supplier supplier = session.get(Supplier.class, c_id);
    if (product != null && supplier != null) product.setSupplier(supplier);
    tx.commit();
}
```

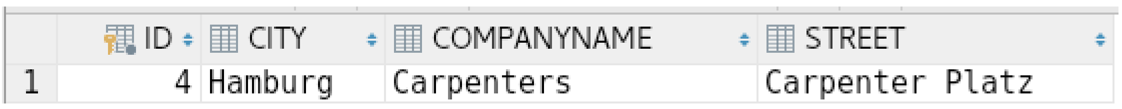
Logi:

```
Hibernate: select product0_.id as id1_0_0_, product0_.price as price2_0_0_,
product0_.productName as productN3_0_0_, product0_.supplier_id as supplier5_0_0_,
product0_.unitsOnStock as unitsOnS4_0_0_, supplier1_.id as id1_1_1_, supplier1_.city as
city2_1_1_, supplier1_.companyName as companyN3_1_1_, supplier1_.street as street4_1_1_ from
Products product0_ left outer join Supplier supplier1_ on product0_.supplier_id=supplier1_.id
where product0_.id=?

Hibernate: select supplier0_.id as id1_1_0_, supplier0_.city as city2_1_0_,
supplier0_.companyName as companyN3_1_0_, supplier0_.street as street4_1_0_ from Supplier
supplier0_ where supplier0_.id=?

Hibernate: update Products set price=?, productName=?, supplier_id=?, unitsOnStock=? where id=?
```

```
SELECT * FROM PRODUCTS;
SELECT * FROM SUPPLIER;
```

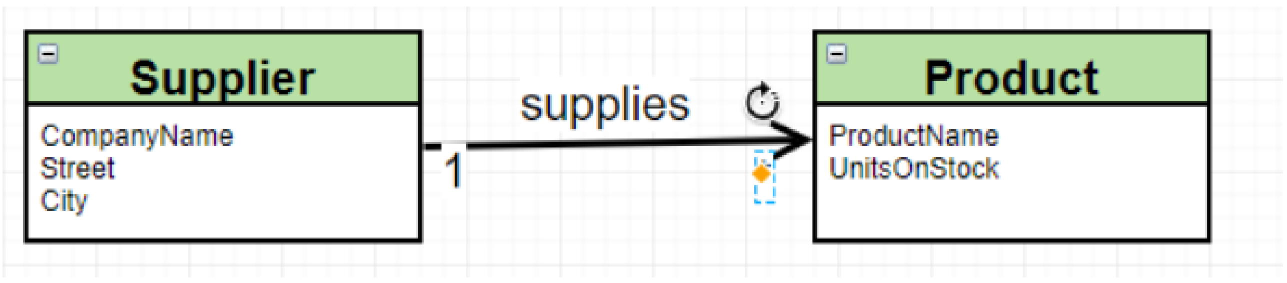


	ID	PRICE	PRODUCTNAME	UNITSONSTOCK	SUPPLIER_ID
1	1	12.4	Computer	12	*<null>*
2	2	2300.320068359375	PC	12	*<null>*
3	3	12.34000015258789	Table	30	4



	ID	CITY	COMPANYNAME	STREET
1	4	Hamburg	Carpenters	Carpenter Platz

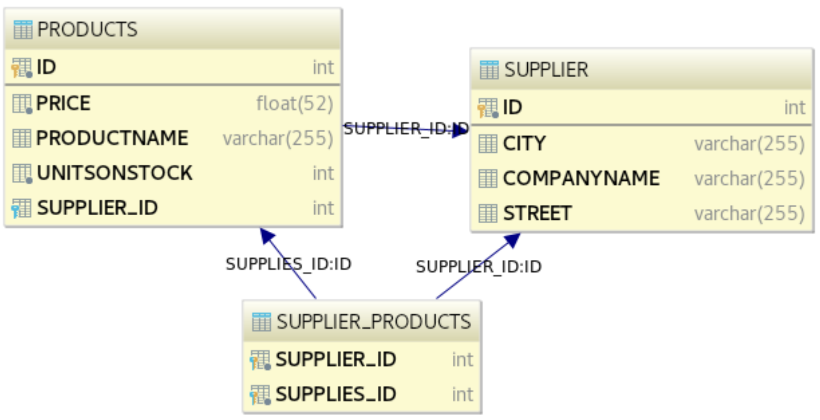
4. Odwrócenie relacji



Zmiana w modelu Supplier:

```
@OneToMany
private Set<Product> supplies;
```

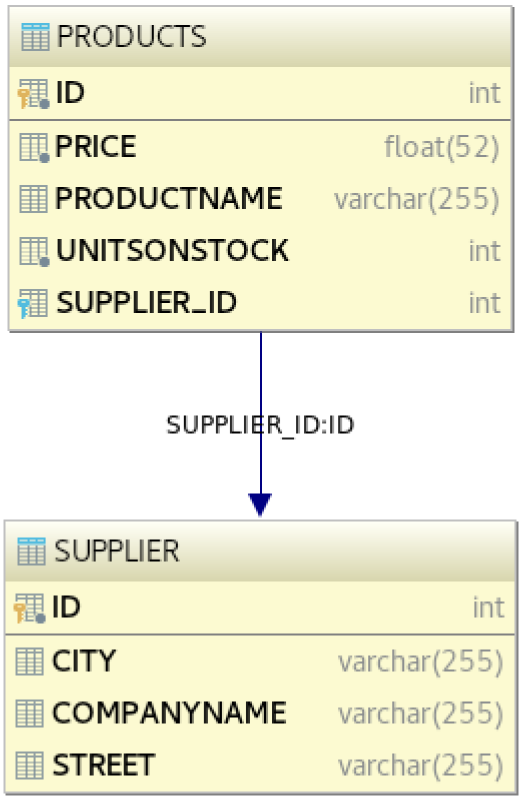
Po zaaktualizowaniu bazy (bez ponownego tworzenia):



Jak widać, jest to przykład z tabelą łącznikową. Zmieńmy teraz model:

W modelu Supplier:

```
@OneToMany
@JoinColumn(name = "SUPPLIER_ID")
private Set<Product> supplies;
```

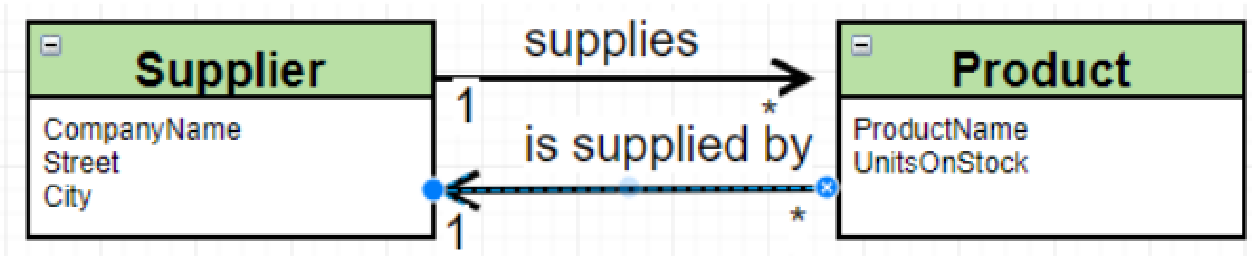


Jak widać, mamy relację bez tabeli łącznikowej.

Koniec zajęć

---

6. Relacja dwustronna:



W modelu Supplier:

```
@OneToMany
@JoinColumn(name = "SUPPLIER_ID")
private Set<Product> supplies;
```

W modelu Product:

```
@ManyToOne
```

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

Prowadzi to jednak to podwójnych updatów w bazie:

```
Hibernate: select product0_.id as id1_0_0_, product0_.price as price2_0_0_,
product0_.productName as productN3_0_0_, product0_.SUPPLIER_ID as
SUPPLIER5_0_0_, product0_.unitsOnStock as unitsOnS4_0_0_, supplier1_.id as
id1_1_1_, supplier1_.city as city2_1_1_, supplier1_.companyName as
companyN3_1_1_, supplier1_.street as street4_1_1_ from Products product0_ left
outer join Supplier supplier1_ on product0_.SUPPLIER_ID=supplier1_.id where
product0_.id=?

Hibernate: select supplier0_.id as id1_1_0_, supplier0_.city as city2_1_0_,
supplier0_.companyName as companyN3_1_0_, supplier0_.street as street4_1_0_ from
Supplier supplier0_ where supplier0_.id=?

Hibernate: select supplies0_.SUPPLIER_ID as SUPPLIER5_0_0_, supplies0_.id as
id1_0_0_, supplies0_.id as id1_0_1_, supplies0_.price as price2_0_1_,
supplies0_.productName as productN3_0_1_, supplies0_.SUPPLIER_ID as
SUPPLIER5_0_1_, supplies0_.unitsOnStock as unitsOnS4_0_1_ from Products
supplies0_ where supplies0_.SUPPLIER_ID=?

Hibernate: update Products set price=?, productName=?, SUPPLIER_ID=?,
unitsOnStock=? where id=?

Hibernate: update Products set SUPPLIER_ID=? where id=?
```

Dlatego podmieniamy adnotacje w modelu Supplier na:

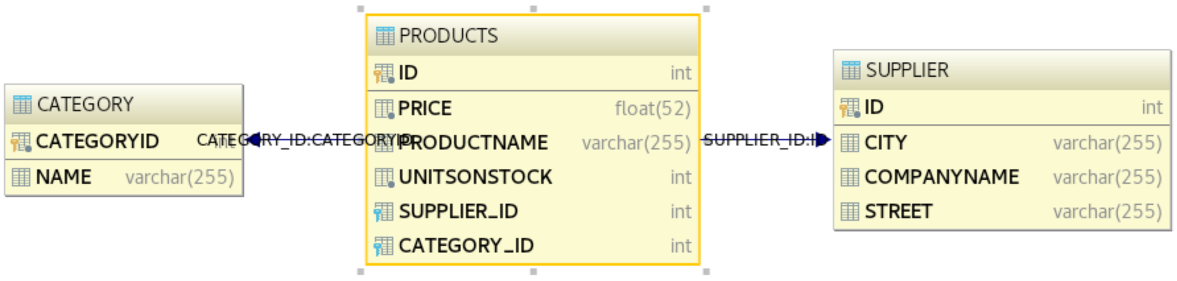
```java
@OneToMany(mappedBy = "supplier")
private Set<Product> supplies;
```

W resultacie otrzymujemy:

```
Hibernate: select product0_.id as id1_0_0_, product0_.price as price2_0_0_,
product0_.productName as productN3_0_0_, product0_.SUPPLIER_ID as
SUPPLIER5_0_0_, product0_.unitsOnStock as unitsOnS4_0_0_, supplier1_.id as
id1_1_1_, supplier1_.city as city2_1_1_, supplier1_.companyName as
companyN3_1_1_, supplier1_.street as street4_1_1_ from Products product0_ left
outer join Supplier supplier1_ on product0_.SUPPLIER_ID=supplier1_.id where
product0_.id=?

Hibernate: select supplier0_.id as id1_1_0_, supplier0_.city as city2_1_0_,
supplier0_.companyName as companyN3_1_0_, supplier0_.street as street4_1_0_ from
Supplier supplier0_ where supplier0_.id=?

Hibernate: select supplies0_.SUPPLIER_ID as SUPPLIER5_0_0_, supplies0_.id as
id1_0_0_, supplies0_.id as id1_0_1_, supplies0_.price as price2_0_1_,
supplies0_.productName as productN3_0_1_, supplies0_.SUPPLIER_ID as
SUPPLIER5_0_1_, supplies0_.unitsOnStock as unitsOnS4_0_1_ from Products
supplies0_ where supplies0_.SUPPLIER_ID=?

Hibernate: update Products set price=?, productName=?, SUPPLIER_ID=?,
unitsOnStock=? where id=?
```

7. Dodanie modelu Category

```java
@Entity
public class Category {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int categoryId;
    private String name;
    @OneToMany
    @JoinColumn(name = "CATEGORY_ID")
    private List<Product> products;
    public Category() {
    }
    public Category(String name) {
        this.name = name;
        products = new LinkedList<>();
    }
    @Override
    public String toString() {
        return String.format("ID: %d, Name: %s", categoryId, name);
    }
    public void addProduct(Product product){
        products.add(product);
    }
}
```



Dodawanie dostawców, produktów i pobieranie danych:

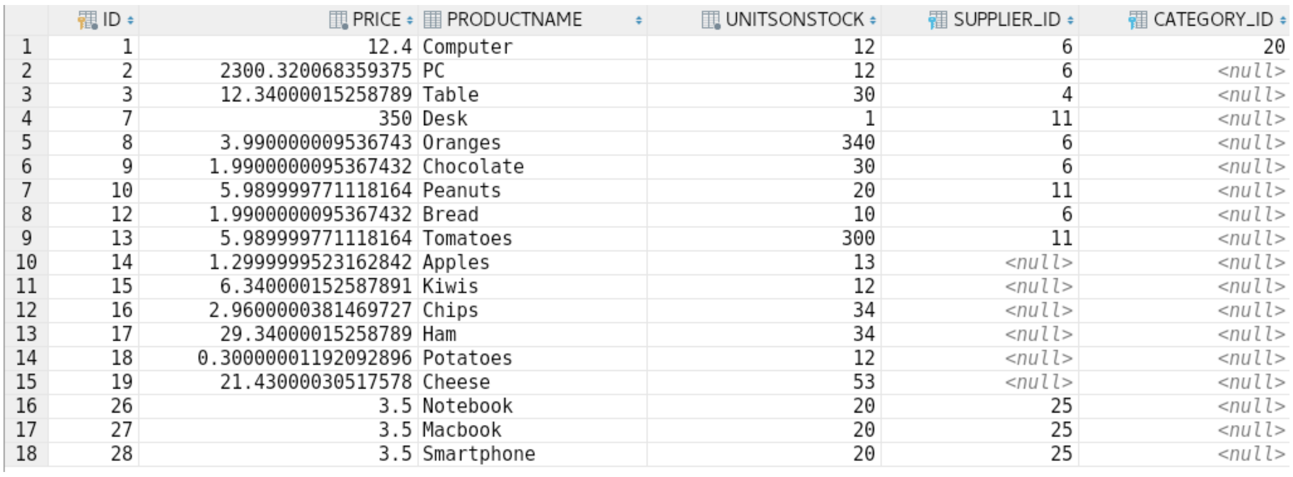
```java
public static void main(final String[] args) throws Exception {
    final Session session = getSession();
    Transaction tx = session.beginTransaction();
    Supplier supplier = new Supplier("XKOM", "Chopina", "Krak?w");
    session.save(supplier);
    Product product = new Product("Notebook", 20, 3.500);
    Product product1 = new Product("Macbook", 20, 3.500);
    Product product2 = new Product("Smartphone", 20, 3.500);
    session.save(product);
    session.save(product1);
    session.save(product2);
    supplier.addSupplied(product);
    supplier.addSupplied(product1);
    supplier.addSupplied(product2);
    Supplier s = session.get(Supplier.class, supplier.getId());
    s.getSupplies().forEach(System.out::println);
    Product p = session.get(Product.class, product.getId());
```

```
        System.out.println(p.getSupplier());
        tx.commit();
        session.close();
    }
```

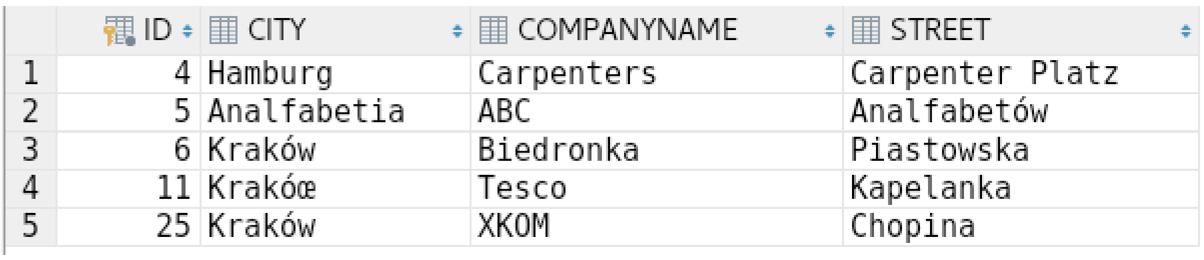
Logi:

```
Hibernate: values next value for hibernate_sequence
Hibernate: values next value for hibernate_sequence
Hibernate: values next value for hibernate_sequence
Hibernate: values next value for hibernate_sequence
ID: 26, Name: Notebook, Units: 20, Price: 3,50
ID: 27, Name: Macbook, Units: 20, Price: 3,50
ID: 28, Name: Smartphone, Units: 20, Price: 3,50
ID: 25, CompanyName: XKOM, Street: Chopina, City: Kraków
Hibernate: insert into Supplier (city, companyName, street, id) values (?, ?, ?, ?)
Hibernate: insert into Products (price, productName, SUPPLIER_ID, unitsOnStock, id) values (?, ?, ?, ?, ?)
Hibernate: insert into Products (price, productName, SUPPLIER_ID, unitsOnStock, id) values (?, ?, ?, ?, ?)
Hibernate: insert into Products (price, productName, SUPPLIER_ID, unitsOnStock, id) values (?, ?, ?, ?, ?)
Hibernate: update Products set price=?, productName=?, SUPPLIER_ID=?, unitsOnStock=? where id=?
Hibernate: update Products set price=?, productName=?, SUPPLIER_ID=?, unitsOnStock=? where id=?
Hibernate: update Products set price=?, productName=?, SUPPLIER_ID=?, unitsOnStock=? where id=?
```

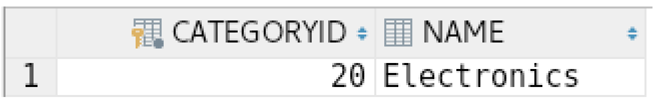
```
SELECT * FROM PRODUCTS;
SELECT * FROM SUPPLIER;
SELECT * FROM CATEGORY;
```



	ID	PRICE	PRODUCTNAME	UNITSONSTOCK	SUPPLIER_ID	CATEGORY_ID
1	1	12.4	Computer	12	6	20
2	2	2300.320068359375	PC	12	6	<null>
3	3	12.34000015258789	Table	30	4	<null>
4	7	350	Desk	1	11	<null>
5	8	3.990000009536743	Oranges	340	6	<null>
6	9	1.9900000095367432	Chocolate	30	6	<null>
7	10	5.989999771118164	Peanuts	20	11	<null>
8	12	1.9900000095367432	Bread	10	6	<null>
9	13	5.989999771118164	Tomatoes	300	11	<null>
10	14	1.2999999523162842	Apples	13	<null>	<null>
11	15	6.340000152587891	Kiwis	12	<null>	<null>
12	16	2.9600000381469727	Chips	34	<null>	<null>
13	17	29.34000015258789	Ham	34	<null>	<null>
14	18	0.30000001192092896	Potatoes	12	<null>	<null>
15	19	21.43000030517578	Cheese	53	<null>	<null>
16	26	3.5	Notebook	20	25	<null>
17	27	3.5	Macbook	20	25	<null>
18	28	3.5	Smartphone	20	25	<null>

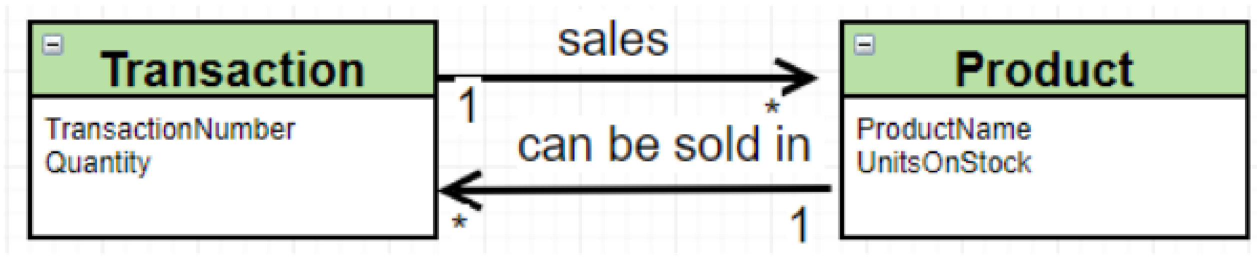


	ID	CITY	COMPANYNAME	STREET
1	4	Hamburg	Carpenters	Carpenter Platz
2	5	Analfabetia	ABC	Analfabetów
3	6	Kraków	Biedronka	Piastowska
4	11	Krakóœ	Tesco	Kapelanka
5	25	Kraków	XKOM	Chopina



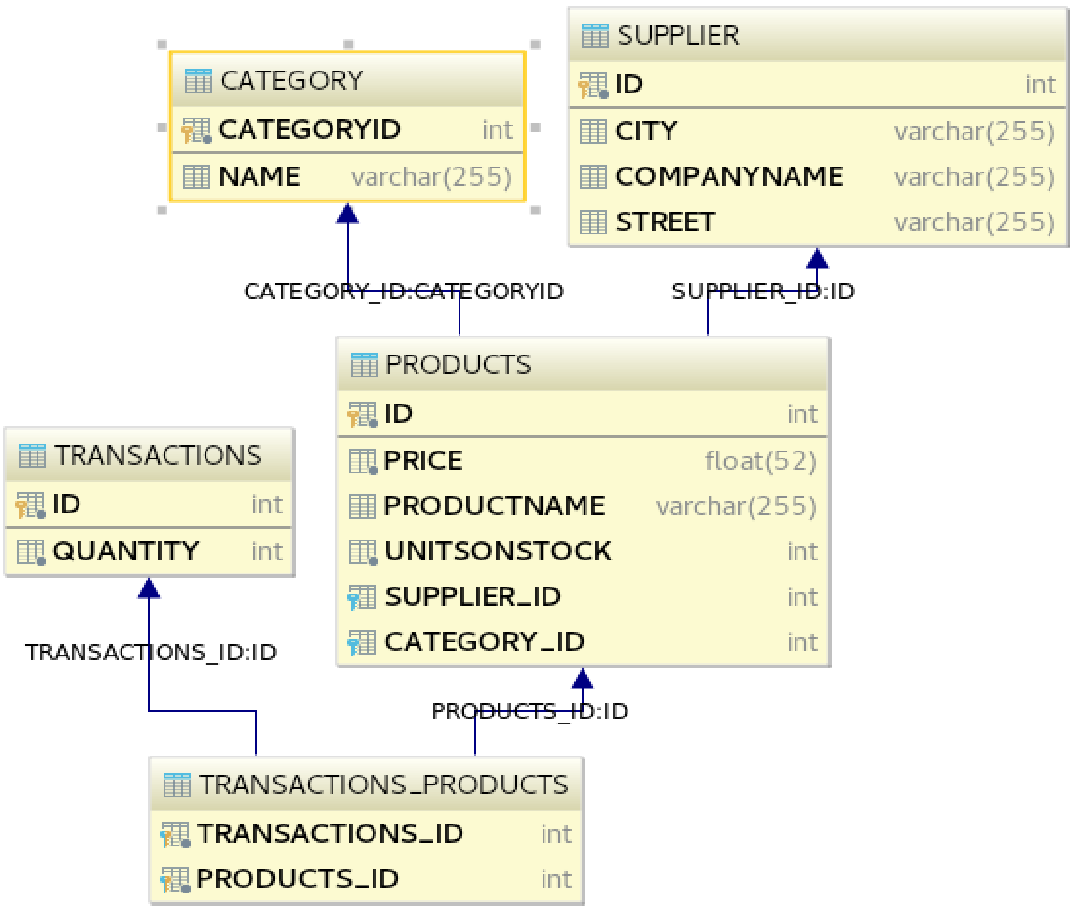
	CATEGORYID	NAME
1	20	Electronics

8. Relacje wiele do wielu



Stworzyłem model Transaction:

```
@Entity(name = "Transactions")
public class Transaction {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;
    private int quantity;
    @ManyToMany
    private Set<Product> products;
    public Transaction() {
    }
    public Transaction(int quantity) {
        this.quantity = quantity;
        this.products = new HashSet<>();
    }
    public void addProduct(Product product) {
        products.add(product);
        product.getTransactions().add(this);
    }
    public Set<Product> getProducts() {
        return products;
    }
    @Override
    public String toString() {
        return String.format("Transation number: %d, Quantity: %d", id,
quantity);
    }
    public int getId() {
        return id;
    }
}
```

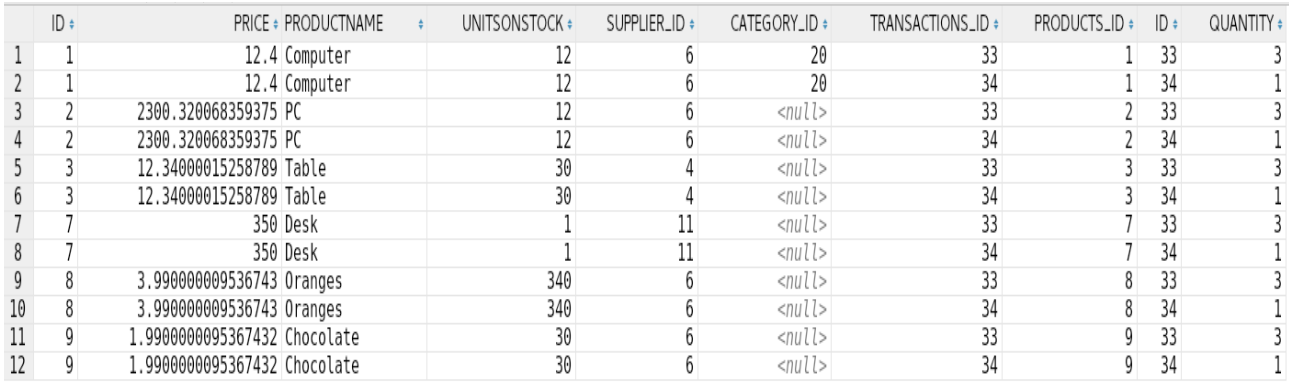


Dodanie produktów do transakcji i pobranie danych:

```
public static void main(final String[] args) throws Exception {
    final Session session = getSession();
    Transaction tx = session.beginTransaction();
    domain.Transaction transaction = new domain.Transaction(3);
    domain.Transaction transaction1 = new domain.Transaction(1);
    session.save(transaction);
    session.save(transaction1);
    List<Product> productList = session
            .createQuery("from Products", Product.class)
            .getResultStream().limit(6).collect(Collectors.toList());
    productList.forEach(product -> {
        transaction.addProduct(product);
        transaction1.addProduct(product);
    });
    Product p = session.get(Product.class, productList.get(0).getId());
    p.getTransactions().forEach(System.out::println);
    System.out.println();
    domain.Transaction t = session.get(domain.Transaction.class,
transaction.getId());
    t.getProducts().forEach(System.out::println);
    tx.commit();
    session.close();
}
```

```
Logi:
Hibernate: create table Transactions (id integer not null, quantity integer not
null, primary key (id))
Hibernate: create table Transactions_Products (transactions_id integer not null,
products_id integer not null, primary key (transactions_id, products_id))
Hibernate: alter table Transactions_Products add constraint
FKnm9r0f3h9sbrx6jbbv263y2t6 foreign key (products_id) references Products
Hibernate: alter table Transactions_Products add constraint
FKa2t7pp8h5r02op4tcj7pwfupv foreign key (transactions_id) references
Transactions
Hibernate: values next value for hibernate_sequence
Hibernate: values next value for hibernate_sequence
Hibernate: select product0_.id as id1_1_, product0_.price as price2_1_,
product0_.productName as productN3_1_, product0_.SUPPLIER_ID as SUPPLIER5_1_,
product0_.unitsOnStock as unitsOnS4_1_ from Products product0_
Hibernate: select supplier0_.id as id1_2_0_, supplier0_.city as city2_2_0_,
supplier0_.companyName as companyN3_2_0_, supplier0_.street as street4_2_0_ from
Supplier supplier0_ where supplier0_.id=?
[...]
Hibernate: select transactio0_.products_id as products2_4_0_,
transactio0_.transactions_id as transact1_4_0_, transactio1_.id as id1_3_1_,
transactio1_.quantity as quantity2_3_1_ from Transactions_Products transactio0_
inner join Transactions transactio1_ on
transactio0_.transactions_id=transactio1_.id where transactio0_.products_id=?
Hibernate: select transactio0_.products_id as products2_4_0_,
transactio0_.transactions_id as transact1_4_0_, transactio1_.id as id1_3_1_,
transactio1_.quantity as quantity2_3_1_ from Transactions_Products transactio0_
inner join Transactions transactio1_ on
transactio0_.transactions_id=transactio1_.id where transactio0_.products_id=?
Transation number: 33, Quantity: 3
Transation number: 34, Quantity: 1

ID: 1, Name: Computer, Units: 12, Price: 12,40
ID: 3, Name: Table, Units: 30, Price: 12,34
ID: 8, Name: Oranges, Units: 340, Price: 3,99
ID: 2, Name: PC, Units: 12, Price: 2300,32
ID: 9, Name: Chocolate, Units: 30, Price: 1,99
ID: 7, Name: Desk, Units: 1, Price: 350,00
Hibernate: insert into Transactions (quantity, id) values (?, ?)
Hibernate: insert into Transactions (quantity, id) values (?, ?)
Hibernate: insert into Transactions_Products (transactions_id, products_id)
values (?, ?)
Hibernate: insert into Transactions_Products (transactions_id, products_id)
values (?, ?)
[...]
Hibernate: insert into Transactions_Products (transactions_id, products_id)
values (?, ?)
```

```sql
SELECT * FROM PRODUCTS as p
  JOIN TRANSACTIONS_PRODUCTS PRODUCT ON p.ID = PRODUCT.PRODUCTS_ID
  JOIN TRANSACTIONS T ON PRODUCT.TRANSACTIONS_ID = T.ID;
```



	ID	PRICE	PRODUCTNAME	UNITSONSTOCK	SUPPLIER_ID	CATEGORY_ID	TRANSACTIONS_ID	PRODUCTS_ID	ID	QUANTITY
1	1	12.4	Computer	12	6	20	33	1	33	3
2	1	12.4	Computer	12	6	20	34	1	34	1
3	2	2300.320068359375	PC	12	6	<null>	33	2	33	3
4	2	2300.320068359375	PC	12	6	<null>	34	2	34	1
5	3	12.34000015258789	Table	30	4	<null>	33	3	33	3
6	3	12.34000015258789	Table	30	4	<null>	34	3	34	1
7	7	350	Desk	1	11	<null>	33	7	33	3
8	7	350	Desk	1	11	<null>	34	7	34	1
9	8	3.990000009536743	Oranges	340	6	<null>	33	8	33	3
10	8	3.990000009536743	Oranges	340	6	<null>	34	8	34	1
11	9	1.9900000095367432	Chocolate	30	6	<null>	33	9	33	3
12	9	1.9900000095367432	Chocolate	30	6	<null>	34	9	34	1

9. Przejście na JPA

```
public static void main(final String[] args) throws Exception {
        EntityManagerFactory emf = Persistence
                .createEntityManagerFactory("WStanekJPAPractice");
        EntityManager em = emf.createEntityManager();
        EntityTransaction etx = em.getTransaction();
        etx.begin();
        Supplier supplier = new Supplier("Komputronik", "Nie wiem", "Krak?w");
        em.persist(supplier);
        Product product = new Product("Notebook", 20, 3.500);
        Product product1 = new Product("Macbook", 20, 3.500);
        Product product2 = new Product("Smartphone", 20, 3.500);
        em.persist(product);
        em.persist(product1);
        em.persist(product2);
        supplier.addSupplied(product);
        supplier.addSupplied(product1);
        supplier.addSupplied(product2);
        Supplier s = em.find(Supplier.class, supplier.getId());
            s.getSupplies().forEach(System.out::println);
        Product p = em.find(Product.class, product.getId());
            System.out.println(p.getSupplier());
        etx.commit();
        em.close();
}
```

10. Kaskady:

W modelu Product:
```
@ManyToMany(mappedBy = "products", cascade = CascadeType.PERSIST)
private Set<Transaction> transactions;
```

W modelu Transaction:
```
@ManyToMany(cascade = CascadeType.PERSIST)
private Set<Product> products;
```

```
public static void main(final String[] args) throws Exception {
    final Session session = getSession();
    Transaction tx = session.beginTransaction();
    domain.Transaction t = new domain.Transaction(3);
    domain.Transaction t1 = new domain.Transaction(1);
    Product p = new Product("Scisors", 12, 2.5);
    Product p1 = new Product("Sofa", 1, 1400);
    t.addProduct(p);
    t.addProduct(p1);
    t1.addProduct(p);
    t1.addProduct(p1);
    session.persist(p);
    tx.commit();
    session.close();
}
```

```
Logs:
```

```
Hibernate: values next value for hibernate_sequence
Hibernate: values next value for hibernate_sequence
Hibernate: values next value for hibernate_sequence
Hibernate: values next value for hibernate_sequence
Hibernate: insert into Products (price, productName, SUPPLIER_ID, unitsOnStock, id) values (?, ?, ?, ?, ?)
Hibernate: insert into Transactions (quantity, id) values (?, ?)
Hibernate: insert into Products (price, productName, SUPPLIER_ID, unitsOnStock, id) values (?, ?, ?, ?, ?)
Hibernate: insert into Transactions (quantity, id) values (?, ?)
Hibernate: insert into Transactions_Products (transactions_id, products_id) values (?, ?)
Hibernate: insert into Transactions_Products (transactions_id, products_id) values (?, ?)
Hibernate: insert into Transactions_Products (transactions_id, products_id) values (?, ?)
Hibernate: insert into Transactions_Products (transactions_id, products_id) values (?, ?)
```

11. Embedded i embeddable:

```java
@Embeddable
public class Address {
    private String street;
    private String city;
    public Address() {
    }
    public Address(String street, String city) {
        this.street = street;
        this.city = city;
    }
    public void setStreet(String street) {
        this.street = street;
    }
    public void setCity(String city) {
        this.city = city;
    }
    public String getStreet() {
        return street;
    }
    public String getCity() {
        return city;
    }
}
```

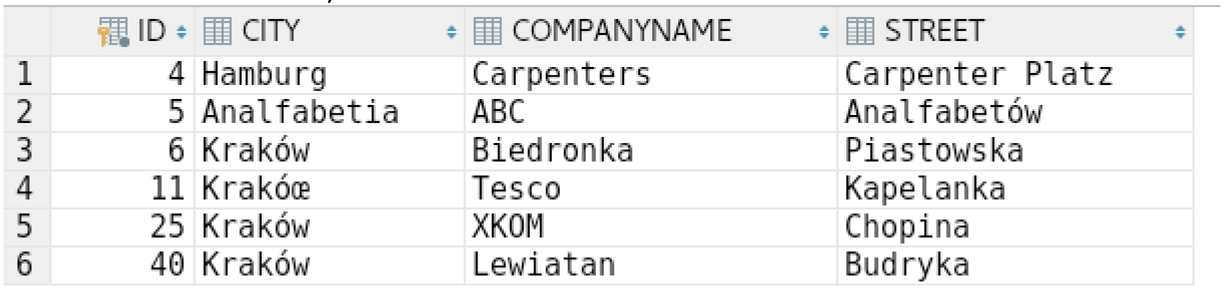
W modelu Supplier:

```java
@Embedded
private Address address;
```

```java
public static void main(final String[] args) throws Exception {
    final Session session = getSession();
    Transaction tx = session.beginTransaction();
    Supplier p = new Supplier("Lewiatan", "Budryka", "Krak?w");
    session.persist(p);
    tx.commit();
    session.close();
}
```

```
Hibernate: insert into Supplier (city, street, companyName, id) values (?, ?, ?, ?)
```

```sql
SELECT * FROM SUPPLIER;
```



	ID	CITY	COMPANYNAME	STREET
1	4	Hamburg	Carpenters	Carpenter Platz
2	5	Analfabetia	ABC	Analfabetów
3	6	Kraków	Biedronka	Piastowska
4	11	Krakóœ	Tesco	Kapelanka
5	25	Kraków	XKOM	Chopina
6	40	Kraków	Lewiatan	Budryka

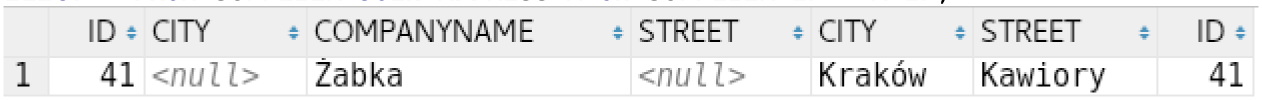
Teraz w drugą stronę:

```java
@Entity
@SecondaryTable(name = "Address")
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;
    private String companyName;
    @Column(table = "Address")
    private String street;
    @Column(table = "Address")
    private String city;
    @OneToMany(mappedBy = "supplier")
    private Set<Product> supplies;
    public Supplier() {
    }

}

public static void main(final String[] args) throws Exception {
    final Session session = getSession();
    Transaction tx = session.beginTransaction();
    Supplier p = new Supplier("?abka", "Kawiory", "Krak?w");
    session.persist(p);
    tx.commit();
    session.close();
}
```

Logs:

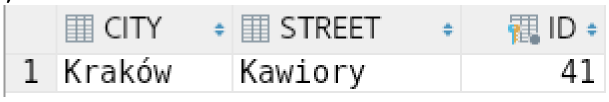
```
Hibernate: create table Address (city varchar(255), street varchar(255), id integer not null, primary key (id))
Hibernate: alter table Address add constraint FKj91l3o9613sfn00sb8yj237f2 foreign key (id) references Supplier
Hibernate: values next value for hibernate_sequence
Hibernate: insert into Supplier (companyName, id) values (?, ?)
Hibernate: insert into Address (city, street, id) values (?, ?, ?)
```

```sql
SELECT * FROM SUPPLIER JOIN ADDRESS A ON SUPPLIER.ID = A.ID;
```

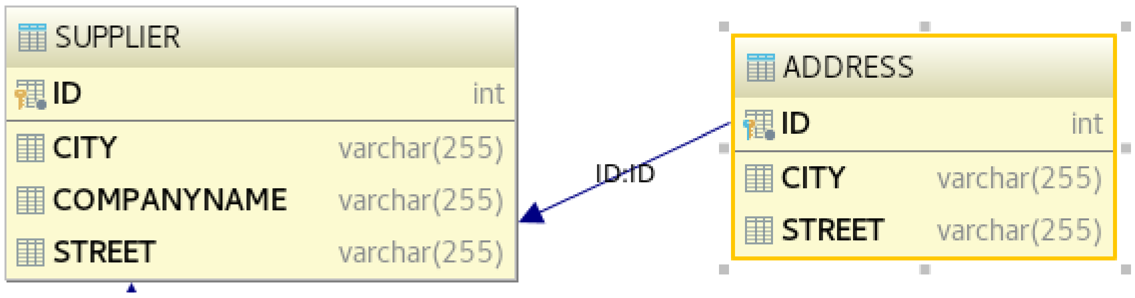


	ID	CITY	COMPANYNAME	STREET	CITY	STREET	ID
1	41	<null>	Żabka	<null>	Kraków	Kawiory	41

```sql
SELECT * FROM ADDRESS;
```

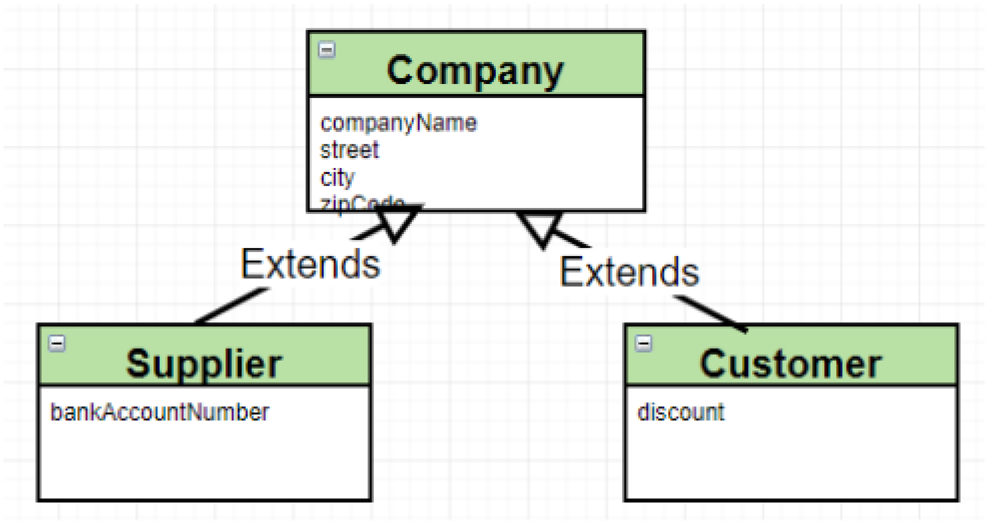


	CITY	STREET	ID
1	Kraków	Kawiory	41



Pola CITY i STREET w tabeli Supplier pozostały z powodu wcześniejszych rekordów.

12. Dziedziczenie

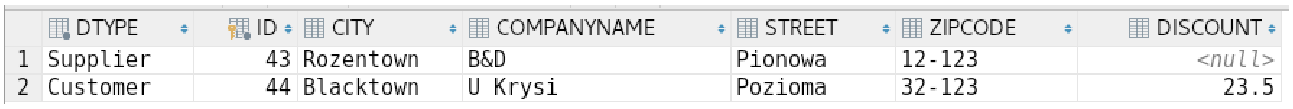


a)
```
@Entity
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
public class Company {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;
    private String companyName;
    private String street;
    private String city;
    private String zipCode;
    public Company() {
    }
    public Company(String companyName, String street, String city, String
zipCode) {
        this.companyName = companyName;
        this.street = street;
        this.city = city;
        this.zipCode = zipCode;
    }
    @Override
    public String toString() {
        return String.format("ID: %d, CompanyName: %s, Street: %s, City: %s,
ZipCode: %s",
                id, companyName, street, city, zipCode);
    }
}
```

```
public static void main(final String[] args) throws Exception {

    final Session session = getSession();
    Transaction tx = session.beginTransaction();
    Supplier s = new Supplier("B&D", "Pionowa", "Rozentown", "12-123");
    Customer c = new Customer("U Krysi", "Pozioma", "Blacktown", "32-123",
23.5);
    session.save(s);
    session.save(c);
    tx.commit();
    session.close();
}
```

```
Hibernate: insert into Company (city, companyName, street, zipCode, DTYPE, id)
values (?, ?, ?, ?, 'Supplier', ?)

Hibernate: insert into Company (city, companyName, street, zipCode, discount,
DTYPE, id) values (?, ?, ?, ?, ?, 'Customer', ?)
```

```
SELECT * FROM COMPANY;
```



	DTYPE	ID	CITY	COMPANYNAME	STREET	ZIPCODE	DISCOUNT
1	Supplier	43	Rozentown	B&D	Pionowa	12-123	<null>
2	Customer	44	Blacktown	U Krysi	Pozioma	32-123	23.5

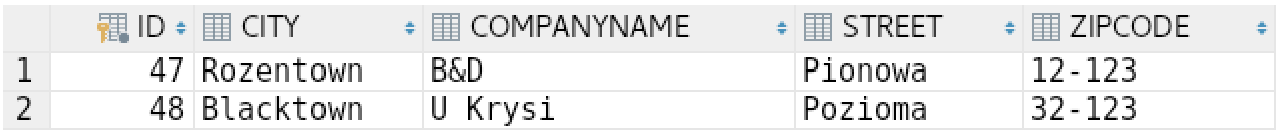
b)

```
@Entity
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
public class Company
```

Funkcja main zostaje taka sama

Hibernate: insert into Company (city, companyName, street, zipCode, id) values (?, ?, ?, ?, ?)
Hibernate: insert into Supplier (id) values (?)
Hibernate: insert into Company (city, companyName, street, zipCode, id) values (?, ?, ?, ?, ?)
Hibernate: insert into Customer (discount, id) values (?, ?)

```
SELECT * FROM COMPANY;
```



	ID	CITY	COMPANYNAME	STREET	ZIPCODE
1	47	Rozentown	B&D	Pionowa	12-123
2	48	Blacktown	U Krysi	Pozioma	32-123

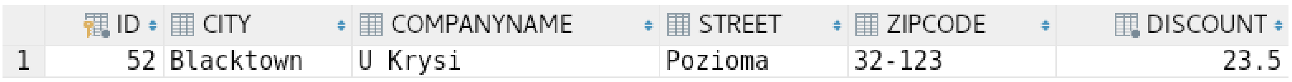
c)

```
@Entity
@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)
public class Company
```

Hibernate: create table Company (id integer not null, city varchar(255), companyName varchar(255), street varchar(255), zipCode varchar(255), primary key (id))
Hibernate: create table Customer (id integer not null, city varchar(255), companyName varchar(255), street varchar(255), zipCode varchar(255), discount double not null, primary key (id))
Hibernate: alter table APP.SUPPLIER add column zipCode varchar(255)
Hibernate: values next value for hibernate_sequence
Hibernate: values next value for hibernate_sequence

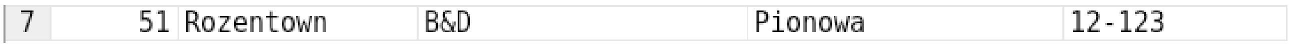
```
Hibernate: insert into Supplier (city, companyName, street, zipCode, id) values (?, ?, ?, ?, ?)
Hibernate: insert into Customer (city, companyName, street, zipCode, discount, id) values (?, ?, ?, ?,
?, ?)
```

```sql
SELECT * FROM CUSTOMER;
```



	ID	CITY	COMPANYNAME	STREET	ZIPCODE	DISCOUNT
1	52	Blacktown	U Krysi	Pozioma	32-123	23.5

```sql
SELECT * FROM SUPPLIER;
```



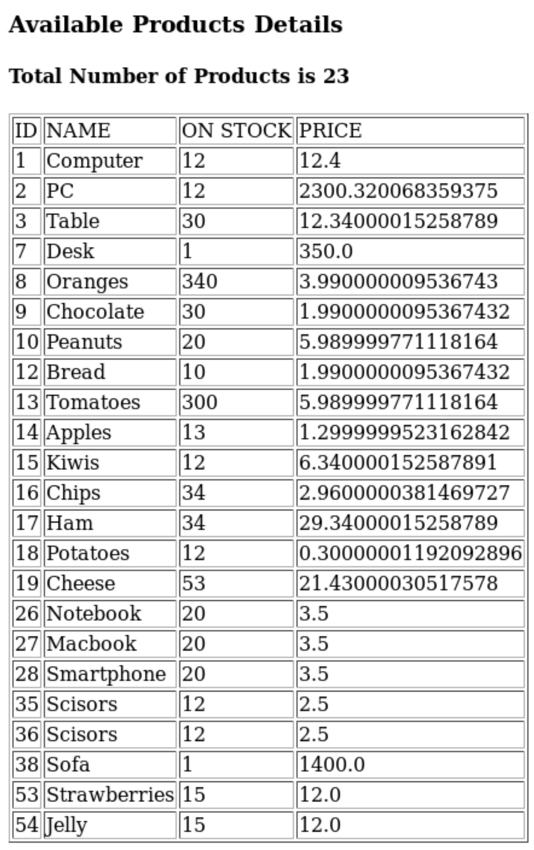
7	51	Rozentown	B&D	Pionowa	12-123

13. Web aplikacja.

Stworzyłem prostą aplikację, która umożliwia pobranie oraz dodanie produktu do bazy

a) getProducts:

```java
@WebServlet("/getProducts")
public class GetProducts extends HttpServlet {
    protected void doGet(HttpServletRequest request,
                         HttpServletResponse response) throws ServletException,
IOException {
        Session session = Config.getSession();
        List<Product> products = session
                .createQuery("from Products", Product.class)
                .getResultList();
        session.close();
        request.setAttribute("products", products);
        request.getRequestDispatcher("productsDetails.jsp").forward(request,
response);
    }
}
```



Available Products Details

Total Number of Products is 23

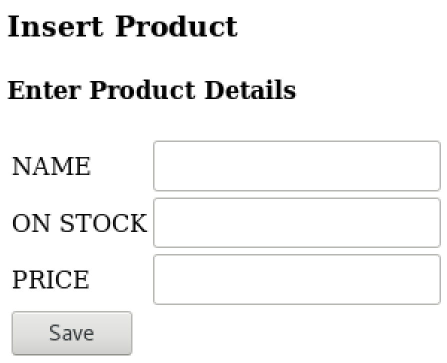
ID	NAME	ON STOCK	PRICE
1	Computer	12	12.4
2	PC	12	2300.320068359375
3	Table	30	12.34000015258789
7	Desk	1	350.0
8	Oranges	340	3.990000009536743
9	Chocolate	30	1.9900000095367432
10	Peanuts	20	5.989999771118164
12	Bread	10	1.9900000095367432
13	Tomatoes	300	5.989999771118164
14	Apples	13	1.2999999523162842
15	Kiwis	12	6.340000152587891
16	Chips	34	2.9600000381469727
17	Ham	34	29.34000015258789
18	Potatoes	12	0.30000001192092896
19	Cheese	53	21.43000030517578
26	Notebook	20	3.5
27	Macbook	20	3.5
28	Smartphone	20	3.5
35	Scisors	12	2.5
36	Scisors	12	2.5
38	Sofa	1	1400.0
53	Strawberries	15	12.0
54	Jelly	15	12.0

Korzystałem tutaj z query.

b)insertProduct

```java
@WebServlet("/insertProduct")

public class InsertProduct extends HttpServlet {
    protected void doPost(HttpServletRequest request,
                          HttpServletResponse response) throws ServletException,
IOException {
        String name = request.getParameter("name");
        double price = Double.parseDouble(request.getParameter("price"));
        int onStock = Integer.parseInt(request.getParameter("onStock"));
        Product p = new Product(name, onStock, price);
        Session session = Config.getSession();
        Transaction tx = session.beginTransaction();
        session.save(p);
        tx.commit();
        session.close();
        response.sendRedirect("getProducts");
    }
}
```

Korzystałem tutaj z transakcji.



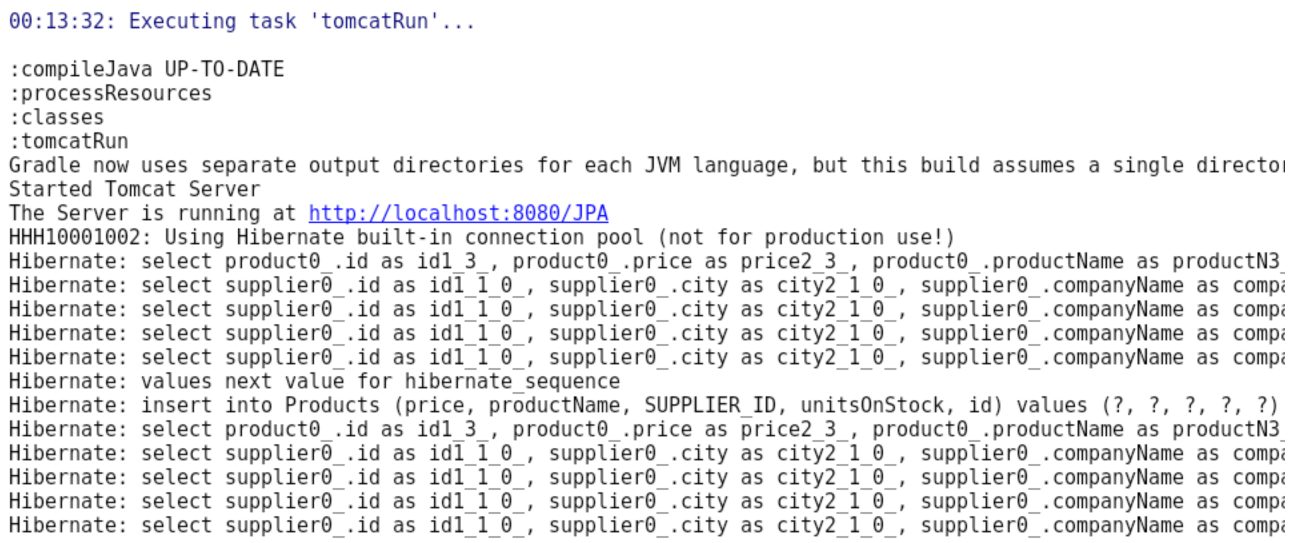
Insert Product

Enter Product Details

NAME	
ON STOCK	
PRICE	

Save

Przykładowe użycie, pobranie danych, dodanie produktu, ponowne pobranie danych:



```
00:13:32: Executing task 'tomcatRun'...

:compileJava UP-TO-DATE
:processResources
:classes
:tomcatRun
Gradle now uses separate output directories for each JVM language, but this build assumes a single directo
Started Tomcat Server
The Server is running at http://localhost:8080/JPA
HHH10001002: Using Hibernate built-in connection pool (not for production use!)
Hibernate: select product0_.id as id1_3_, product0_.price as price2_3_, product0_.productName as productN3_
Hibernate: select supplier0_.id as id1_1_0_, supplier0_.city as city2_1_0_, supplier0_.companyName as compa
Hibernate: select supplier0_.id as id1_1_0_, supplier0_.city as city2_1_0_, supplier0_.companyName as compa
Hibernate: select supplier0_.id as id1_1_0_, supplier0_.city as city2_1_0_, supplier0_.companyName as compa
Hibernate: select supplier0_.id as id1_1_0_, supplier0_.city as city2_1_0_, supplier0_.companyName as compa
Hibernate: values next value for hibernate_sequence
Hibernate: insert into Products (price, productName, SUPPLIER_ID, unitsOnStock, id) values (?, ?, ?, ?, ?)
Hibernate: select product0_.id as id1_3_, product0_.price as price2_3_, product0_.productName as productN3_
Hibernate: select supplier0_.id as id1_1_0_, supplier0_.city as city2_1_0_, supplier0_.companyName as compa
Hibernate: select supplier0_.id as id1_1_0_, supplier0_.city as city2_1_0_, supplier0_.companyName as compa
Hibernate: select supplier0_.id as id1_1_0_, supplier0_.city as city2_1_0_, supplier0_.companyName as compa
Hibernate: select supplier0_.id as id1_1_0_, supplier0_.city as city2_1_0_, supplier0_.companyName as compa
```

Aby uruchomić server należy:

1. Ustawić odpowieni adres bazy w pliku konfiguracyjnym Hibernate.

2. Z terminala wykonać polecenie:

```
>gradle tomcatRun
```