Raport z laboratorium Hibernate/JPA

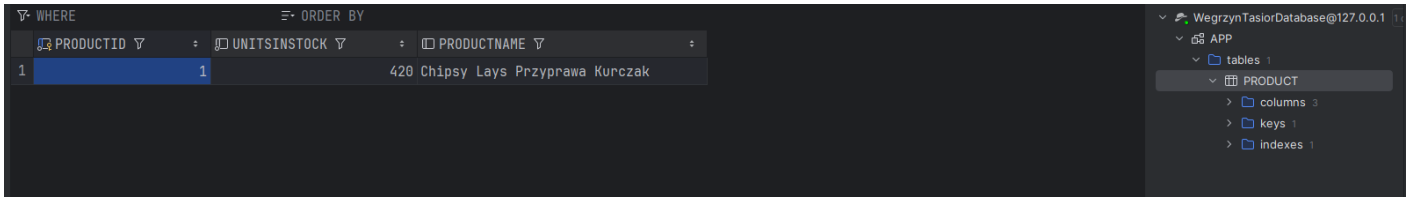
Autorzy:

Filip Węgrzyn
Seweryn Tasior

Zadanie 0

```
public static void main(String[] args) {
    sessionFactory = getSessionFactory();
    Session session = sessionFactory.openSession();
    Product product = new Product("Chipsy Lays Przyprawa Kurczak", 420);
    Transaction tx = session.beginTransaction();
    session.persist(product);
    tx.commit();
    session.close();
    }
```

```
Hibernate:
    drop sequence Product_SEQ restrict
Hibernate:
    create sequence Product_SEQ start with 1 increment by 50
Hibernate:
    create table Product (
        productId integer not null,
        unitsInStock integer not null,
        productName varchar(255),
        primary key (productId)
    )
Hibernate:

values
    next value for Product_SEQ
Hibernate:
    /* insert for
        org.example.model.Product */insert
    into
        Product (productName, unitsInStock, productId)
    values
        (?, ?, ?)
```



Zadanie 1

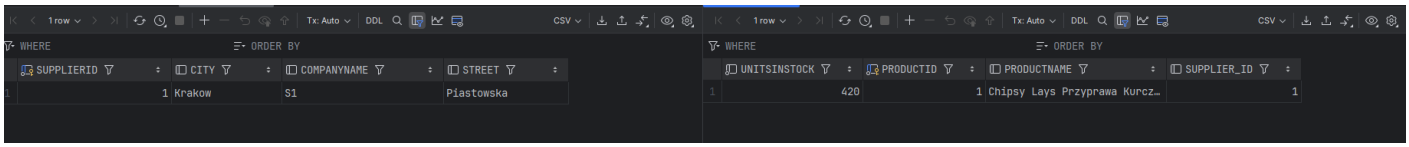
```
@Entity
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long productId;
    private String productName;
    private int unitsInStock;
    @ManyToOne
    @JoinColumn(name = "supplier_id")
    private Supplier supplier;

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

    public Product() {
    }

    public Supplier getSupplier() {
        return supplier;
    }

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

```
@Entity
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long supplierId;
    private String companyName;
    private String city;
    private String street;
```

```
    public Supplier() {
    }

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

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

    Transaction tx = session.beginTransaction();

    Product product = session.get(Product.class, 1);
    Supplier supplier = new Supplier("S1", "Krakow", "Piastowska");
    product.setSupplier(supplier);

    session.persist(supplier);
    session.persist(product);
    tx.commit();
    session.close();
}}
```

```
Hibernate:
    alter table Product
       add column supplier_id bigint
Hibernate:
    alter table Product
       add constraint FK11uleikow9eaenolp88xnaudd
       foreign key (supplier_id)
       references Supplier
Hibernate:
    select
        p1_0.productId,
        p1_0.productName,
        s1_0.supplierId,
        s1_0.city,
        s1_0.companyName,
        s1_0.street,
        p1_0.unitsInStock
    from
        Product p1_0
    left join
        Supplier s1_0
            on s1_0.supplierId=p1_0.supplier_id
    where
        p1_0.productId=?
Hibernate:

values
    next value for Supplier_SEQ
Hibernate:
    /* insert for
        org.example.model.Supplier */insert
    into
        Supplier (city, companyName, street, supplierId)
    values
        (?, ?, ?, ?)
Hibernate:
    /* update
        for org.example.model.Product */update Product
    set
        productName=?,
        supplier_id=?,
        unitsInStock=?
    where
        productId=?

Process finished with exit code 0
```



SUPPLIERID	CITY	COMPANYNAME	STREET
1	Krakow	S1	Piastowska

UNITSINSTOCK	PRODUCTID	PRODUCTNAME	SUPPLIER_ID
420	1	Chipsy Lays Przyprawa Kurcz…	1

Zadanie 2

Zadanie 2a

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

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

    public Product() {
    }
}
```

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

```
    private String companyName;
    private String city;
    private String street;
    @OneToMany(cascade = CascadeType.ALL)
    @JoinColumn(name = "supplier_id")
    private List<Product> supplies = new ArrayList<>();

    public Supplier() {}

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

    public List<Product> getSupplies() {
        return supplies;
    }
}
```

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

    Supplier s = new Supplier("S2", "Warszawa", "Długa");

    Product p1 = new Product("Pepsi", 100);
    Product p2 = new Product("Lays", 50);

    s.getSupplies().add(p1);
    s.getSupplies().add(p2);

    session.persist(s);
    tx.commit();
    session.close();
}
```

Zadanie 2b

```
@Entity
public class Supplier {
    @Id
    @GeneratedValue
    private Long supplierId;
    private String companyName;
    private String city;
    private String street;

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

    public Supplier() {
    }

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

    public List<Product> getSupplies() {
        return supplies;
    }
}
```

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

    Supplier s = new Supplier("Nestle", "Krakow", "Pawia");

    Product p1 = new Product("Lion", 30);
    Product p2 = new Product("KitKat", 70);

    s.getSupplies().add(p1);
    s.getSupplies().add(p2);

    session.persist(s);
    tx.commit();
    session.close();
}
```

```
Hibernate:

values
    next value for Supplier_SEQ
Hibernate:

values
    next value for Product_SEQ
Hibernate:
    /* insert for
        org.example.model.Supplier */insert
    into
        Supplier (city, companyName, street, supplierId)
    values
        (?, ?, ?, ?)
Hibernate:
    /* insert for
        org.example.model.Product */insert
    into
        Product (productName, unitsInStock, productId)
    values
        (?, ?, ?)
Hibernate:
    /* insert for
        org.example.model.Product */insert
    into
        Product (productName, unitsInStock, productId)
    values
        (?, ?, ?)
Hibernate:
    /* insert for
        org.example.model.Supplier.supplies */insert
    into
        supplier_products (supplier_id, product_id)
    values
        (?, ?)
Hibernate:
    /* insert for
        org.example.model.Supplier.supplies */insert
    into
        supplier_products (supplier_id, product_id)
    values
        (?, ?)
```

Main.java | SUPPLIER_PRODUCTS | code.java | cons

2 rows | Tx: Auto | DDL

WHERE | ORDER BY

	SUPPLIER_ID	PRODUCT_ID
1	102	102
2	102	103

PRODUCT

5 rows | Tx: Auto | DDL | CSV

WHERE | ORDER BY

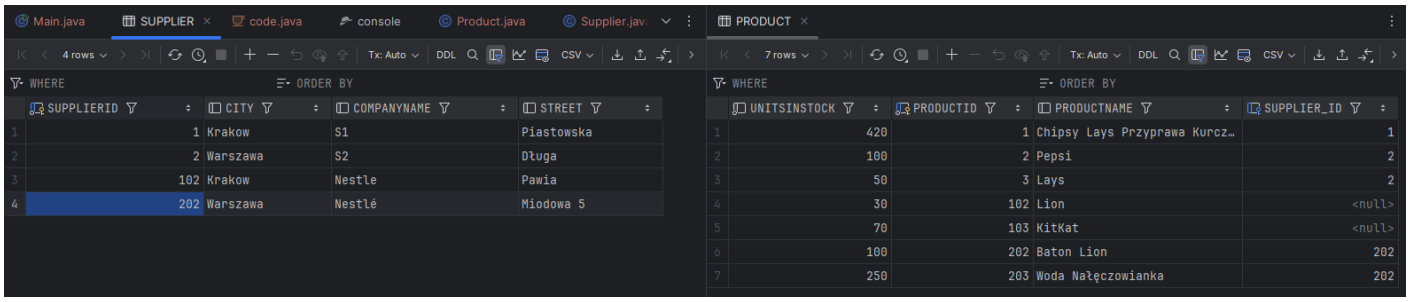
	UNITSINSTOCK	PRODUCTID	PRODUCTNAME	SUPPLIER_ID
1	420	1	Chipsy Lays Przyprawa Kurczak	1
2	100	2	Pepsi	2
3	50	3	Lays	2
4	30	102	Lion	<null>
5	70	103	KitKat	<null>

Zadanie 3

```java
@Entity
public class Supplier {
    @Id @GeneratedValue(strategy = GenerationType.AUTO)
    private Long supplierId;
    private String companyName;
    private String city;
    private String street;
    @OneToMany(mappedBy = "supplier", cascade = CascadeType.ALL, orphanRemoval = true)
    private List<Product> products = new ArrayList<>();

    public Supplier() {}

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

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

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

    Supplier supplier = new Supplier("Nestlé", "Warszawa", "Miodowa 5");

    Product p1 = new Product("Baton Lion", 100);
    Product p2 = new Product("Woda Nałęczowianka", 250);

    supplier.getProducts().add(p1);
    supplier.getProducts().add(p2);
    p1.setSupplier(supplier);
    p2.setSupplier(supplier);

    session.persist(supplier);

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

```
Hibernate:

values
    next value for Supplier_SEQ
Hibernate:

values
    next value for Product_SEQ
Hibernate:
    /* insert for
        org.example.model.Supplier */insert
    into
        Supplier (city, companyName, street, supplierId)
    values
        (?, ?, ?, ?)
Hibernate:
    /* insert for
        org.example.model.Product */insert
    into
        Product (productName, supplier_id, unitsInStock, productId)
    values
        (?, ?, ?, ?)
Hibernate:
    /* insert for
        org.example.model.Product */insert
    into
        Product (productName, supplier_id, unitsInStock, productId)
    values
        (?, ?, ?, ?)
```



Main.java | SUPPLIER | code.java | console | Product.java | Supplier.jav

4 rows | Tx: Auto | DDL | CSV

WHERE | ORDER BY

	SUPPLIERID	CITY	COMPANYNAME	STREET
1	1	Krakow	S1	Piastowska
2	2	Warszawa	S2	Długa
3	102	Krakow	Nestle	Pawia
4	202	Warszawa	Nestlé	Miodowa 5

PRODUCT

7 rows | Tx: Auto | DDL | CSV

WHERE | ORDER BY

	UNITSINSTOCK	PRODUCTID	PRODUCTNAME	SUPPLIER_ID
1	420	1	Chipsy Lays Przyprawa Kurcz…	1
2	100	2	Pepsi	2
3	50	3	Lays	2
4	30	102	Lion	<null>
5	70	103	KitKat	<null>
6	100	202	Baton Lion	202
7	250	203	Woda Nałęczowianka	202

Zadanie 4

```
@Entity
public class Category {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long categoryId;
    private String name;
    @OneToMany(mappedBy = "category", cascade = CascadeType.ALL)
    private List<Product> products = new ArrayList<>();

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

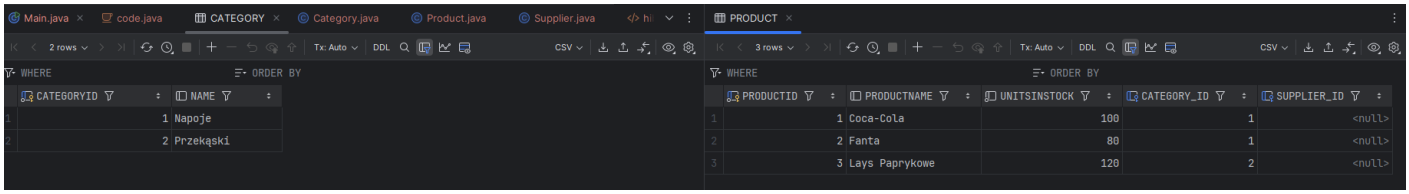
```
@Entity
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long productId;
    private String productName;
    private int unitsInStock;
    @ManyToOne
    @JoinColumn(name = "supplier_id")
    private Supplier supplier;
    @ManyToOne
    @JoinColumn(name = "category_id")
    private Category category;

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

    public Product() {
    }

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

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

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

```
    Category drinks = new Category("Napoje");
    Category snacks = new Category("Przekąski");

    Product cola = new Product("Coca-Cola", 100);
    Product fanta = new Product("Fanta", 80);
    Product lays = new Product("Lays Paprykowe", 120);

    drinks.getProducts().add(cola);
    drinks.getProducts().add(fanta);
    snacks.getProducts().add(lays);
    cola.setCategory(drinks);
    fanta.setCategory(drinks);
    lays.setCategory(snacks);

    session.persist(drinks);
    session.persist(snacks);

    tx.commit();
    session.close();

    Category loadedCategory = session.get(Category.class, 1);
    for (Product p : loadedCategory.getProducts()) {
        System.out.println(p.getProductName());
    }

    Product p = session.get(Product.class, 1L);
    System.out.println("Kategoria: " + p.getCategory().getName());
}
```

```
Hibernate:
    select
        c1_0.categoryId,
        c1_0.name
    from
        Category c1_0
    where
        c1_0.categoryId=?
Hibernate:
    select
        p1_0.category_id,
        p1_0.productId,
        p1_0.productName,
        s1_0.supplierId,
        s1_0.city,
        s1_0.companyName,
        s1_0.street,
        p1_0.unitsInStock
    from
        Product p1_0
    left join
        Supplier s1_0
            on s1_0.supplierId=p1_0.supplier_id
    where
        p1_0.category_id=?
Coca-Cola
Fanta
```

```
Hibernate:
    select
        p1_0.productId,
        c1_0.categoryId,
        c1_0.name,
        p1_0.productName,
        s1_0.supplierId,
        s1_0.city,
        s1_0.companyName,
        s1_0.street,
        p1_0.unitsInStock
    from
        Product p1_0
    left join
        Category c1_0
            on c1_0.categoryId=p1_0.category_id
    left join
        Supplier s1_0
            on s1_0.supplierId=p1_0.supplier_id
    where
        p1_0.productId=?
Kategoria: Napoje

Process finished with exit code 0
```



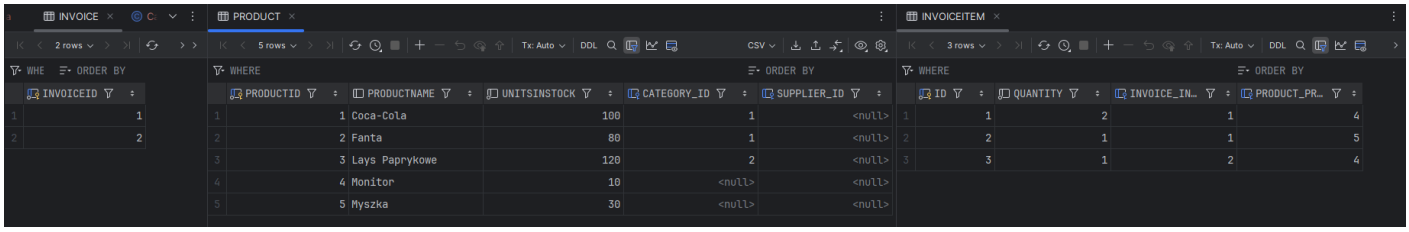
Zadanie 5

```
@Entity
public class Invoice {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long invoiceId;
    @OneToMany(mappedBy = "invoice", cascade = CascadeType.ALL, orphanRemoval = true)
    private List<InvoiceItem> items = new ArrayList<>();

    public List<InvoiceItem> getItems() {
        return items;
    }
}
```

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

    @ManyToOne
    private Invoice invoice;

    @ManyToOne(cascade = CascadeType.PERSIST)
    @JoinColumn(name = "product_id")
    private Product product;

    private int quantity;
}
```

```
@Entity
public class Product {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long productId;
    private String productName;
    private int unitsInStock;
    @ManyToOne
    @JoinColumn(name = "supplier_id")
    private Supplier supplier;
    @ManyToOne
    @JoinColumn(name = "category_id")
    private Category category;
    @OneToMany(mappedBy = "product")
    private List<InvoiceItem> invoiceItems = new ArrayList<>();


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

    public Product() {
    }
}
```

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

    Product p1 = new Product();
    p1.setProductName("Monitor");
    p1.setUnitsInStock(10);

    Product p2 = new Product();
    p2.setProductName("Myszka");
    p2.setUnitsInStock(30);

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

    //Tworzenie faktury z pozycjami
    Invoice invoice1 = new Invoice();

    InvoiceItem item1 = new InvoiceItem();
    item1.setProduct(p1);
    item1.setInvoice(invoice1);
    item1.setQuantity(2);

    InvoiceItem item2 = new InvoiceItem();
    item2.setProduct(p2);
    item2.setInvoice(invoice1);
    item2.setQuantity(1);

    invoice1.getItems().add(item1);
    invoice1.getItems().add(item2);

    session.persist(invoice1);

    // Druga faktura
    Invoice invoice2 = new Invoice();

    InvoiceItem item3 = new InvoiceItem();
    item3.setProduct(p1);
    item3.setInvoice(invoice2);
    item3.setQuantity(1);

    invoice2.getItems().add(item3);

    session.persist(invoice2);

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



INVOICE

	INVOICEID
1	1
2	2

PRODUCT

	PRODUCTID	PRODUCTNAME	UNITSINSTOCK	CATEGORY_ID	SUPPLIER_ID
1	1	Coca-Cola	100	1	<null>
2	2	Fanta	80	1	<null>
3	3	Lays Paprykowe	120	2	<null>
4	4	Monitor	10	<null>	<null>
5	5	Myszka	30	<null>	<null>

INVOICEITEM

	ID	QUANTITY	INVOICE_IN…	PRODUCT_PR…
1	1	2	1	4
2	2	1	1	5
3	3	1	2	4

Zadanie 6

```
emf = Persistence.createEntityManagerFactory("my-persistence-unit");
EntityManager em = emf.createEntityManager();

EntityTransaction tx = em.getTransaction();
tx.begin();

// reszta kodu bez zmian
// ...
```

```
tx.commit();
em.close();
emf.close();
```

Zadanie 7

```
@ManyToOne(cascade = CascadeType.PERSIST)
@JoinColumn(name = "product_id")
private Product product;

@OneToMany(mappedBy = "invoice", cascade = CascadeType.ALL, orphanRemoval = true)
private List<InvoiceItem> items = new ArrayList<>();
```

Zadanie 8

Zadanie 8a

```
@Entity
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long supplierId;
    private String companyName;
    @Embedded
    private Address address;
    @OneToMany(mappedBy = "supplier", cascade = CascadeType.ALL, orphanRemoval = true)
    private List<Product> products = new ArrayList<>();
}
```

```
@Embeddable
public class Address {
    private Long id;
    private String street;
    private String city;

    public Address() {
    }

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

Zadanie 8b

```
@Entity
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long supplierId;
    private String companyName;
    @OneToOne(cascade = CascadeType.ALL)
    @JoinColumn(name = "address_id")
    private Address address;
    @OneToMany(mappedBy = "supplier", cascade = CascadeType.ALL, orphanRemoval = true)
    private List<Product> products = new ArrayList<>();
}
```

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

    private String street;
    private String city;
}
```

```
Hibernate:
    create table Address (
        id bigint generated by default as identity,
        city varchar(255),
        street varchar(255),
        primary key (id)
    )
Hibernate:
    alter table Supplier
       add column address_id bigint
Hibernate:
    alter table Supplier
       drop constraint UK_ggs7vq4d52vpjiqc9oleue96i
Hibernate:
    alter table Supplier
       add constraint UK_ggs7vq4d52vpjiqc9oleue96i unique (address_id)
Hibernate:
    alter table Supplier
       add constraint FKl89csswjw3604alcbijxr3hrq
       foreign key (address_id)
       references Address

Process finished with exit code 0
```

Zadanie 9

Single table

```java
@Entity
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name = "company_type")
public class Company {
    @Id @GeneratedValue(strategy = GenerationType.AUTO)
    private Long companyId;
    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;
    }
}
```

```java
@Entity
@DiscriminatorValue("CUSTOMER")
public class Customer extends Company{
    private double discount;

    public Customer(double discount, String companyName, String street, String city, String zipCode) {
        super(companyName, street, city, zipCode);
        this.discount = discount;
    }
    public Customer() {}

}
```

```java
@Entity
@DiscriminatorValue("SUPPLIER")
public class Supplier extends Company {
    private String accountNumber;

    public Supplier() {}

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

```
Hibernate:
    /* insert for
        org.example.model.Supplier */insert
    into
        Company (city, companyName, street, zipCode, accountNumber, company_type, companyId)
    values
        (?, ?, ?, ?, ?, 'SUPPLIER', ?)
Hibernate:
    /* insert for
        org.example.model.Supplier */insert
    into
        Company (city, companyName, street, zipCode, accountNumber, company_type, companyId)
    values
        (?, ?, ?, ?, ?, 'SUPPLIER', ?)
Hibernate:
    /* insert for
        org.example.model.Customer */insert
    into
        Company (city, companyName, street, zipCode, discount, company_type, companyId)
    values
        (?, ?, ?, ?, ?, 'CUSTOMER', ?)
Hibernate:
    /* insert for
        org.example.model.Customer */insert
    into
        Company (city, companyName, street, zipCode, discount, company_type, companyId)
    values
        (?, ?, ?, ?, ?, 'CUSTOMER', ?)
Hibernate:
    /*
from
    Company */ select
        c1_0.companyId,
        c1_0.company_type,
        c1_0.city,
        c1_0.companyName,
        c1_0.street,
        c1_0.zipCode,
        c1_0.discount,
        c1_0.accountNumber
    from
        Company c1_0

=== Wczytane firmy z bazy ===
Supplier: org.example.model.Supplier@322204dc
Supplier: org.example.model.Supplier@562919fe
Customer: org.example.model.Customer@5d5c41e5
Customer: org.example.model.Customer@72b0a004
```

Table per class

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

@Entity
public class Customer extends Company

@Entity
public class Supplier extends Company
```

```
values
    next value for Company_SEQ
Hibernate:
    /* insert for
        org.example.model.Supplier */insert
    into
        Supplier (city, companyName, street, zipCode, accountNumber, companyId)
    values
        (?, ?, ?, ?, ?, ?)
Hibernate:
    /* insert for
        org.example.model.Supplier */insert
    into
        Supplier (city, companyName, street, zipCode, accountNumber, companyId)
    values
        (?, ?, ?, ?, ?, ?)
Hibernate:
    /* insert for
        org.example.model.Customer */insert
    into
        Customer (city, companyName, street, zipCode, discount, companyId)
    values
        (?, ?, ?, ?, ?, ?)
Hibernate:
    /* insert for
        org.example.model.Customer */insert
    into
        Customer (city, companyName, street, zipCode, discount, companyId)
    values
        (?, ?, ?, ?, ?, ?)
Hibernate:
    /*
from
    Company */ select
        c1_0.companyId,
        c1_0.clazz_,
        c1_0.city,
        c1_0.companyName,
        c1_0.street,
        c1_0.zipCode,
        c1_0.discount,
        c1_0.accountNumber
    from
        (select
            companyId,
            city,
            companyName,
            street,
```

```
    from
        (select
            companyId,
            city,
            companyName,
            street,
            zipCode,
            nullif('', '') as accountNumber,
            nullif(0, 0) as discount,
            0 as clazz_
        from
            Company
        union
        all select
            companyId,
            city,
            companyName,
            street,
            zipCode,
            accountNumber,
            nullif(0, 0) as discount,
            1 as clazz_
        from
            Supplier
        union
        all select
            companyId,
            city,
            companyName,
            street,
            zipCode,
            nullif('', '') as accountNumber,
            discount,
            2 as clazz_
        from
            Customer
    ) c1_0

=== Wczytane firmy z bazy ===
Supplier: org.example.model.Supplier@72028a45
Supplier: org.example.model.Supplier@5e0c4f21
Customer: org.example.model.Customer@2c18a3ea
Customer: org.example.model.Customer@5927f904
```

Joined

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

```
values
    next value for Company_SEQ
Hibernate:
    /* insert for
        org.example.model.Supplier */insert
    into
        Company (city, companyName, street, zipCode, companyId)
    values
        (?, ?, ?, ?, ?)
Hibernate:
    /* insert for
        org.example.model.Supplier */insert
    into
        Supplier (accountNumber, companyId)
    values
        (?, ?)
Hibernate:
    /* insert for
        org.example.model.Supplier */insert
    into
        Company (city, companyName, street, zipCode, companyId)
    values
        (?, ?, ?, ?, ?)
Hibernate:
    /* insert for
        org.example.model.Supplier */insert
    into
        Supplier (accountNumber, companyId)
    values
        (?, ?)
Hibernate:
    /* insert for
        org.example.model.Customer */insert
    into
        Company (city, companyName, street, zipCode, companyId)
    values
        (?, ?, ?, ?, ?)
Hibernate:
    /* insert for
        org.example.model.Customer */insert
    into
        Customer (discount, companyId)
    values
        (?, ?)
```

```
Hibernate:
    /* insert for
        org.example.model.Customer */insert
    into
        Company (city, companyName, street, zipCode, companyId)
    values
        (?, ?, ?, ?, ?)
Hibernate:
    /* insert for
        org.example.model.Customer */insert
    into
        Customer (discount, companyId)
    values
        (?, ?)
Hibernate:
    /*
from
    Company */ select
        c1_0.companyId,
        case
            when c1_2.companyId is not null
                then 1
            when c1_1.companyId is not null
                then 2
            when c1_0.companyId is not null
                then 0
            end,
        c1_0.city,
        c1_0.companyName,
        c1_0.street,
        c1_0.zipCode,
        c1_1.discount,
        c1_2.accountNumber
    from
        Company c1_0
    left join
        Customer c1_1
            on c1_0.companyId=c1_1.companyId
    left join
        Supplier c1_2
            on c1_0.companyId=c1_2.companyId

=== Wczytane firmy z bazy ===
Supplier: org.example.model.Supplier@23e3f5cd
Supplier: org.example.model.Supplier@3b97907c
Customer: org.example.model.Customer@59096b66
Customer: org.example.model.Customer@6d87cd177
```

Main class

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

    Transaction tx = session.beginTransaction();

    Supplier supplier1 = new Supplier("ACC123", "Tech Supplies", "First St", "CityA", "12345");
    Supplier supplier2 = new Supplier("ACC124", "Build Co", "Second St", "CityB", "23456");

    Customer customer1 = new Customer(10.5, "Happy Buyer", "Third St", "CityC", "34567");
    Customer customer2 = new Customer(7.0, "Budget Buyer", "Fourth St", "CityD", "45678");

    session.persist(supplier1);
    session.persist(supplier2);
    session.persist(customer1);
    session.persist(customer2);

    tx.commit();

    List<Company> companies = session.createQuery("from Company", Company.class).list();

    System.out.println("\n=== Wczytane firmy z bazy ===");
    companies.forEach(company -> {
        System.out.println(company.getClass().getSimpleName() + ": " + company);
    });

    session.close();
}
```