

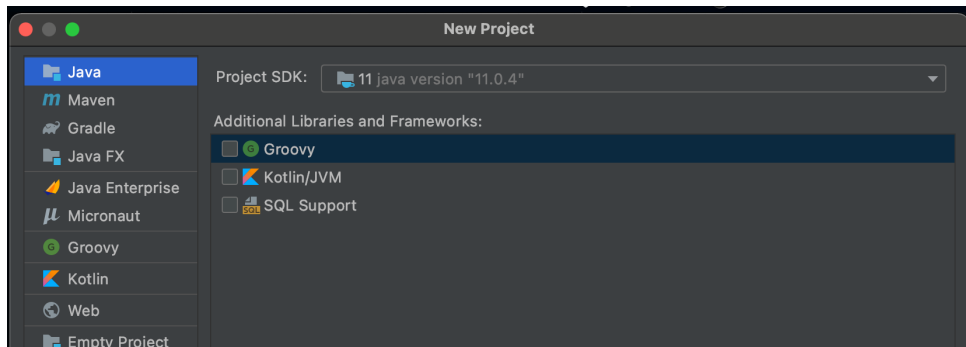
Hibernate, JPA – laboratorium

I. Basics

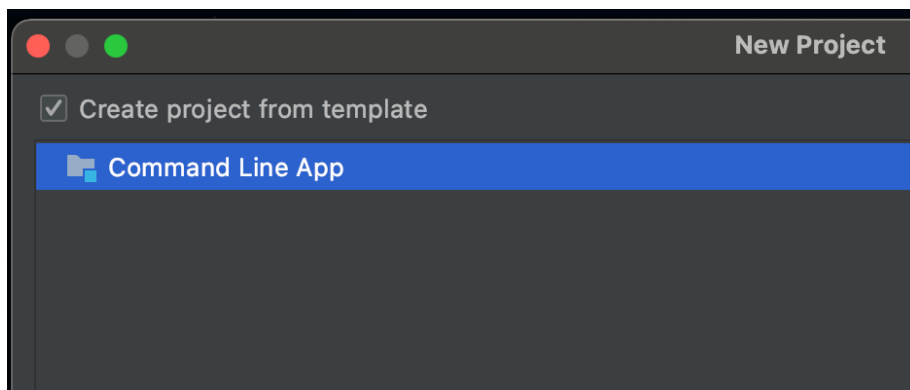
- Sciągnij i rozpakuj serwer bazodanowy Apache Derby
<https://ftp.man.poznan.pl/apache//db/derby/db-derby-10.15.2.0/db-derby-10.15.2.0-bin.zip>
- Uruchom serwer Derby (skrypt startNetworkServer z podkatalogu bin ściągniętej paczki). Powinieneś uzyskać efekt podobny do poniższego:

```
-lwxr-xr-x@ 1 macbookpro staff 1389 6 sty 2019 sysinfo.bat
(base) MacBook-Pro-macbook:bin macbookpro$ ./startNetworkServer
Wed Apr 28 21:17:11 CEST 2021 : Security manager installed using the Basic server security policy.
Wed Apr 28 21:17:22 CEST 2021 : Serwer sieciowy Apache Derby - 10.15.2.0 - (1873585) uruchomiony i gotowy do zaakceptowania połączeń na porcie 1527 w {3}
```

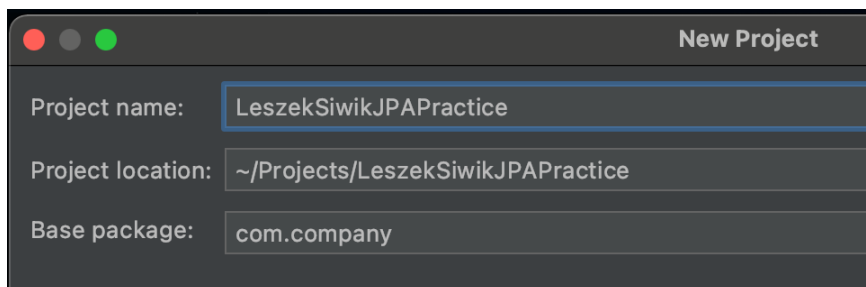
- Wędrujemy do IntelliJ'a. Tworzymy nowy projekt typu Java,



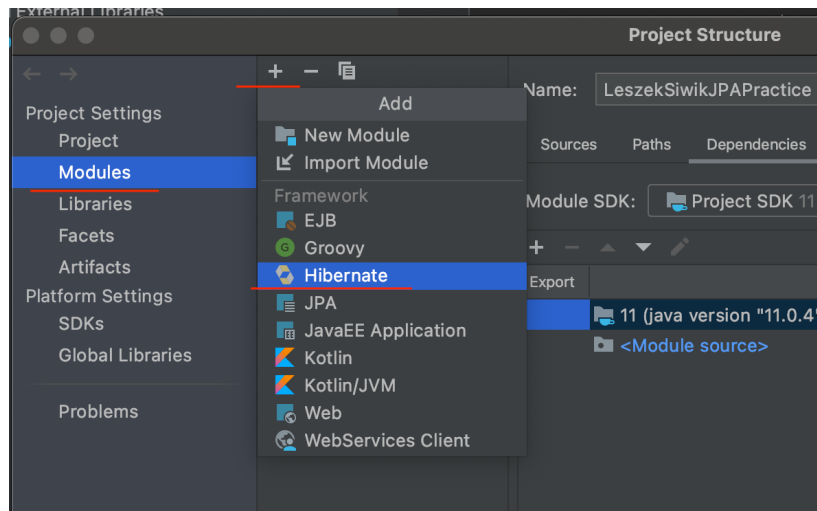
- Jako wzorec wybierzmy sobie aplikację command line'owa



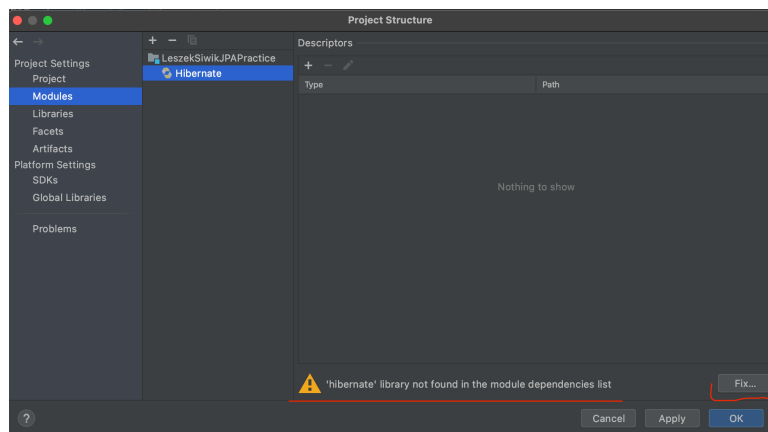
- Nazwijmy projekt ImieNazwiskoJPAPractice



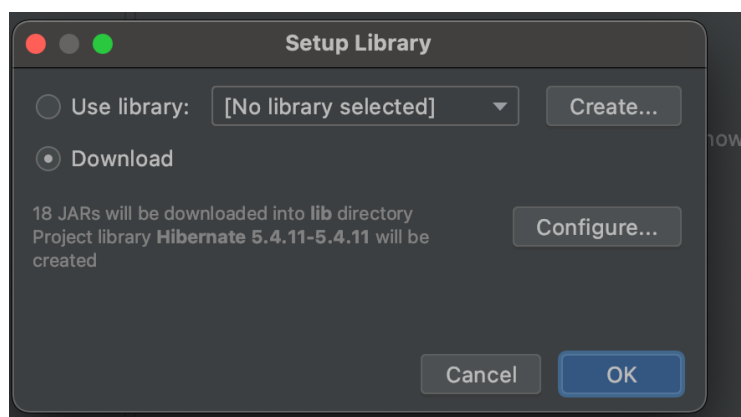
- f. Po utworzeniu projektu wędrujemy do okna zarządzania zależnościami (File->Project Structure). Do modułów dodajemy Hibernate'a



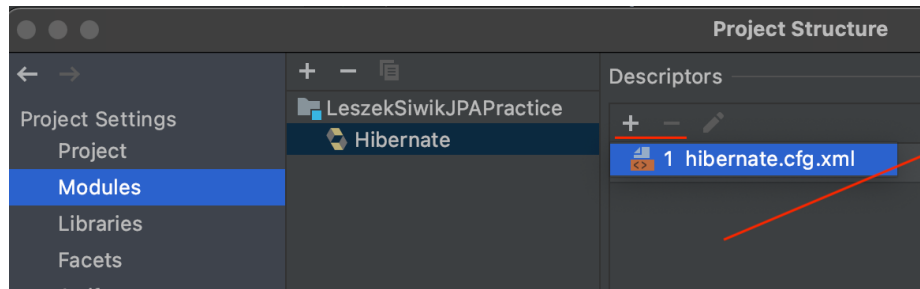
- g.
- h. Następnie fiksujemy problem braku hibernate na liście zależności



- i.
- j. I dociągamy hibernate'a do projektu

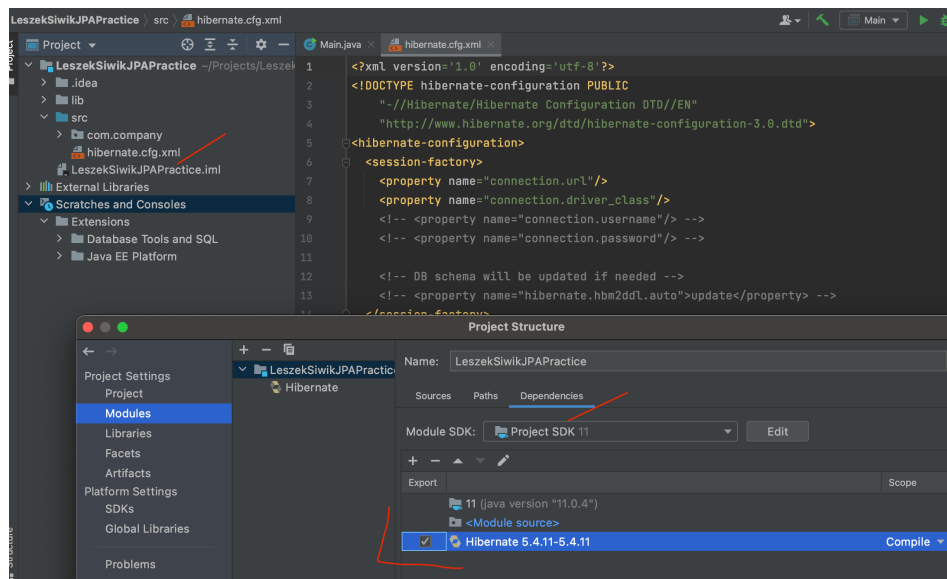


- k. Od razu możemy wygenerować sobie plik konfiguracyjny



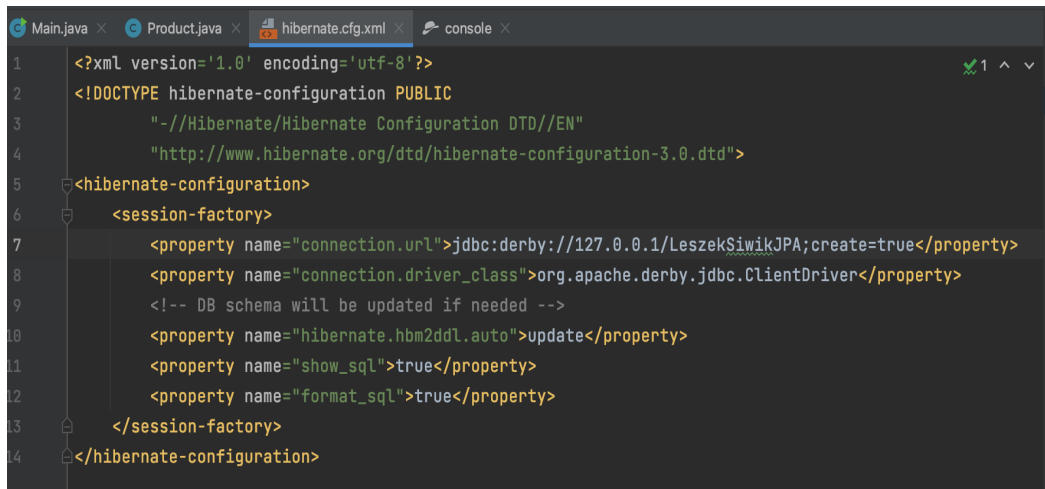
l.

- m. Po dotychczasowych krokach sytuacja powinna być taka, że w zależnościach projektu mamy Hibernate'a, i dodatkowo w źródłach mamy wygenerowany plik konfiguracyjny



n.

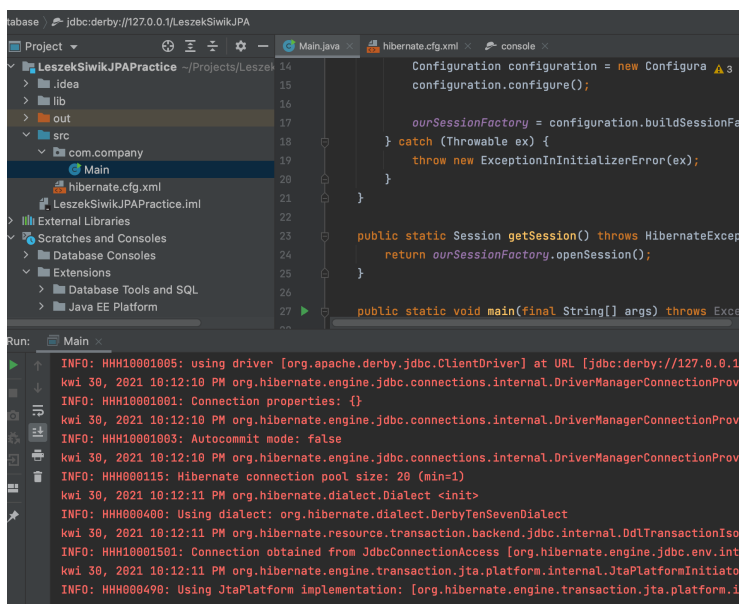
- o. Dołącz do projektu (File→Project Structure → Modules→ Dependencies) Jar-ki Związane z obsługą/komunikacją z Derby (derby.jar, derbyclient.jar, derbynet.jar, derbytools.jar). Znajdziesz je w podkatalogu lib ściągniętego Apache Derby.
- p. Uzupełnij wpisy w hibernate.cfg.xml podając driver, connection_url, dodaj opcje show_sql oraz format_sql, a także hbm2ddl na update. Przed pierwszym uruchomieniem connection_url powinien wyglądać jak poniżej
 jdbc:derby://127.0.0.1/INazwiskoJPA;create=true; (z dokładnością do nazwy bazy danych). Po pierwszym uruchomieniu dopisek create=true może zostać usunięty. Reasumując config powinien wyglądać podobnie do poniższego:



```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD//EN"
    "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="connection.url">jdbc:derby://127.0.0.1/LeszekSiwikJPA;create=true</property>
        <property name="connection.driver_class">org.apache.derby.jdbc.ClientDriver</property>
        <!-- DB schema will be updated if needed -->
        <property name="hibernate.hbm2ddl.auto">update</property>
        <property name="show_sql">true</property>
        <property name="format_sql">true</property>
    </session-factory>
</hibernate-configuration>
```

q.

- r. Uzupełnij klasę main zgodnie ze wzorcem dostępnym na platformie UPEL
- s. Uruchom projekt, Na razie nie będzie się wiele działo ale na konsoli powinieneś zobaczyć wpisy hibernate, bez żadnych wyjątków, czyli mniej więcej stan jak poniżej:



```
Configuration configuration = new Configuration();
configuration.configure();

SessionFactory sessionFactory = configuration.buildSessionFactory();
} catch (Throwable ex) {
    throw new ExceptionInInitializerError(ex);
}

public static Session getSession() throws HibernateException {
    return sessionFactory.openSession();
}

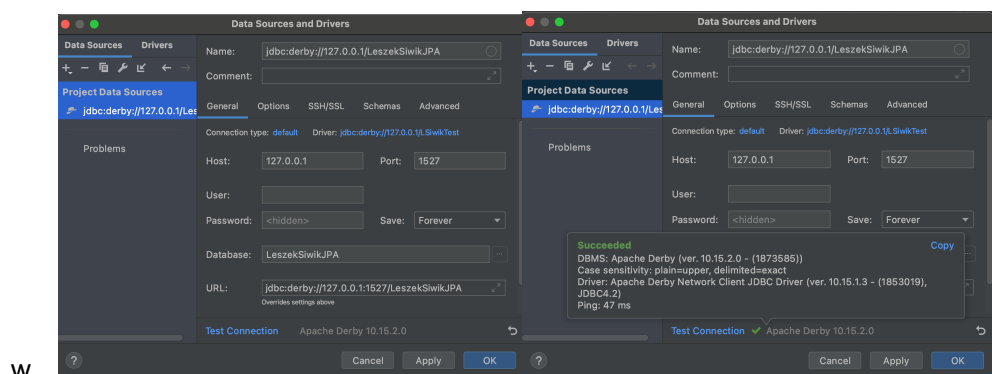
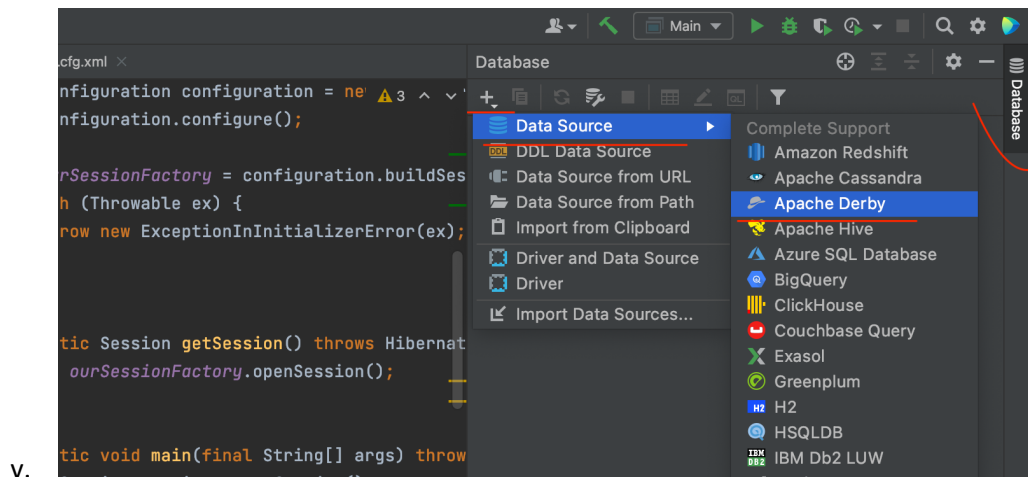
public static void main(final String[] args) throws Exception {
    // ...
}
```

Run: Main

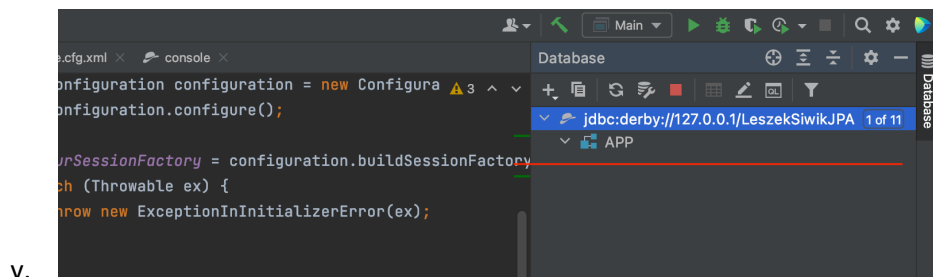
```
INFO: HHH10001005: using driver [org.apache.derby.jdbc.ClientDriver] at URL [jdbc:derby://127.0.0.1/LeszekSiwikJPA;create=true]
kwi 30, 2021 10:12:10 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProvider
INFO: HHH10001001: Connection properties: {}
kwi 30, 2021 10:12:10 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProvider
INFO: HHH10001003: Autocommit mode: false
kwi 30, 2021 10:12:10 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProvider
INFO: HHH000115: Hibernate connection pool size: 20 (min=1)
kwi 30, 2021 10:12:11 PM org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.DerbyTenSevenDialect
kwi 30, 2021 10:12:11 PM org.hibernate.resource.transaction.backend.jdbc.internal.DdlTransactionIsolator
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.inte
kwi 30, 2021 10:12:11 PM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator
INFO: HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.in
```

t.

- u. a na serwerze powinna się założyć baza o zdefiniowanej w konfigu nazwie. Możesz to zweryfikować podpinając się np. z poziomu IntelliJ do uruchomionego serwera Derby do bazy o podanej nazwie i to połączenie powinno się powieść.



- x. Po podpięciu do bazy na serwerze powinien być widoczny schemat APP – na razie pusty bo nie dodawaliśmy tam żadnych tabel – czyli stan jak poniżej



I. Praca z modelem

- Stwórz klasę produktu z polami ProductName, UnitsOnStock
- Uzupełnij definicję klasy o elementy niezbędne do jej zmapowania do bazy danych przez Hibernate (adnotacja @Entity, nominowanie pola ID, pusty konstruktor)
- Rozszerzamy maina o stworzenie nowego produktu i zapisanie go w bazie danych z wykorzystaniem hibernate'a. Uruchamiamy i testujemy projekt. Efekty powinny być następujące:

d.

```

// Product.java
final Session session = getSession();
Product product = new Product("Krzyszto", unitsOnStock: 111);
try {
    Transaction tx = session.beginTransaction();
    session.save(product);
    tx.commit();
} catch (Exception e) {
    // ...
}

// Main.java
// ...

// Console
INFO: HHH18081581: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnvironment]
Hibernate:
    create table Product (
        ProductID integer not null,
        ProductName varchar(255),
        UnitsOnStock integer,
        primary key (ProductID)
    )
Hibernate: create sequence hibernate_sequence start with 1 increment by 1
kwi 30, 2021 10:51:05 PM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator initiateService
INFO: HHH080490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
Hibernate:
    values
        next value for hibernate_sequence
Hibernate:
    insert
    into
        Product
        (ProductName, UnitsOnStock, ProductID)
    values
        (?, ?, ?)
  
```

e.

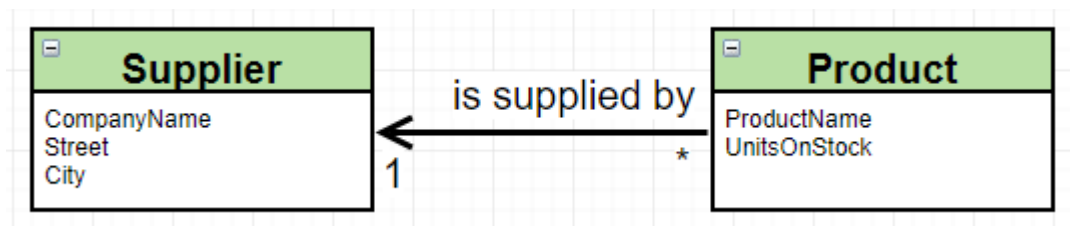
PRODUCTID	PRODUCTNAME	UNITSONSTOCK
1	Krzyszto	111

```

-- Schema
PRODUCT (
  PRODUCTID INTEGER,
  PRODUCTNAME VARCHAR(255),
  UNITSONSTOCK INTEGER,
  PRIMARY KEY (PRODUCTID)
)
  
```

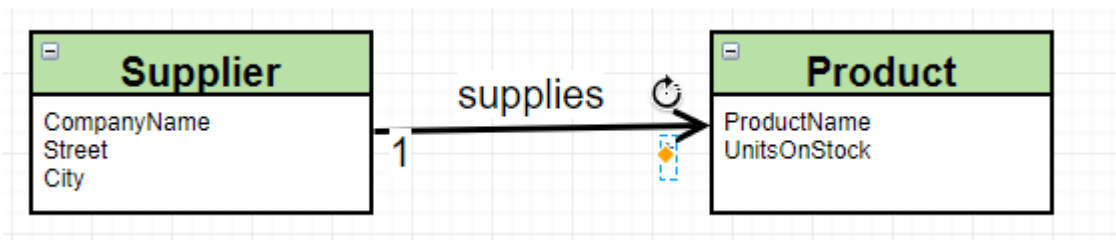
f. Udokumentuj wykonane kroki oraz uzyskany rezultat (logi wywołań słowych,describe table/schemat bazy, select * from....)

II. Zmodyfikuj model wprowadzając pojęcie Dostawcy jak poniżej



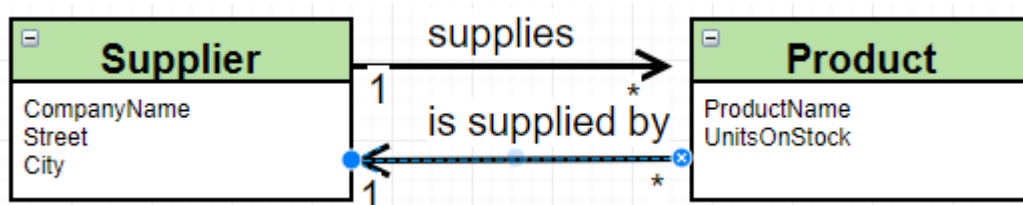
- Stworz nowego dostawce.
- Znajdz poprzednio wprowadzony produkt i ustaw jego dostawce na właśnie dodanego.
- Udokumentuj wykonane kroki oraz uzyskany rezultat (ogi wywołań słowych,describe table/schemat bazy danych, select * from....)

III. Odwróć relacje zgodnie z poniższym schematem



- Zamodeluj powyższe w dwóch wariantach „z” i „bez” tabeli łącznikowej
- Stwórz kilka produktów
- Dodaj je do produktów dostarczanych przez nowo stworzonego dostawcę
- Udokumentuj wykonane kroki oraz uzyskane rezultaty w obu wariantach (logi wywołań sqlowych, describe table/schemat bazy danych, select * from....)**

IV. Zamodeluj relację dwustronną jak poniżej:

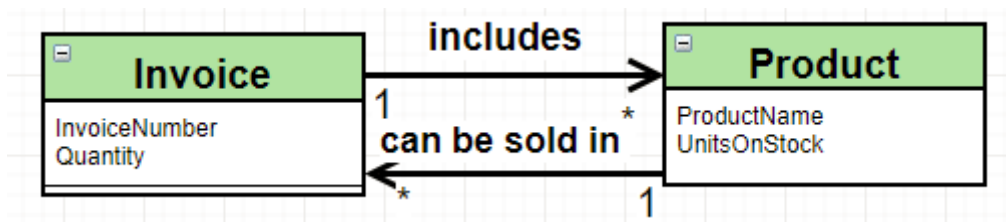


- Tradycyjnie: Stwórz kilka produktów
- Dodaj je do produktów dostarczanych przez nowo stworzonego dostawcę (pamiętaj o poprawnej obsłudze dwustronności relacji)
- Udokumentuj wykonane kroki oraz uzyskane rezultaty (logi wywołań sqlowych, describe table/schemat bazy danych, select * from....)**

V. Dodaj klasę Category z property int CategoryID, String Name oraz listą produktów List<Product> Products

- Zmodyfikuj produkty dodając wskazanie na kategorii do której należy.
- Stwórz kilka produktów i kilka kategorii
- Dodaj kilka produktów do wybranej kategorii
- Wydobądź produkty z wybranej kategorii oraz kategorię do której należy wybrany produkt
- Udokumentuj wykonane kroki oraz uzyskane rezultaty (logi wywołań sqlowych, describe table/schemat bazy danych, select * from....)**

VI. Zamodeluj relację wiele-do-wielu, jak poniżej:



- a. Stórz kilka produktów i “sprzedaj” je na kilku transakcjach.
- b. Pokaż produkty sprzedane w ramach wybranej faktury/transakcji
- c. Pokaż faktury w ramach których był sprzedany wybrany produkt
- d. **Udokumentuj wykonane kroki oraz uzyskane rezultaty (logi wywołań sqlowych, describe table/schemat bazy danych, select * from....)**

VII. JPA

- a. Stwórz nowego maina w którym zrobisz to samo co w poprzednim ale z wykorzystaniem JPA
- b. **Udokumentuj wykonane kroki oraz uzyskane rezultaty (logi wywołań sqlowych, describe table/schemat bazy danych, select * from....)**

VIII. Kaskady

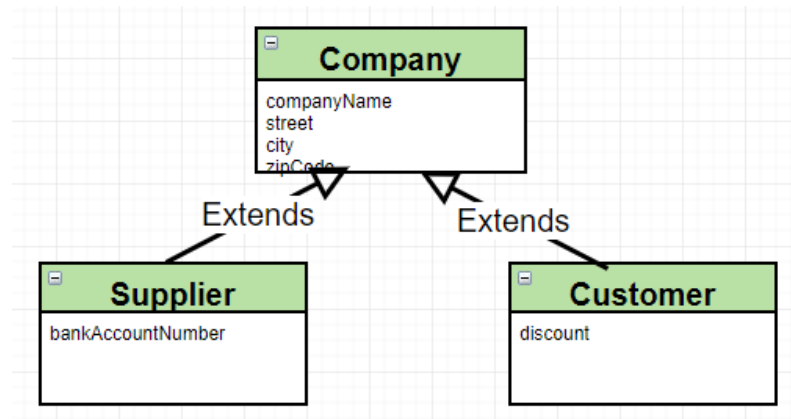
- a. Zmodyfikuj model w taki sposób aby było możliwe kaskadowe tworzenie faktur wraz z nowymi produktami, oraz produktów wraz z nową fakturą
- b. **Udokumentuj wykonane kroki oraz uzyskane rezultaty (logi wywołań sqlowych, describe table/schemat bazy danych, select * from....)**

IX. Embedded class

- a. Dodaj do modelu klasę adres. „Wbuduj” ją do tabeli Dostawców.
- b. **Udokumentuj wykonane kroki oraz uzyskane rezultaty (logi wywołań sqlowych, describe table/schemat, select * from....)**
- c. Zmodyfikuj model w taki sposób, że dane adresowe znajdują się w klasie dostawców. Zmapuj to do dwóch osobnych tabel.
- d. **Udokumentuj wykonane kroki oraz uzyskane rezultaty (logi wywołań sqlowych, describe table/schemat bazy danych, select * from....)**

X. Dziedziczenie

- a. Wprowadź do modelu następującą hierarchię:



- b. Dodaj i pobierz z bazy kilka firm obu rodzajów stosując po kolei trzy różne strategie mapowania dziedziczenia.
- c. **Udokumentuj wykonane kroki oraz uzyskane rezultaty (logi wywołań sqlowych, describe table/schemat bazy danych, select * from....)**