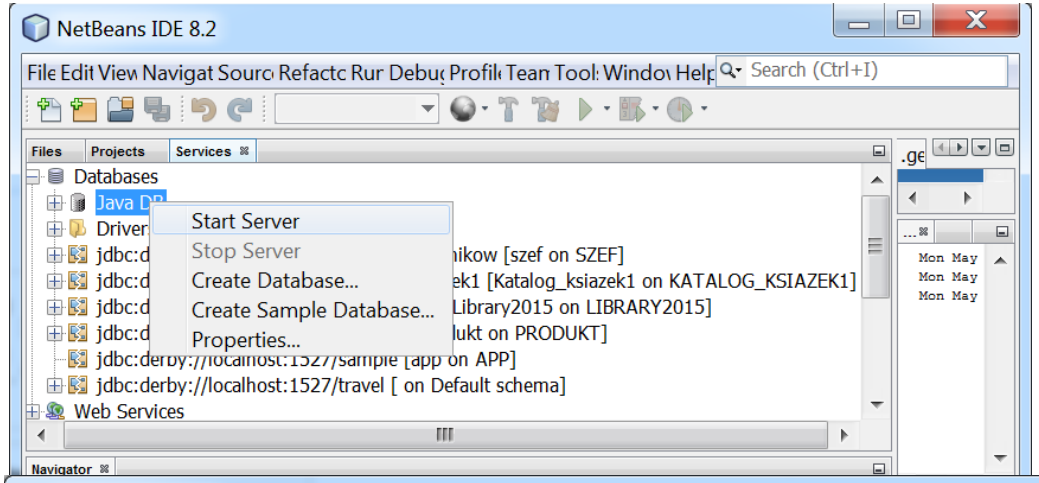
Protokół JDBC – współpraca z relacyjnymi bazami danych lab3

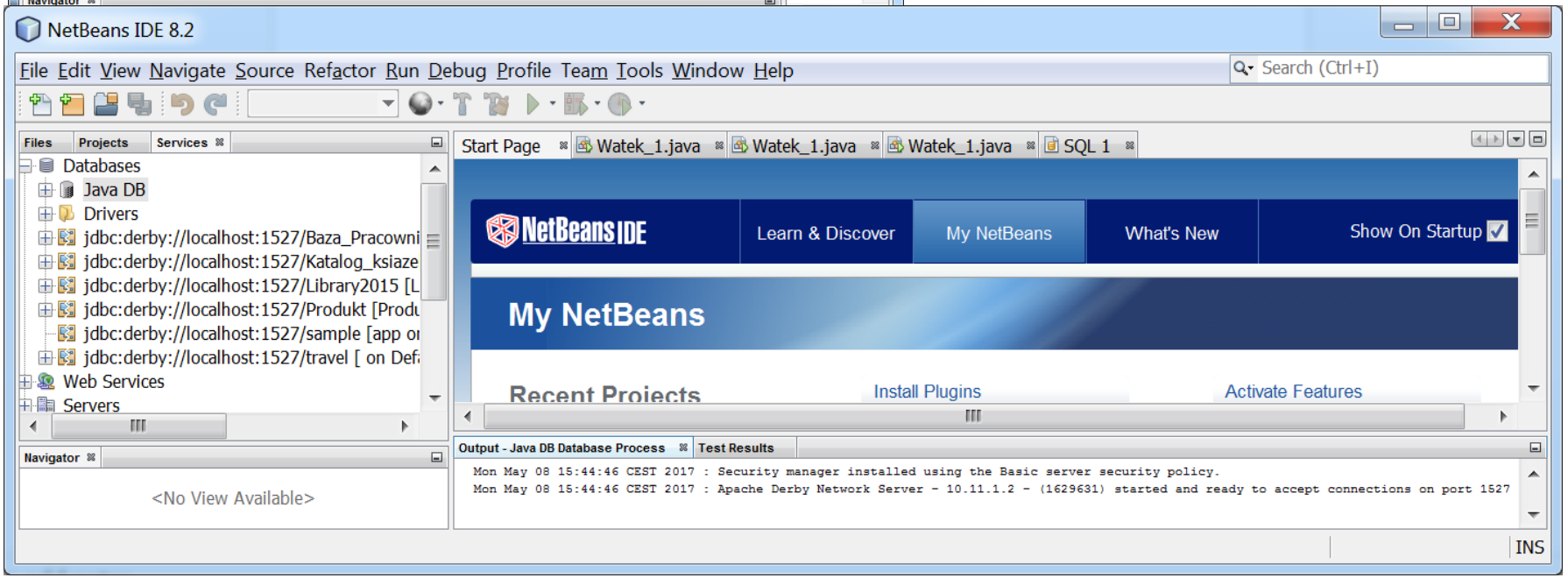
Dr inż. Zofia Kruczkiewicz

Programowanie aplikacji internetowych

Zadanie1 – Połączenie z bazą danych Sample systemu bazodanowego Derby (metoda void polaczenie_z_baza()), wyświetlanie zawartości tabeli CUSTOMER (metoda ArrayList<String> dane_tablicy_osob())

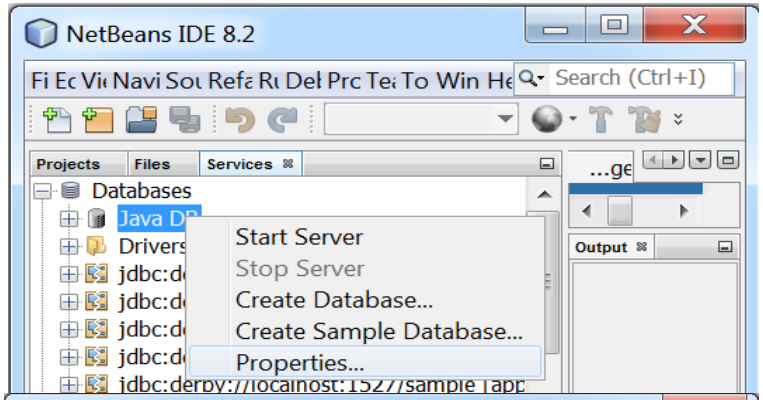
1a. Uruchom program Baza_1 w środowisku NetBeans 8.2 – załącznik do laboratorium. Należy uruchomić serwer bazy danych.W tym celu należy wykonać czynności pokazane poniżej w okienku Services. W p. 1b pokazano, jak zmienić program Java DB, gdy wystapią problemy uruchomieniem serwera Java DB.



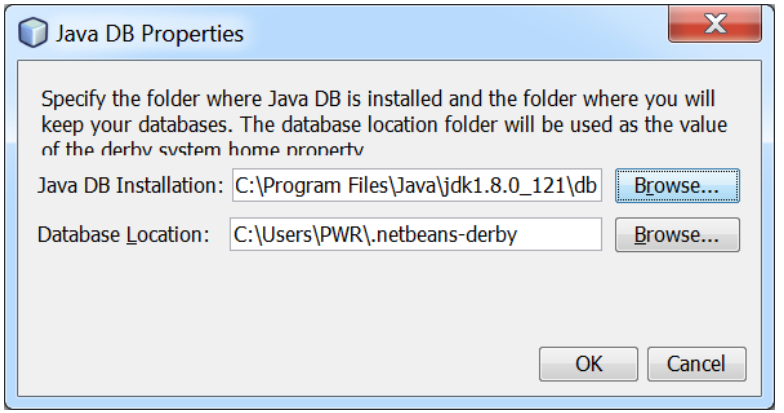


1b. Jeśli serwer **Java DB** nie startuje, należy zmienić jego program pobrany z zainstalowanego serwera glassfish 4.1.1 i ponownie uruchomić wg p. 1a.

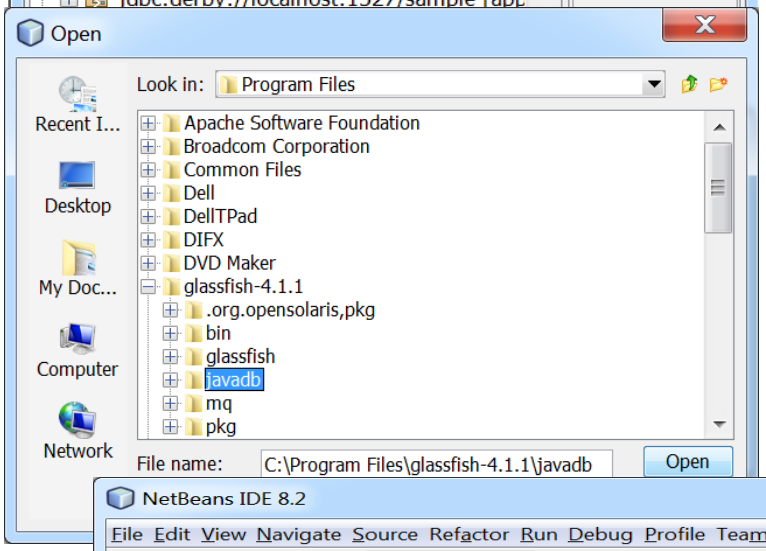
1



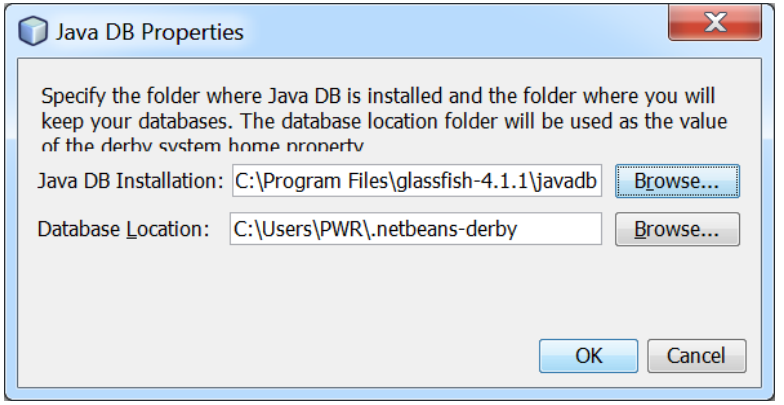
2



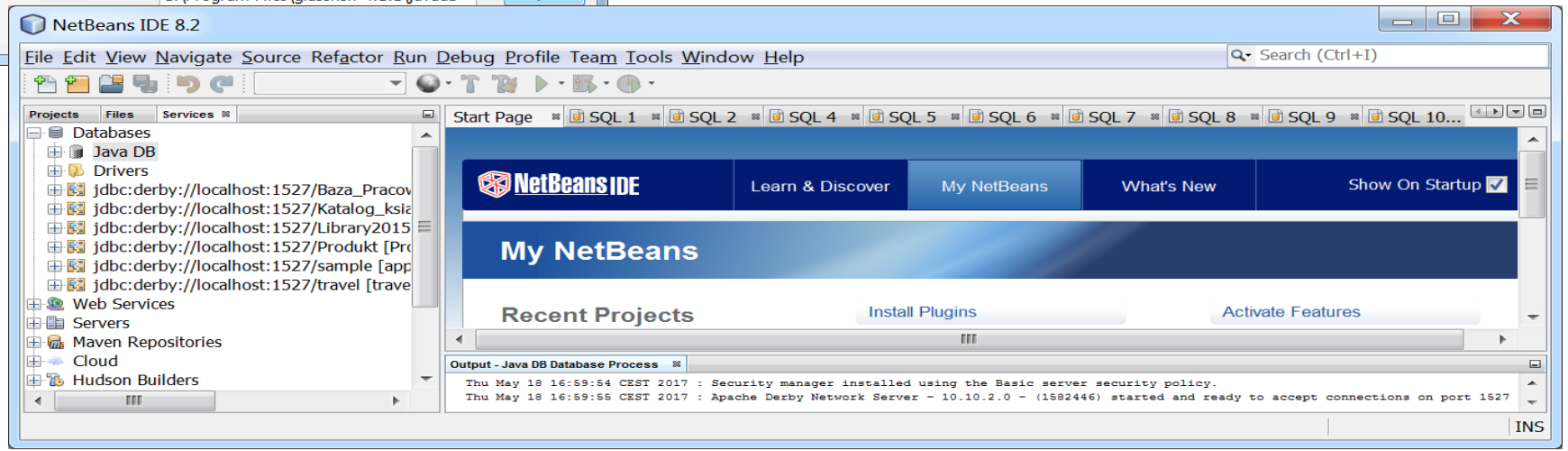
3



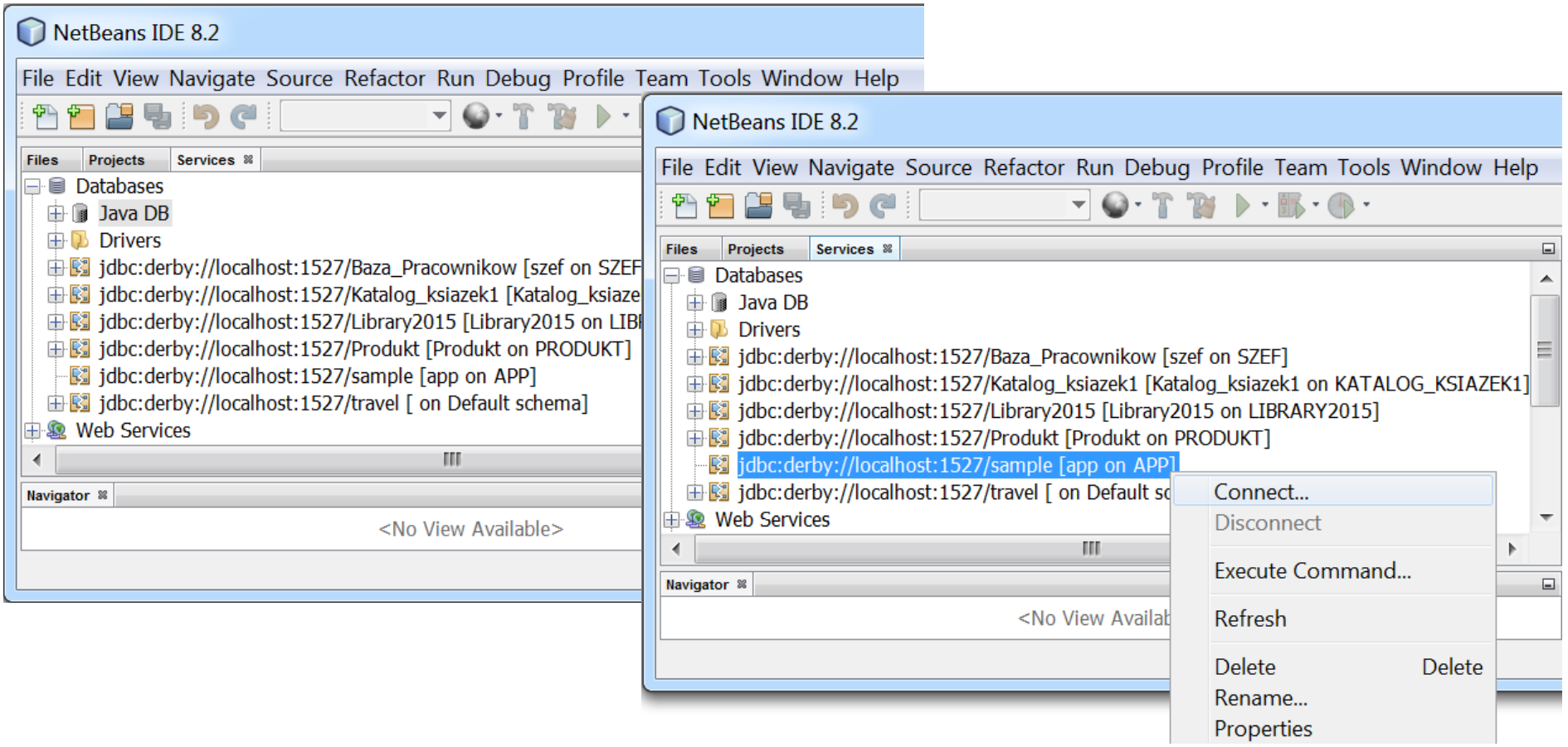
4

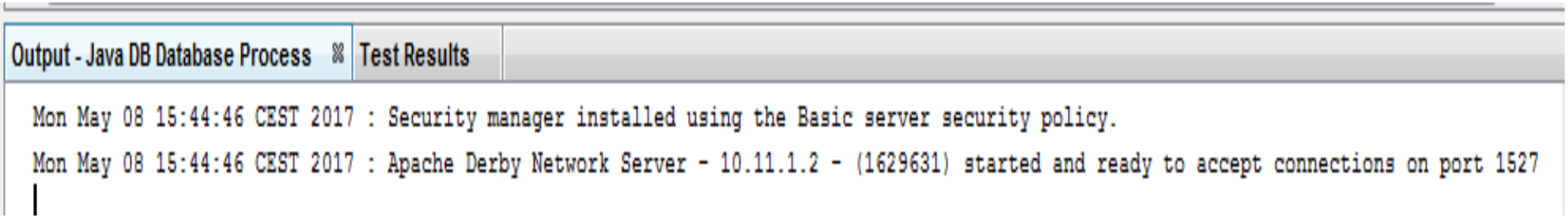


5

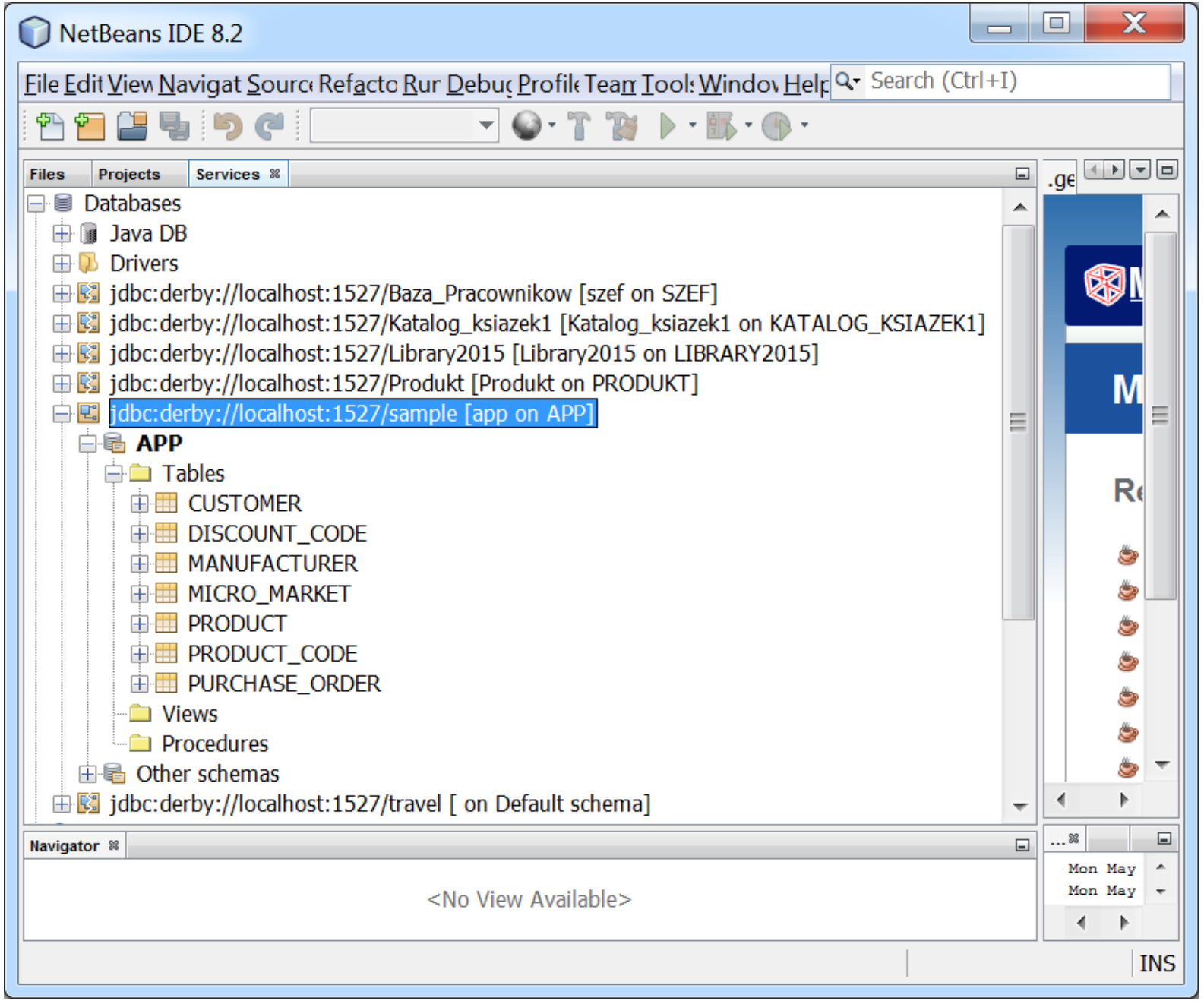


2. W okienku Services należy otworzyć pozycję Databases. Następnie należy kliknąć prawym klawiszem myszy na rozerwany prostokąt bazy danych **jdbc:derby://localhost:1527/sample**. W menu podręcznym należy kliknąć na pozycję Connect. Po chwili prostokąt zostanie scalony, co symbolizuje połączenie z bazą danych.

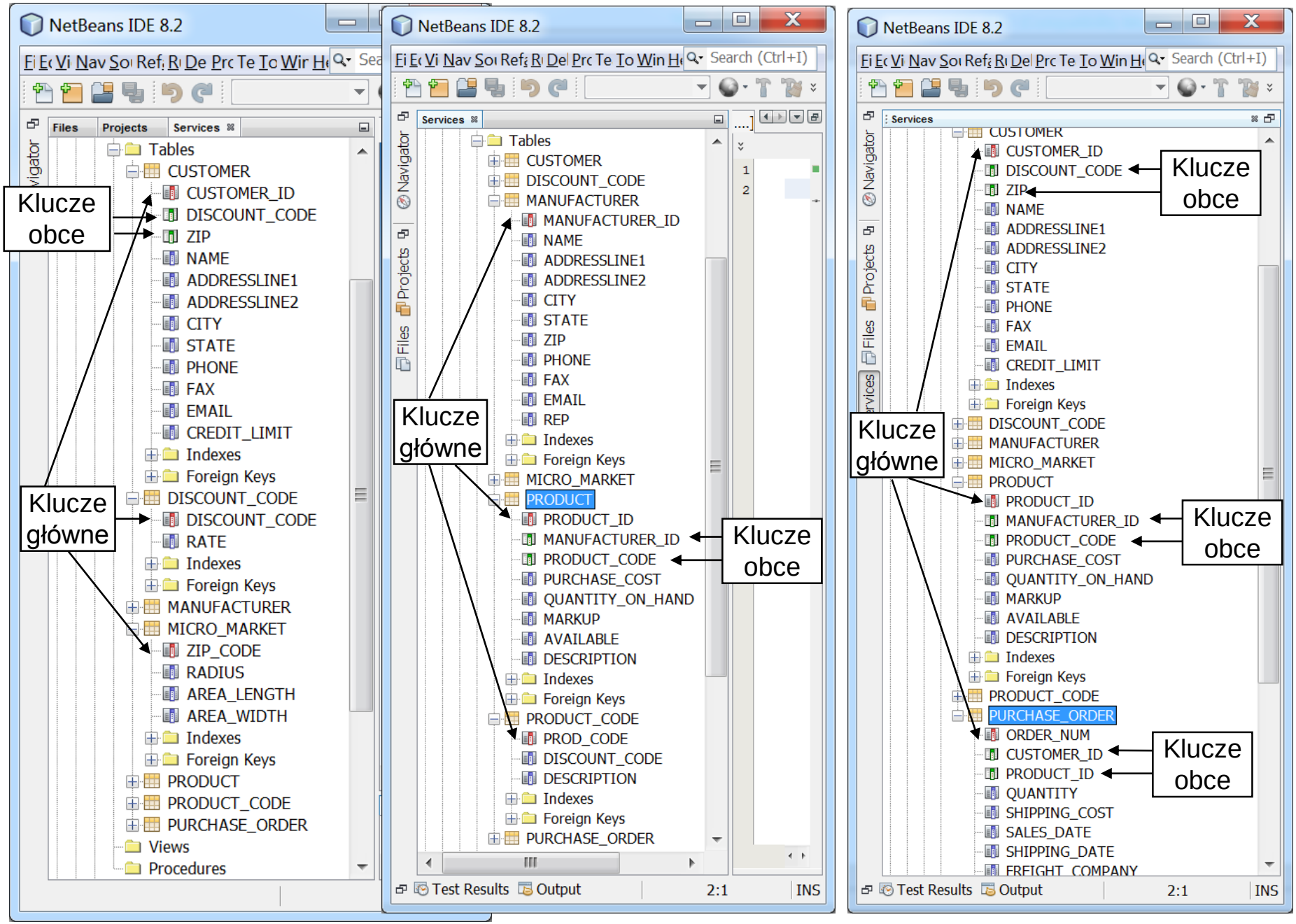




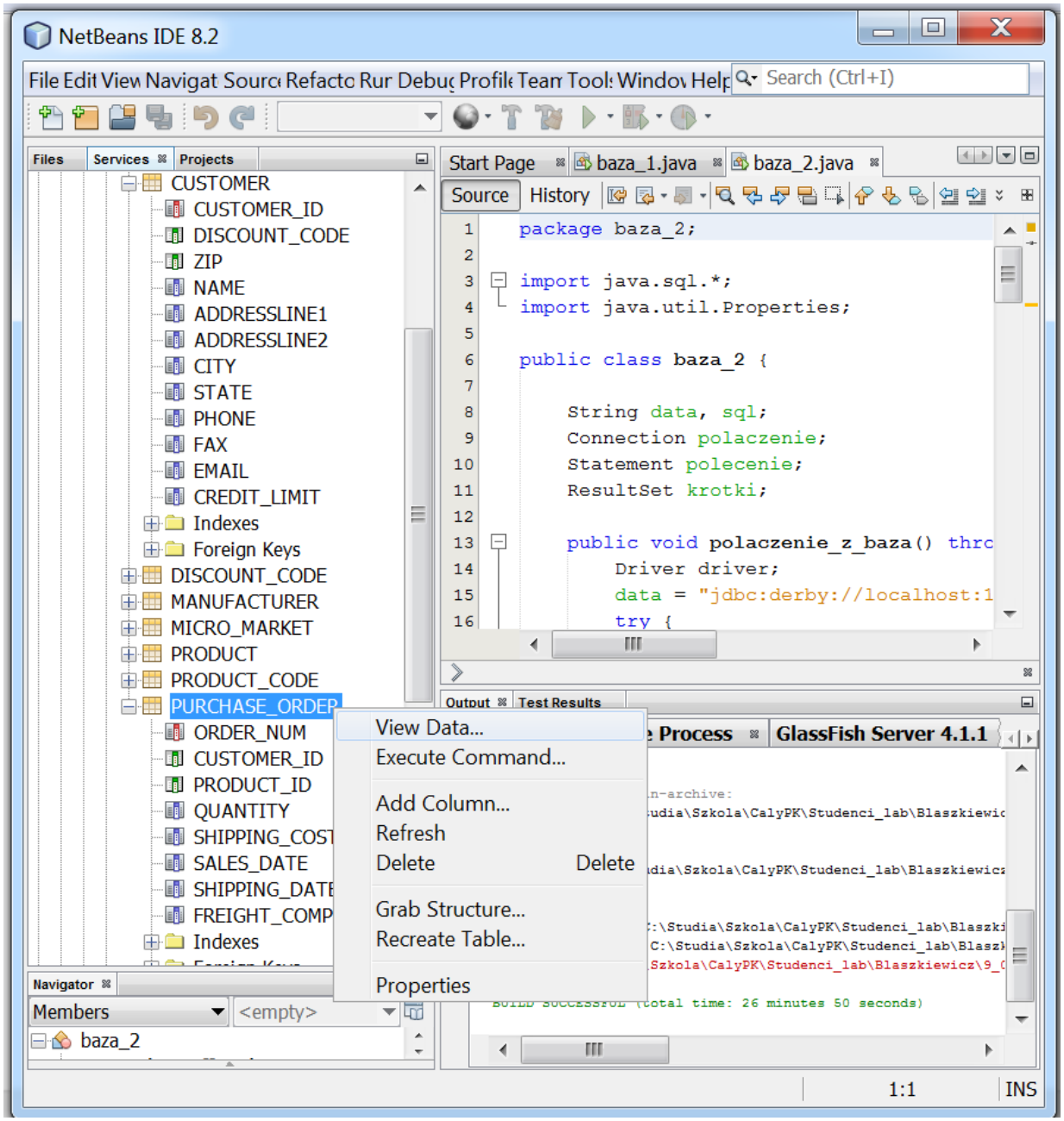
Widok bazy danych **Sample**



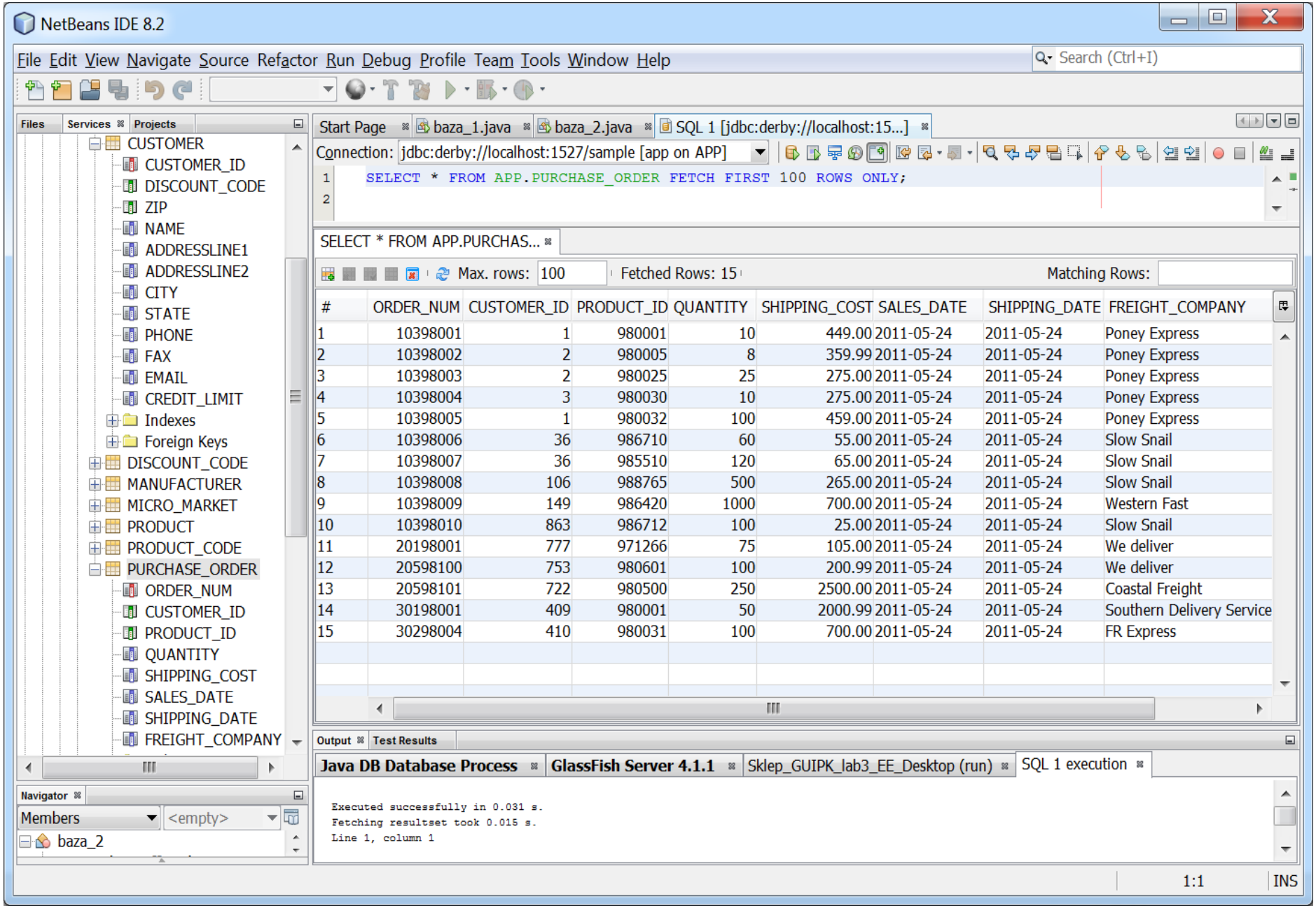
Schemat bazy danych **Sample**



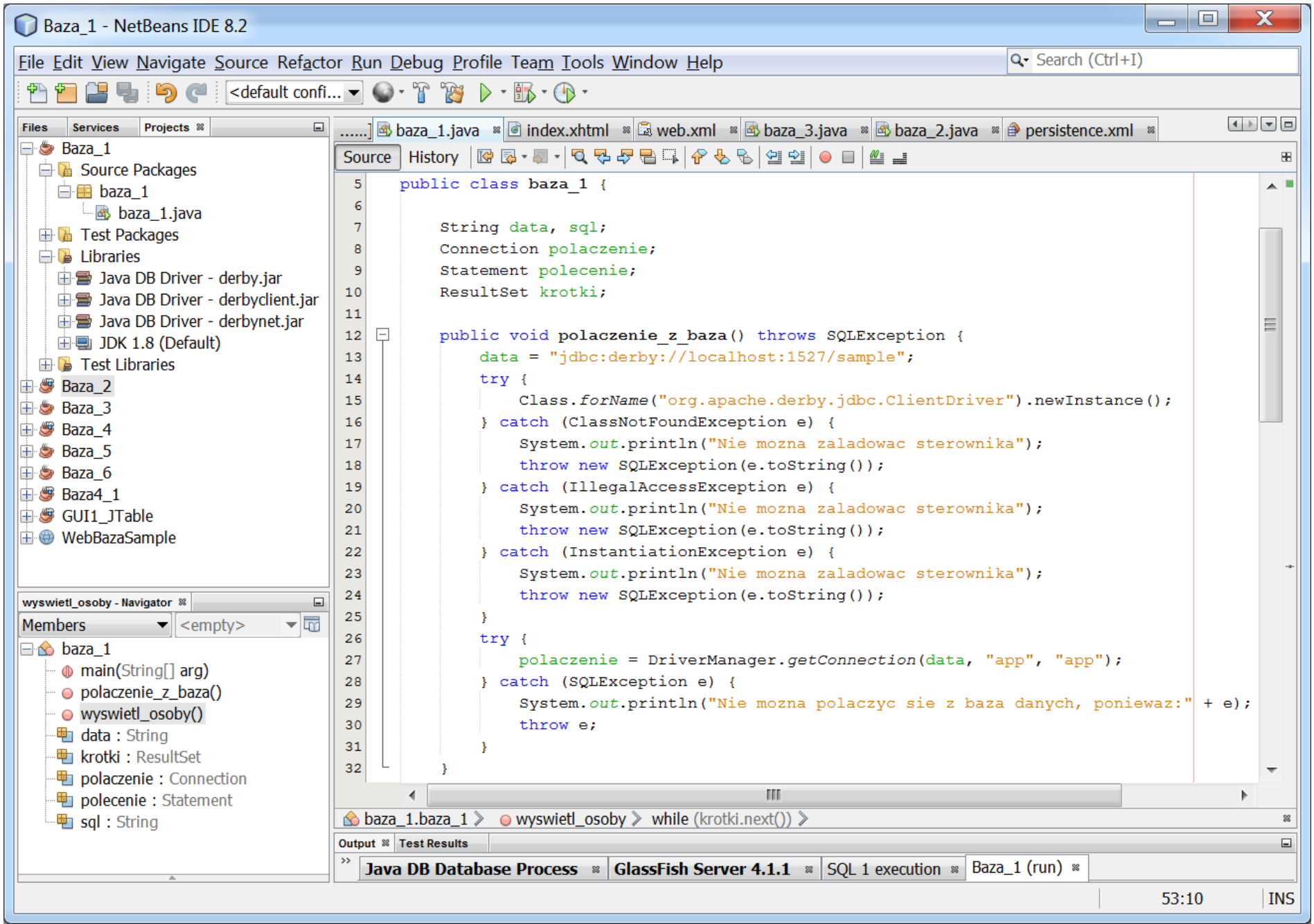
Wyświetlenie danych z tabeli **PURCHASE_ORDER**



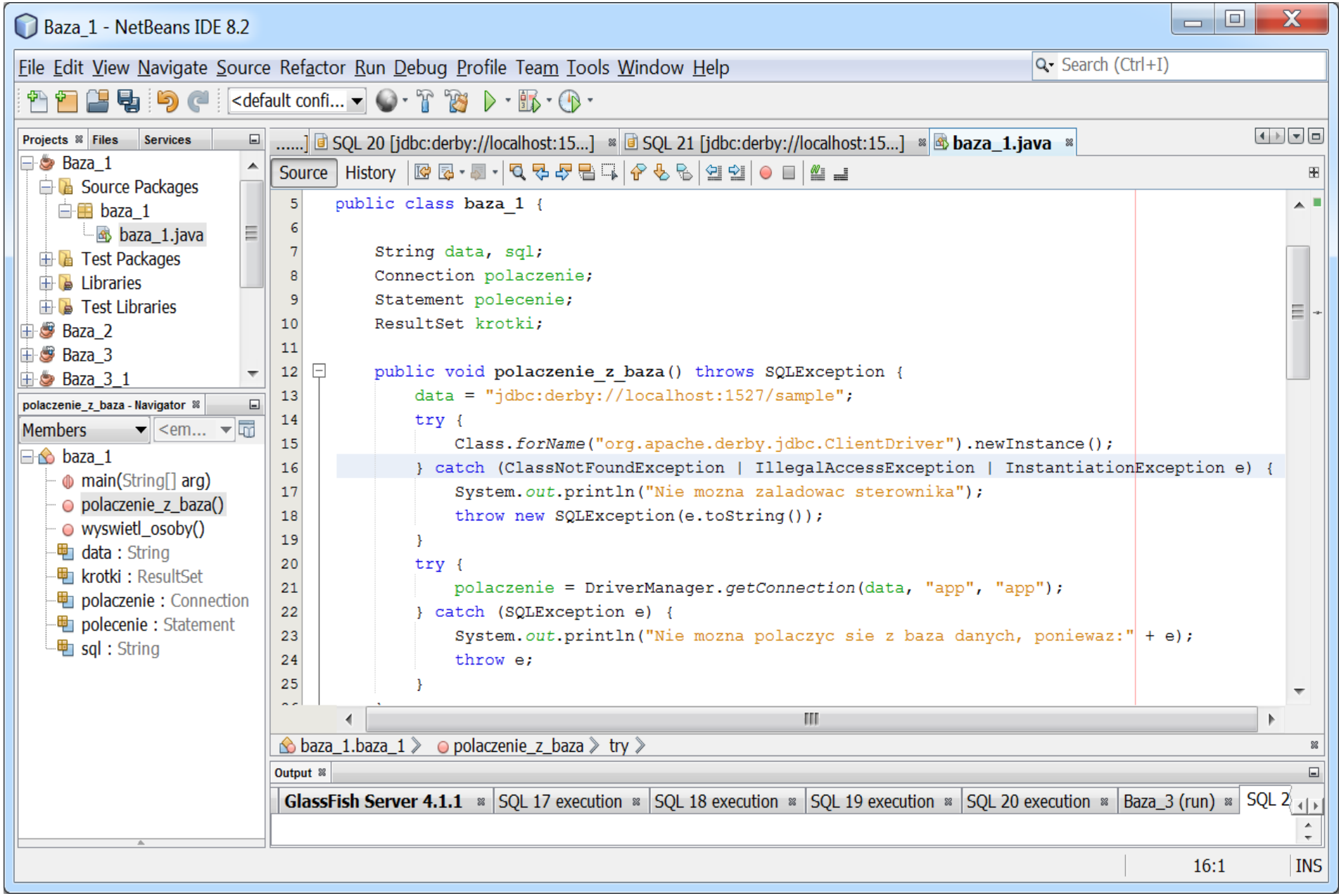
Zawartość tabeli **PURCHASE_ORDER**

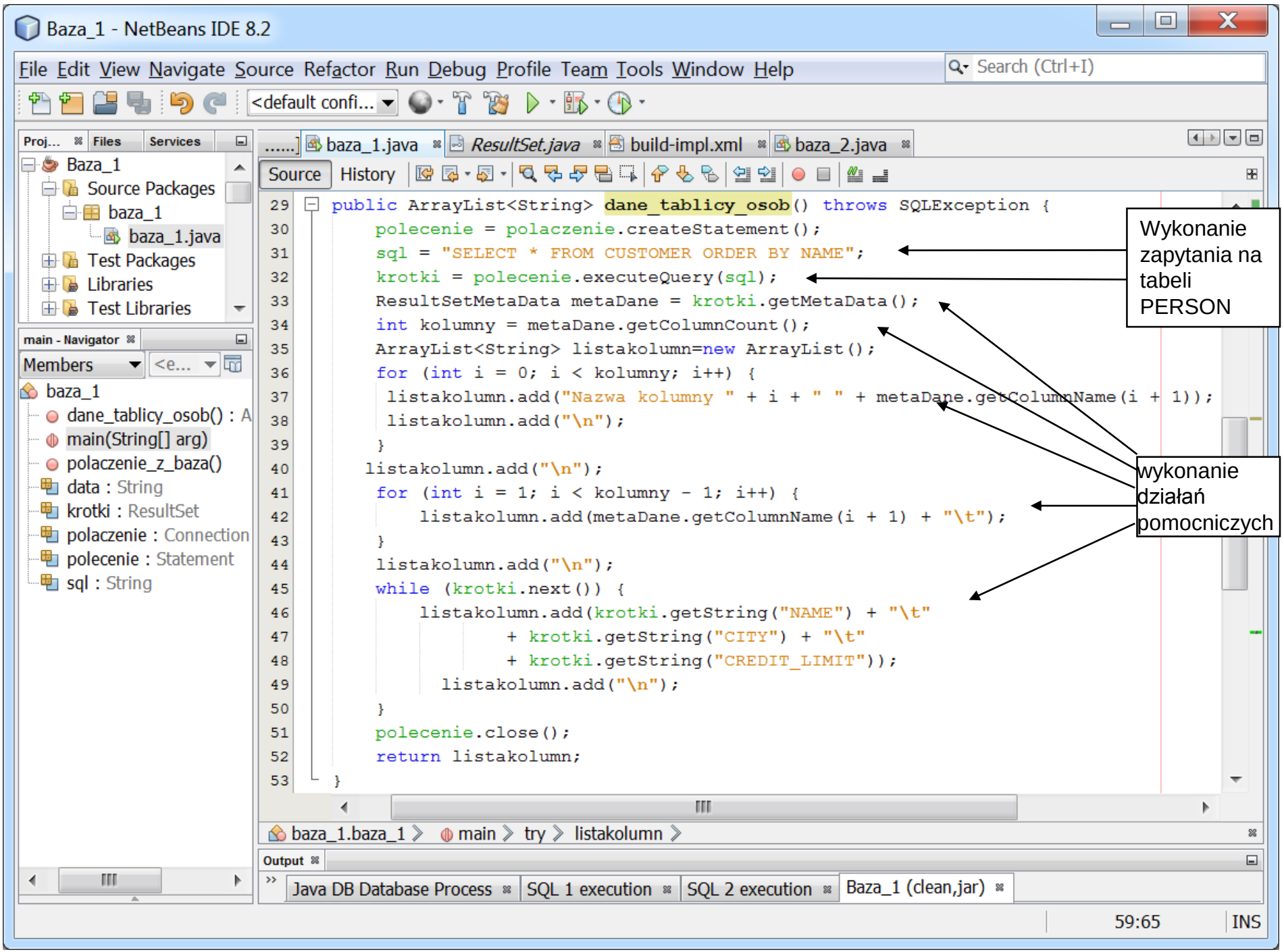


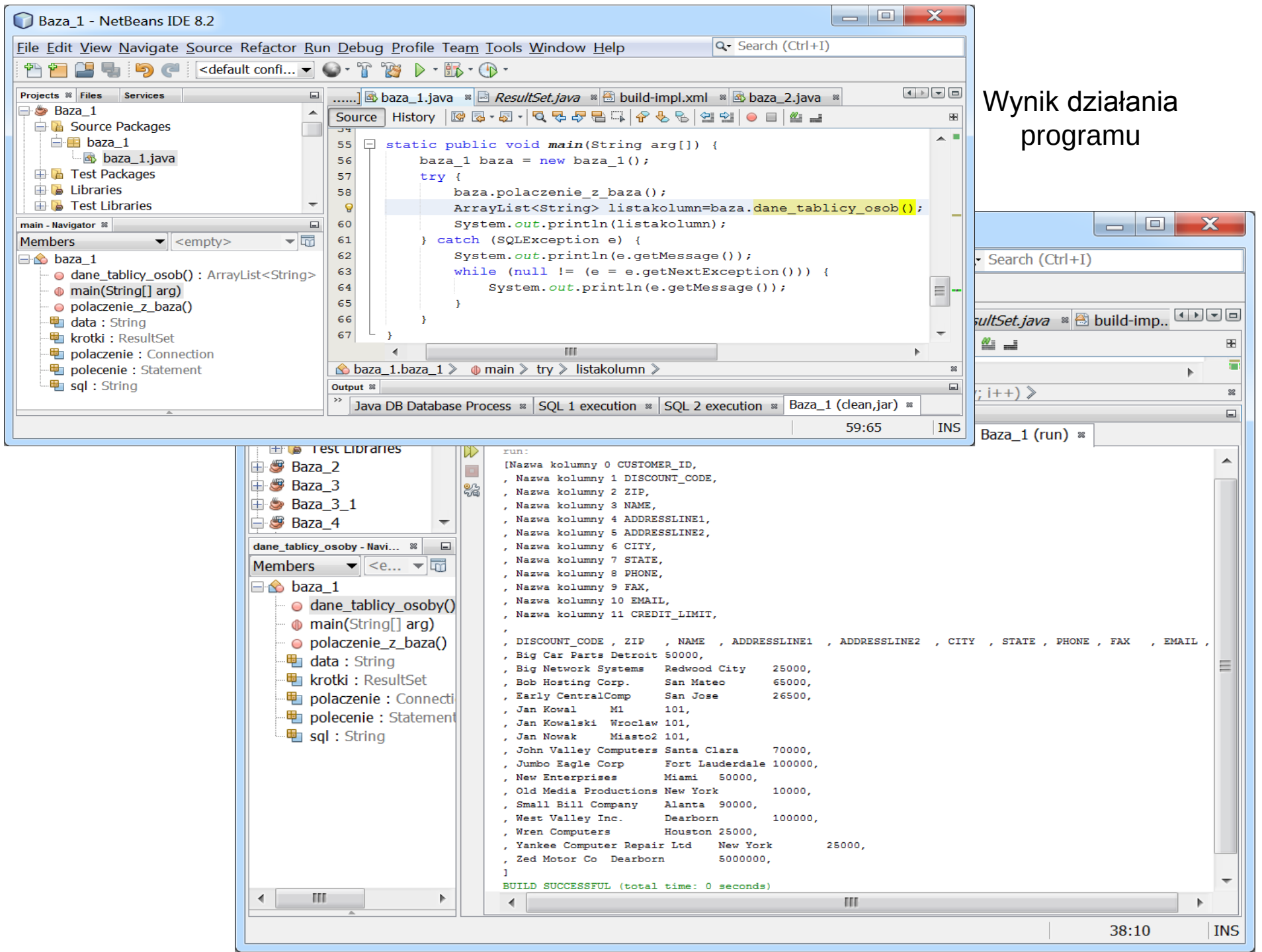
Metoda do obsługi połączenia z bazą danych – 1-y sposób obsługi wyjątków



Metoda do obsługi połączenia z bazą danych – 2-i sposób obsługi wyjątków



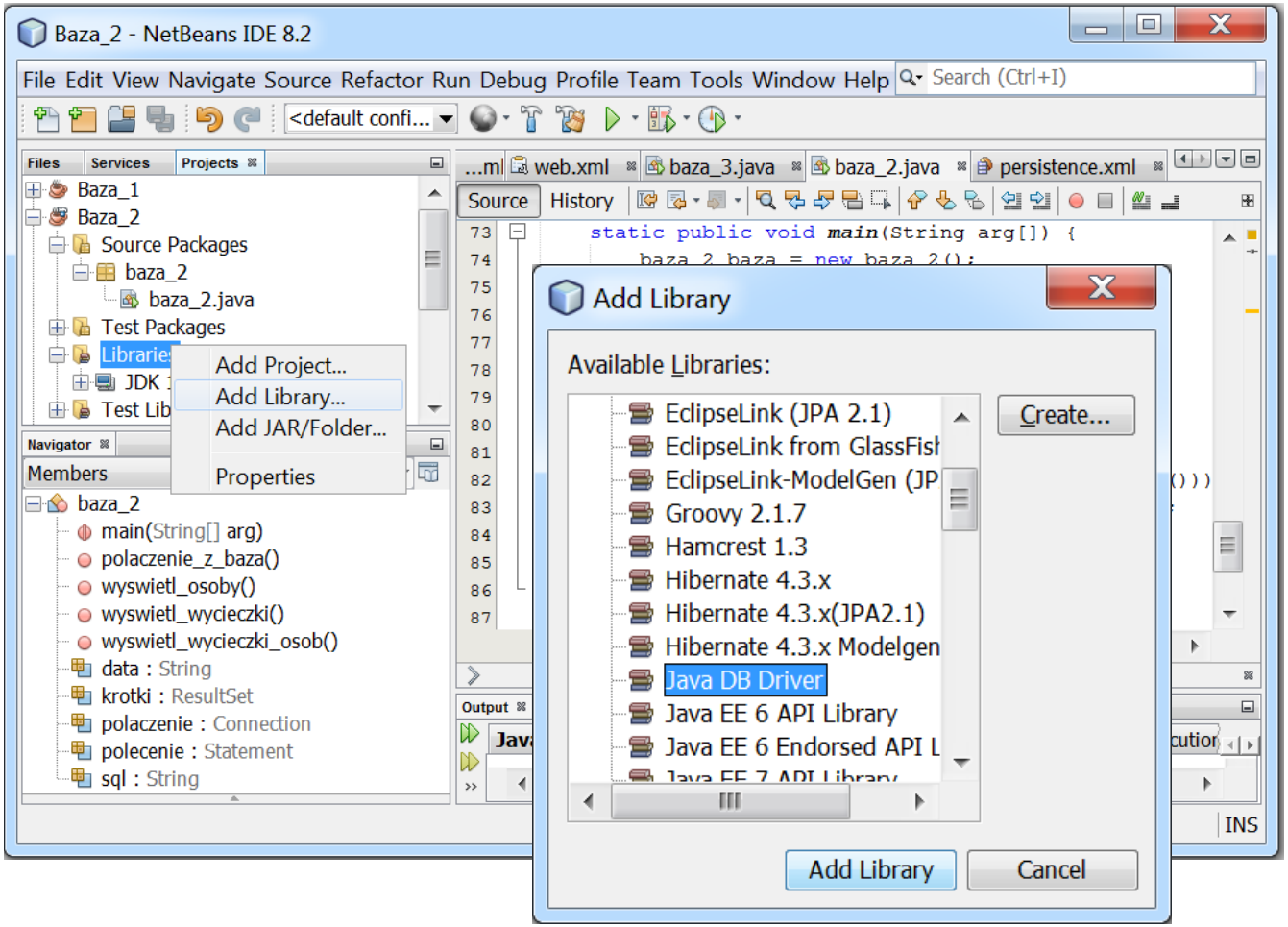

Baza_1 - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
Proj... Files Services
Baza_1
Source Packages
baza_1
baza_1.java
Test Packages
Libraries
Test Libraries
main - Navigator
Members
baza_1
dane_tablicy_osob() : A
main(String[] arg)
polaczenie_z_baza()
data : String
krotki : ResultSet
polaczenie : Connection
polecenie : Statement
sql : String
baza_1.java
ResultSet.java
build-impl.xml
baza_2.java
Source
History
29 public ArrayList<String> dane_tablicy_osob() throws SQLException {
30 polecenie = polaczenie.createStatement();
31 sql = "SELECT * FROM CUSTOMER ORDER BY NAME";
32 krotki = polecenie.executeQuery(sql);
33 ResultSetMetaData metaDane = krotki.getMetaData();
34 int kolumny = metaDane.getColumnCount();
35 ArrayList<String> listakolumn=new ArrayList();
36 for (int i = 0; i < kolumny; i++) {
37 listakolumn.add("Nazwa kolumny " + i + " " + metaDane.getColumnName(i + 1));
38 listakolumn.add("\n");
39 }
40 listakolumn.add("\n");
41 for (int i = 1; i < kolumny - 1; i++) {
42 listakolumn.add(metaDane.getColumnName(i + 1) + "\t");
43 }
44 listakolumn.add("\n");
45 while (krotki.next()) {
46 listakolumn.add(krotki.getString("NAME") + "\t"
47 + krotki.getString("CITY") + "\t"
48 + krotki.getString("CREDIT_LIMIT"));
49 listakolumn.add("\n");
50 }
51 polecenie.close();
52 return listakolumn;
53 }
Wykonanie zapytania na tabeli PERSON
wykonanie działań pomocniczych
baza_1.baza_1 main try listakolumn
Output
Java DB Database Process
SQL 1 execution
SQL 2 execution
Baza_1 (clean,jar)
59:65
INS



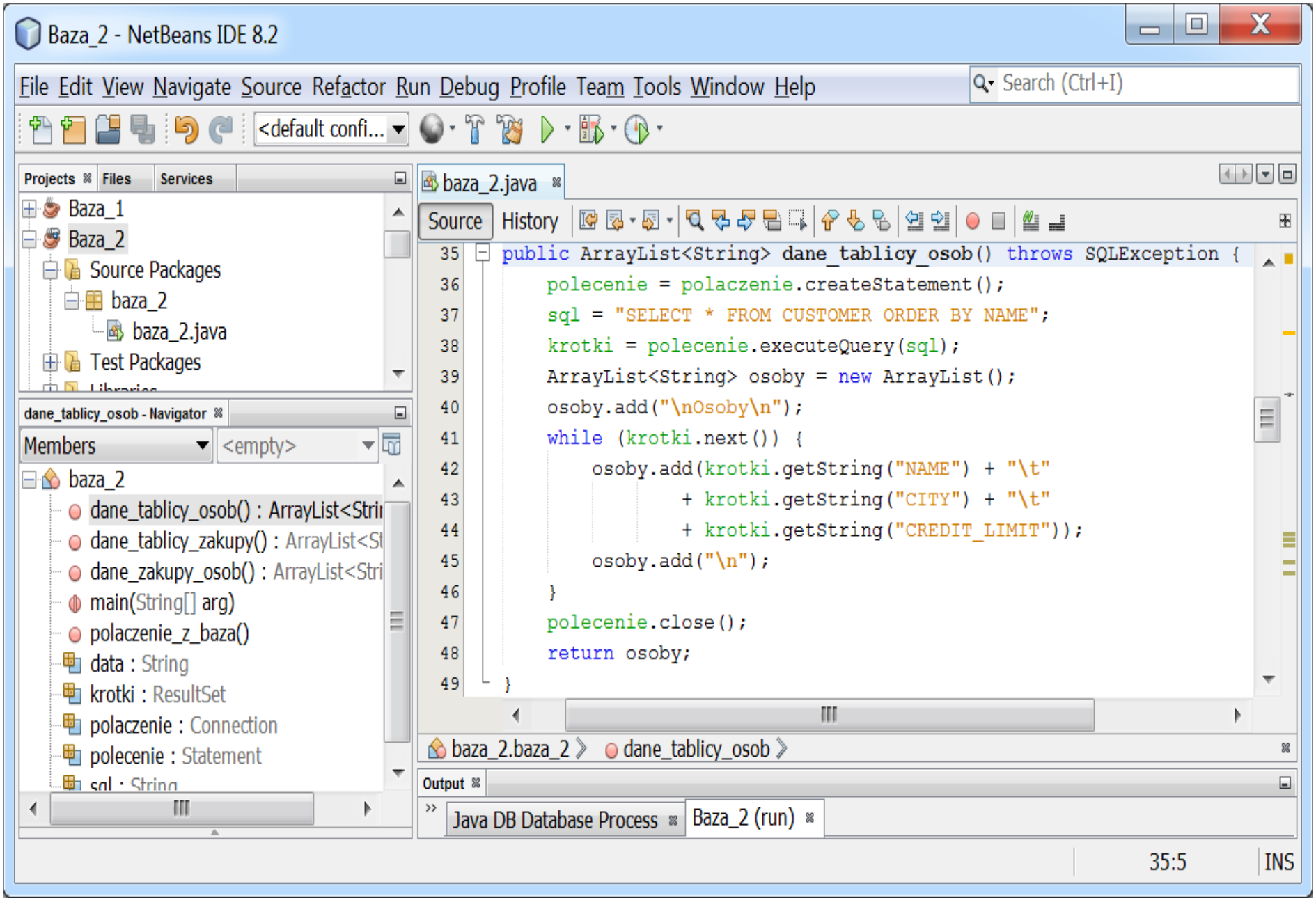
Wynik działania programu

Zadanie 2
cd zadania 1- należy dodać:
wyświetlanie zawartość tabeli PURCHASE_ORDER (metoda dane_tablicy_zakupy()),
wyświetlanie zakupów każdej osoby (metoda dane_zakupy_osob())

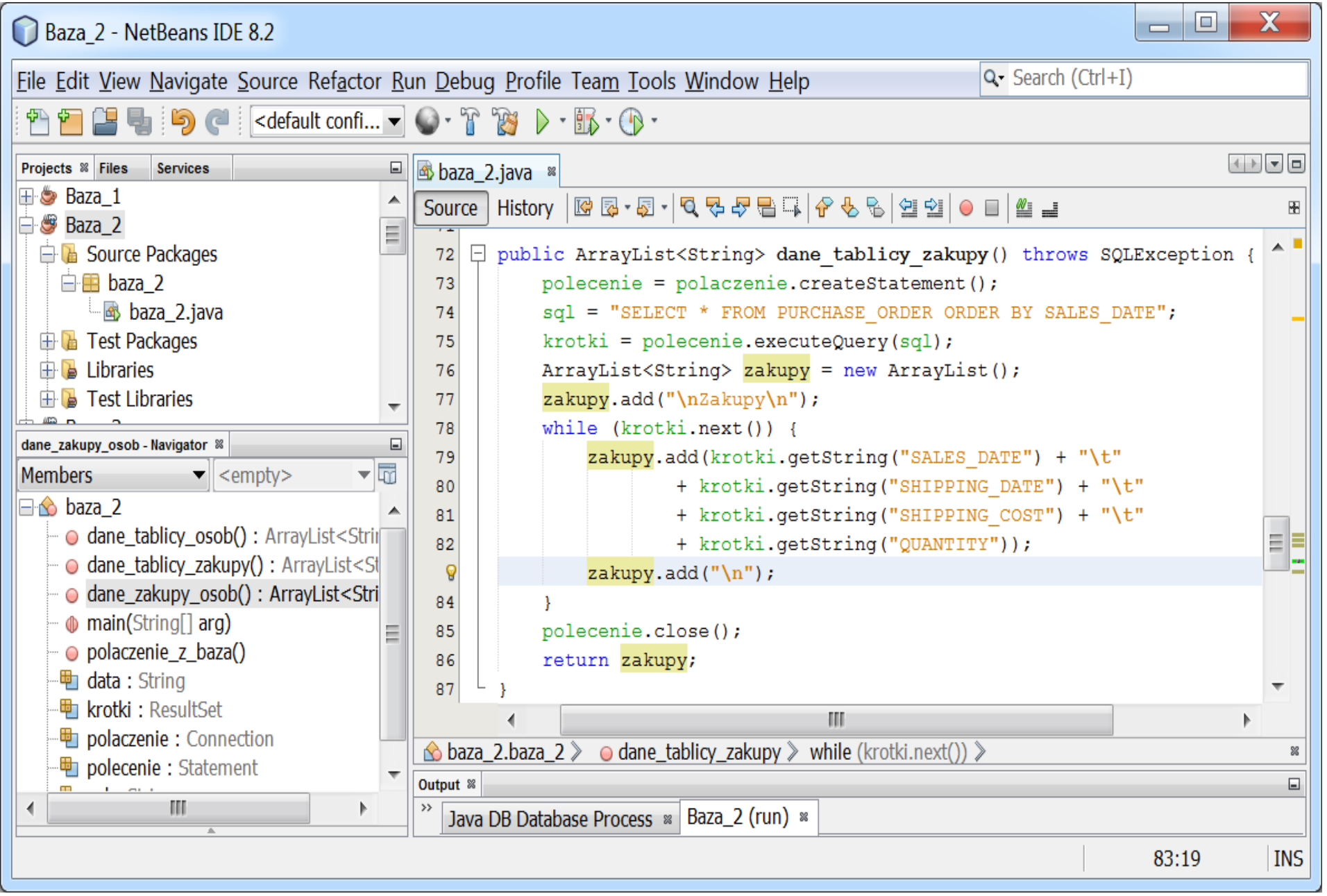
1. Wykonaj kopię programu Baza_1 jako Baza_2 (patrz instrukcja do lab3 - zad.2, pkt. 1)
2. Zmień nazwę pliku baza_1 na baza_2 (patrz instrukcja do lab3 – zad.2, pkt. 2)
3. Dodaj bibliotekę sterownika JavaDBDriver jak poniżej.
4. Wykonaj połączenie z bazą danych Sample wg Zadanie 1, p.2.



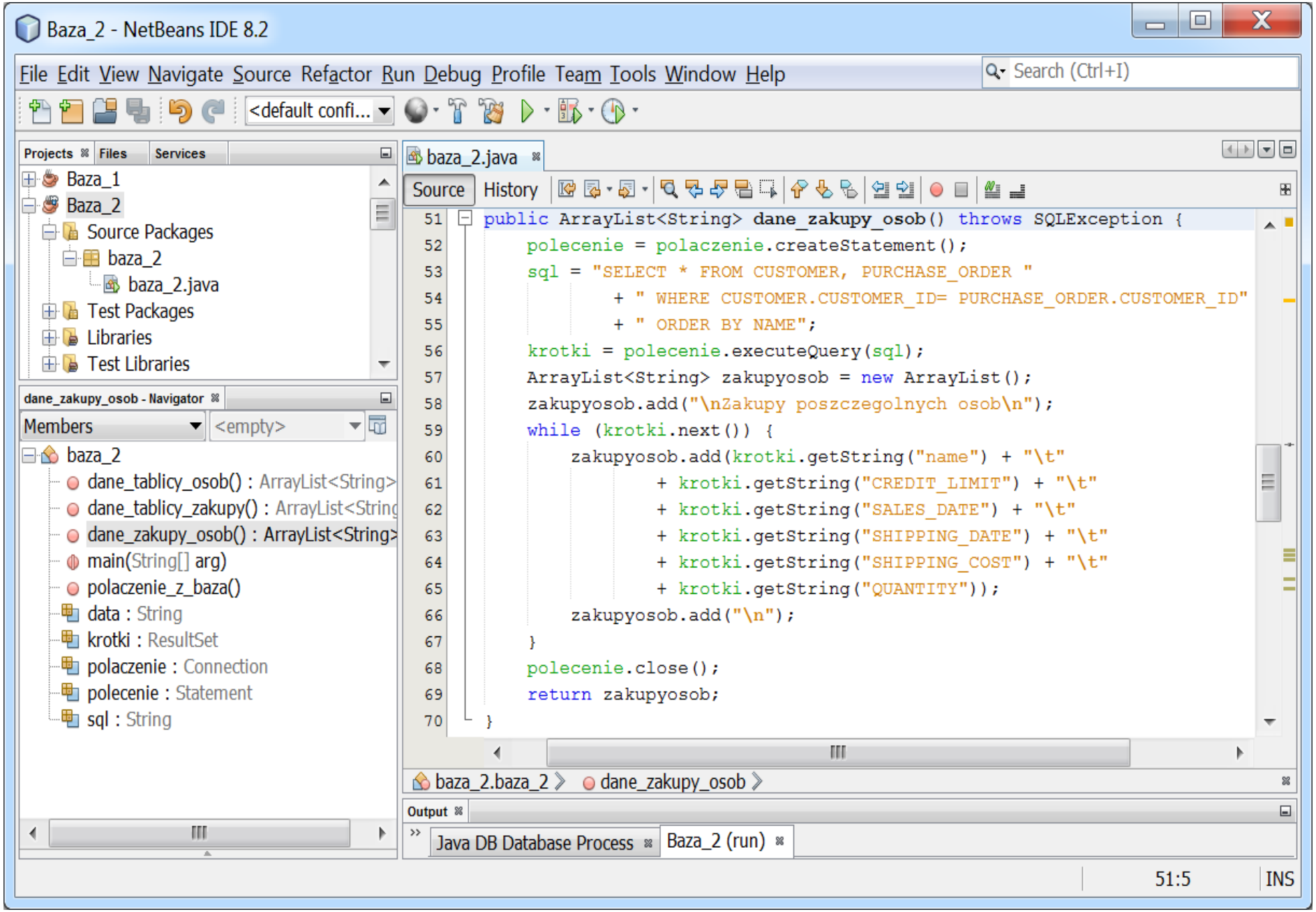
5. Zmodyfikuj zawartość metody **public ArrayList<String> dane_tablicy_osob()** wg slajdu poniżej.



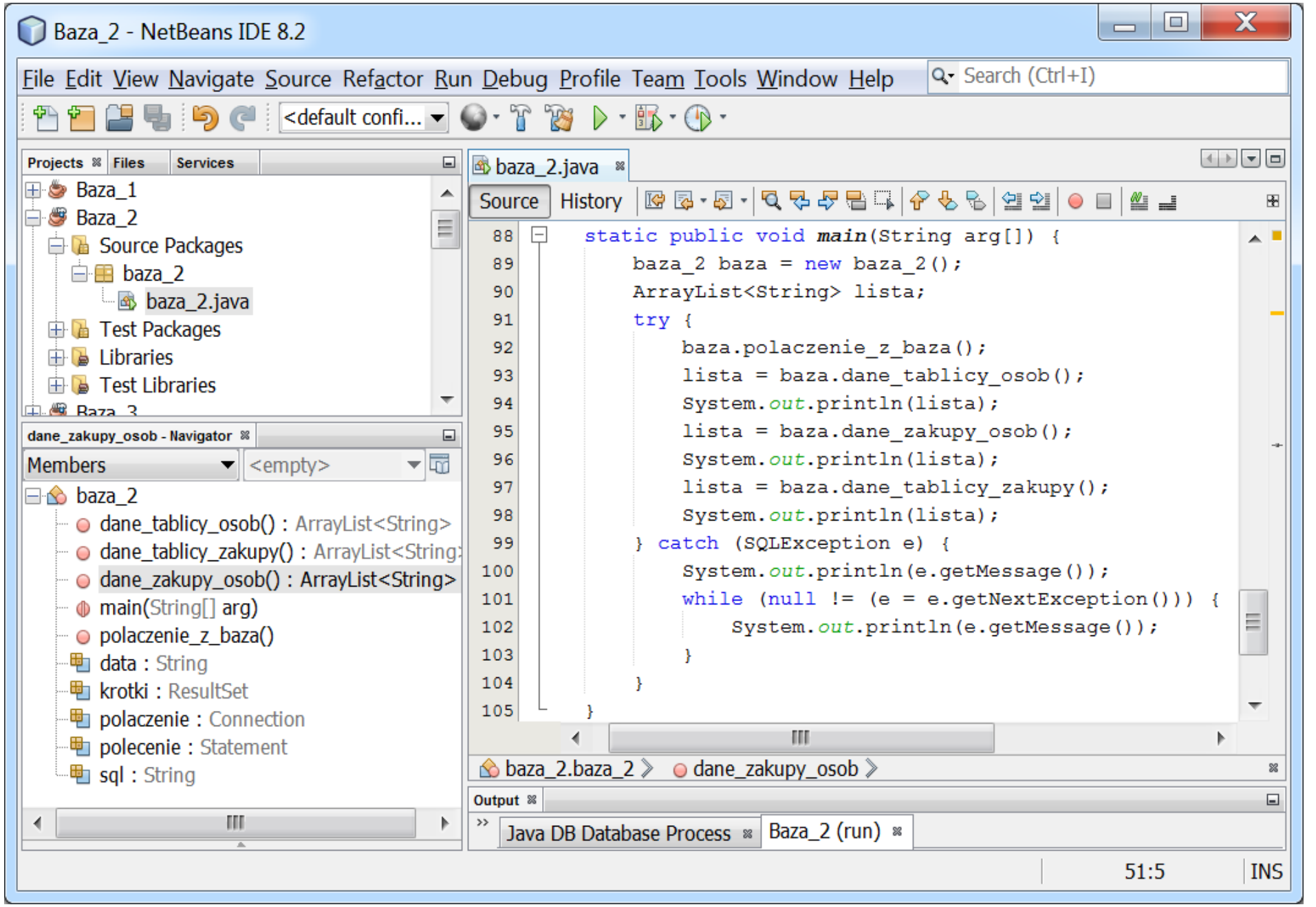
6. Dodaj metodę **public ArrayList<String> dane_tablicy_zakupy()** - wyświetlanie zawartości tabeli **PURCHASE_ORDER**



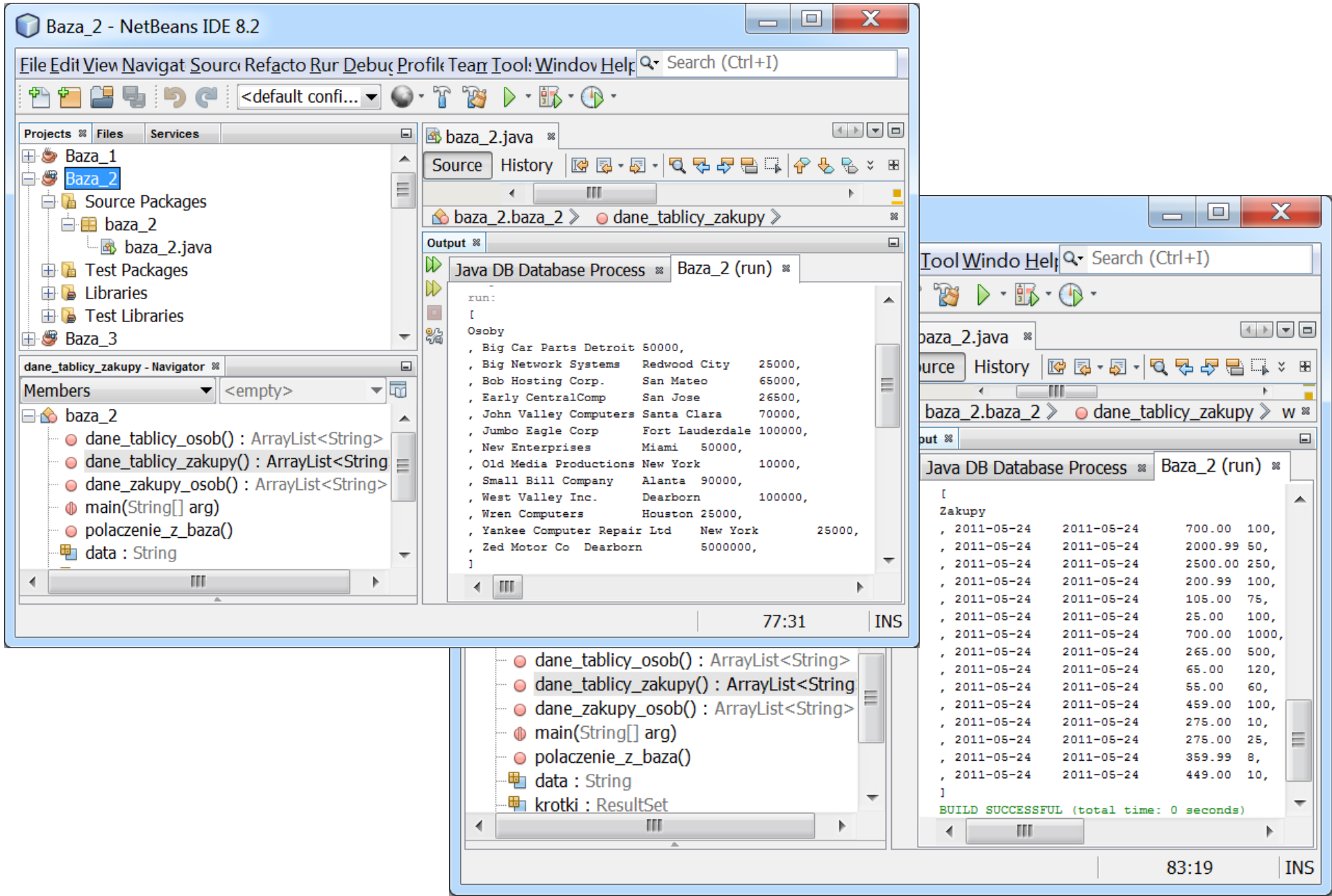
7. Dodaj metodę **public ArrayList<String> dane_zakupy_osob()** -- wyświetlanie zakupów każdej osoby



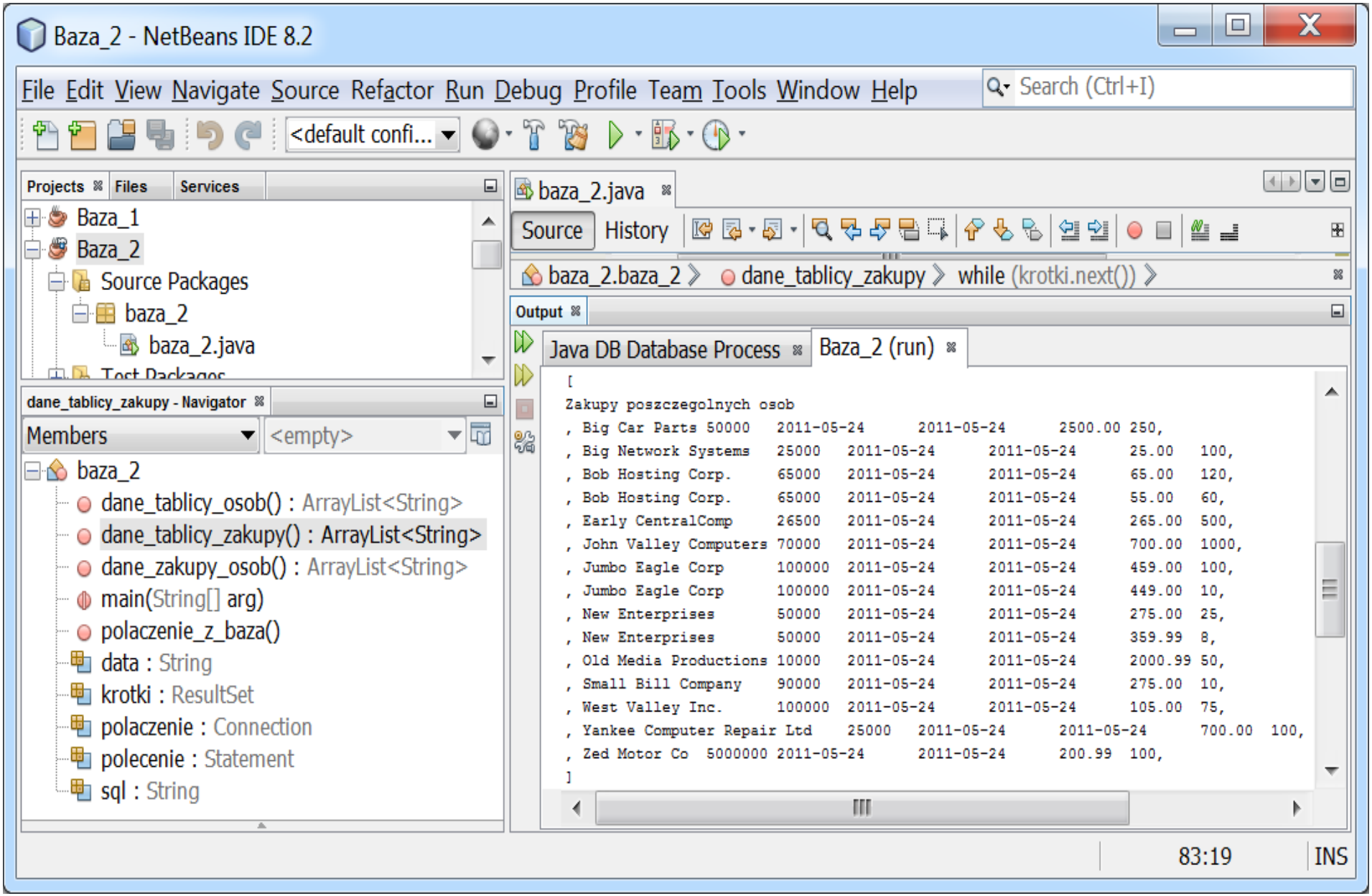
8. Uruchom program



Wynik działania programu (z lewej strony: zawartość tabeli **CUSTOMER**, z prawej strony zawartość tabeli **PURCHASE_ORDER**)



Wynik działania programu (zawartość złączonych tabel CUSTOMER i PURCHASE_ORDER)



Zadanie 3 - cd zadania 2
dodawanie nowych osób do tabeli CUSTOMER oraz dodawanie nowych zakupów wybranej osoby do tabeli PURCHASE_ORDER

1. Wykonaj kopię programu Baza_2 jako Baza_3 (patrz instrukcja do lab3 - zad.2, pkt. 1)
2. Zmień nazwę pliku baza_2 na baza_3 (patrz instrukcja do lab3 – zad.2, pkt. 2)
3. Dodaj sterownik JavaDBDrive wg pkt3 z zad2.
4. Wykonaj połączenie z bazą danych Sample wg Zadanie 1, p.2.
5. Wykonaj pakiet GUI

   5.1. dodaj pomocniczą klasę WeWy do pakietu GUI – do wprowadzania danych z klawiatury. Podczas edycji kodu zaimportuj brakujące biblioteki za pomocą Fix Imports (kliknij prawym klawiszem w oknie Edytora Javy i wybór opcji Fix Imports)

```
public class WeWy {
   static public String weString(String menu) {
      InputStreamReader wejscie = new InputStreamReader(System.in);
      BufferedReader bufor = new BufferedReader(wejscie);
      try {
         System.out.print(menu);
         return bufor.readLine();
      } catch (IOException e) {
         System.err.println("Blad IO String");
         return "";
      }
   }
 static public void wyString(ArrayList<String> lista) {
      lista.forEach((wiersz) -> {
         System.out.print(wiersz);
      });
   }
}
```

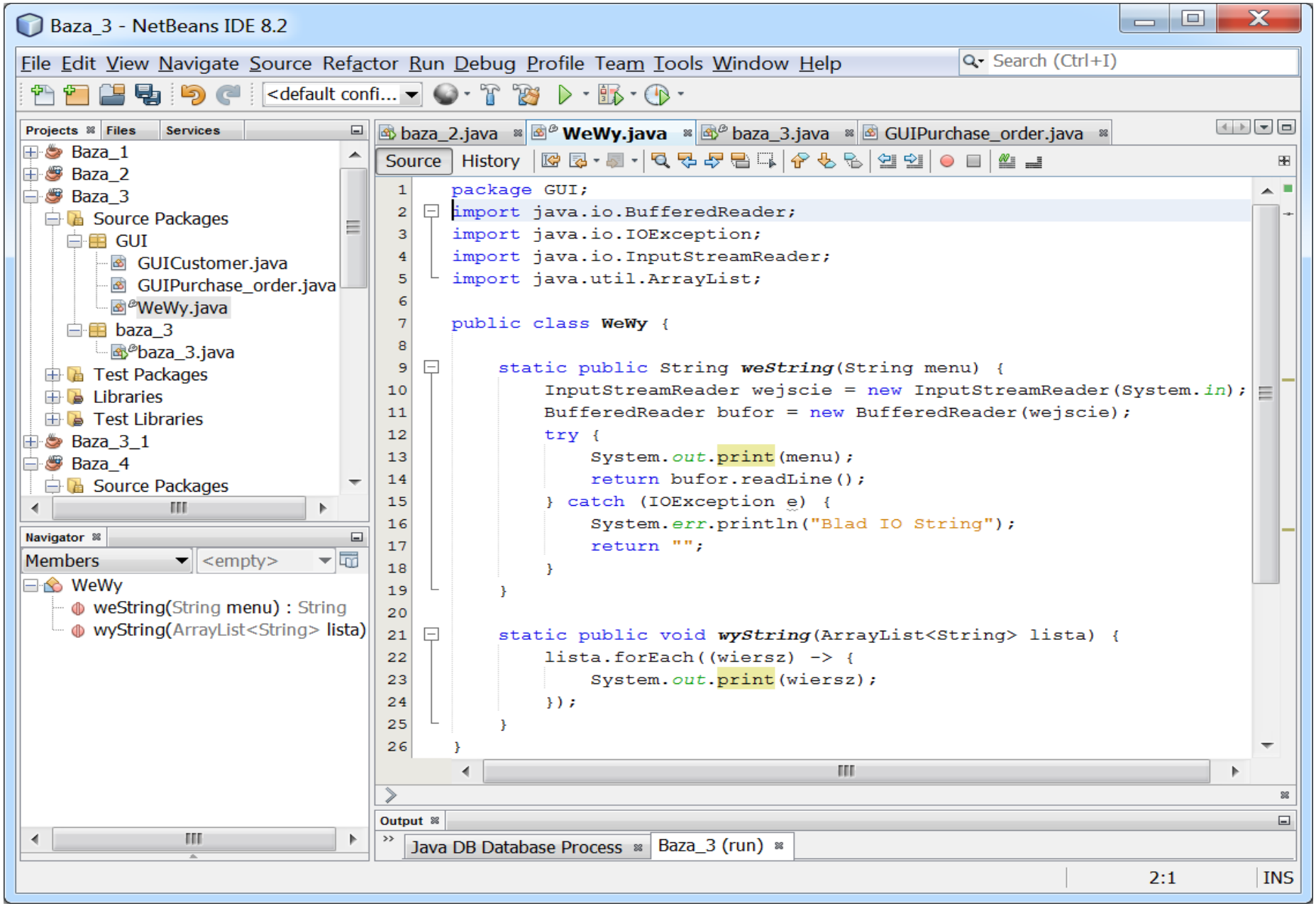
5.2. Dodaj klasę **GUICustomer** w pakiecie GUI w celu wprowadzania danych do tabeli **CUSTOMER**

```
public class GUICustomer {

    public String name, addressline1, adressline2, city, state, phone, fax, email;
    static public int credit_limit = 100;

    public String[] wstaw_dane() {
        name          = WeWy.weString("Podaj nazwisko: ");
        addressline1  = WeWy.weString("Podaj adres1: ");
        adressline2   = WeWy.weString("Podaj adres2: ");
        city          = WeWy.weString("Podaj miasto: ");
        state         = WeWy.weString("Podaj stan: ");
        phone         = WeWy.weString("Podaj numer telefonu: ");
        fax           = WeWy.weString("Podaj numer faksu: ");
        email         = WeWy.weString("Podaj email: ");
        credit_limit++;
        String[] dane = {name, addressline1, adressline2, city, state, phone, fax, email};
        return dane;
    }
}
```

5.2. cd. Dodana klasa **WeWy** do wprowadzania danych typu String (**weString**) i wyświetlania listy łańcuchów (**wyString**)



5.3. Dodaj metodę **public String klucz_glowny(String sql1, String sql2)** do klasy **baza_3** w celu wyznaczenia wartości kolejnego klucza głównego. Metoda jest uniwersalna, ponieważ łańcuch sql2 + "."+sql1 tworzy nazwę pola klucza głównego w dowolnej tabeli (sql1 jest nazwą klucza głównego, a sql2 jest nazwą tabeli). Metoda zwraca wartość kolejnego klucza głównego, jakim można nadać nowej krotce lub null.

```
sql = " SELECT MAX(" + sql2 + "." + sql1 + ")+1 AS MAXID FROM " + sql2;
```

```
public String klucz_glowny(String sql1, String sql2) {
    try {
        sql = " SELECT MAX(" + sql2 + "."+sql1 + ")+1 AS MAXID FROM " + sql2;
        krotki = polecenie.executeQuery(sql);
        krotki.next();
        String pom = krotki.getString("MAXID");
        return pom;
    } catch (SQLException e) {
        System.out.println("Blad w wyznaczaniu klucza" + e);
    }
    return null;
}
```

5.4. Dodaj metodę **void blad() w klasie baza3** do obsługi błędów wstawiania danych do tabeli **CUSTOMER** i **PURCHASE_ORDER**

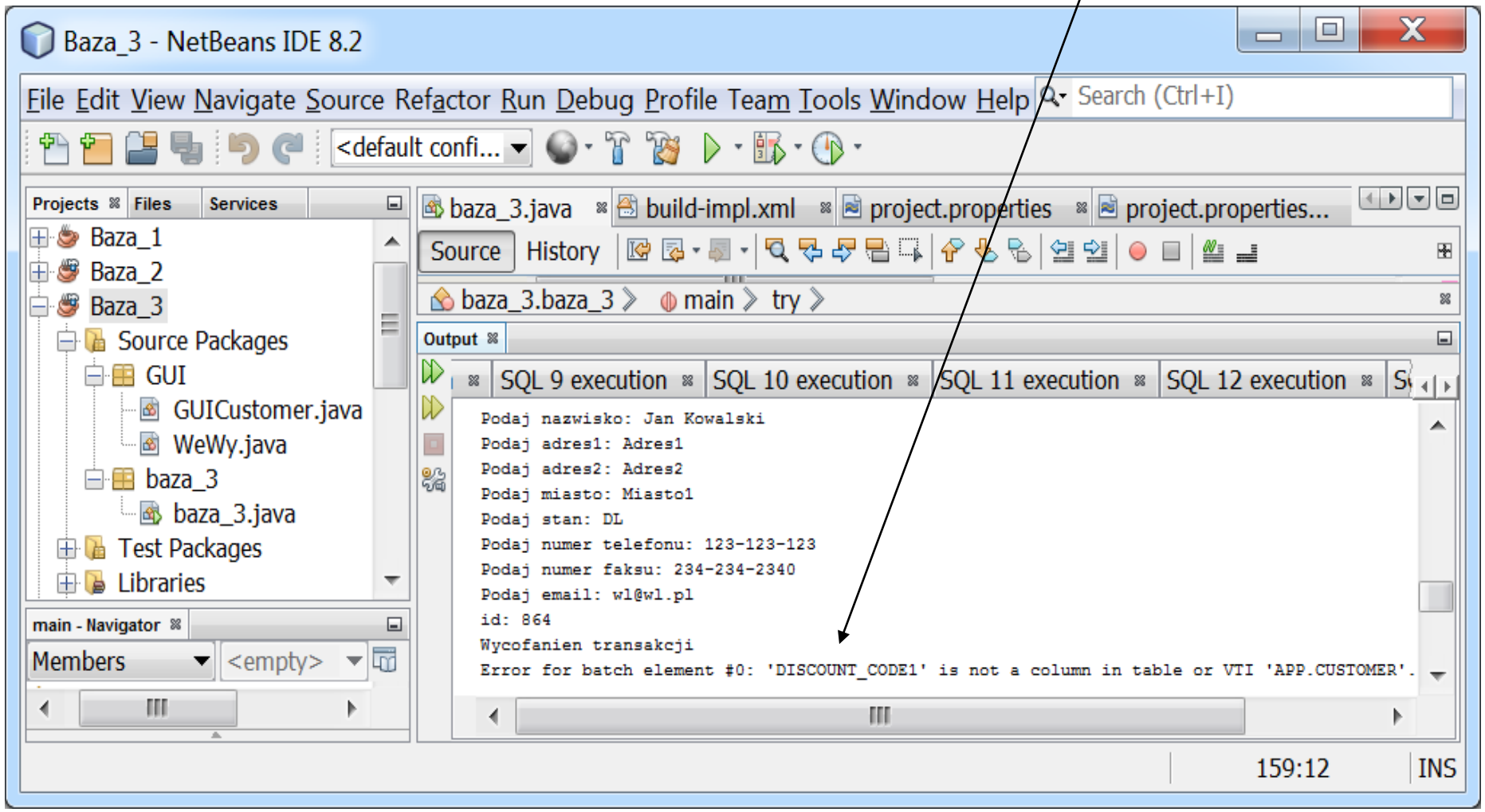
```
void blad(BatchUpdateException e) {
      System.out.println("Wycofanie transakcji ");
      SQLException e1 = e.getNextException();
      while (e1 != null) {
         System.out.println(e1.getMessage());
         e1 = e1.getNextException();
      }
   }
```

5.5. Dodaj metodę **public void wstaw_osobe(String[] dane)** do klasy **baza_3** w celu dodania nowej krotki do tabeli **CUSTOMER**

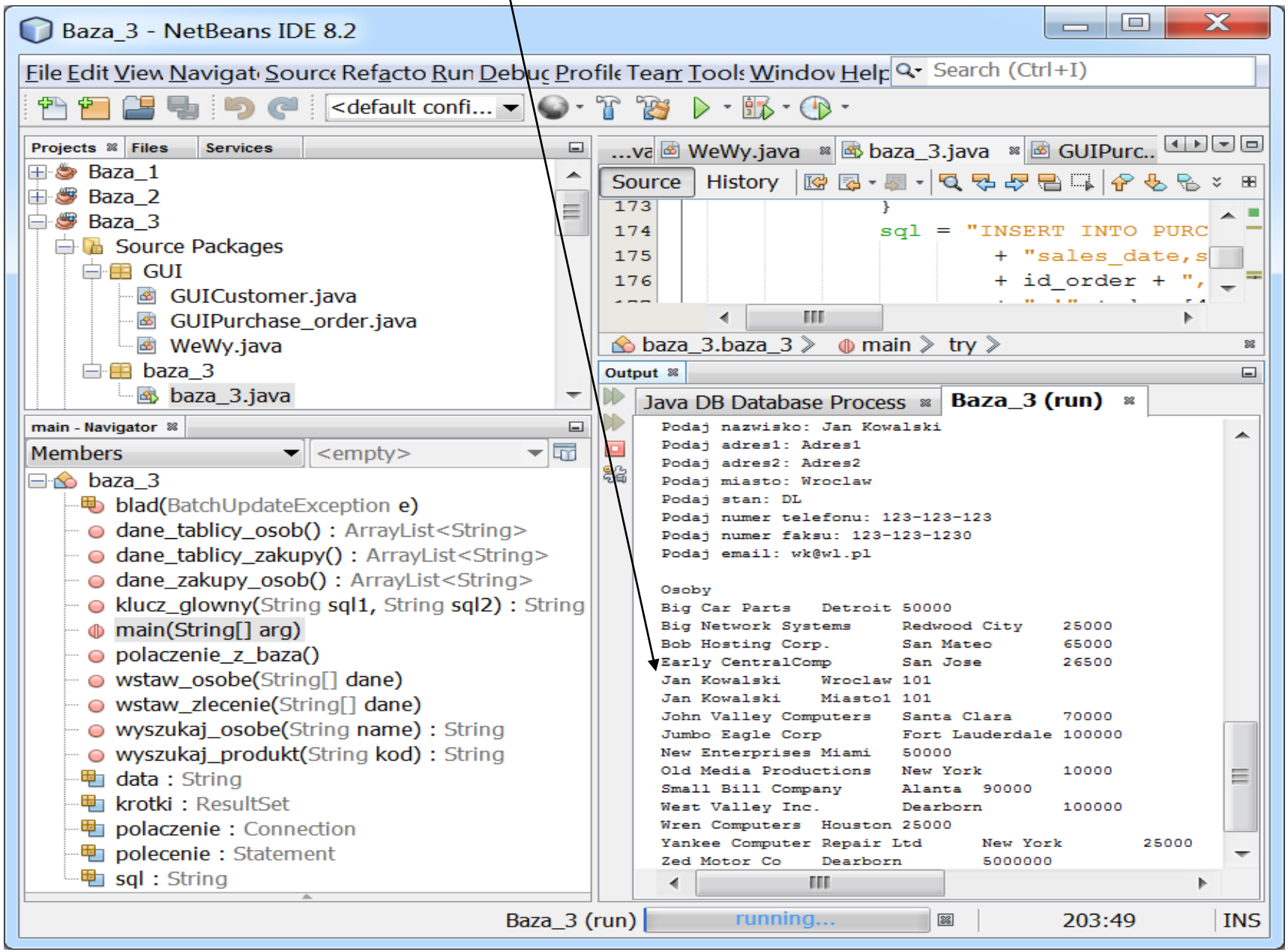
```
public void wstaw_osobe(String[] dane) throws SQLException {
    String id_osoby;
    polaczenie.setAutoCommit(false);
    try {
       polecenie = polaczenie.createStatement();
       if ((id_osoby = klucz_glowny("CUSTOMER_ID", "CUSTOMER")) == null)
          return;
      sql = "INSERT INTO CUSTOMER (customer_id, discount_code1, zip, name, "
            +"addressline1,addressline2, city, state, phone,fax, email, credit_limit)"
            + " VALUES (" + id_osoby + ",'" + "N" + "','" + "48128" + "','" + dane[0] + "','"
            + dane[1] + "','"+ dane[2] + "','" + dane[3] + "','" + dane[4] + "','"
            + dane[5] + "','" + dane[6] + "','" + dane[7] + "','"+ GUICustomer.credit_limit +"')";
       polecenie.addBatch(sql);
       polecenie.executeBatch();
       polaczenie.commit();
    } catch (BatchUpdateException e) {
      blad(e);
      polaczenie.rollback();
    }
}
```

```
sql = "INSERT INTO CUSTOMER (CUSTOMER_ID, discount_code, zip, name, addressline1,"
    + "addressline2, city, state, phone,fax, email, credit_limit)"
    + " VALUES (" + id_osoby + ",'" + "N" + "','" + "48128" + "','" + dane[0] + "','"
    + dane[1] + "','" + dane[2] + "','" + dane[3] + "','" + dane[4] + "','"
    + dane[5] + "','" + dane[6] + "','" + dane[7] + "',"
    + GUICustomer.credit_limit + ")";
```

5.5. cd. Działanie metody **public void wstaw_osobe(String[] dane)**, gdy wystapił błąd nazwy jednej z kolumn tabeli **CUSTOMER**



5.5. cd. Działanie metody **public void wstaw_osobe(String[] dane)**, gdy usunięto błąd nazwy jednej z kolumn tabeli **CUSTOMER**



6. Dodawanie nowej krotki do tablicy **PURCHASE_ORDER**
6.1. Dodaj klasę **GUIPurchase_order** do pakietu GUI do wprowadzania danych do tabeli **PURCHASE_ORDER**. Metoda **public String wstaw_date(String komunikat)** otrzymuje liczbę dni metodą **weString()** i dodaje je do daty bieżącej metodą **setTime**. Następnie wykorzystuje metodą **SimpleDateFormat** do nadania dacie formatu wymaganego w tabeli **PURCHASE_ORDER** w bazie danych.

```
package GUI;

import java.util.Date;
import java.text.SimpleDateFormat;

public class GUIPurchase_order {

    private String quantity, shipping_cost, freight_company, name, kod_produktu,
                   sales_date, shipping_date;

    public String wstaw_date(String komunikat) {
        Date pom1 = new Date();
        String pom2 = WeWy.weString(komunikat);       // liczbe dni od dnia bieżącego
        pom1.setTime(pom1.getTime() + Long.parseLong(pom2) * 24 * 60 * 60 * 1000);
        SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd");
        String data = format.format(pom1.getTime());
        return data;
    }
}
```

6. Dodawanie nowej krotki do tablicy **PURCHASE_ORDER cd**
6.2. Dodaj metodę **String[] wstaw_dane_zlecenia()** do wprowadzania danych do tabeli **PURCHASE_ORDER**. Metoda umożliwia pobranie wszystkich danych do utworzenia nowej krotki w tabeli **PURCHASE_ORDER**

```
    public String[] wstaw_dane_zlecenia() {
        sales_date          = wstaw_date("Podaj date sprzedazy (za ile dni): ");
        shipping_date       = wstaw_date("Podaj date wysylki (za ile dni): ");
        quantity            = WeWy.weString("Podaj koszt zakupu: ");
        shipping_cost       = WeWy.weString("Podaj koszt wysylki: ");
        freight_company     = WeWy.weString("Podaj firme przewozowa: ");
        name                = WeWy.weString("Podaj nazwisko osoby: ");
        kod_produktu        = WeWy.weString("Podaj kod produktu: ");
        String [] dane = {name, kod_produktu, quantity, shipping_cost,
                          sales_date, shipping_date, freight_company};
        return dane;
    }
}
```

6.3. Należy dodać metodę **public String wyszukaj_osobe(String name)** w celu wyszukania **klucza głównego krotki z tabeli CUSTOMER** oraz metodę **public String wyszukaj_produkt(String kod)** w celu wyszukania **klucza głównego krotki z tabeli PRODUCT** - do klasy baza_3

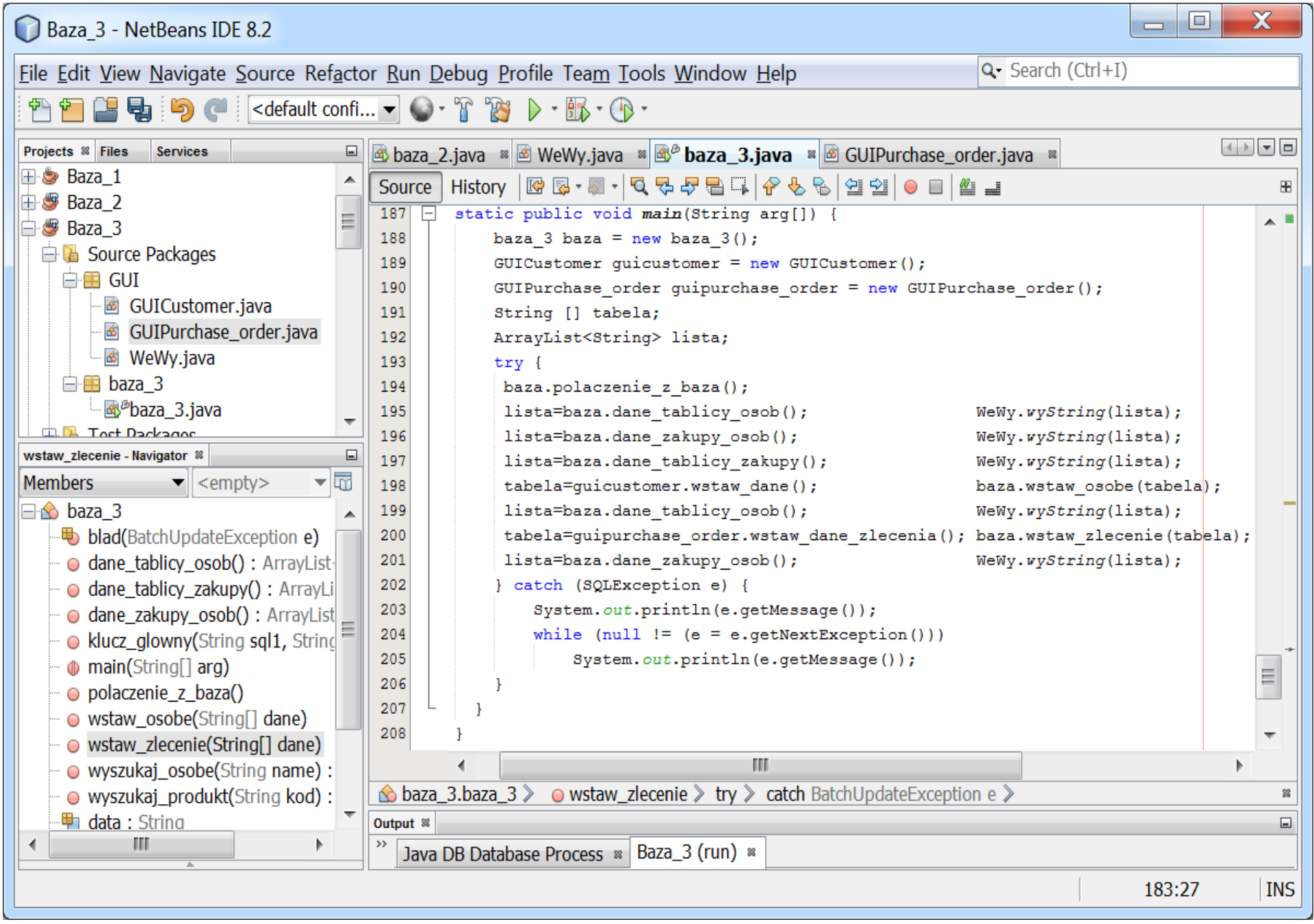
```
                    sql = "SELECT * FROM CUSTOMER WHERE CUSTOMER.NAME = '" + name + "'";
public String wyszukaj_osobe(String name) {
    try {
        sql = "SELECT * FROM CUSTOMER WHERE CUSTOMER.NAME = '" + name + "'";
        krotki = polecenie.executeQuery(sql);
        if (krotki.next()) {
            return krotki.getString("CUSTOMER_ID");
        }
    } catch (SQLException e) {
        System.out.println("Blad w wyszukaniu osoby" + e);
    }
    return null;
}
                    sql = "SELECT * FROM Product WHERE PRODUCT_CODE = '" + kod + "'";
public String wyszukaj_produkt(String kod) {
    try {
        sql = "SELECT * FROM Product WHERE PRODUCT_CODE = '" + kod + "'";
        krotki = polecenie.executeQuery(sql);
        if (krotki.next()) {
            return krotki.getString("Product_ID");
        }
    } catch (SQLException e) {
        System.out.println("Blad w wyszukaniu produktu" + e);
    }
    return null;
}
```

6.4. Należy dodać metodę **public void wstaw_zlecenie(String[] dane)** do klasy **baza_3** w celu wstawienia nowej krotki do tabeli **PURCHASE_ORDER**

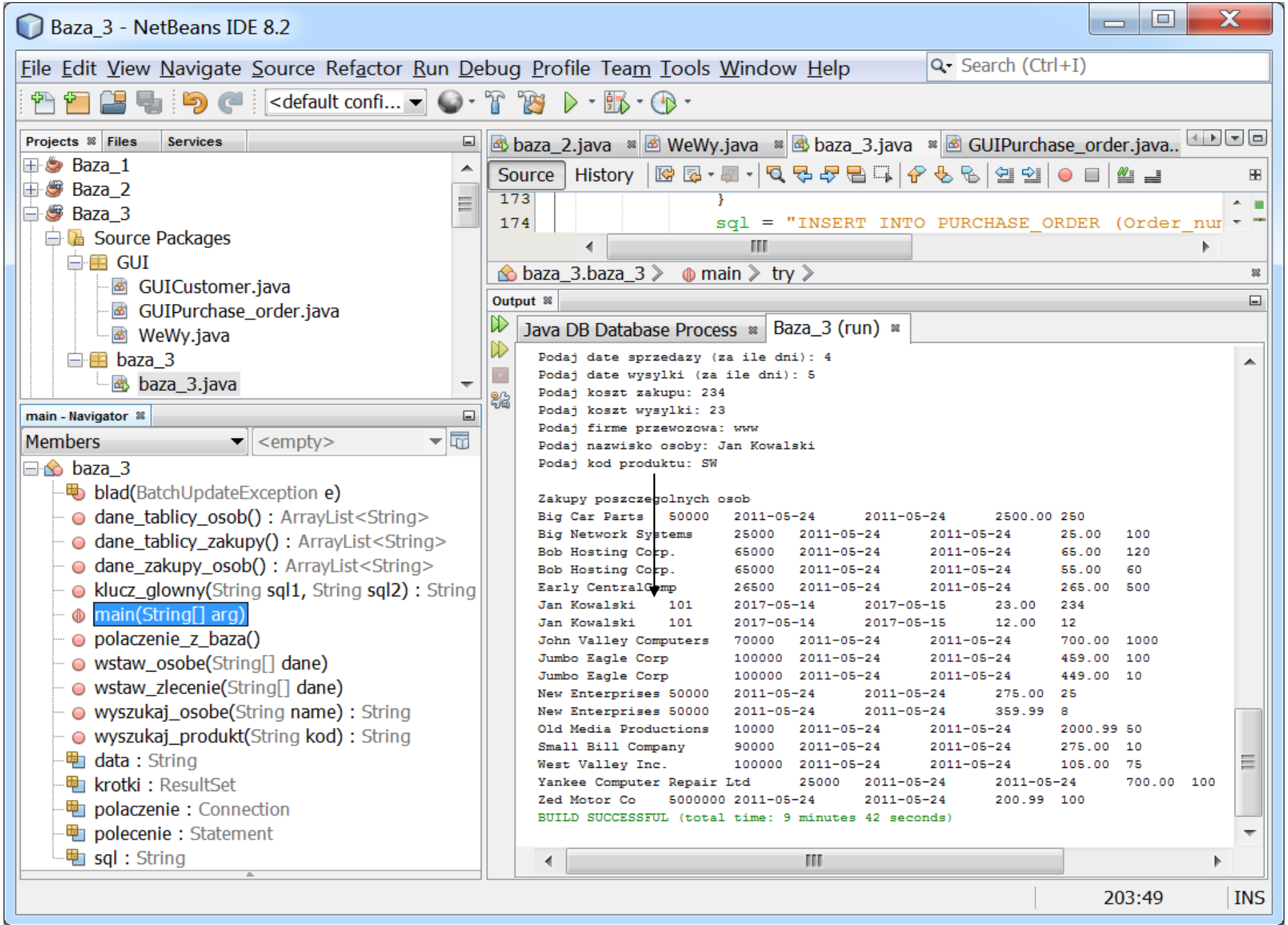
```
public void wstaw_zlecenie(String[] dane) throws SQLException {
    String id_osoby, id_produkt, id_order;
    polaczenie.setAutoCommit(false);
    try {
        polecenie = polaczenie.createStatement();
        if ((id_osoby = wyszukaj_osobe(dane[0])) == null) {
            System.out.println("Brak osoby");
            return;    }
        if ((id_produkt = wyszukaj_produkt(dane[1])) == null) {
              System.out.println("Brak produktu");
            return;   }
        if ((id_order = klucz_glowny("Order_num", "Purchase_order")) == null) {
              System.out.println("null");
            return;  }
       sql = "INSERT INTO PURCHASE_ORDER (Order_num,customer_id,product_id,Quantity,shipping_cost,"
              + "sales_date,shipping_date,freight_company) VALUES ("
              + id_order + "," + id_osoby + "," + id_produkt + "," + dane[2] + "," + dane[3]
              + ",'" + dane[4] + "','" + dane[5] + "','" + dane[6] + "')";
        polecenie.executeUpdate(sql);
        polecenie.executeBatch();
        polaczenie.commit();
    } catch (BatchUpdateException e) {
        blad(e);
        polaczenie.rollback();
    }
}
```

```
sql = "INSERT INTO PURCHASE_ORDER (Order_num,customer_id,product_id,Quantity,shipping_cost,"
      + "sales_date,shipping_date,freight_company) VALUES ("
      + id_order + "," + id_osoby + "," + id_produkt + "," + dane[2] + "," + dane[3]
      + ",'" + dane[4] + "','" + dane[5] + "','" + dane[6] + "')";
```

6.5. Metoda main



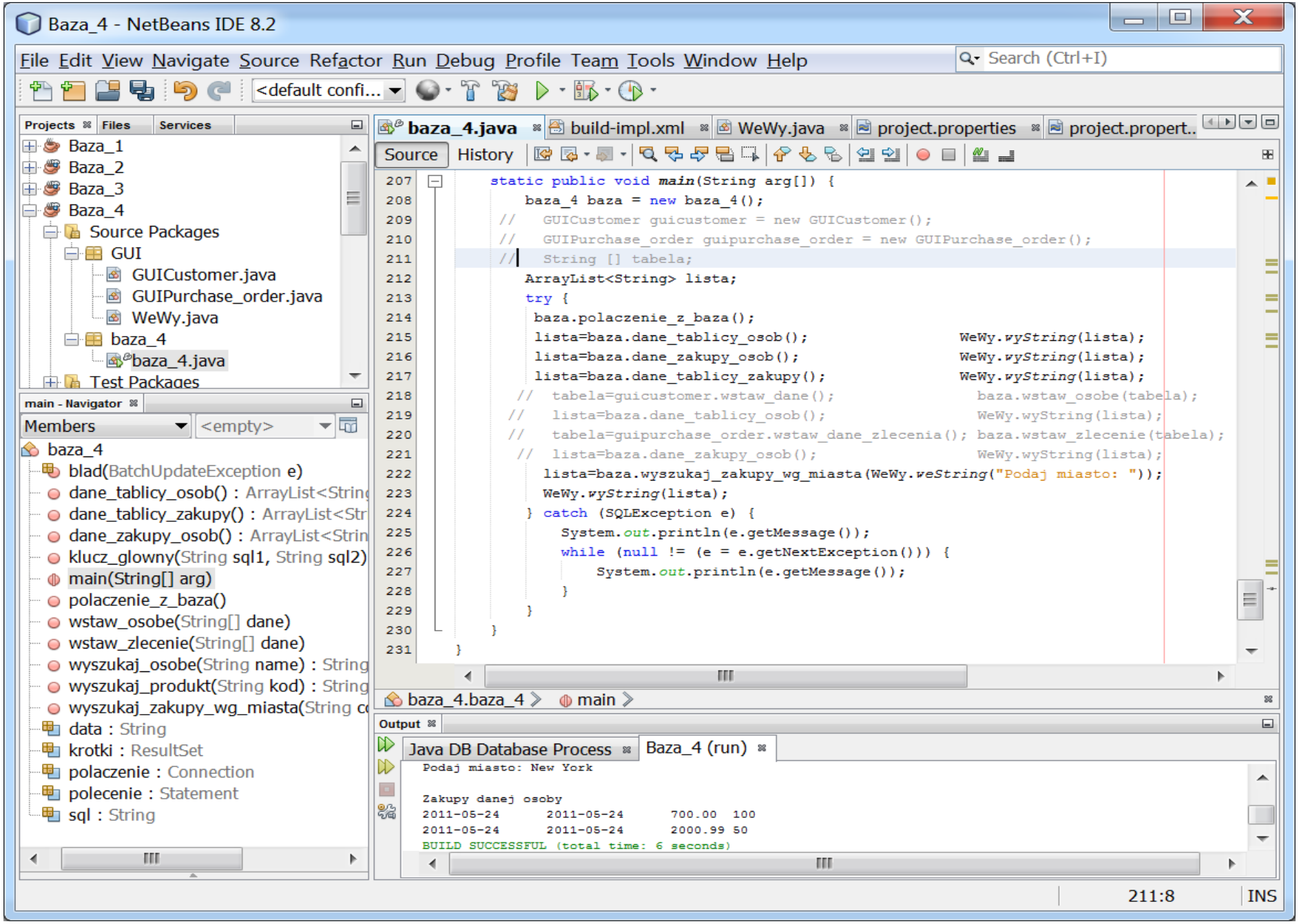
6.6. Wynik działania nowej metody **public void wstaw_zlecenie(String[] dane)**



Zadanie 4
cd zadania 3
dodawanie przeszukiwania tabel CUSTOMER i PURCHASE_ORDER
dodawanie nowych krotek do wybranej tabeli z: PRODUCT, MANUFACTURER, lub MICRO_MARKET
dodanie dowolnej operacji na tabelach bazy danych Sample

1. Wykonaj kopię programu Baza_3 jako Baza_4 (patrz instrukcja do lab3 - zad.2, pkt. 1)
2. Zmień nazwę pliku baza_3 na baza_4 (patrz instrukcja do lab3 – zad.2, pkt. 2)
3. Dodaj sterownik JavaDBDrive wg pkt3 z zad2.
4. Wykonaj połączenie z bazą danych Sample wg Zadanie 1, p.2.
5. Dodaj metodę **public void wyszukaj_wg_miasta() throws SQLException**, która wykonuje zapytanie wyszukiwania osób wg miasta w tablicy **CUSTOMER** posiadających zakup w tablicy **PURCHASE_ORDER**
6. Dodaj metodę, która wstawia nową krotkę do tabeli **PRODUCT lub MANUFACTURER lub MICRO_MARKET**
7. Wykonaj dowolną operację w bazie danych Sample.

Ilustracja do p.5 - Dodaj metodę **public void wyszukaj_wg_miasta() throws SQLException**, która wykonuje zapytanie wyszukiwania osób wg miasta w tablicy **CUSTOMER** posiadających zakup w tablicy **PURCHASE_ORDER**

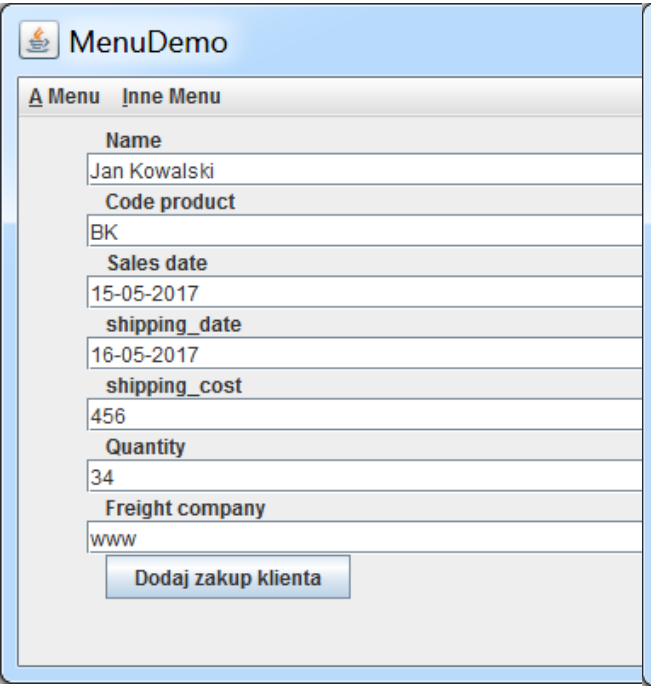


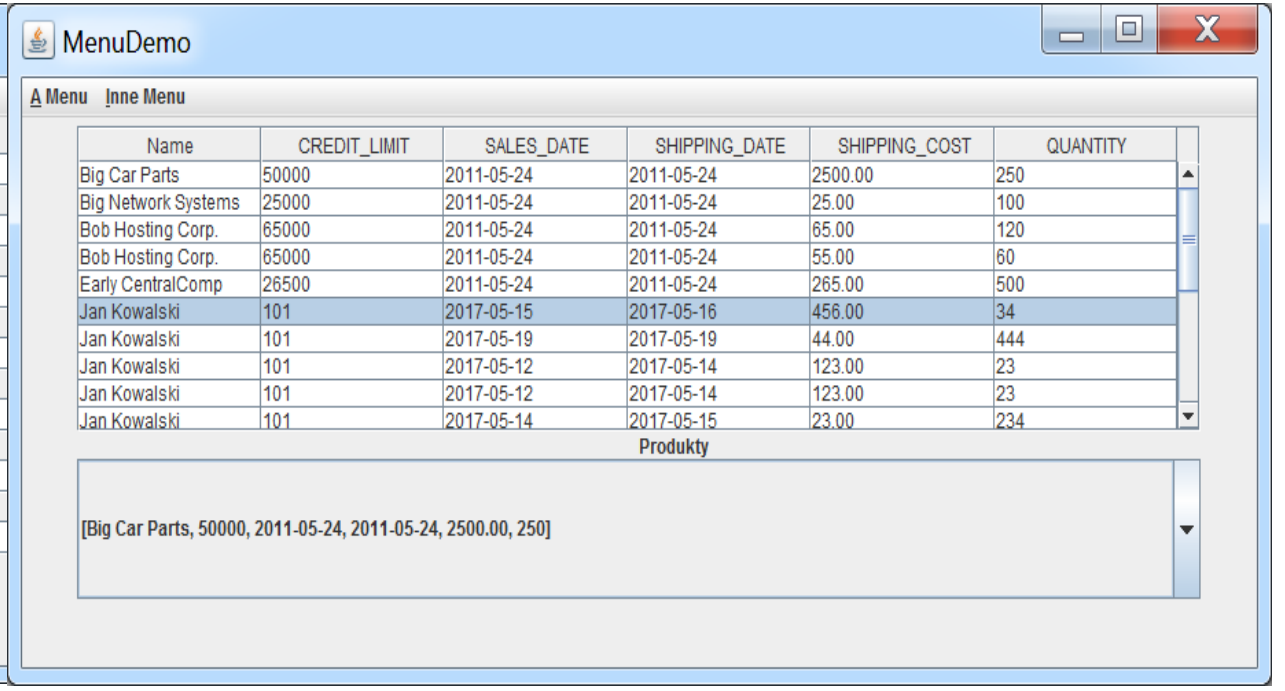
Zadanie 5

1) Wykonaj interfejs graficzny użytkownika, **podobnie jak dalej pokazano w p. 2-6**, do wybranych operacji na bazie danych wykonanych w zadaniach1-4 np **public void wstaw_osobe(String[] dane)** i dodanej **items_** wyswietlajacej dane tablicy **CUSTOMER** (podobnie jak metoda **public ArrayList<String> dane_tablicy_osob()**).

2) Przykłady programów z GUI do wprowadzania danych do tabeli PURCHASE_ORDER oraz wyświetlania złączonych tabel CUSTOMER i PURCHASE_ORDER (po wykonaniu połączenia z bazą danych Sample wg Zadanie 1, p.2) zamieszczono w projekcie Sklep_Gui_Baza3 i powiązanego z projektem Baza_LB, dołączonych do laboratorium (wg programu z lab6, Języki i metody programowania-Java) - http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/java/Zad5.rar

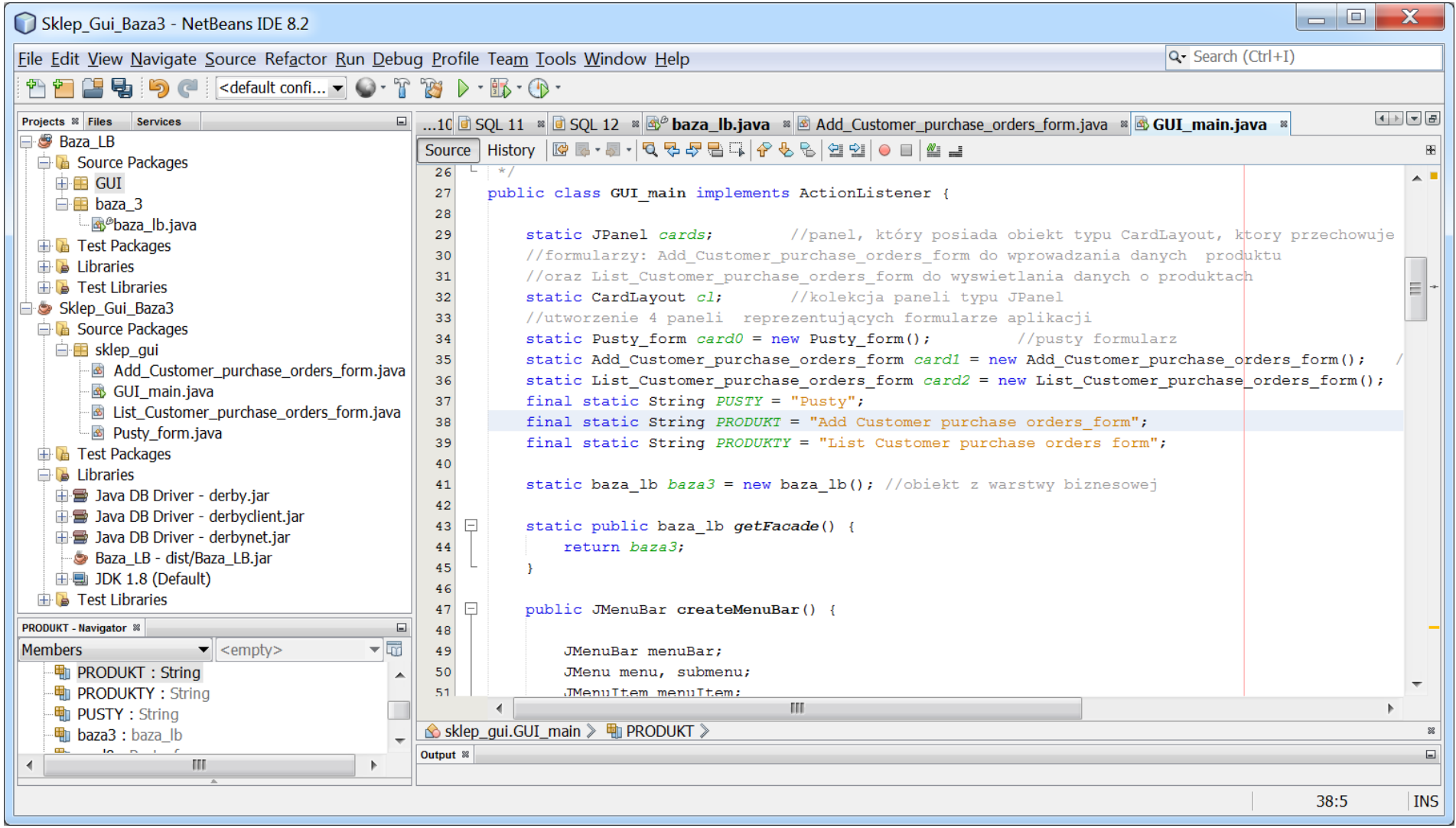
4. **Aplikacje współpracujące z bazami danych - sterowniki JDBC (2017).**
   1. **Instrukcja 4.**
   2. **Program pomocniczy Baza_1 do p. 1 instrukcji 4.**
   3. **Program pomocniczy do p. 5 instrukcji 4.**





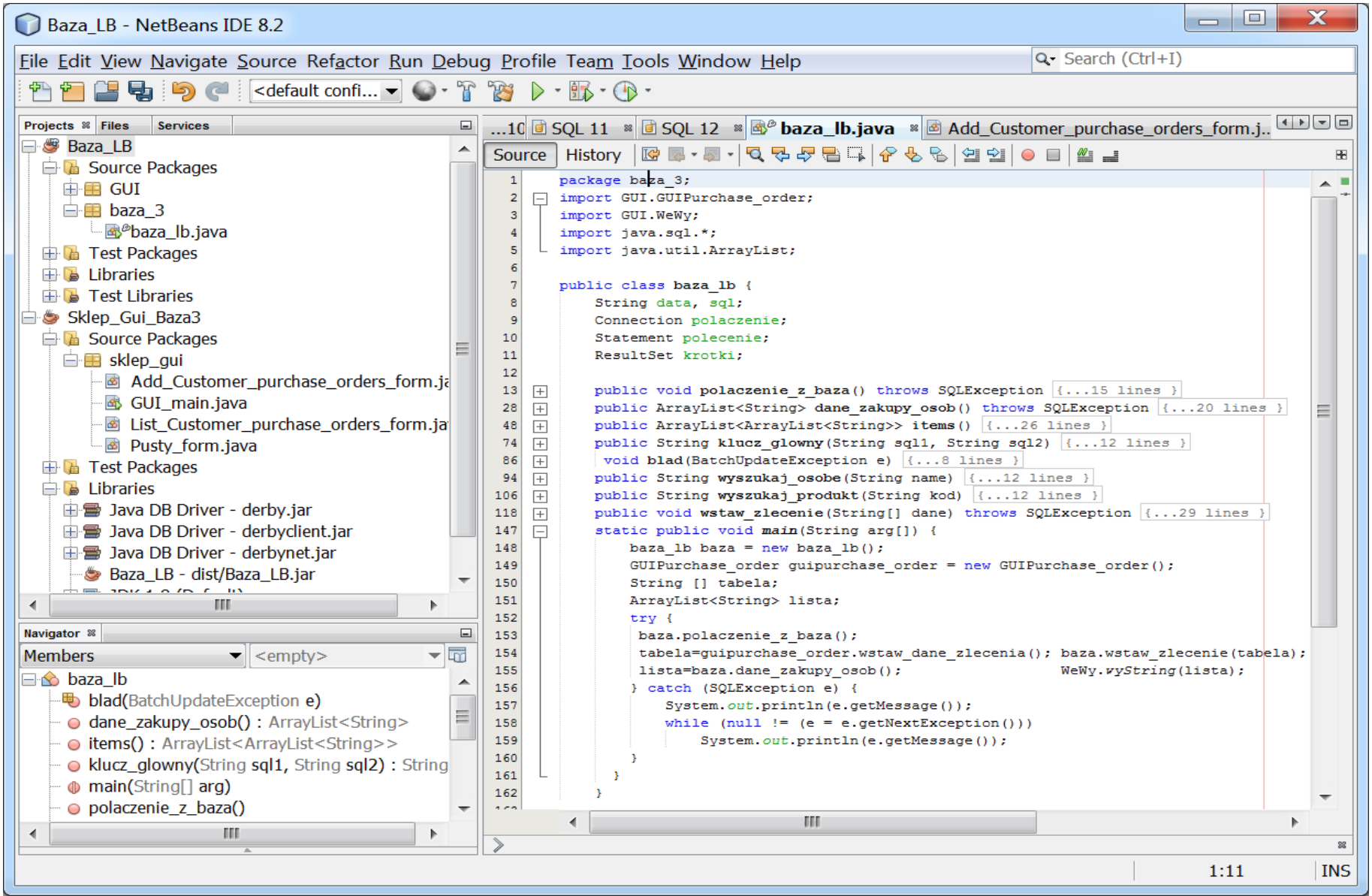
Zadanie 5 cd

3) Program **Sklep_Gui_Baza3** – warstwa klienta programu i projekt **Baza_LB** – warstwa biznesowa programu. Projekt **Baza_LB** jest okrojoną wersja programu **Baza_3** z z zadanie 3.



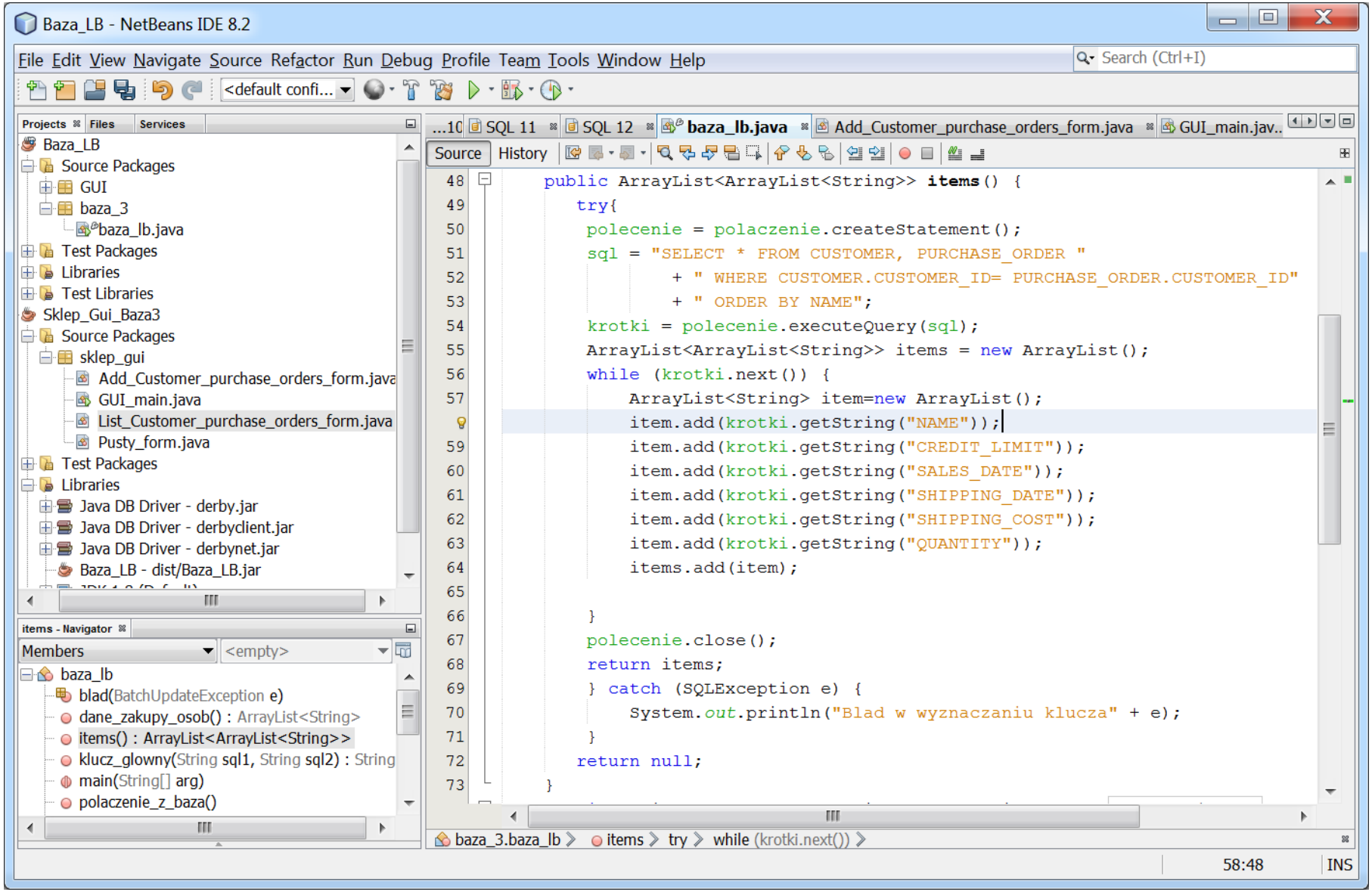
Zadanie 5 cd

4) Program **Sklep_Gui_Baza3** – warstwa klienta programu i projekt **Baza_LB** – warstwa biznesowa programu. Projekt **Baza_LB** jest okrojoną wersją programu **Baza_3** z z zadanie 3 i posiada dodatkową metodę **items**



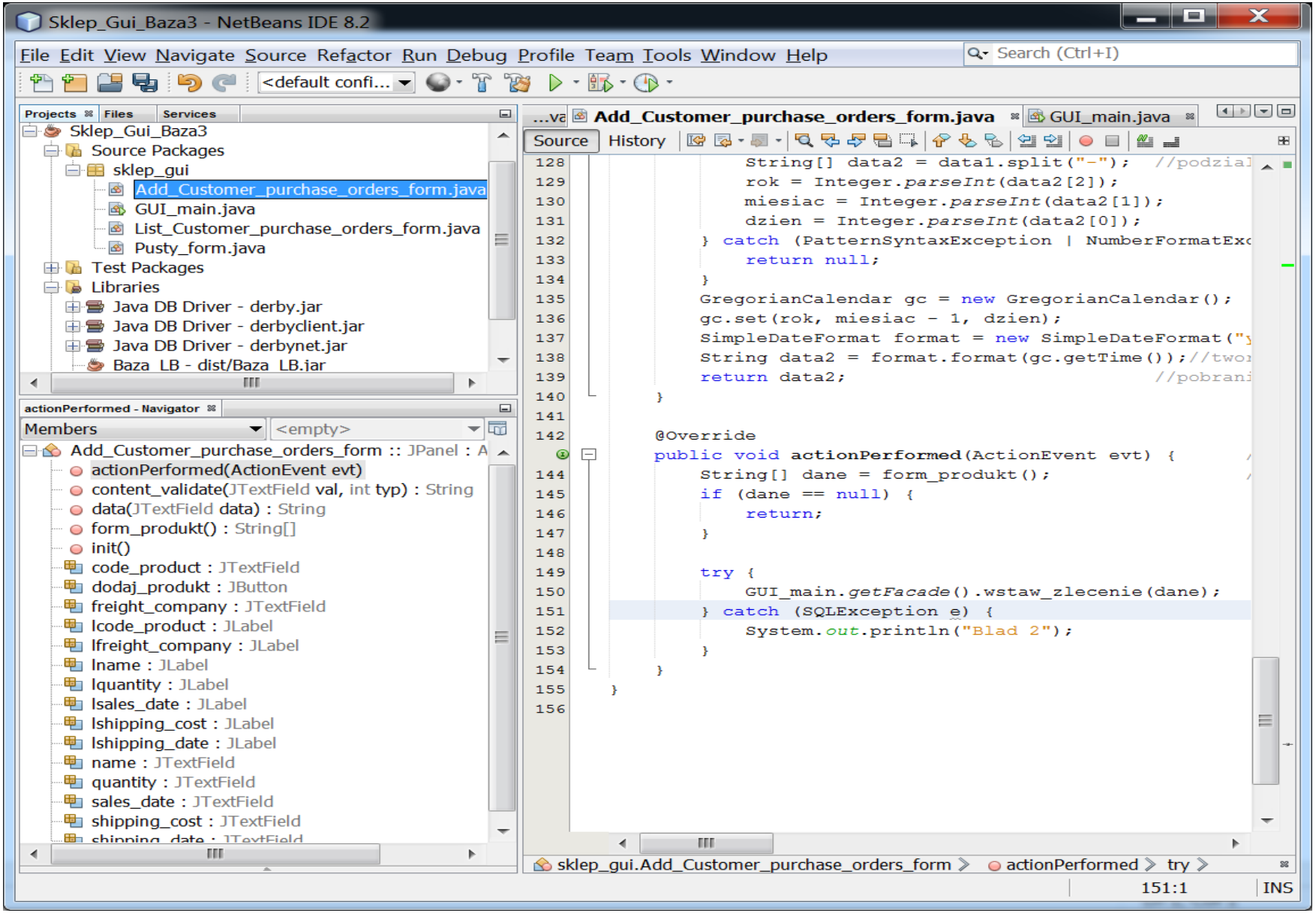
Zadanie 5 cd

5) Dodana metoda **items** (podona do metody **dane_zakupy_osob**) dostarcza model do komponentu **JTable** w formularzu **List_Customer_purchase_orders_form** w projekcie **Sklep_Gui_Baza3**



Zadanie 5 cd

6) Teraz metoda **wstaw_zlecenie** wywołana w formularzu **Add_Customer_purchase_orders_form** w projekcie **Sklep_Gui_Baza3** pozwala wstawic kolejną krotkę do tabeli PURCHASE_ORDER



Schemat bazy danych **Sample**

A. Wyszukiwanie w tabeli wszystkich krotek Product wg Manufacturer

B. Wyszukiwanie w tabeli wszystkich krotek Product wg Product_Code