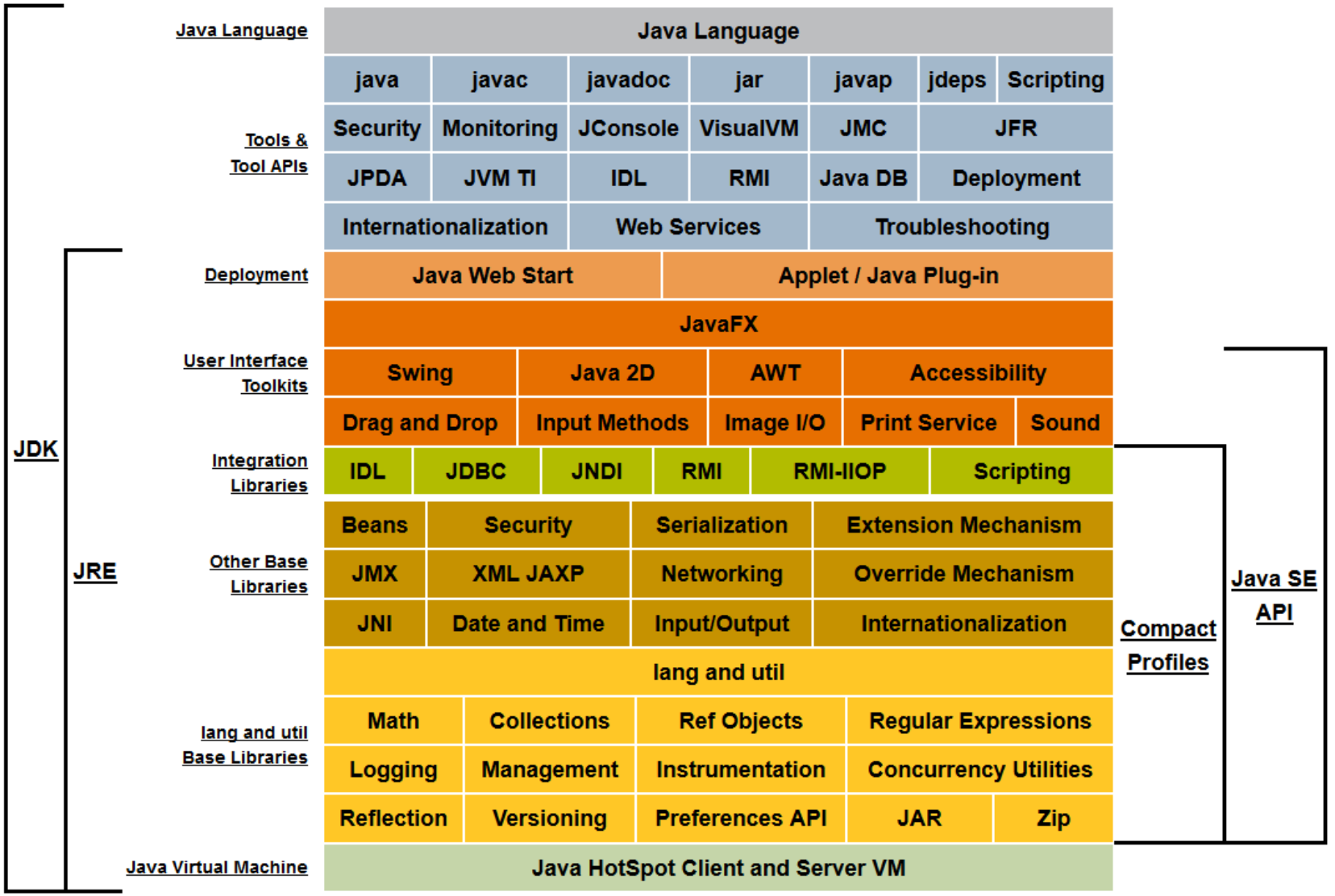
Protokół JDBC – współpraca z relacyjnymi bazami danych

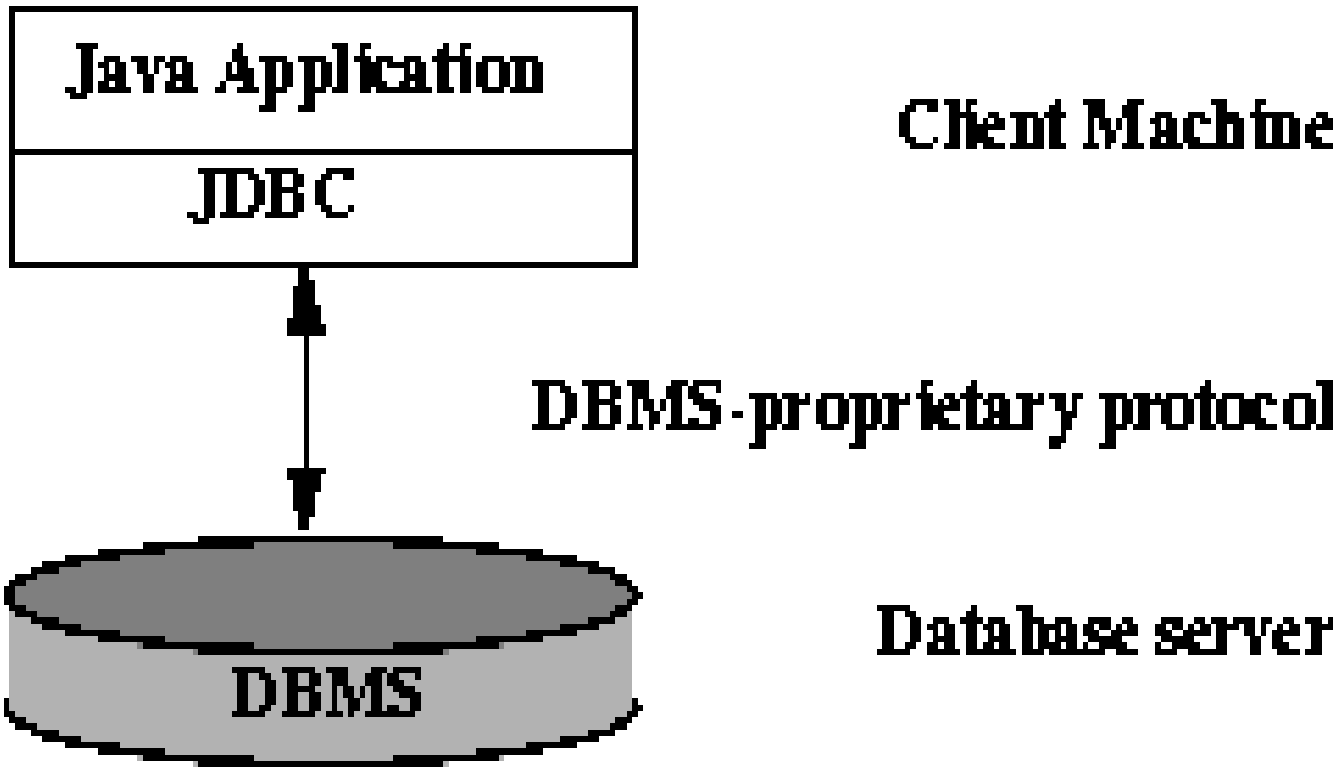
Dr inż. Zofia Kruczkiewicz

Programowanie aplikacji internetowych – wykład 2

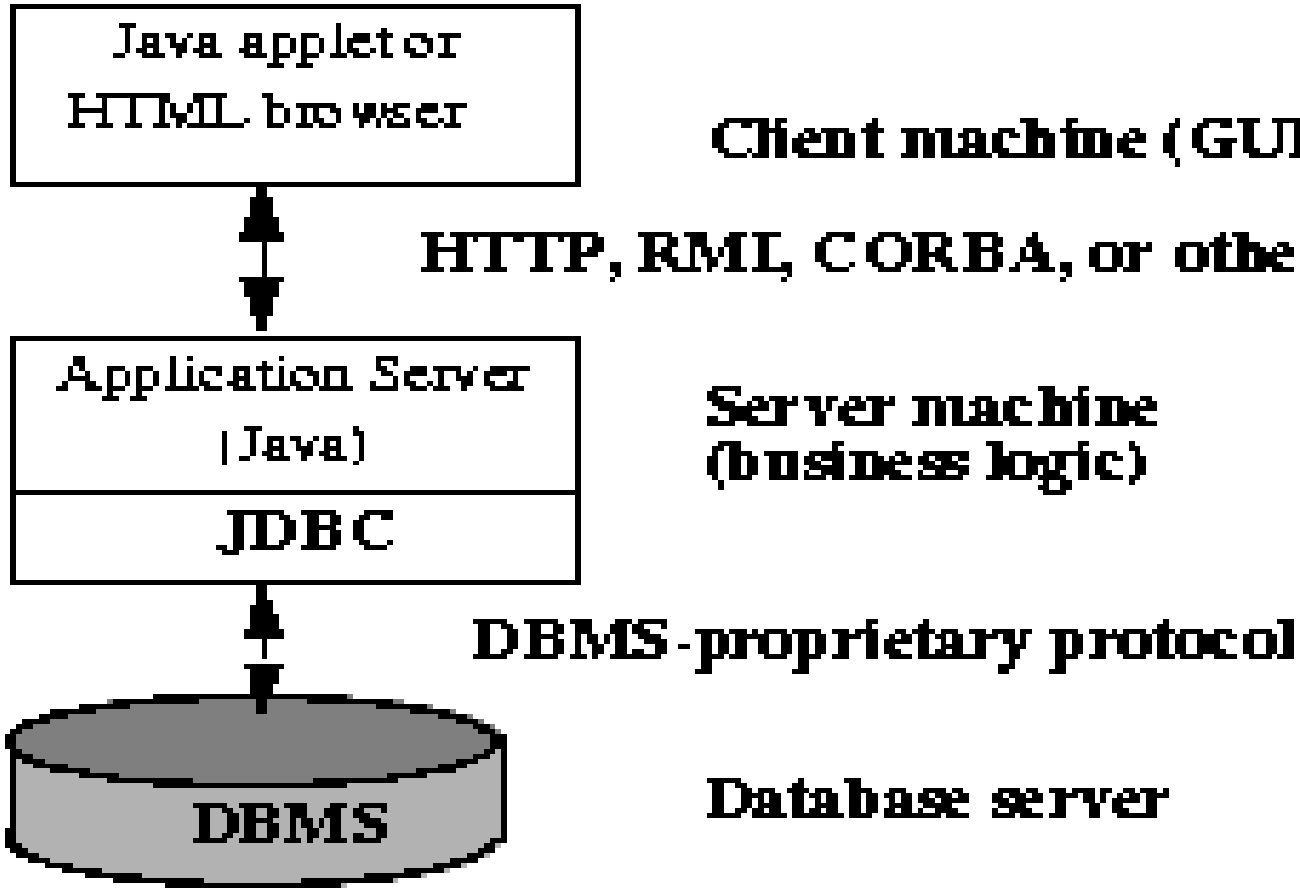
Java SE 8.0


Java Language
Java Language
java
javac
javadoc
jar
javap
jdeps
Scripting
Tools & Tool APIs
Security
Monitoring
JConsole
VisualVM
JMC
JFR
JPDA
JVM TI
IDL
RMI
Java DB
Deployment
Internationalization
Web Services
Troubleshooting
Deployment
Java Web Start
Applet / Java Plug-in
JavaFX
User Interface Toolkits
Swing
Java 2D
AWT
Accessibility
Drag and Drop
Input Methods
Image I/O
Print Service
Sound
Integration Libraries
IDL
JDBC
JNDI
RMI
RMI-IIOP
Scripting
Other Base Libraries
Beans
Security
Serialization
Extension Mechanism
JMX
XML JAXP
Networking
Override Mechanism
JNI
Date and Time
Input/Output
Internationalization
lang and util
lang and util Base Libraries
Math
Collections
Ref Objects
Regular Expressions
Logging
Management
Instrumentation
Concurrency Utilities
Reflection
Versioning
Preferences API
JAR
Zip
Java Virtual Machine
Java HotSpot Client and Server VM
JDK
JRE
Compact Profiles
Java SE API

2 Architektura JDBC
2.1.Dwuwarstwowa architektura dostępu do baz danych



2. 2. Trójwarstwowa architektura dostępu do baz danych

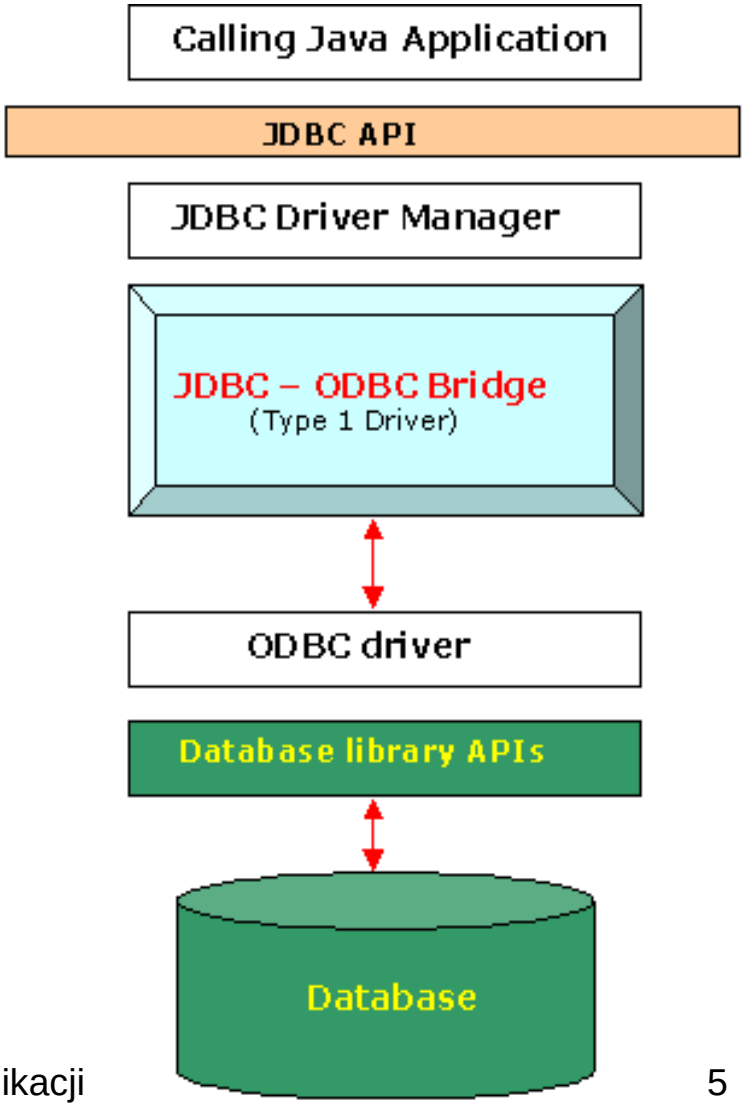


3. Cztery kategorie sterowników JDBC

3.1. JDBC-ODBC bridge plus ODBC driver (*Open Database Connectivity*)

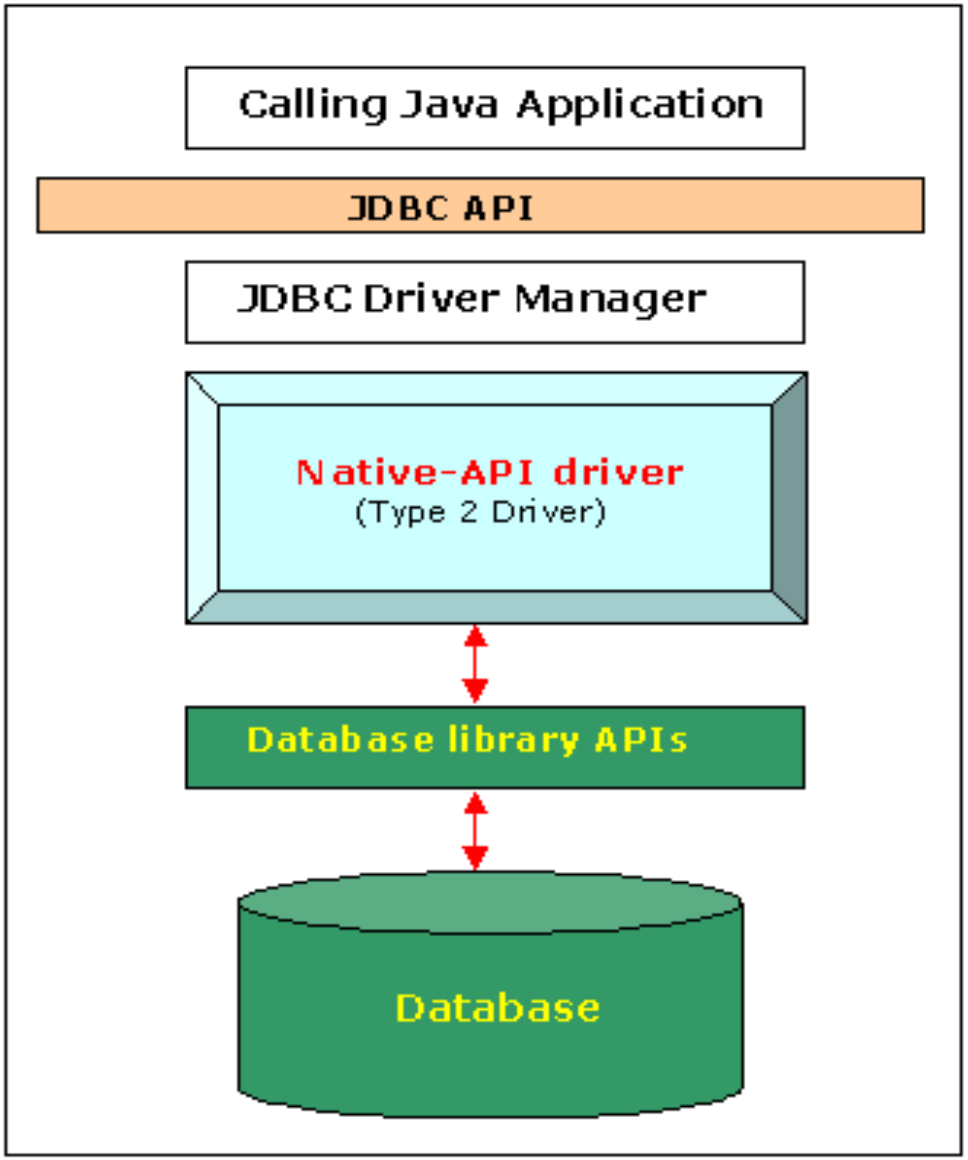
Rodzaj sterownika wspieranego sterownikiem ODBC w dostępie do baz danych (polecenia JDBC są tłumaczone na polecenia ODBC) – zależny od platformy. Potrzebny jest sterownik ODBC wspierający dostęp do wybranej bazy danych oraz dodatkowy kod na każdej maszynie klienta.

- Najprostsza metoda, jednak należy ją stosować wtedy, gdy brakuje sterownika Javy,
- Obecnie nie jest obsługiwany przez firmę *ORACLE.*



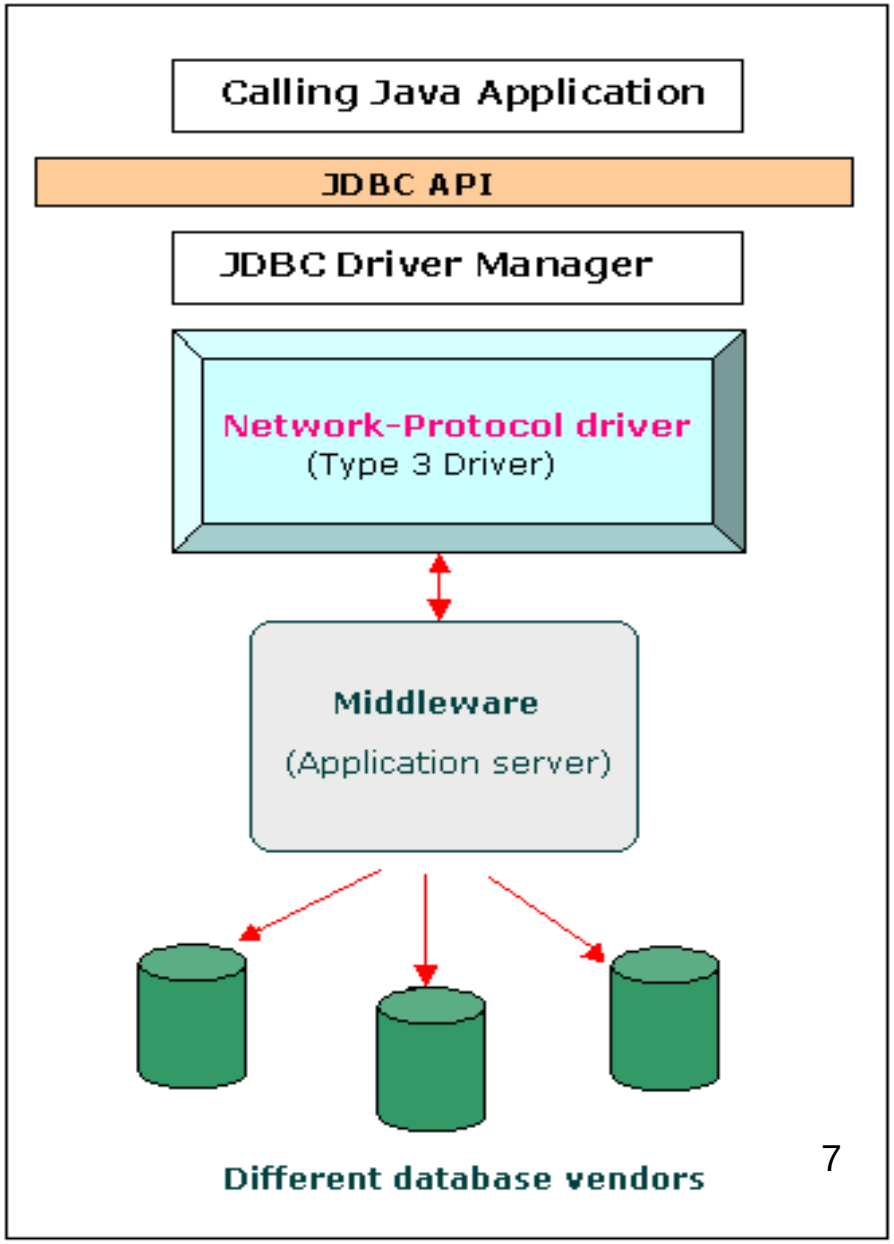
3.2. Native-API partly-Java driver:

- Rodzaj sterownika, skompilowanego dla wybranego systemu operacyjnego (zależny od platformy), który tłumaczy wywołania JDBC na wywołania API klienta Na każdej maszynie klienta jest potrzebny dodatkowy kod, podobnie jak w pierwszym przypadku.
- Bardziej funkcjonalna i wydajna metoda niż pierwsza metoda, jednak gorsza od sterownika Javy, bezpośrednio łączącego z wybraną bazą danych (4-ty typ).



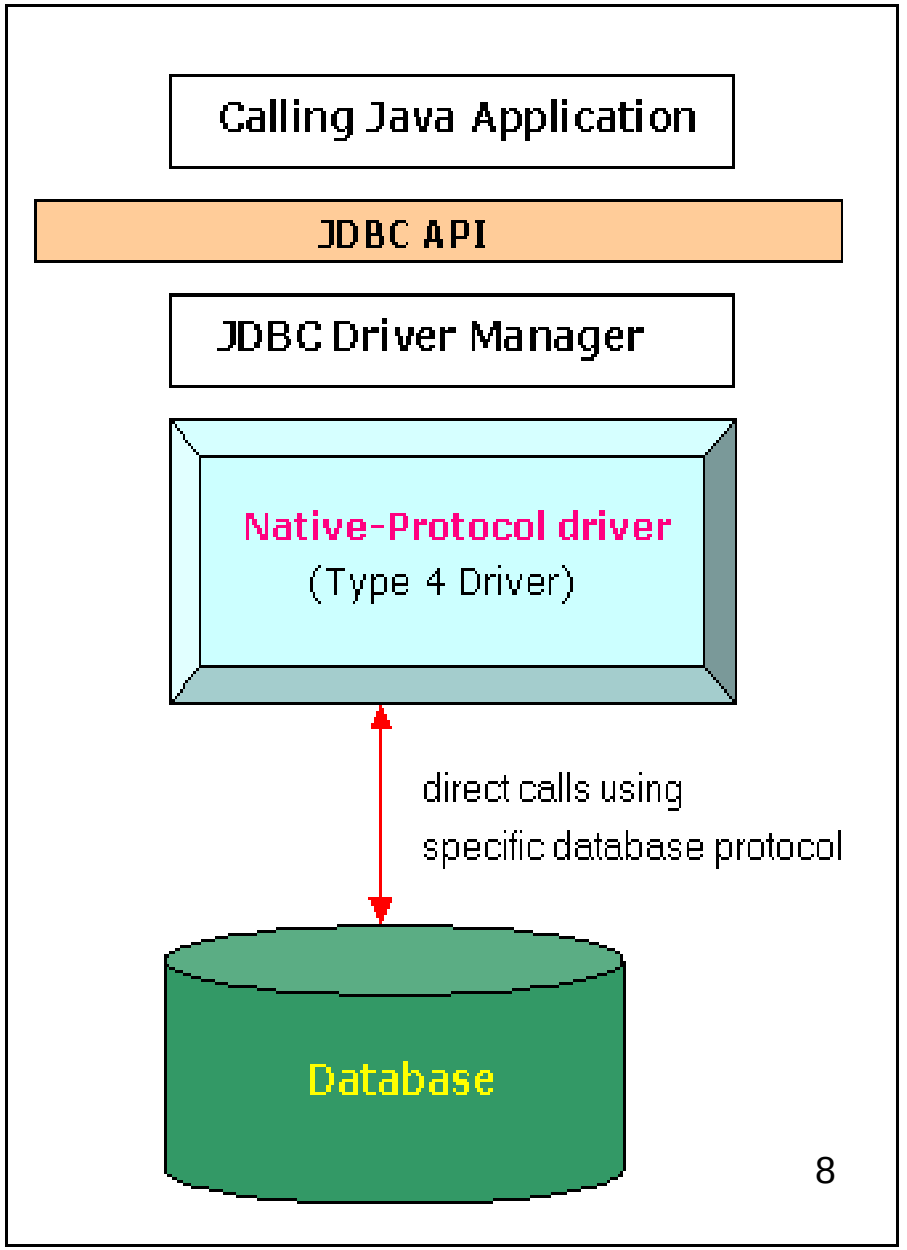
3.3. JDBC-Net pure Java driver

- Rodzaj sterownika, który tłumaczy wywołania JDBC na niezależny od baz danych protokół sieciowy, który jest dopiero przez specjalny serwer tłumaczony na język danej DBMS. Sterownik jest w pełni napisany w Javie, stanowi elastyczne połączenie do baz danych i jest niezależny od systemu operacyjnego. Nie wymaga instalowania dodatkowego oprogramowania po stronie klienta.



3.4. Native-protocol pure Java driver

- Rodzaj sterownika bezpośrednio łączący wywołania JDBC do protokołu używanego przez DBMSs. Ten sterownik warstwy pośre-dniczącej jest w pełni napisany w Javie, jest niezależny od systemu operacyjnego (instalowany wewnątrz maszyny virtualnej Javy) i jest wydajniejszy od pierwszej i drugiej metody łączenia z bazami danych. Nie wymaga instalowania dodatkowego oprogramo-wania po stronie klienta.



4. Wykonanie aplikacji z wykorzystaniem JDBC

1) wykonanie łańcucha typu String *data* identyfikującego źródło danych i rodzaj połączenia do bazy danych

Łańcuch *data* składa się z trzech członów:

<protokół>:<podprotokół>:<nazwa_źródła_danych>

np.

data =jdbc:derby://sprocket.ict.pwr.wroc.pl:5000/Katalog

lub data ="jdbc:derby:Katalog";

2) załadowanie sterownika za pomocą wywołania

Class.forName("org.apache.derby.jdbc.ClientDriver").newInstance();

,gdzie metoda *forName* ładuje klasę o nazwie podanej jako parametr do interpretera (java.lang)

4. cd. Wykonanie aplikacji z wykorzystaniem JDBC

3) połączenie ze źródłem danych za pomocą klasy Driver Manager (java.sql)

```
polaczenie = DriverManager.getConnection(data, "", "");
```

gdzie *połączenie* jest referencją do obiektu typu Connection. Metoda getConnection ma trzy parametry: *data* , nazwę użytkownika (w przykładzie pusta), hasło (w przykładzie puste)

4) przygotowanie polecenia SQL za pomocą obiektu typu *Statement:*

```
polecenie = polaczenie.createStatement();
```

5) przygotowanie zapytania SQL

```
sql="SELECT * FROM Tytul ORDER BY tytul;";
```

np. do wyboru wszystkich kolumn tabeli *Tytul*

6) wykonanie zapytania SQL:

```
krotka = polecenie.executeQuery(sql);
```

4. cd. Wykonanie aplikacji z wykorzystaniem JDBC

7) zostanie zwrócony wynik zapytania w postaci obiektu typu *ResultSet*, który ma następujące metody:

getDate(String s) zwraca wartość typu *Date* z kolumny s wiersza

getDouble(String s) zwraca wartość typu *double* z kolumny s wiersza

getFloat(String s) zwraca wartość typu float z kolumny s wiersza

getInt(String s) zwraca wartość typu *int* z kolumny s wiersza

getLong(String s) zwraca wartość typu *long* z kolumny s wiersza

getString(String s) zwraca wartość typu *String* z kolumny s wiersza

Do nawigacji po pozostałych rekordach uzyskanych po wykonaniu zapytania SQL:

next() przejście do następnego rekordu

afterLast() przejście do ostatniego rekordu

afterFirst() przejście do pierwszego rekordu

first() przejście do pierwszego rekordu

last() przejście do ostatniego rekordu

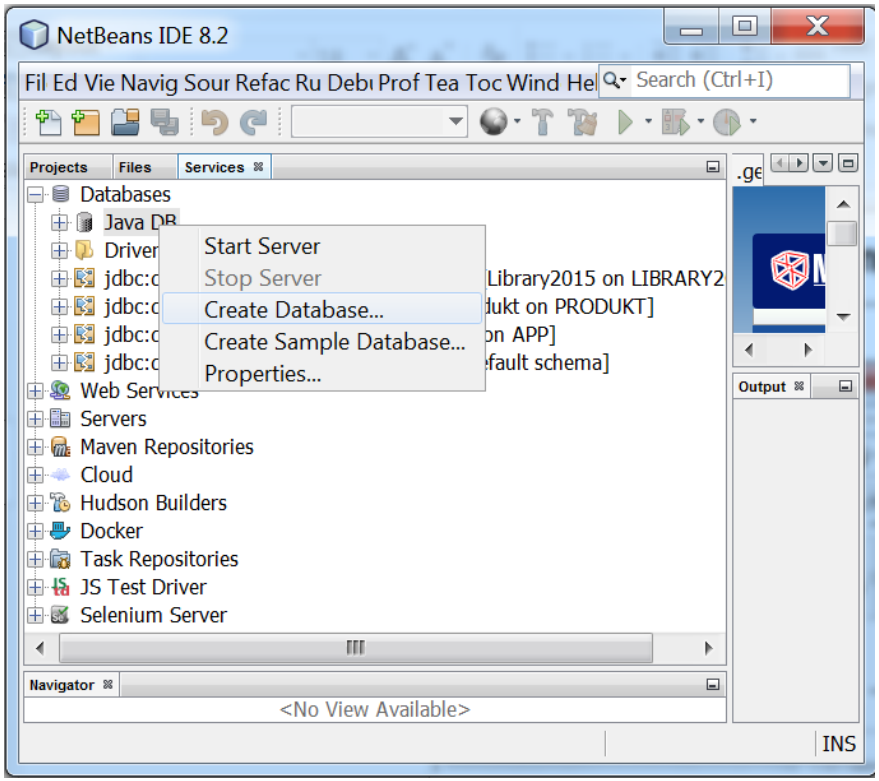
previous() przejście do poprzedniego rekordu

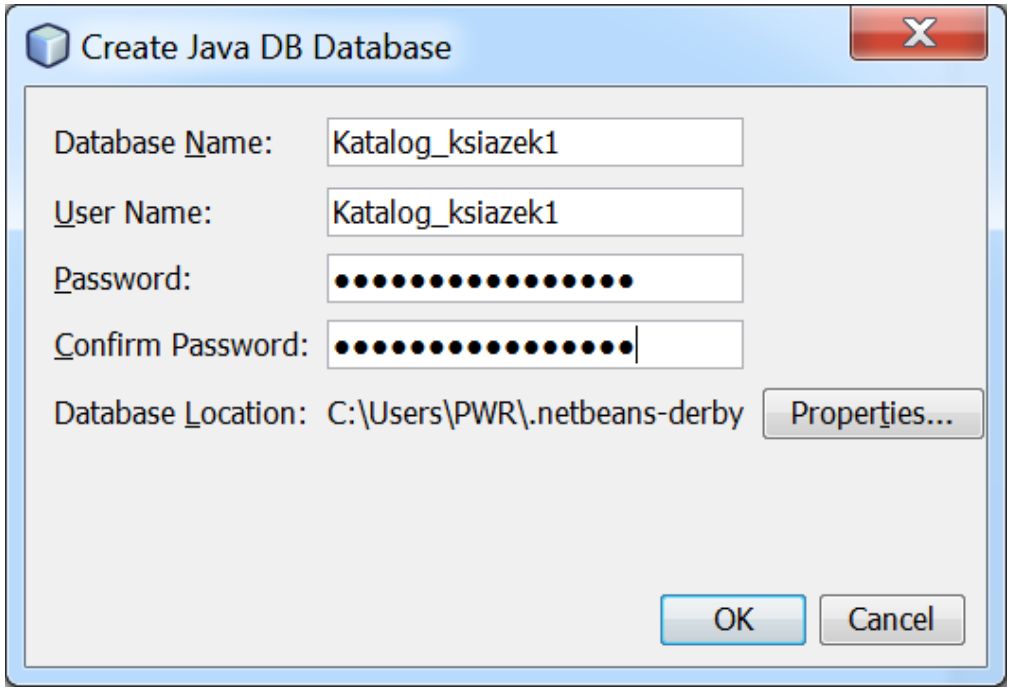
4. cd. Wykonanie aplikacji z wykorzystaniem JDBC

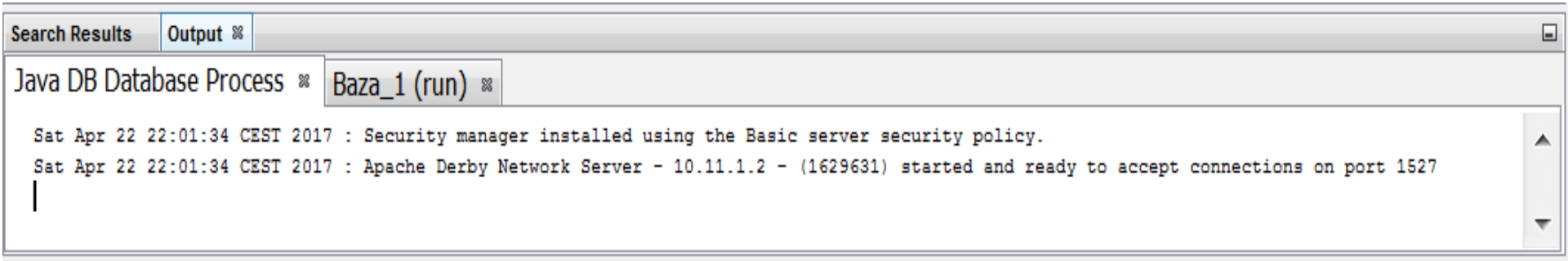
8) Po zakończeniu pracy ze źródłem danych zamknięcie połączenia

polecenie.close();

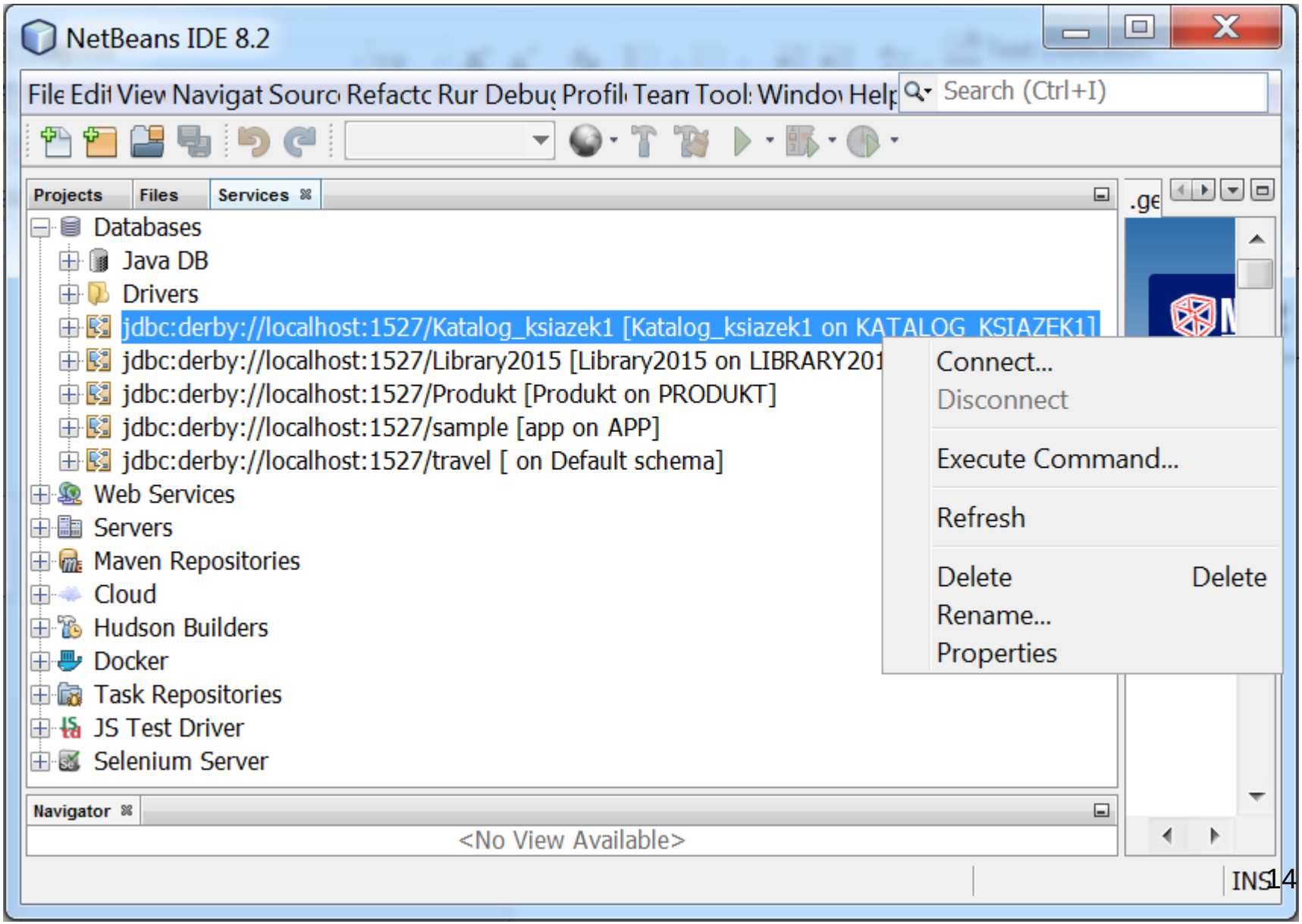
5. Tworzenie bazy danych z wykorzystaniem silnika Java DB







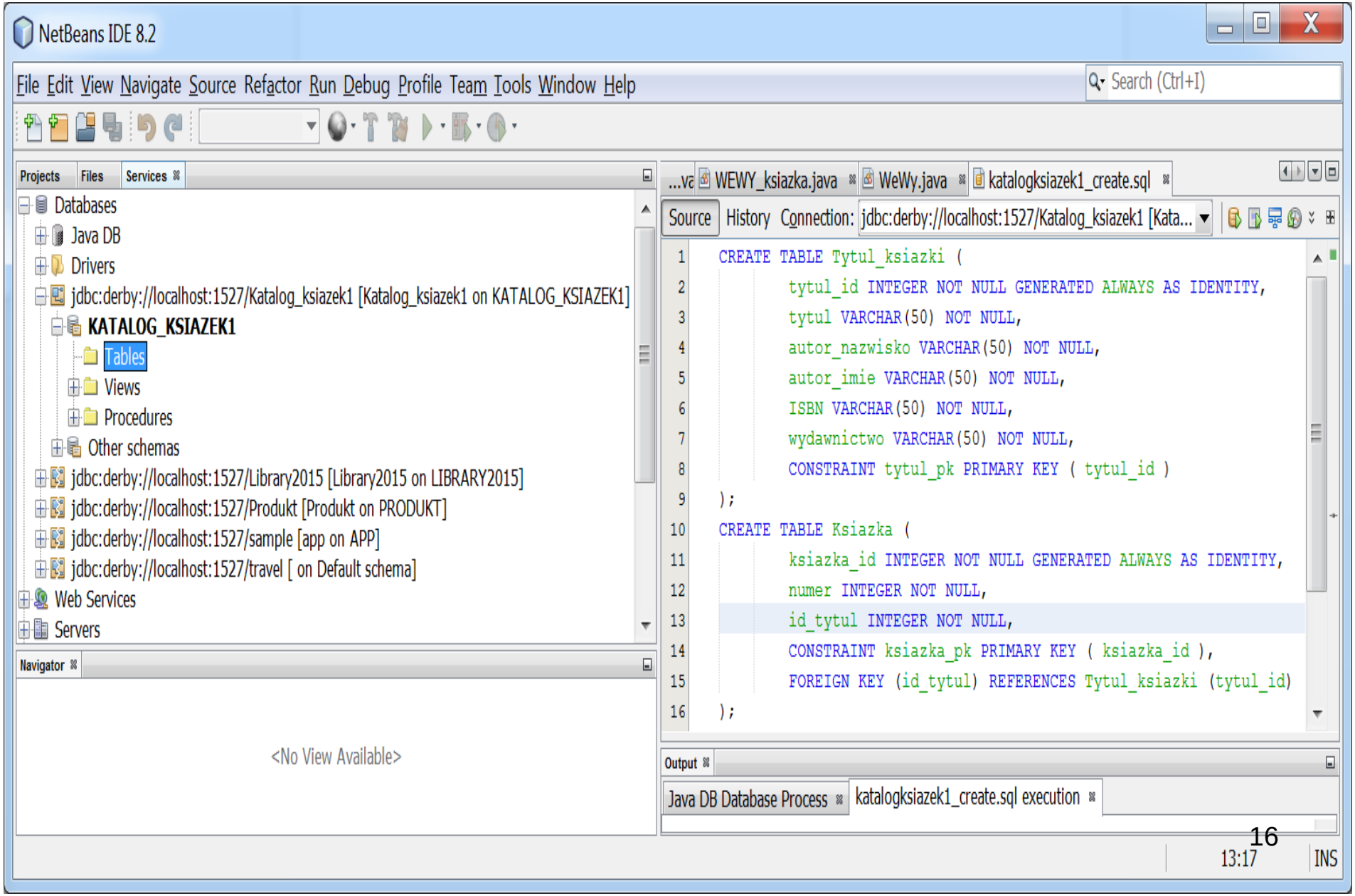
5. cd. Połączenie z bazą danych



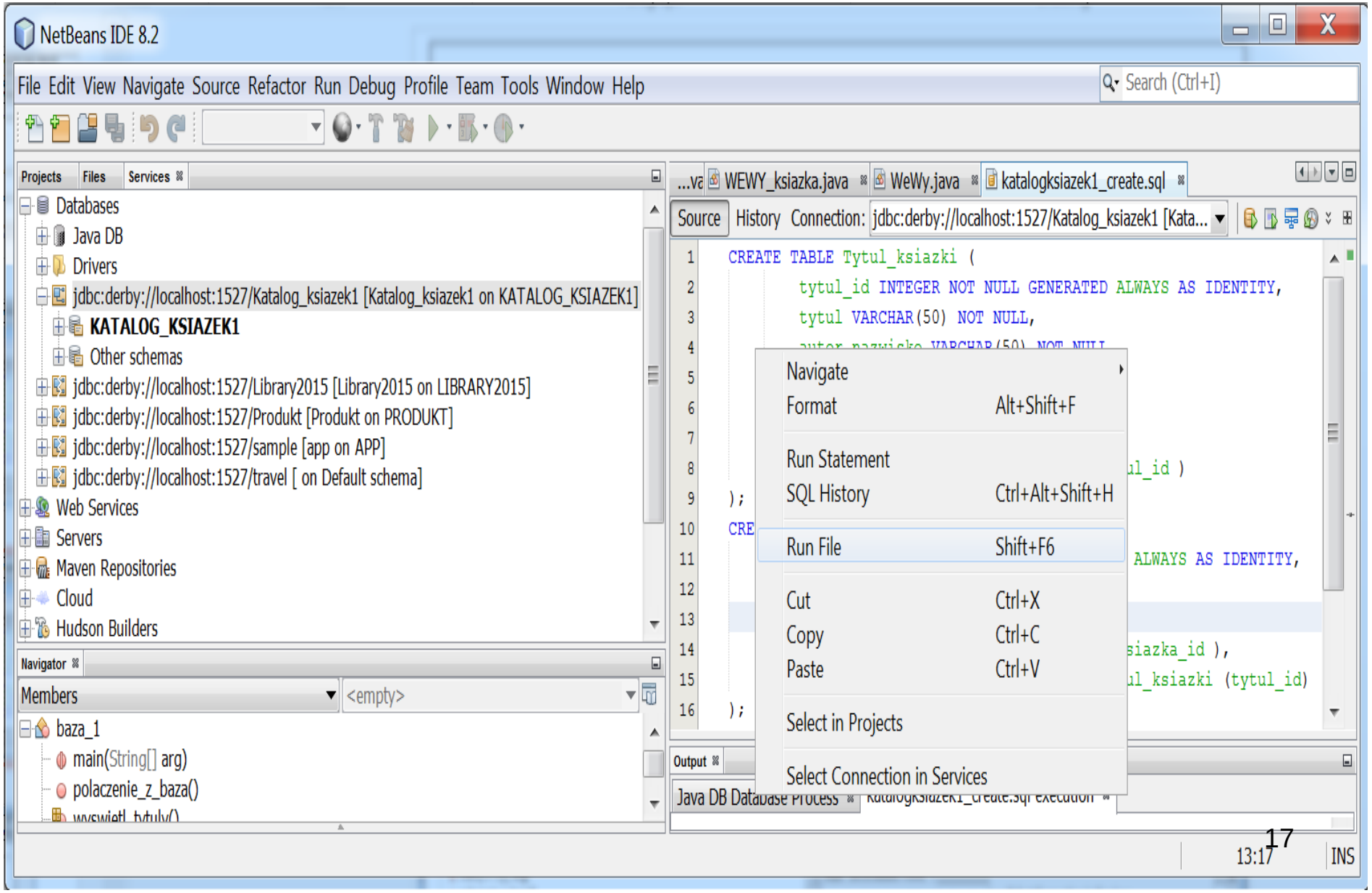
6. W celu utworzenia tabel relacyjnej bazy danych zdefiniowano skrypt w języku SQL *katalogksiazek1_create.sql* – skrypt utworzy dwie tabele: *Tytul_ksiazki* oraz *Ksiazka* powiązane relacją 1..*.

```
CREATE TABLE Tytul_ksiazki (
        tytul_id INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY,
        tytul VARCHAR(50) NOT NULL,
        autor_nazwisko VARCHAR(50) NOT NULL,
        autor_imie VARCHAR(50) NOT NULL,
        ISBN VARCHAR(50) NOT NULL,
        wydawnictwo VARCHAR(50) NOT NULL,
        CONSTRAINT tytul_pk PRIMARY KEY ( tytul_id )
);
CREATE TABLE Ksiazka (
        ksiazka_id INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY,
        numer INTEGER NOT NULL,
      id_tytul INTEGER NOT NULL,
        CONSTRAINT ksiazka_id PRIMARY KEY ( ksiazka_id ),
    FOREIGN KEY (id_tytul) REFERENCES Tytul_ksiazki (tytul_id)
);
```

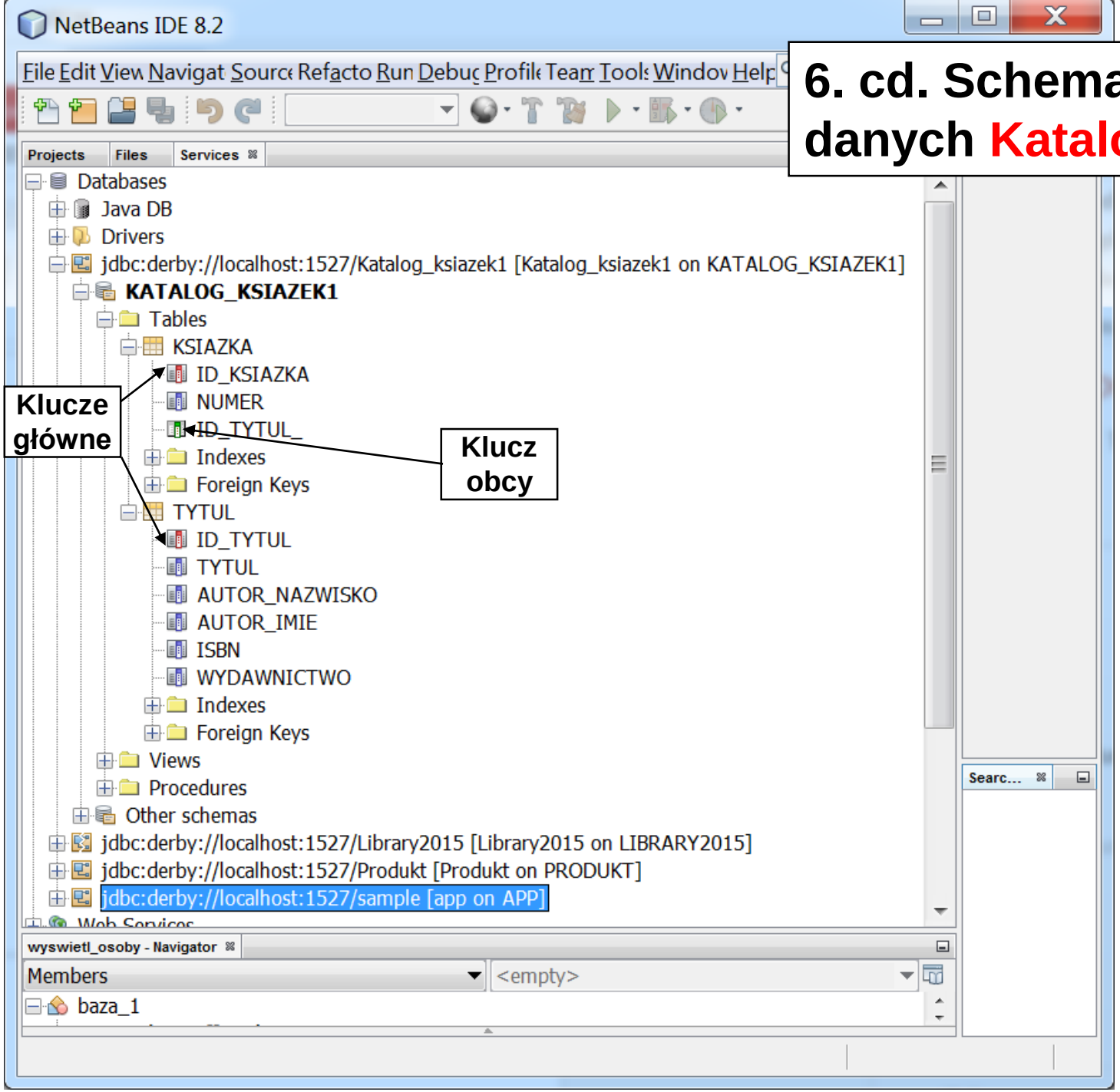
6. cd. Uruchomienie skryptu w środowisku NetBeans



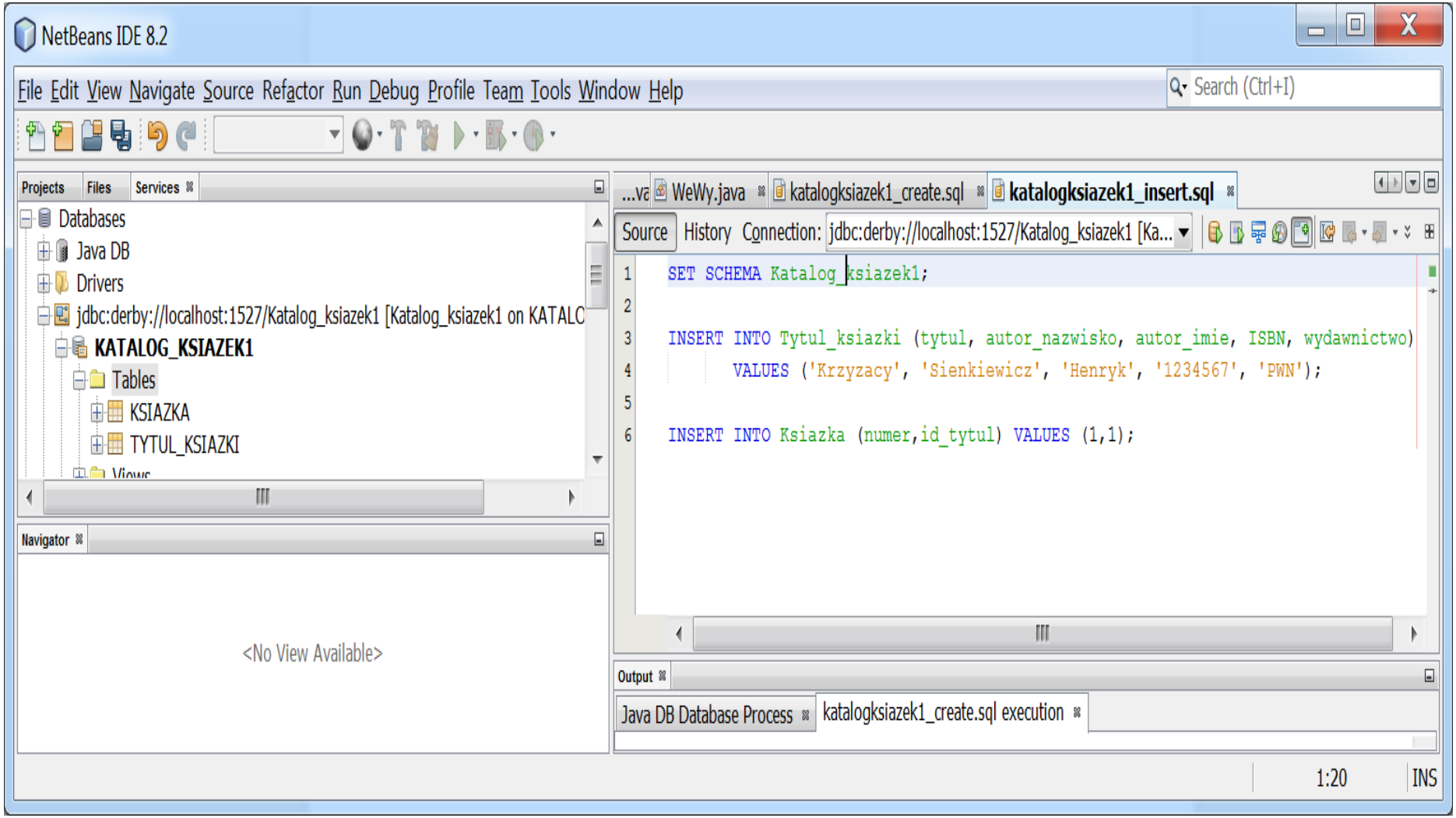
6. cd. Uruchomienia skryptu *katalogksiazek1_create.sql*



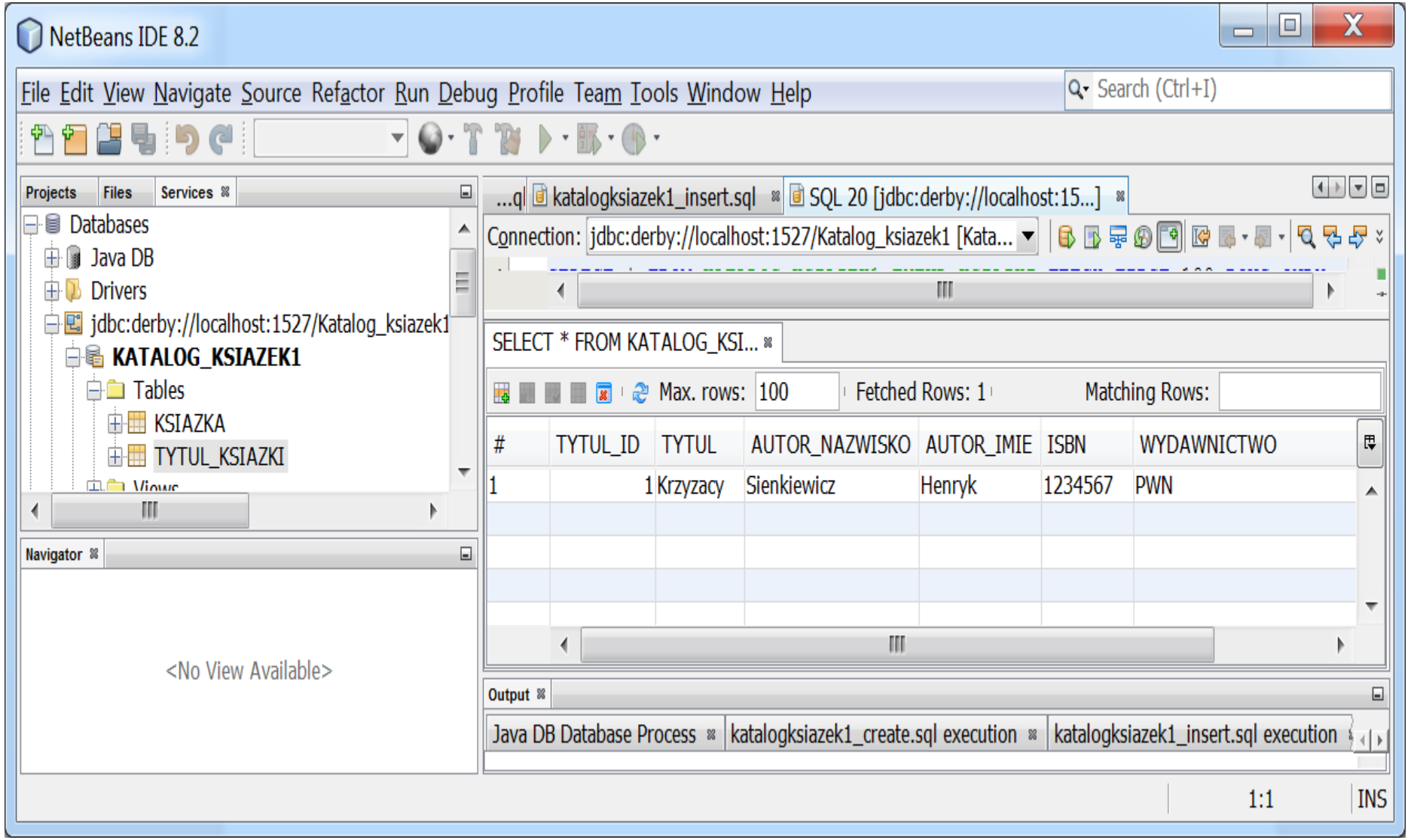
6. cd. Schemat bazy danych Katalog_ksiazek1



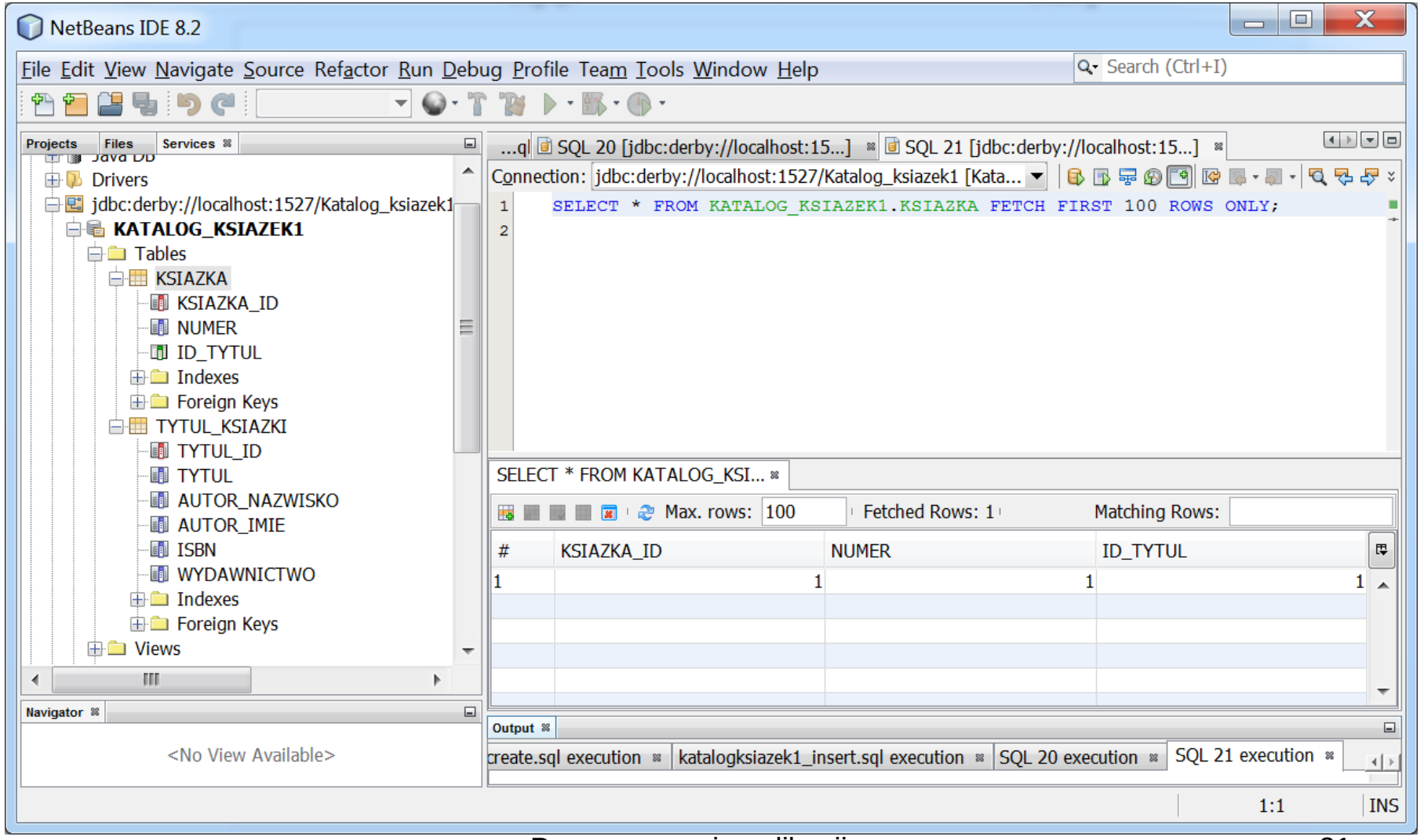
6. cd. Uruchomienia skryptu ***katalogksiazek1_insert.sql***



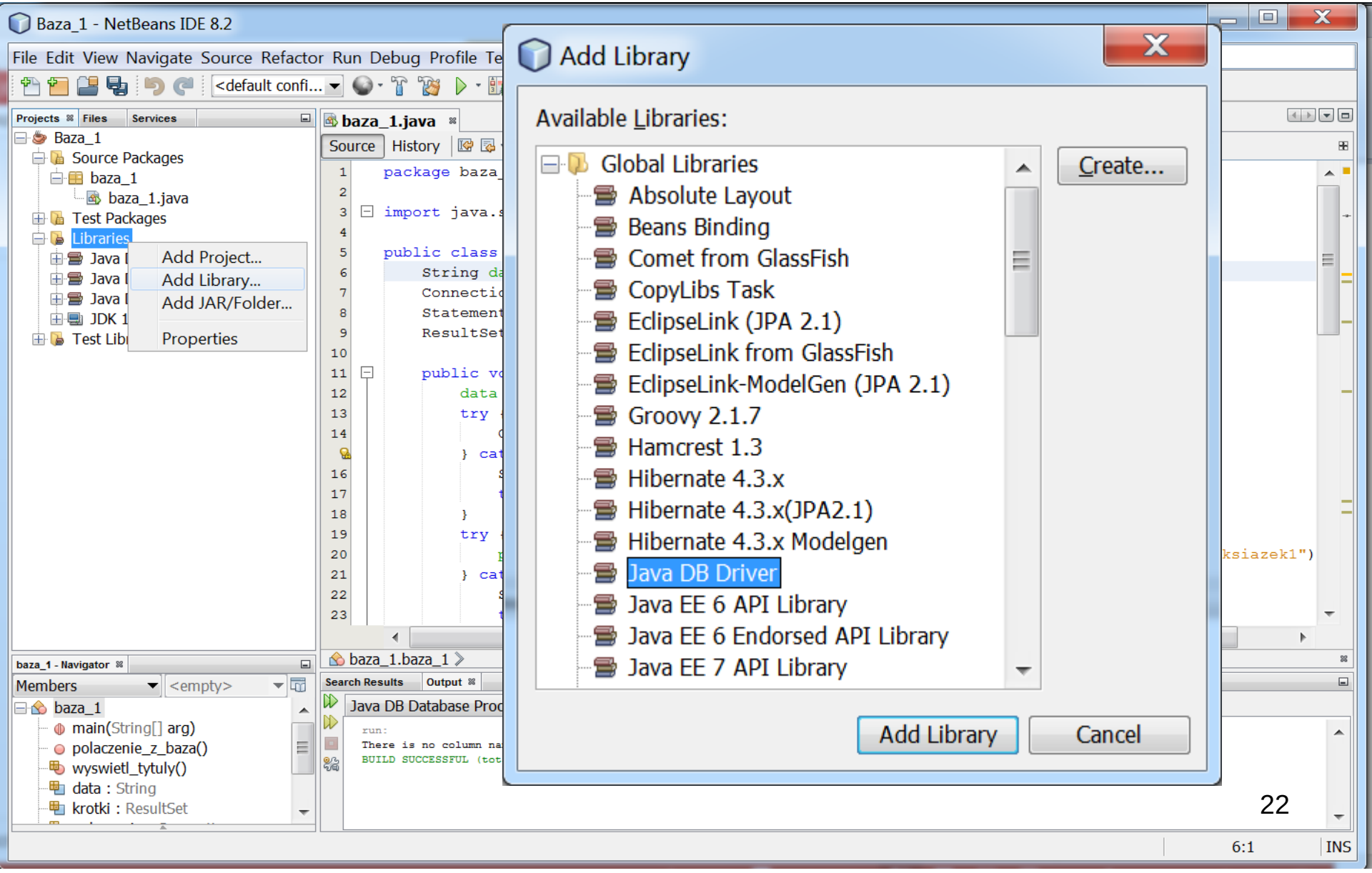
6. cd. Zawartość tabeli Tytul_ksiazki



6. cd. Zawartość tabeli Ksiazka



7. Utworzenie projektu Java Application Baza_1 i dodanie bliblioteki Java DB Driver (typ 4)



7. cd. Kod programu Baza_1 – klasa baza_1

```
package baza_1;
import java.sql.*;

public class baza_1 {
    String data, sql;
    Connection polaczenie;
    Statement polecenie;
    ResultSet krotki;
    public void polaczenie_z_baza() throws SQLException {
        data = "jdbc:derby://localhost:1527/KATALOG_KSIAZEK1";
        try {
            Class.forName("org.apache.derby.jdbc.ClientDriver").newInstance();
        } catch (Exception e) {
            System.out.println("Nie mozna zaladowac sterownika");
            throw new SQLException(e.toString()); }
        try {
             polaczenie = DriverManager.getConnection(data, "Katalog_ksiazek1",
                                                      "Katalog_ksiazek1");
        } catch (SQLException e) {
            System.out.println("Nie mozna polaczyc sie z baza danych, poniewaz:" + e);
            throw e;  }
    }
```

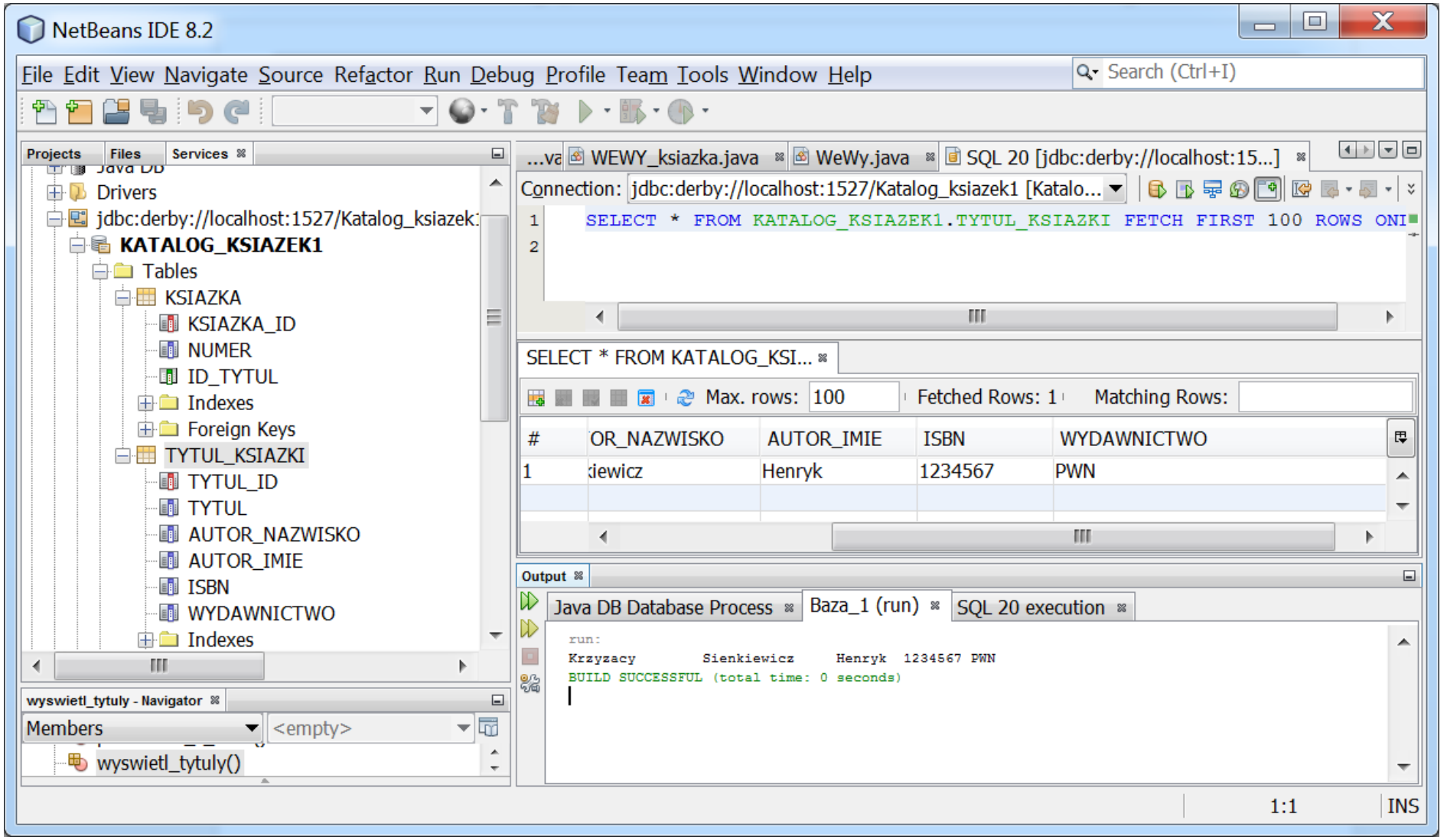
7. cd. Kod programu Baza_1 – klasa baza_1

```
void wyswietl_tytuly() throws SQLException {
        polecenie = polaczenie.createStatement();
        sql = "SELECT * FROM Tytul_ksiazki ORDER BY tytul";
        krotki = polecenie.executeQuery(sql);
        while (krotki.next()) {
            System.out.println(
                    krotki.getString("tytul") + "\t"
                    + krotki.getString("autor_nazwisko") + "\t"
                    + krotki.getString("autor_imie") + "\t"
                    + krotki.getString("ISBN")+ "\t"
                    + krotki.getString("Wydawnictwo"));
        }
        polecenie.close();
    }
}
```

7. cd. Kod programu Baza_1 – klasa baza_1

```
    static public void main(String arg[]) {
        baza_1 baza = new baza_1();
        try {
            baza.polaczenie_z_baza();
            baza.wyswietl_tytuly();
        } catch (SQLException e) {
            System.out.println(e.getMessage());
            while (null != (e = e.getNextException())) {
                System.out.println(e.getMessage());
            }
        }
    }
}
```

7. cd. Kod programu Baza_1 – klasa baza_1



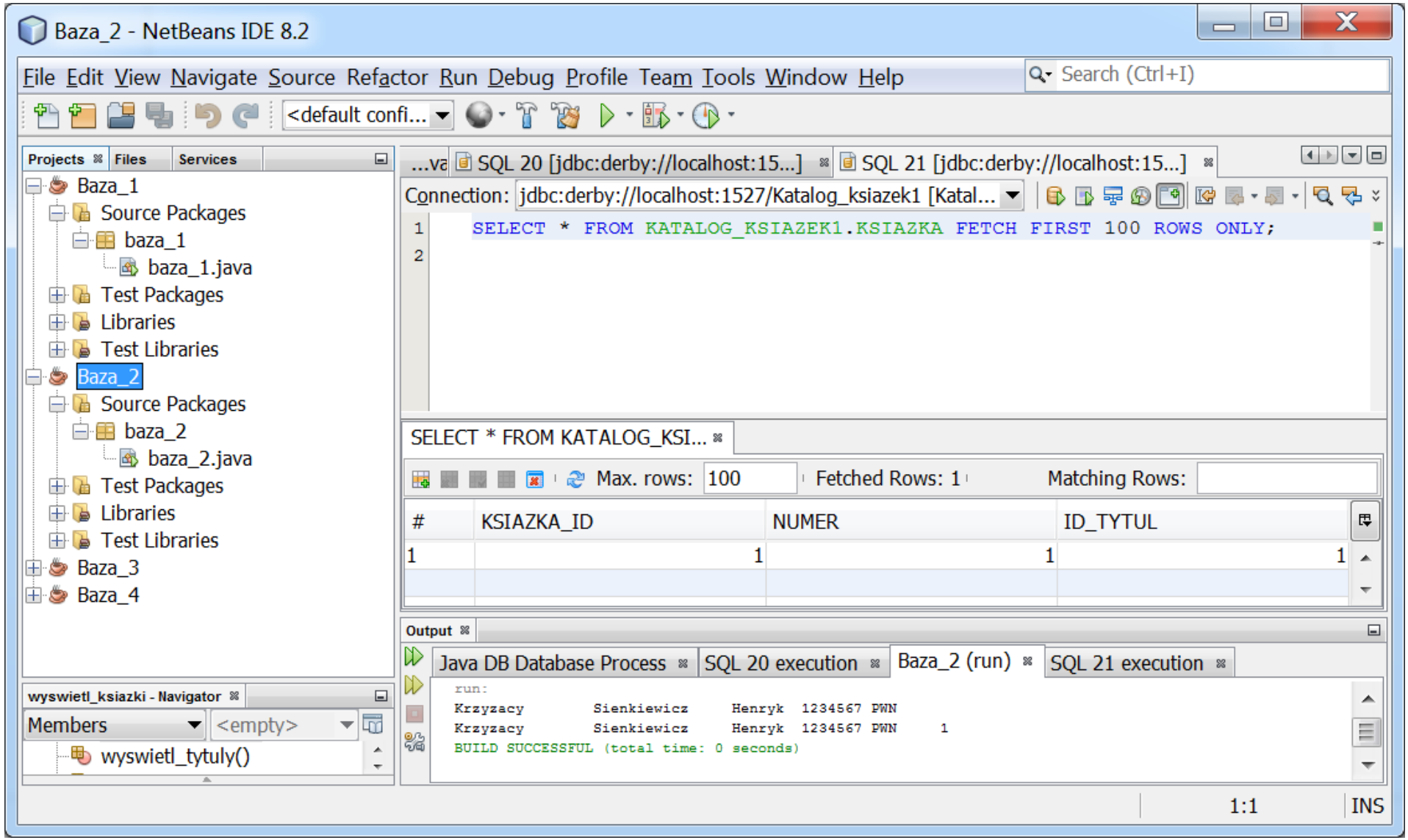
8. Utworzenie projektu Java Application Baza_2 i dodanie bliblioteki Java DB Driver (typ 4) – rozwinięcie programu Baza_1

```
void wyswietl_ksiazki() throws SQLException {
    polecenie = polaczenie.createStatement();
    sql = "SELECT * FROM Tytul_ksiazki, Ksiazka WHERE tytul_id=id_tytul "
          + " ORDER BY tytul";
    krotki = polecenie.executeQuery(sql);
    while (krotki.next()) {
       System.out.println(krotki.getString("tytul") + "\t"
             + krotki.getString("autor_nazwisko") + "\t"
             + krotki.getString("autor_imie") + "\t"
             + krotki.getString("ISBN") + "\t"
             + krotki.getString("Wydawnictwo")+ "\t"
             + krotki.getString("numer"));
    }
    polecenie.close();
}
```

8. Utworzenie projektu Java Application Baza_2 i dodanie bliblioteki Java DB Driver (typ 4) – rozwinięcie programu Baza_1

```
  static public void main(String arg[]) {
        baza_2 baza = new baza_2();
        try {
            baza.polaczenie_z_baza();
            baza.wyswietl_tytuly();
            baza.wyswietl_ksiazki();
        } catch (SQLException e) {
            System.out.println(e.getMessage());
            while (null != (e = e.getNextException())) {
                System.out.println(e.getMessage());
            }
        }
    }
}
```

8. cd. Wynik działania programu Baza_2



9. Przykład programu: **Baza_3 – klasa baza_3 – usuwanie tabel, tworzenie nowego schematu bazy danych**

```
package baza_3;
import java.sql.*;

public class baza_3 {
    String data, sql;
    Connection polaczenie;
    Statement polecenie;
    ResultSet krotki;

    public void polaczenie_z_baza() throws SQLException {
        data = "jdbc:derby://localhost:1527/KATALOG_KSIAZEK1";
        try {
            Class.forName("org.apache.derby.jdbc.ClientDriver").newInstance();
        } catch (Exception e) {
            System.out.println("Nie mozna zaladowac sterownika");
            throw new SQLException(e.toString());  }
        try {
            polaczenie = DriverManager.getConnection(data, "Katalog_ksiazek1",
                                                     "Katalog_ksiazek1");
        } catch (SQLException e) {
            System.out.println("Nie mozna polaczyc sie z baza danych, poniewaz:" + e);
            throw e;     }    }
```

9. cd. Przykład programu: **Baza_3 – klasa baza_3**

```
void tabele() throws SQLException {
    polecenie = polaczenie.createStatement();
    try {
        polecenie.executeUpdate("DROP TABLE Ksiazka");
        polecenie.executeUpdate("DROP TABLE Tytul_ksiazki");
    } catch (SQLException e) {
        System.out.println("Nie mozna usunac tabeli");
    }

   try {
        polecenie.executeUpdate(
          "CREATE TABLE Tytul (id_tytul INTEGER, tytul VARCHAR(50),"
        + "autor VARCHAR(50), ISBN INTEGER, PRIMARY KEY (id_tytul))");
    } catch (SQLException e) {
        System.out.println("Nie mozna zalozyc tabeli Tytul");
    }
```

Usuwanie tabel **Ksiazka** i **Tytul_ksiazki**

Tworzenie tabeli **Tytul**

9. cd. Przykład programu: **Baza_3 – klasa baza_3**

Tworzenie tabeli **Ksiazka**

```
try {
      polecenie.executeUpdate(
        "CREATE TABLE Ksiazka (id_ksiazka INTEGER, numer INTEGER, "
              + "id_tytul_ INTEGER, PRIMARY KEY (id_ksiazka), "
              + "FOREIGN KEY (id_tytul_) REFERENCES Tytul (id_tytul))");
      } catch (SQLException e) {
         System.out.println("Nie mozna zalozyc tabeli Ksiazka");
      }
```

Wstawianie krotek do tabeli **Tytul**

```
 for (int i = 1; i < 10; i++) {
   polecenie.executeUpdate("INSERT INTO Tytul (id_tytul,tytul, autor, ISBN)"
                    + " VALUES (" + i + ",'Tytul" + i + "','Autor" + i + "'," + i + ")");
      }
    }
```

```
void wyswietl_tytuly() throws SQLException {
    polecenie = polaczenie.createStatement();
    sql = "SELECT * FROM Tytul ORDER BY tytul";
    krotki = polecenie.executeQuery(sql);
    ResultSetMetaData metaDane = krotki.getMetaData();
    int kolumny = metaDane.getColumnCount();

    for (int i = 0; i < kolumny; i++) {
       System.out.println("Nazwa kolumny " + i + " " +
                                 metaDane.getColumnName(i + 1));
    }
    System.out.println();

    for (int i = 1; i < kolumny; i++) {
       System.out.print(metaDane.getColumnName(i + 1) + "\t");
    }
    System.out.println("\n");
    while (krotki.next()) {
        System.out.println(krotki.getString("tytul") + "\t" +
                                 krotki.getString("autor") + "\t"+
                                 krotki.getString("ISBN"));
    }
    polecenie.close();
}
```

Pobranie liczby kolumn z tabeli **Tytul**

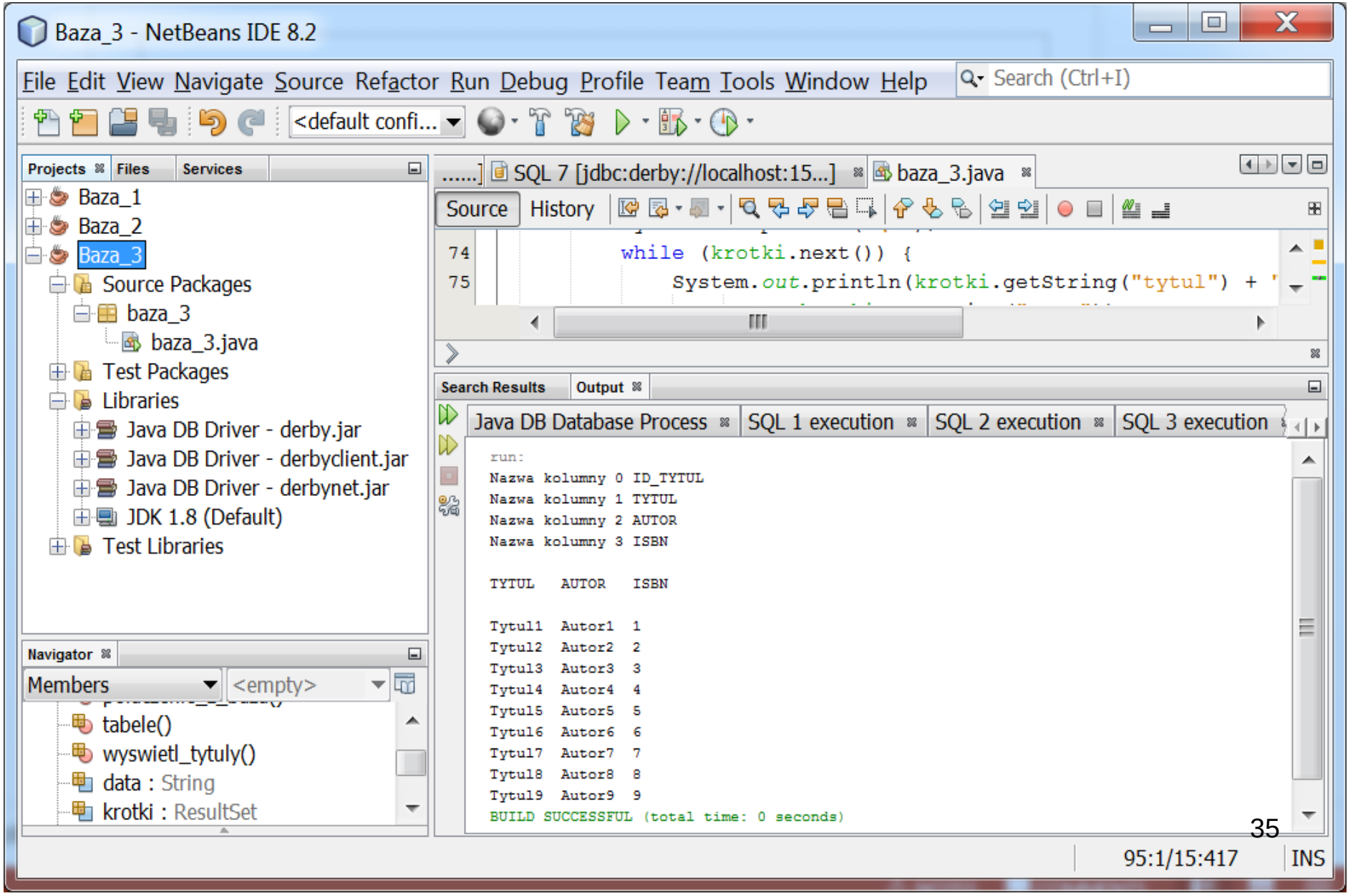
Wyświetlenie nazw kolumn z tabeli **Tytul**

Wyświetlenie danych z krotek tabeli **Tytul**

9. cd. Przykład programu: **Baza_3 – klasa baza_3**

```
    static public void main(String arg[]) {
        baza_3 baza = new baza_3();
        try {
            baza.polaczenie_z_baza();
            baza.tabele();
            baza.wyswietl_tytuly();
        } catch (SQLException e) {
            System.out.println(e.getMessage());
            while (null != (e = e.getNextException())) {
                System.out.println(e.getMessage());
            }
        }
    }
}
```

9. cd. Przykład programu: **Baza_3 – klasa baza_3**



10. Przykład programu: Baza_4 – klasa baza_4

```
package baza_4;
import java.sql.*;

public class baza_4 {
    String data, sql;
    Connection polaczenie;
    Statement polecenie;
    ResultSet krotki;
    static int id=14;
    public void polaczenie_z_baza() throws SQLException {
        data = "jdbc:derby://localhost:1527/KATALOG_KSIAZEK1";
        try {
            Class.forName("org.apache.derby.jdbc.ClientDriver").newInstance();
        } catch (Exception e) {
            System.out.println("Nie mozna zaladowac sterownika");
            throw new SQLException(e.toString());  }
        try {
            polaczenie = DriverManager.getConnection(data, "Katalog_ksiazek1",
                                                     "Katalog_ksiazek1");
        } catch (SQLException e) {
           System.out.println("Nie mozna polaczyc sie z baza danych, poniewaz:" + e);
            throw e; }
}
```

```
void wyswietl_tytuly() throws SQLException {
    polecenie = polaczenie.createStatement();
    sql = "SELECT * FROM Tytul ORDER BY tytul";
    krotki = polecenie.executeQuery(sql);
    while (krotki.next()) {
        System.out.println(krotki.getString("tytul") + "\t"
                + krotki.getString("autor") + "\t" + krotki.getString("ISBN"));
    }
    polecenie.close();
}
 void wyswietl_ksiazki() throws SQLException {
    polecenie = polaczenie.createStatement();
    sql = "SELECT * FROM Tytul, Ksiazka  WHERE id_tytul=id_tytul_ "
          + " ORDER BY tytul";
    krotki = polecenie.executeQuery(sql);
    while (krotki.next()) {
        System.out.println(krotki.getString("tytul") + "\t"
                + krotki.getString("autor") + "\t"
                + krotki.getString("ISBN") + "\t"
                + krotki.getString("numer"));
    }
    polecenie.close();
}
```

10. cd. Przykład programu: Baza_4 – klasa baza_4

```
void wyszukaj() throws SQLException {
    String co = WeWy.weString("Podaj autora: ");
    polecenie = polaczenie.createStatement();
    sql = "SELECT * FROM Tytul, Ksiazka "
        + " WHERE id_tytul=id_tytul_  AND  autor = '" + co + "' ORDER BY tytul";
    krotki = polecenie.executeQuery(sql);
    while (krotki.next()) {
       System.out.println(krotki.getString("tytul") + "\t"
            + krotki.getString("autor") + "\t"
            + krotki.getString("ISBN") + "\t"
            + krotki.getString("numer"));
    }
    polecenie.close();
}
```

10. cd. Przykład programu: Baza_4 – klasa baza_4

```
void wstaw_tytul() throws SQLException {
    WEWY_tytul t = new WEWY_tytul();
    t.wstaw_tytul();
    polaczenie.setAutoCommit(false);    //wyłączenie trybu transakcji auto-commit
    try {
        polecenie = polaczenie.createStatement();
        sql="INSERT INTO Tytul (id_tytul,tytul, autor, ISBN)
            VALUES ("+(++id)+",'"+t.tytul + "','"+ t.autor+"',"+ t.ISBN+")";
        polecenie.addBatch(sql);
         //wprowadzenie 1 operacji SQL (można podać więcej operacji
         // w kolejnych wywołanych metodach addBATCH)
        polecenie.executeBatch();
            //wywołanie wykonania operacji (lub wielu operacji)
        polaczenie.commit();                                    //zamknięcie transakcji
    } catch (BatchUpdateException e)      //wyjątek dziedziczący po SQLException
    {
        System.out.println(e+"Wycofanie transakcji");
        polaczenie.rollback();
    }                          //jeśli wystąpiły problemy, należy odwołać transakcję
}
```

```
void wstaw_ksiazke() throws SQLException {
    WEWY_ksiazka k = new WEWY_ksiazka();
    k.wstaw_ksiazke();
    polaczenie.setAutoCommit(false);
    try {
        polecenie = polaczenie.createStatement();
        String tytul = WeWy.weString("Podaj tytul ksiazki: ");
        sql = "SELECT * FROM Tytul  WHERE tytul= '" + tytul + "'";
        krotki = polecenie.executeQuery(sql);
        if (!krotki.next()) {
            System.out.println("brak tytulu");
            return;
        }
        sql = "INSERT INTO Ksiazka (id_ksiazka,numer, id_tytul_)
              VALUES ("+(++id)+","+ k.numer + "," + krotki.getString("id_tytul") + ")";
       polecenie.addBatch(sql);
       polecenie.executeBatch();
       polaczenie.commit();
    } catch (BatchUpdateException e) {
        System.out.println("Wycofanie transakcji");
        polaczenie.rollback();
    }
}
```

```
void operacje_na_bazie() throws SQLException {
    int opcja;
    do {
        System.out.println("1  -  wyswietl tytuly");
        System.out.println("2  -  wyswietl ksiazki");
        System.out.println("3  -  wyszukaj ksiazki danego autora");
        System.out.println("4  -  wstaw tytul");
        System.out.println("5  -  wstaw ksiazke");
        System.out.println("-1 -  koniec programu");
        opcja = WeWy.weInteger("Podaj opcje: ");
        switch (opcja) {
            case 1:   System.out.println("Tytuly"); wyswietl_tytuly();      break;
            case 2:   System.out.println("Ksiazki"); wyswietl_ksiazki();  break;
            case 3:   wyszukaj();                                          break;
            case 4:   wstaw_tytul();                                       break;
            case 5:   wstaw_ksiazke();                                     break;
            case -1:  System.out.println("Koniec programu");               break;
            default:  System.out.println("Zla opcja");
        }
    } while (opcja != -1);
}
```

10. cd. Przykład programu: Baza_4 – klasa baza_4

```
static public void main(String arg[]) {
        baza_4 baza = new baza_4();
        try {
            baza.polaczenie_z_baza();
            baza.operacje_na_bazie();
        } catch (SQLException e) {
            System.out.println("Blad bazy " + e);
        }
    }
}
```

10. cd. Przykład programu: Baza_4 – klasa WEWY_ksiazka

```
package baza_4;

public class WEWY_ksiazka {

    public int numer;

    void wstaw_ksiazke() {
        numer = WeWy.weInteger("Podaj numer ksiazki: ");
    }

}
```

10. cd. Przykład programu: : Baza_4 – klasa WEWY_tytul

```
package baza_4;

public class WEWY_tytul {

   public String tytul, autor;
   public int ISBN;

   void wstaw_tytul() {
      tytul = WeWy.weString("Podaj tytul: ");
      autor = WeWy.weString("Podaj autora: ");
      ISBN = WeWy.weInteger("Podaj ISBN: ");
   }
}
```

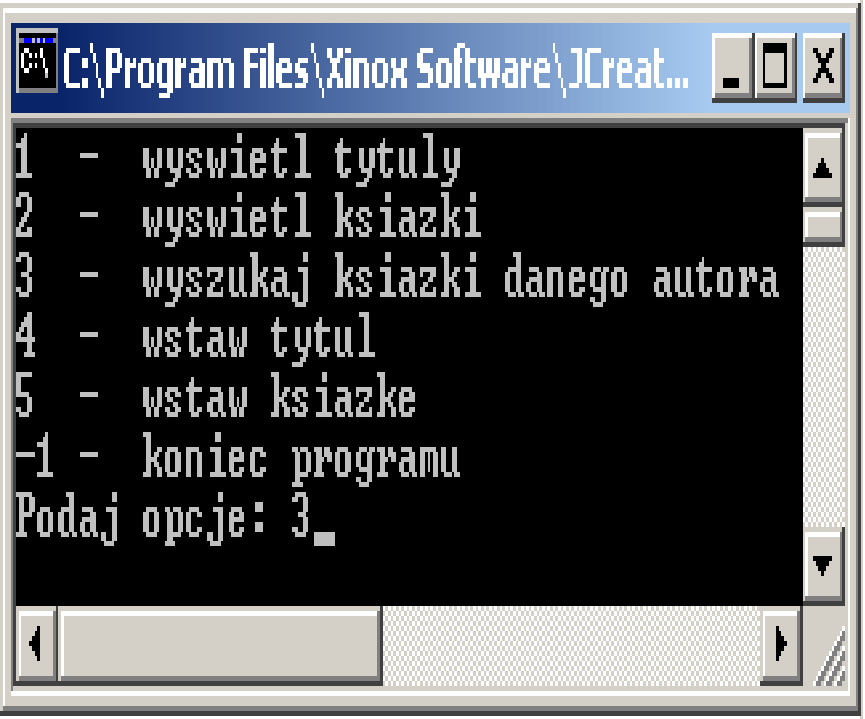
10. cd. Przykład programu: : Baza_4 – klasa WyWy

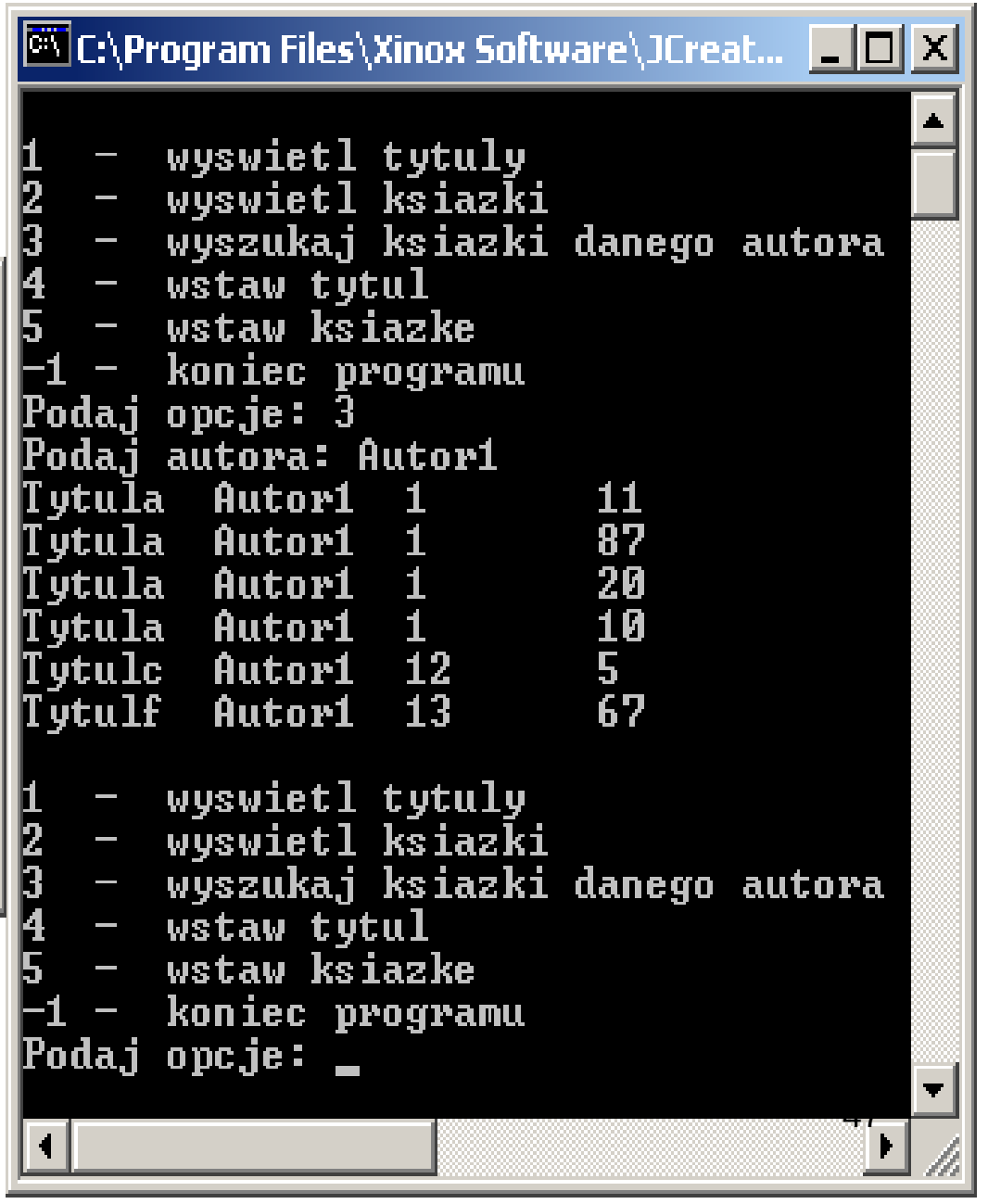
```
package baza_4;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.StringTokenizer;

public class WeWy {

  static 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 "";        }
  }
```

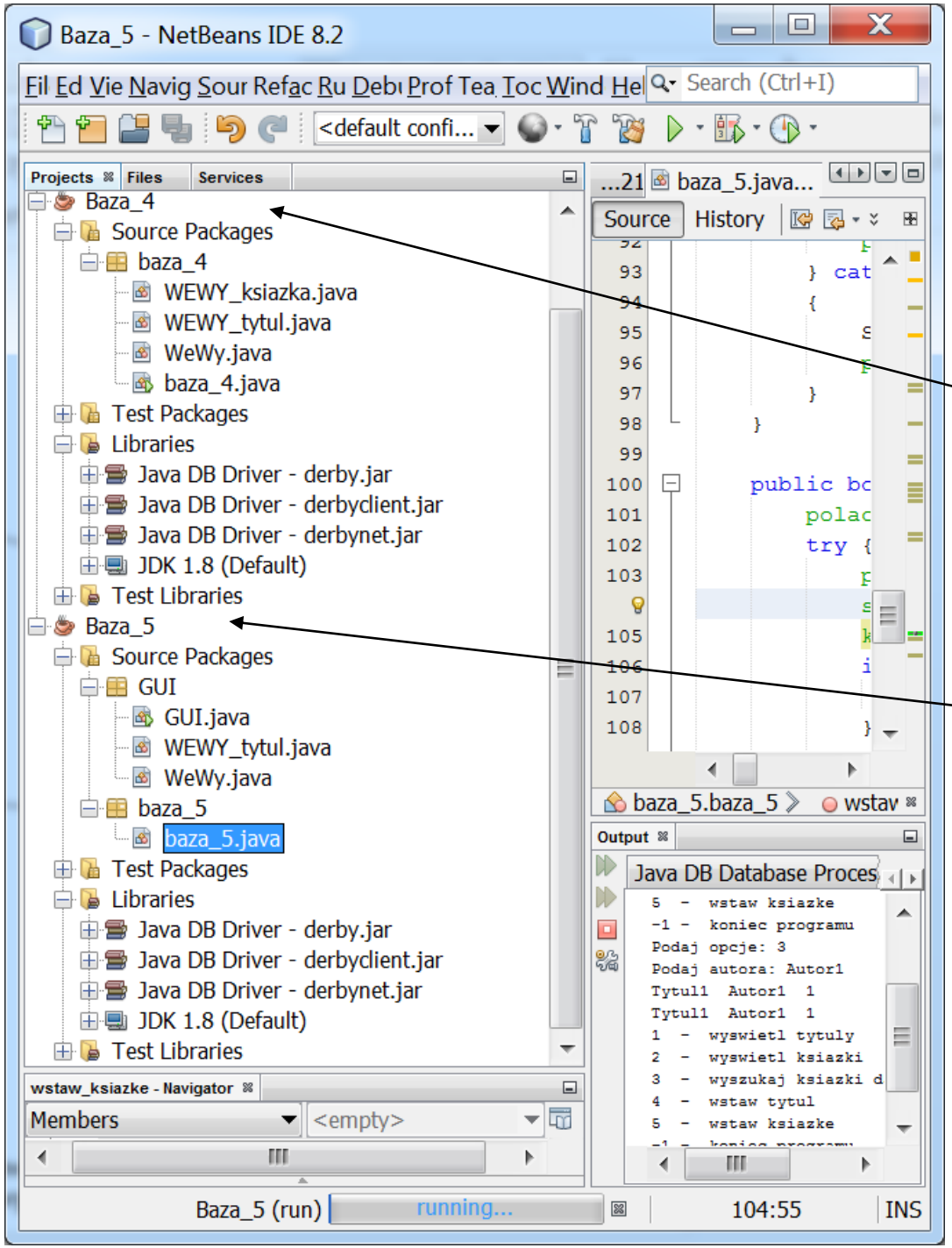
10. cd. Przykład programu: : Baza_4 – klasa WyWy

```
static byte weInteger(String menu)
    {
      InputStreamReader wejscie = new InputStreamReader( System.in );
      BufferedReader bufor = new BufferedReader( wejscie );
      StringTokenizer zeton;
      try
        {
        System.out.print(menu);
        zeton = new StringTokenizer(bufor.readLine());
        return Byte.parseByte(zeton.nextToken());
         }
       catch (Exception e)
         {
           System.err.println("Blad Integer "+e);
           return 0;
         }
     }
}
```

10. cd. Przykład programu: Baza_4







Refaktoryzacja kodu programu Baza_4

(wydzielenie logiki biznesowej do klasy **baza_5**, a operacji GUI do klas z pakietu GUI)

- utworzenie programu Baza_5

11. Przykład programu: Baza_5 – klasa baza_5

```
package baza_5;

import java.sql.*;
import java.util.ArrayList;

public class baza_5 {
    String data, sql;
    Connection polaczenie;
    Statement polecenie;
    ResultSet krotki;
    static int id=14;
    public void polaczenie_z_baza() throws SQLException {
        data = "jdbc:derby://localhost:1527/KATALOG_KSIAZEK1";
        try {
            Class.forName("org.apache.derby.jdbc.ClientDriver").newInstance();
        } catch (Exception e) {
            System.out.println("Nie mozna zaladowac sterownika");
            throw new SQLException(e.toString());  }
        try {
            polaczenie = DriverManager.getConnection(data, "Katalog_ksiazek1",
                                                           "Katalog_ksiazek1");
        } catch (SQLException e) {
           System.out.println("Nie mozna polaczyc sie z baza danych, poniewaz:" + e);
            throw e; }
}
```

11. cd. Przykład programu: Baza_5 – klasa baza_5

```
public ArrayList<String> wyswietl_tytuly() throws SQLException {
    polecenie = polaczenie.createStatement();
    sql = "SELECT * FROM Tytul ORDER BY tytul";
    krotki = polecenie.executeQuery(sql);
    ArrayList<String> tytuly =tytuly(krotki);
    polecenie.close();
    return tytuly;
}

public ArrayList<String> tytuly(ResultSet krotki_) throws SQLException {
    ArrayList<String> tytuly = new ArrayList();
    while (krotki_.next()) {
        String krotka = (krotki_.getString("tytul") + "\t"
                + krotki_.getString("autor") + "\t" + krotki.getString("ISBN"));
        tytuly.add(krotka);
    }
    return tytuly;
}
```

11. cd. Przykład programu: Baza_5 – klasa baza_5

```
public ArrayList<String> wyswietl_ksiazki() throws SQLException {
    polecenie = polaczenie.createStatement();
    sql = "SELECT * FROM Tytul, Ksiazka  WHERE id_tytul=id_tytul_ "
          + " ORDER BY tytul";
    krotki = polecenie.executeQuery(sql);
    ArrayList<String> ksiazki=ksiazki(krotki);
    polecenie.close();
    return ksiazki;
}

public ArrayList<String> ksiazki(ResultSet krotki_) throws SQLException {
    ArrayList<String> ksiazki = new ArrayList();
    while (krotki.next()) {
       String krotka=krotki.getString("tytul") + "\t"
             + krotki.getString("autor") + "\t"
             + krotki.getString("ISBN") + "\t"
             + krotki.getString("numer");
       ksiazki.add(krotka);
    }
    return ksiazki;
}
```

11. cd. Przykład programu: Baza_5 – klasa baza_5

```
public ArrayList<String> wyszukaj(String co) throws SQLException {
    polecenie = polaczenie.createStatement();
    sql = "SELECT * FROM Tytul, Ksiazka "
       + " WHERE id_tytul=id_tytul_  AND  autor =  '" + co + "'  ORDER BY tytul";
    krotki = polecenie.executeQuery(sql);
    ArrayList<String> ksiazki=ksiazki(krotki);
    polecenie.close();
    return ksiazki;
}
```

11. cd. Przykład programu: Baza_5 – klasa baza_5

```
public void wstaw_tytul(String [] tytul) throws SQLException {
    polaczenie.setAutoCommit(false);  //wyłączenie trybu transakcji auto-commit
    try {
        polecenie = polaczenie.createStatement();
        sql = "INSERT INTO Tytul (id_tytul,tytul, autor, ISBN)  VALUES (" + (++id) +
                ",'" + tytul[0] + "','" + tytul[1] + "'," + Integer.parseInt(tytul[2]) + ")";

        polecenie.addBatch(sql);
                //wprowadzenie 1 operacji SQL (można podać więcej operacji
                // w kolejnych wywołanych metodach addBATCH)
        polecenie.executeBatch();
                //wywołanie wykonania operacji (lub wielu operacji)
        polaczenie.commit();                    //zamknięcie transakcji
    } catch (BatchUpdateException e)     //wyjątek dziedziczący po SQLException
    {
        System.out.println(e + "Wycofanie transakcji");
        polaczenie.rollback(); //jeśli wystąpiły problemy, należy odwołać transakcję
     }
}
```

11. cd. Przykład programu: Baza_5 – klasa baza_5

```
public boolean wstaw_ksiazke(String tytul, int numer) throws SQLException {
    polaczenie.setAutoCommit(false);
    try {
        polecenie = polaczenie.createStatement();
        sql = "SELECT * FROM Tytul  WHERE tytul= '" + tytul + "'";
        krotki = polecenie.executeQuery(sql);
        if (!krotki.next()) {
           return false;
        }
       sql = "INSERT INTO Ksiazka (id_ksiazka,numer, id_tytul_) VALUES (" + (++id) +
                ","+ numer + "," + Integer.parseInt(krotki.getString("id_tytul")) + ")";
        polecenie.addBatch(sql);
        polecenie.executeBatch();
        polaczenie.commit();
    } catch (BatchUpdateException e) {
        System.out.println("Wycofanie transakcji");
        polaczenie.rollback();
        return false;
    }
    return true;
}
```

11. cd. Przykład programu: Baza_5 – klasa GUI

```
package GUI;

import baza_5.baza_5;
import java.sql.SQLException;
import java.util.ArrayList;

public class GUI {

    baza_5 baza = new baza_5();

    public void wyswietl_kolekcje(ArrayList<String> kolekcja) {
        for (String dana : kolekcja) {
            System.out.println(dana);
        }
    }

    public void polaczenie_z_baza() throws SQLException {
        baza.polaczenie_z_baza();
    }
```

11. cd. Przykład programu: Baza_5 – klasa GUI

```
public void operacje_na_bazie() throws SQLException {
        int opcja;
        do {
            System.out.println("1  -  wyswietl tytuly");
            System.out.println("2  -  wyswietl ksiazki");
            System.out.println("3  -  wyszukaj ksiazki danego autora");
            System.out.println("4  -  wstaw tytul");
            System.out.println("5  -  wstaw ksiazke");
            System.out.println("-1 -  koniec programu");
            opcja = WeWy.weInteger("Podaj opcje: ");
            switch (opcja) {
                case 1:
                    System.out.println("Tytuly");
                    ArrayList<String> tytuly = baza.wyswietl_tytuly();
                    wyswietl_kolekcje(tytuly);
                    break;
                case 2:
                    System.out.println("Ksiazki");
                    ArrayList<String> ksiazki = baza.wyswietl_ksiazki();
                    wyswietl_kolekcje(ksiazki);
                    break;
```

11. cd. Przykład programu: Baza_5 – klasa GUI

```
        case 3:
            String co = WeWy.weString("Podaj autora: ");
            ArrayList<String> ksiazki_ = baza.wyszukaj(co);
            wyswietl_kolekcje(ksiazki_);
            break;
        case 4:
            WEWY_tytul wewy_tytul = new WEWY_tytul();
            String[] dane_tytul = wewy_tytul.wstaw_tytul();
            baza.wstaw_tytul(dane_tytul);
            break;
        case 5:
            int numer = WeWy.weInteger("Podaj numer: ");
            String tytul = WeWy.weString("Podaj tytul: ");
            baza.wstaw_ksiazke(tytul, numer);
            break;
        case -1:
            System.out.println("Koniec programu");
            break;
        default:
            System.out.println("Zla opcja");
      }
   } while (opcja != -1);
}
```

11. cd. Przykład programu: Baza_5 – klasa GUI

```
 static public void main(String arg[]) {
        GUI gui = new GUI();
        try {
            gui.polaczenie_z_baza();
            gui.operacje_na_bazie();
        } catch (SQLException e) {
            System.out.println("Blad bazy " + e);
        }
    }
}
```

11. cd. Przykład programu: Baza_5 – klasa WEWY_tytul

```
package GUI;

public class WEWY_tytul {

    public String tytul, autor;
    public int ISBN;

    public String[] wstaw_tytul() {
        tytul = WeWy.weString("Podaj tytul: ");
        autor = WeWy.weString("Podaj autora: ");
        ISBN = WeWy.weInteger("Podaj ISBN: ");
        String []dane= {tytul, autor, String.valueOf(ISBN)};
        return dane;
    }

}
```

11. cd. Przykład programu: Baza_5 – klasa WeWy

```
package GUI;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.StringTokenizer;

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 "";  }
  }
```

11. cd. Przykład programu: Baza_5 – klasa WeWy

```
static public int weInteger(String menu)
   {
     InputStreamReader wejscie = new InputStreamReader( System.in );
     BufferedReader bufor = new BufferedReader( wejscie );
     StringTokenizer zeton;
     try
       {
       System.out.print(menu);
       zeton = new StringTokenizer(bufor.readLine());
       return Integer.parseInt(zeton.nextToken());
        }
      catch (Exception e)
       {
         System.err.println("Blad Integer "+e);
         return 0;
       }
    }
}
```

11. cd. Przykład programu: Baza_5