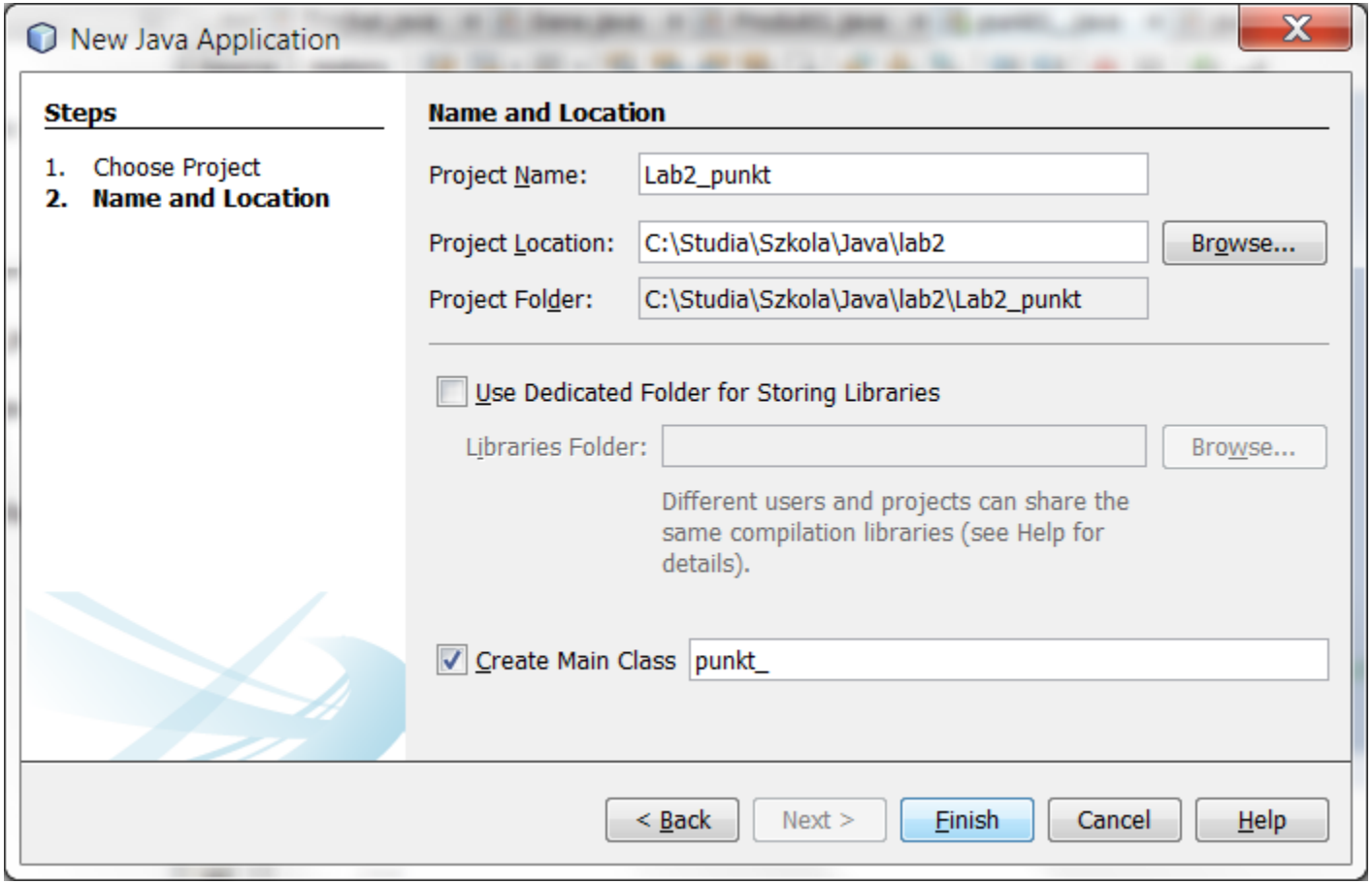
Języki i metody programowania Java
Lab2 – podejście obiektowe

https://docs.oracle.com/javase/tutorial/

http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/pojava/javazk4_2.pdf

Zofia Kruczkiewicz

Zadanie 1. Należy wykonać projekt typu **Java Application** (wg instrukcji z Lab1)- należy wybrać **File/New Project/Java/Java Application** i wpisać nazwę projektu w polu **Project Name** i położenie w polu **Project Location.** Należy w polu **Create Main Class** pozostawić nazwę klasy po usunięciu nazwy pakietu.

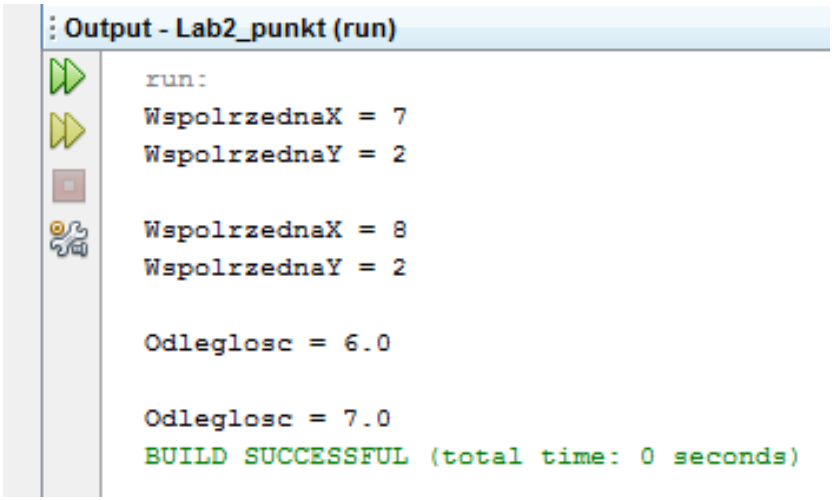


W pliku projektu **punkt_** należy wkleić kod klasy **Punkt1** bez słowa **public** przy słowie **class**

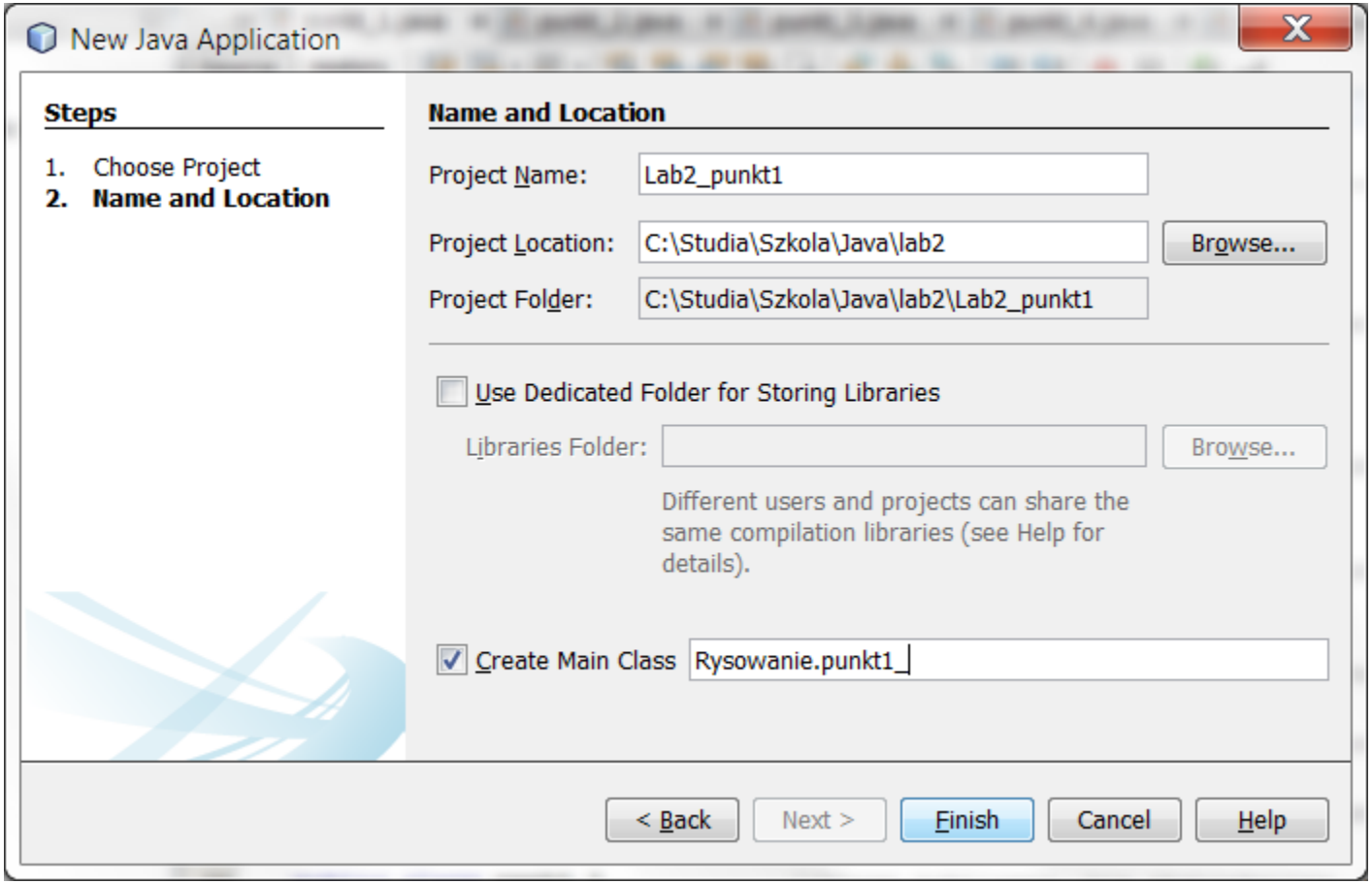
```
class Punkt1 {
    protected int x, y;
    public Punkt1(int wspX, int wspY) {
        x = wspX;
        y = wspY;
    }
    public void zmien(int wspX, int wspY) {
        x = wspX;
        y = wspY;
    }
    public int podajX() {       return x;    }
    public int podajY() {       return y;    }
    public void przesun(int dx, int dy) {
        x += dx;
        y += dy;
    }
    public double odleglosc(Punkt1 p) {
          return Math.sqrt((x - p.x) * (x - p.x) + (y - p.y) * (y - p.y));    }
}
```

oraz klasy głównej **punkt_ (słowo public przy nazwie klasy i obowiązkowa nazwa pliku typu java!).** Należy uruchomić program – wynik działania programu poniżej.

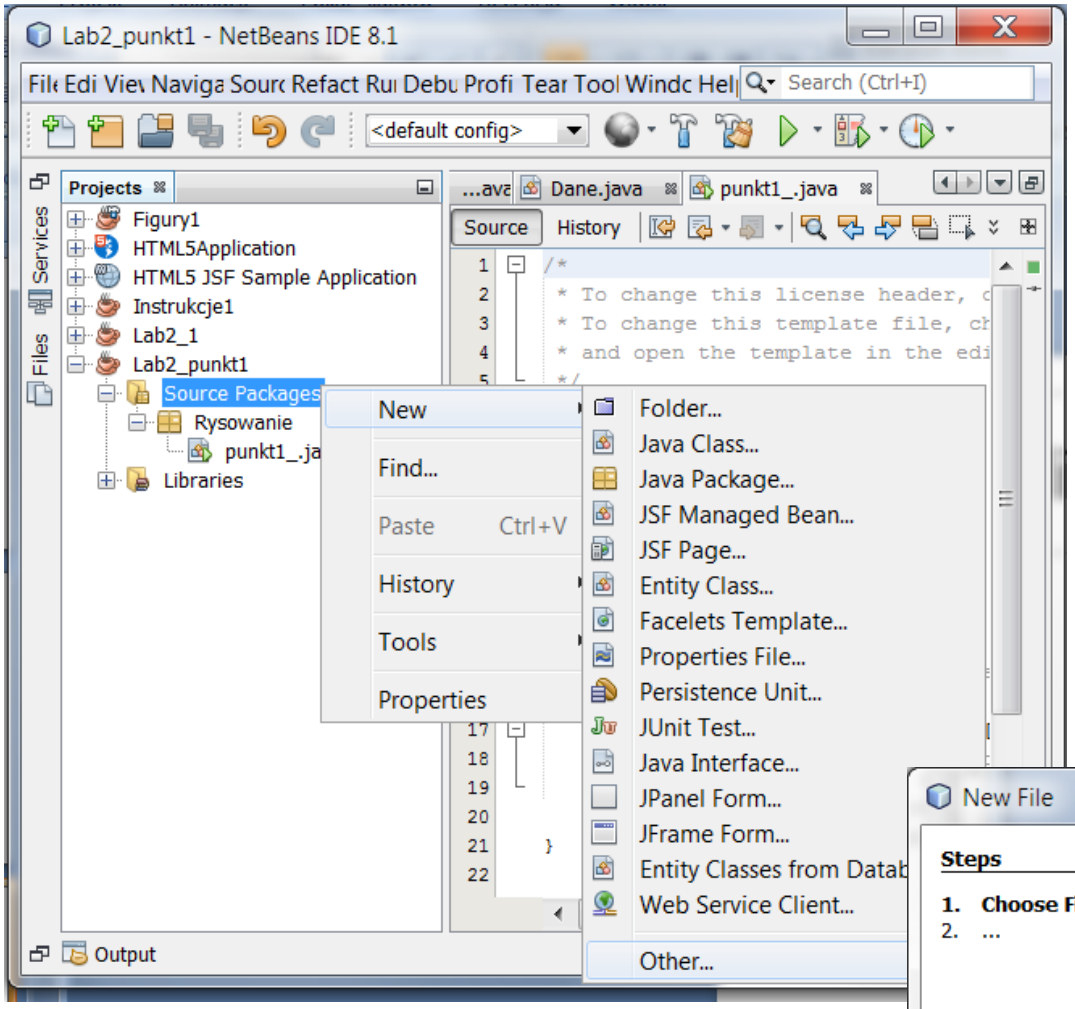
```
public class punkt_ {
    public static void main(String[] args) {
      Punkt1 p1 = new Punkt1(7, 2);
      System.out.println("WspolrzednaX = " + p1.podajX());
      System.out.println("WspolrzednaY = " + p1.podajY());
      Punkt1 p2 = new Punkt1(8, 2);
      System.out.println("\nWspolrzednaX = " + p2.podajX());
      System.out.println("WspolrzednaY = " + p2.podajY());
      p2.zmien(1, 2);
      System.out.println("\nOdleglosc = " + p2.odleglosc(p1));
      p1.przesun(1, 0);
      System.out.println("\nOdleglosc = " + p1.odleglosc(p2));
   }
}
```

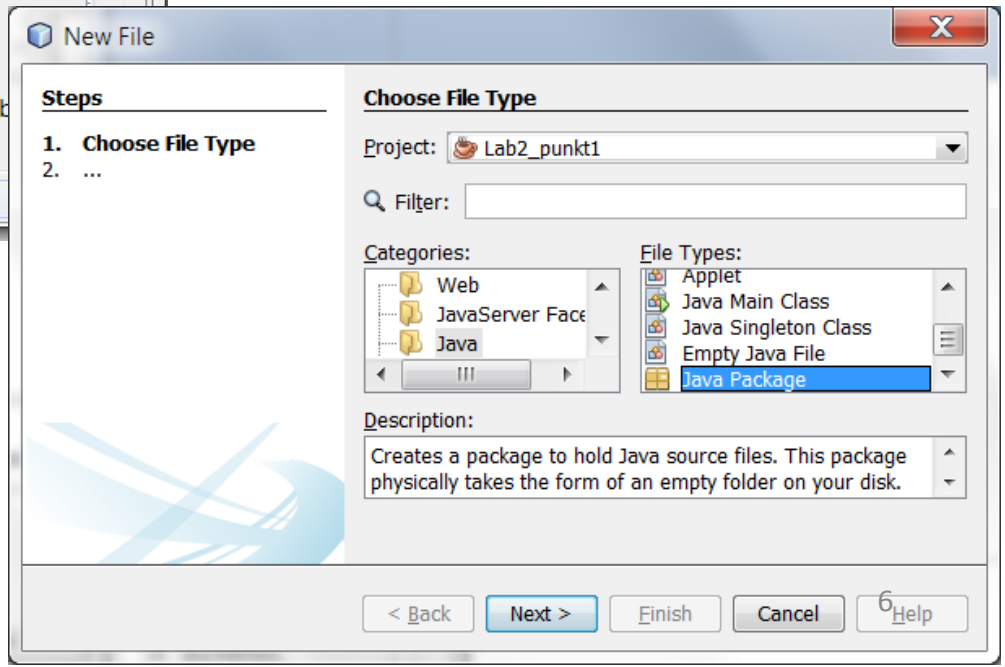


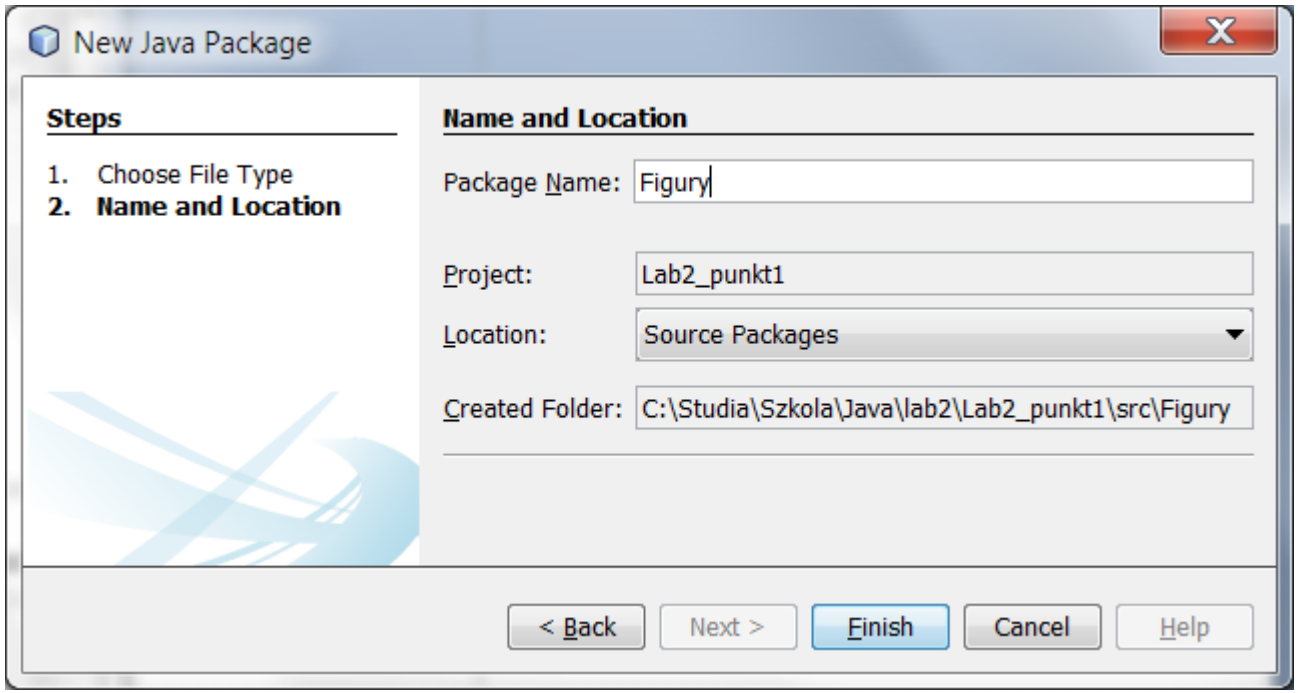
Należy wykonać kolejny nowy projekt typu **Java Application** (wg instrukcji z Lab1)- należy wybrać **File/New Project/Java/Java Application** i wpisać nazwę projektu w polu **Project Name** i położenie w polu **Project Location.** Należy w polu **Create Main Class** podać nazwę pakietu (tutaj **Rysowanie**) i nazwę klasy głównej (tutaj **punkt1_**)



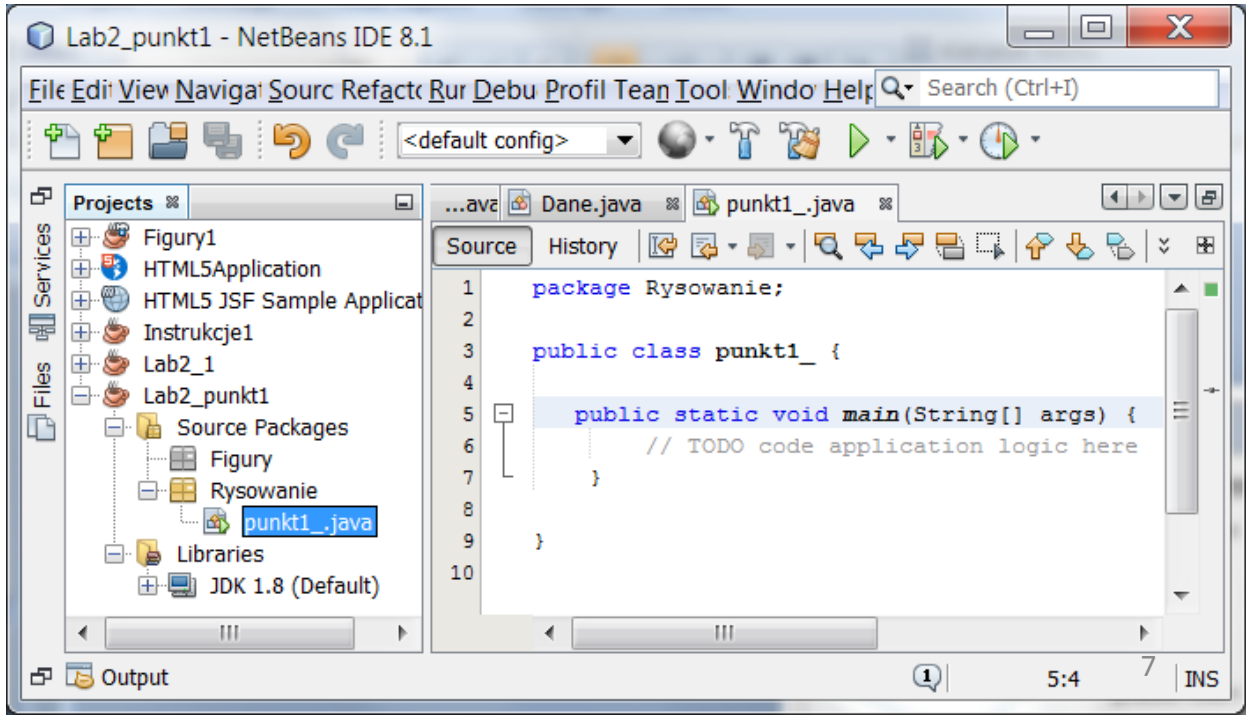
Należy wykonać nowy pakiet w utworzonym projekcie: po kliknięciu prawym klawiszem na nazwę projektu w oknie **Projects** wybrać : **New/Other/Java/Java Packages/Next**

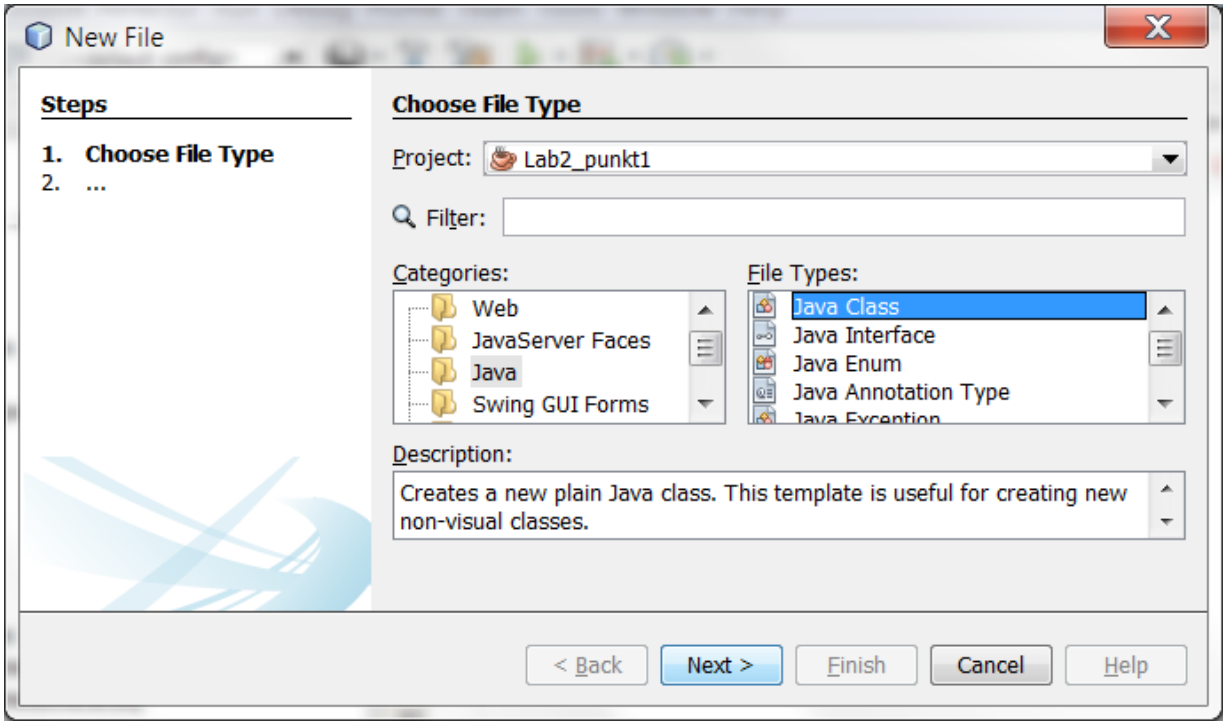


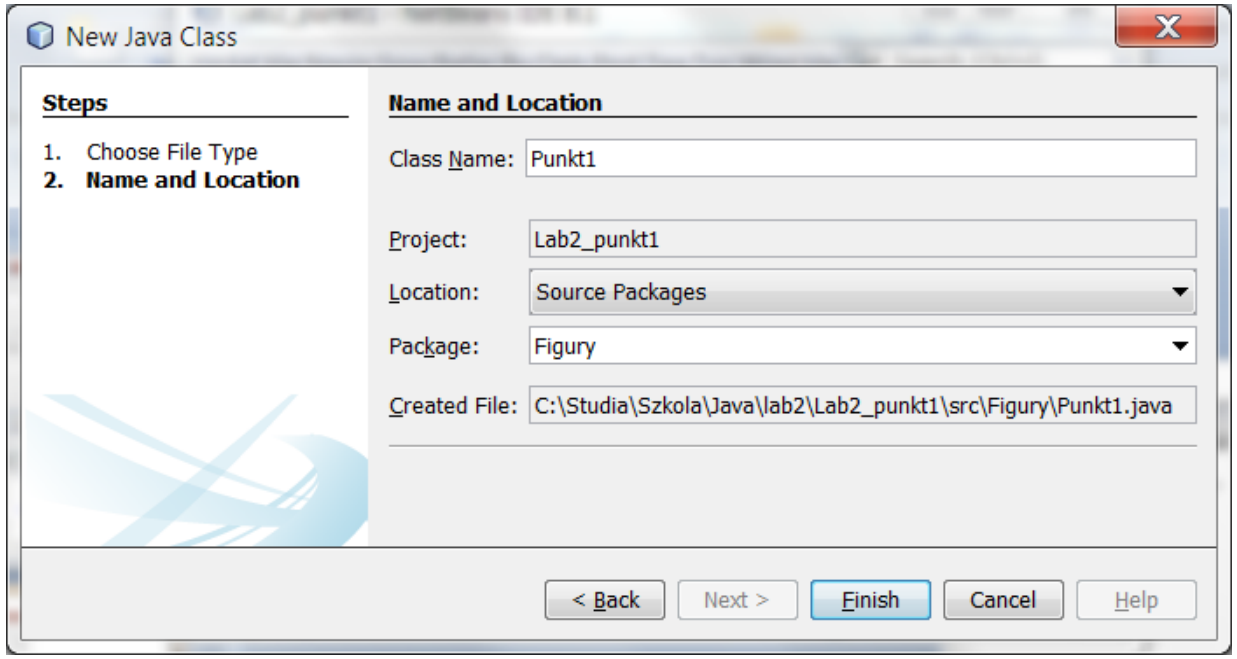




i nadać nazwę nowemu pakietowi: w polu **Package Name.** Poniżej pokazano wynik dotychczasowych działań.

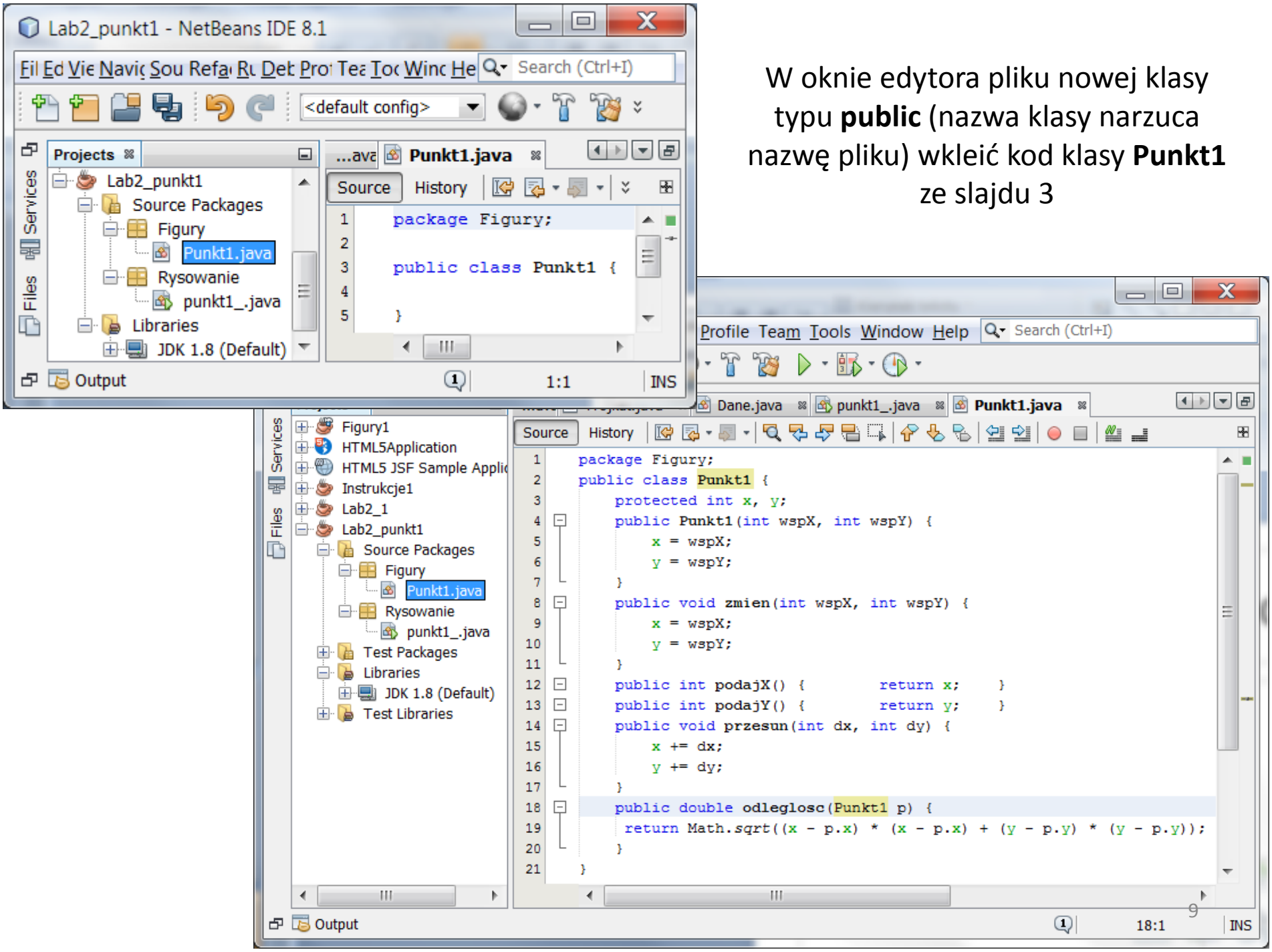




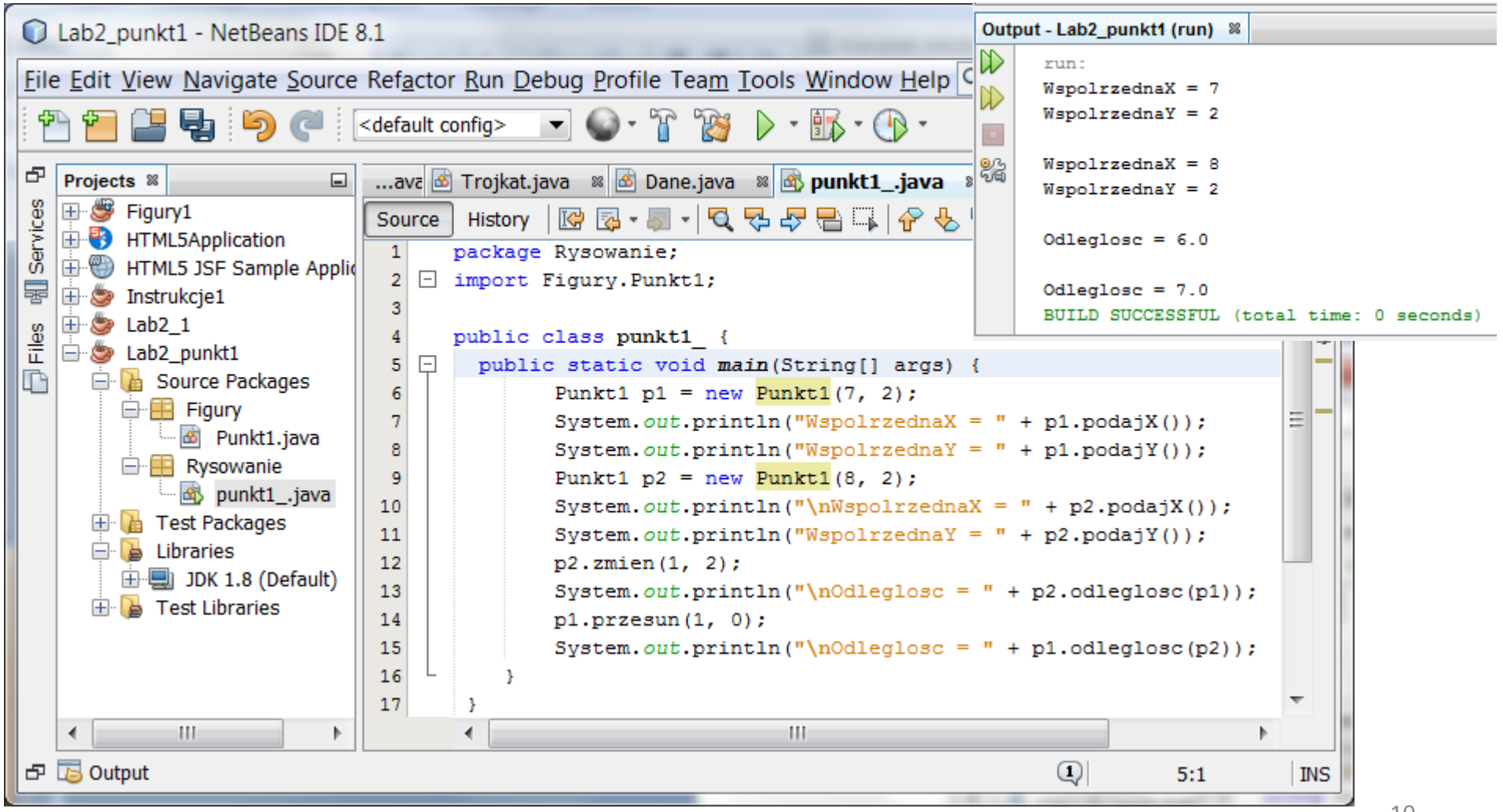


W nowym pakiecie **Figury należy** dodać nową klasę **Punkt1** w wyniku następujących działań: po kliknięciu w oknie **Projects** na nazwę pakietu **Figury**, wybrać: **New/Other/Java/Java Class /Next** i w polu **Class Name** wpisać nazwę klasy w pakiecie **Figury** (tutaj **Punkt1**), wybranym wcześniej. Możliwość wybrania innego pakietu istnieje w polu **Package**. Poniżej z lewej strony pokazano wynik działań

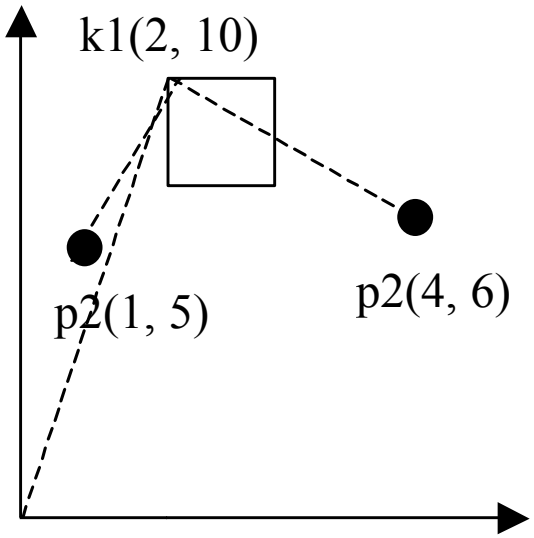
W oknie edytora pliku nowej klasy typu **public** (nazwa klasy narzuca nazwę pliku) wkleić kod klasy **Punkt1** ze slajdu 3



Podobnie, należy wkleić kod klasy **punkt_** do klasy **punkt1_**, utworzonej wcześnie (slajd 4). Należy usunąć błędy kompilacji wynikające z braku importu kodu klasy **Punkt1** (kliknąć na powierzchnię klasy **punkt1_** i wybrać z listy pozycję **Fix Imports**). Na koniec należy uruchomić program. Wynik podano poniżej, z prawej strony..



Zadanie 2 – dziedziczenie metod w klasie pochodnej – część 1. Przedstawiony program zawiera klasy: **Punkt**, pochodną klasę typu **Kwadrat** oraz klasę **punkt_,** która pozwala uruchomić program. **Należy wykonać podobny program, w którym jest klasa Punkt1, a klasą pochodną powinien być Prostokąt - w tym celu należy wykonać kopię programu Lab2_punkt1** jako **Lab2_punkt2** (kliknąć na nazwę projektu w oknie **Projects**, wybrać pozycję **Copy…** i wpisać w polu **Project Name** nową nazwę projektu i położenie w polu **Project Location**). Klasa **Punkt1** może pozostać bez zmian. Na kolejnych slajdach podano informację, co należy zmienić w programie, aby wykonać zadanie 3.



DZIEDZICZENIE
PRZECIĄŻANIE METOD, PRZEDEFINIOWANIE METOD

1) Dziedziczenie
Przykład 1 – dziedziczenie metod w klasie pochodnej

```
public class Punkt
{  protected int x, y;

  public Punkt(int wspX, int wspY)
     {   x = wspX;      y = wspY;}

  public void zmien(int wspX, int wspY)
     {   x = wspX;      y = wspY;}

  public int podajX()
     { return x;  }

public  int podajY()
     { return y; }

  public void przesun(int dx, int dy)
     {   x+=dx;   y+=dy;  }

  public  double odlegloscl(Punkt p)
     {return Math.sqrt((x-p.x)*(x-p.x)+(y-p.y)*(y-p.y));   }
}
```

Klasa **Punkt** odpowiada klasie **Punkt1** w pakiecie **Figury**.

```
public class Kwadrat extends Punkt          //klasa dziedziczy od klasy Punkt
 {
    protected int dlugosc;

  public Kwadrat(int wspX, int wspY, int dlugosc_)
   {
     super(wspX,wspY);                      //wywołanie dziedziczonego konstruktora
    dlugosc=dlugosc_;
   }          //ponieważ nie ma konstruktora domyślnego (jawnego lub niejawnego konstruktora bez parametrów) w klasie
                                  // bazowej Punkt-konstruktora domyślnego się nie wywołuje
  public int podajDl()
    {
      return dlugosc;
    }

  public int pole()
    {
      return dlugosc*dlugosc;
    }
}
```

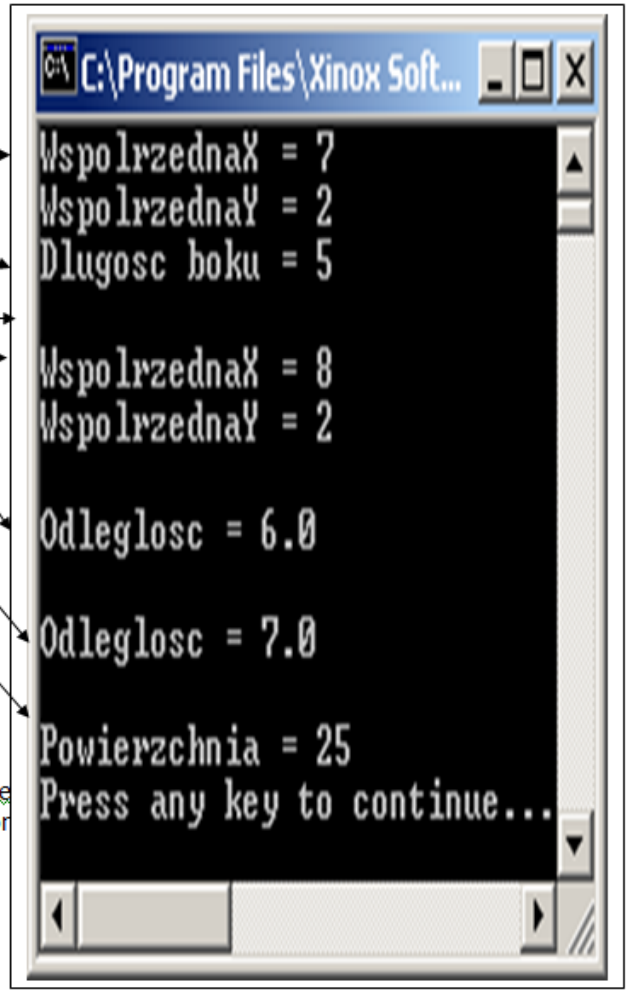
Należy dodać klasę **Prostokat** zamiast klasy **Kwadrat** w pakiecie **Figury**, podobnie jak podano na slajdzie 29. Należy odpowiednio zmienić nazwę klasy **Punkt** na **Punkt1** i uzupełnić definicję klasy **Prostokat** opierając się na definicji klasy **Kwadrat** dodając atrybut **wysokosc**, zmodyfikowany konstruktor (dodatkowy parametr w liście parametrów konstruktora podający wartość wysokości prostokąta oraz w ciele konstruktora dodatkowe przypisanie wartości tego parametru do atrybutu **wysokosc**), metodę **podajWysokosc** oraz zmienić definicję metody pole na **dlugosc*wysokosc**

Program główny przetwarzający obiekty typu Punkt1 i Kwadrat.

Należy w klasie **punkt1_** w pakiecie **Rysowanie**, wpisać kod podanego programu, zmieniając nazwę klasy **Kwadrat** na **Prostokat** i **Punkt** na **Punkt1** i dodając np. polecenie
System.out.println(„Wyskosc prostokata= " + p1.podajWysokosc());

```
public class punkt_
{
  public static void main (String[] args)
  {
    Kwadrat k1 = new Kwadrat(7,2,5);
    System.out.println("WspolrzednaX = "+ k1.podajX());
    System.out.println("WspolrzednaY = "+ k1.podajY());
     System.out.println("Dlugosc boku = "+ k1.podajDl());
     Punkt p2 = new Punkt(8,2);
     System.out.println("\nWspolrzednaX = "+ p2.podajX());
     System.out.println("WspolrzednaY = "+ p2.podajY());
     p2.zmien(1,2);
     System.out.println("\nOdleglosc = "+ p2.odleglosc(k1));
     k1.przesun(1,0);
     System.out.println("\nOdleglosc = "+ k1.odleglosc(p2));
     System.out.println("\nPowierzchnia = "+ k1.pole());
 }
}
```

Te metody są wywołane dzięki dziedziczeniu składowych od klasy Punkt



Obiekt klasy *Kwadrat* posiada metody klasy Punkt i swoje własne (podajDl(), pole() oraz konstruktor Kwadrat(**int, int, int**). W programie w konstruktorze klasy Kwadrat polecenie **super** pozwala wywołać konstrutor dziedziczonej klasy *Punkt*. Konstruktory domyślne nie trzeba wywoływać (są wywoływane niejawnie), pozostałe konstruktory, jeśli nie ma domyślnych, trze zawsze wywołać za pomocą słowa **super**, jako pierwsze wywołanie w konstruktor klasy pochodnej.

Wywołanie p2.odleglosc(k1) zawiera parametr aktualny metody jako referencję do obiektu klasy pochodnej Kwadrat, która może być podstawiona za parametr typu klasy bazowej Punkt.

Zadanie 3 – dziedziczenie metod w klasie pochodnej część 2. Przedstawiony program zawiera klasy: **Punkt**, pochodną klasę typu **Kwadrat** oraz klasę **punkt_,** która pozwala uruchomić program. **Należy wykonać podobny program, w którym jest klasa Punkt1, a klasami pochodnymi powinna być nadal klasa Kwadrat i dodatkowo klasa Prostokat, dziedzicząca od klasy Kwadrat - w tym celu należy wykonać kopię programu Lab2_punkt2** jako **Lab2_punkt3 z zadania 2 .** Na kolejnych slajdach podano informację, co należy zmienić w programie, aby wykonać zadanie 3.

DZIEDZICZENIE
PRZECIĄŻANIE METOD, PRZEDEFINIOWANIE METOD

1) Dziedziczenie
Przykład 1 – dziedziczenie metod w klasie pochodnej

```
public class Punkt
{  protected int x, y;

   public Punkt(int wspX, int wspY)
      {   x = wspX;      y = wspY;}

   public void zmien(int wspX, int wspY)
      {   x = wspX;      y = wspY;}

   public int podajX()
      { return x;  }

   public  int podajY()
      { return y; }

   public void przesun(int dx, int dy)
      {   x+=dx;   y+=dy;  }

   public  double odlegloscPunkt p)
      {return Math.sqrt((x-p.x)*(x-p.x)+(y-p.y)*(y-p.y));  }
}
```

Klasa **Punkt** odpowiada klasie **Punkt1** w pakiecie **Figury**, należy jednak dodać dwie metody:
1) metodę **pole**, która zwraca wartość 0:

```
public int pole ()
 { return 0;}
```

2) metodę **odleglosc**, która jest metodą przeciążoną i mierzy odległość punktu od początku środka współrzędnych (slajd 11)

```
public double odleglosc()
                //przeciążenie nazwy metody dziedziczonej
  {  return Math.sqrt(x*x+y*y);   }
```

```
public class Kwadrat extends Punkt          //klasa dziedziczy od klasy Punkt
 {
    protected int dlugosc;

  public Kwadrat(int wspX, int wspY, int dlugosc_)
   {
     super(wspX,wspY);                      //wywołanie dziedziczonego konstruktora
    dlugosc=dlugosc_;
   }          //ponieważ nie ma konstruktora domyślnego (jawnego lub niejawnego konstruktora bez parametrów) w klasie
                                  // bazowej Punkt-konstruktora domyślnego się nie wywołuje
  public int podajDl()
    {
      return dlugosc;
    }

  public int pole()
    {
      return dlugosc*dlugosc;
    }
}
```

1) Należy do pakietu **Figury** dodać klasę **Kwadrat** o kodzie podanym na slajdzie. Należy odpowiednio zmienić nazwę klasy **Punkt** na **Punkt1 .**

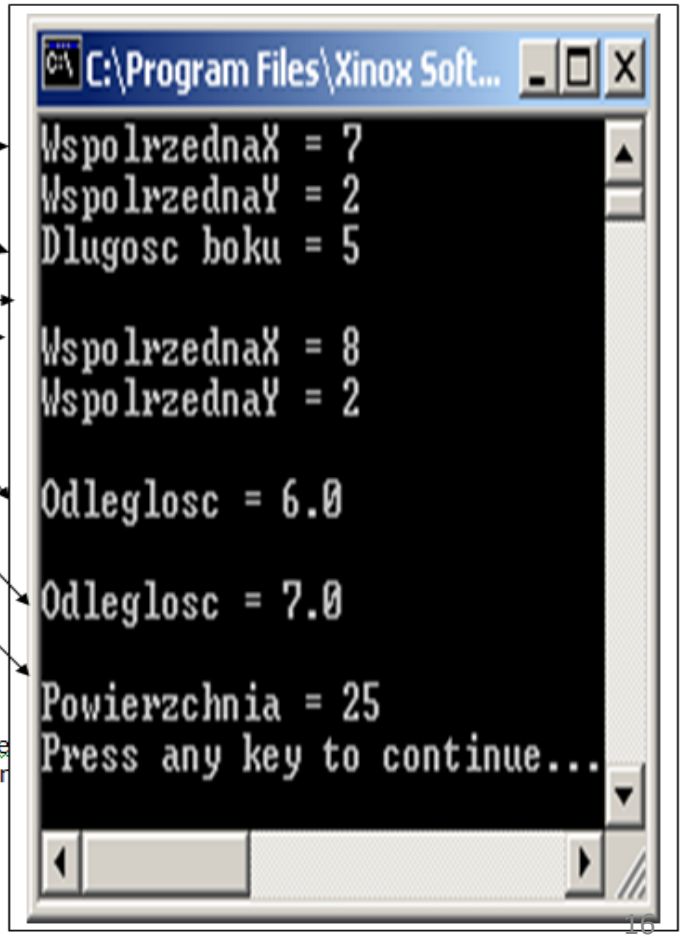
2) Należy dodać klasę **Prostokat** w pakiecie **Figury.** Klasa ta powinna dziedziczyć od klasy **Kwadrat.** Należy w definicji klasy **Prostokat** opierać się na definicji klasy **Kwadrat** dodając atrybut **wyskosc**, zmodyfikowany konstruktor (dodatkowy parametr w liście parametrów konstruktora podający wartość wysokości prostokąta oraz w ciele konstruktora dodatkowe przypisanie wartości tego parametru do atrybutu **wysokosc**), metodę **podajWysokosc** oraz przedefiniować definicję metody **pole** na **dlugosc*wysokosc.** Pozostałe metody powinny być dziedziczone w klasie **Prostokat** od klasy **Kwadrat.**

Program główny przetwarzający obiekty typu **Punkt i Kwadrat**, który należy zmodyfikować.

Należy w klasie **punkt1_** w pakiecie **Rysowanie**, wpisać kod podanego programu, dodając wywołanie nowych metod klasy **Punkt1,** oraz metod klasy **Prostokat** i dodając np. polecenie:
System.out.println(„Wysokosc prostokata= " + p1.podajWysokosc()) lub wywołanie dziedziczonej metody odleglosc(), przeciążonej w klasie **Punkt1**. Należy zmienić nazwę **Punkt** na **Punkt1.**

```
public class punkt_
{
  public static void main (String[] args)
  {
    Kwadrat k1 = new Kwadrat(7,2,5);
    System.out.println("WspolrzednaX = "+ k1.podajX());
    System.out.println("WspolrzednaY = "+ k1.podajY());
     System.out.println("Dlugosc boku = "+ k1.podajDl());
     Punkt p2 = new Punkt(8,2);
     System.out.println("\nWspolrzednaX = "+ p2.podajX());
     System.out.println("WspolrzednaY = "+ p2.podajY());
     p2.zmien(1,2);
     System.out.println("\nOdleglosc = "+ p2.odleglosc(k1));
     k1.przesun(1,0);
     System.out.println("\nOdleglosc = "+ k1.odleglosc(p2));
     System.out.println("\nPowierzchnia = "+ k1.pole());
 }
}
```

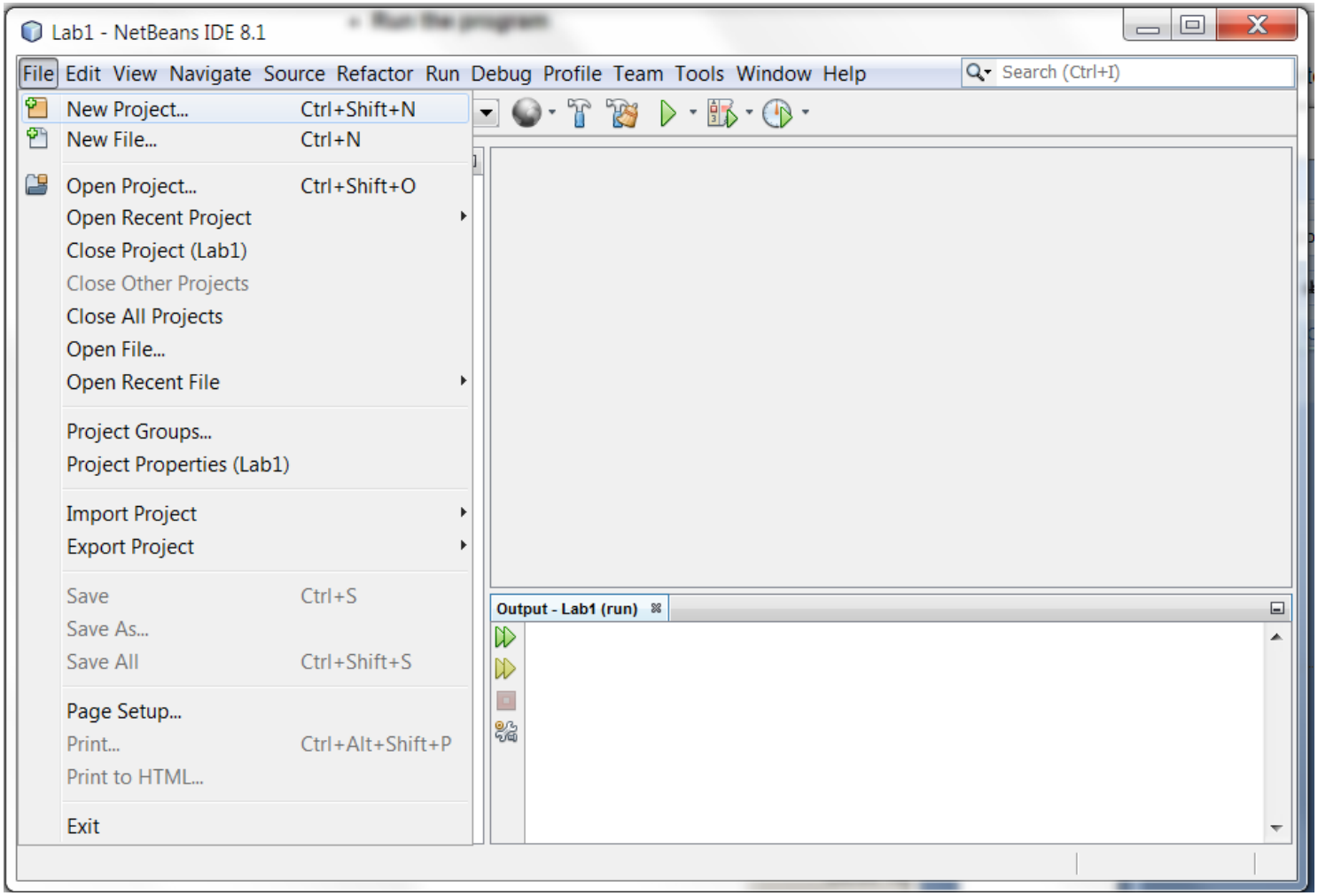
Te metody są wywołane dzięki dziedziczeniu składowych od klasy Punkt



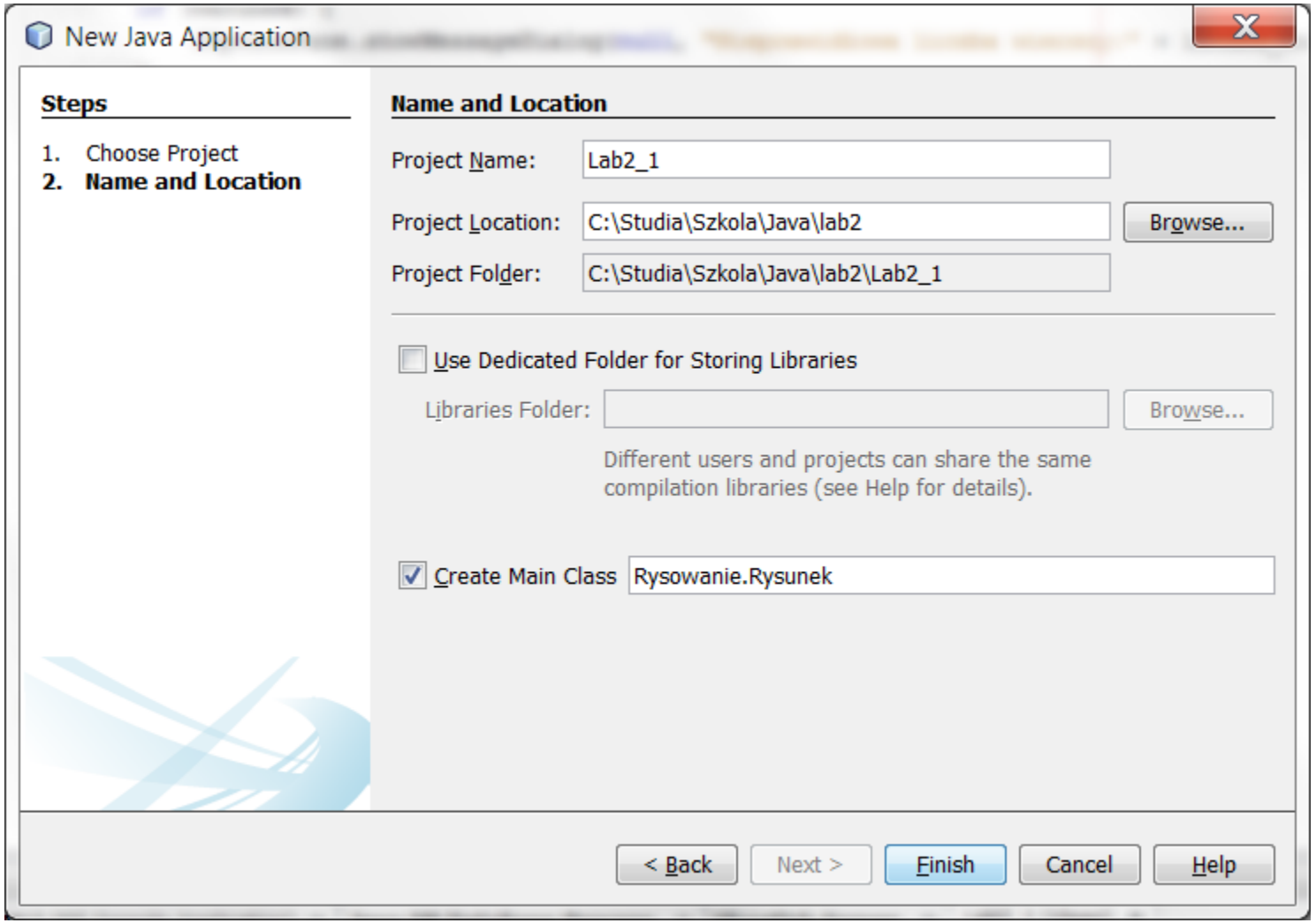
Obiekt klasy *Kwadrat* posiada metody klasy Punkt i swoje własne (podajDl(), pole() oraz konstruktor Kwadrat(**int, int, int**). W programie w konstruktorze klasy Kwadrat polecenie **super** pozwala wywołać konstrutor dziedziczonej klasy *Punkt*. Konstruktory domyślne nie trzeba wywoływać (są wywoływane niejawnie), pozostałe konstruktory, jeśli nie ma domyślnych, trze zawsze wywołać za pomocą słowa **super**, jako pierwsze wywołanie w konstruktor klasy pochodnej.

Wywołanie p2.odleglosc(k1) zawiera parametr aktualny metody jako referencję do obiektu klasy pochodnej Kwadrat, która może być podstawiona za parametr typu klasy bazowej Punkt.

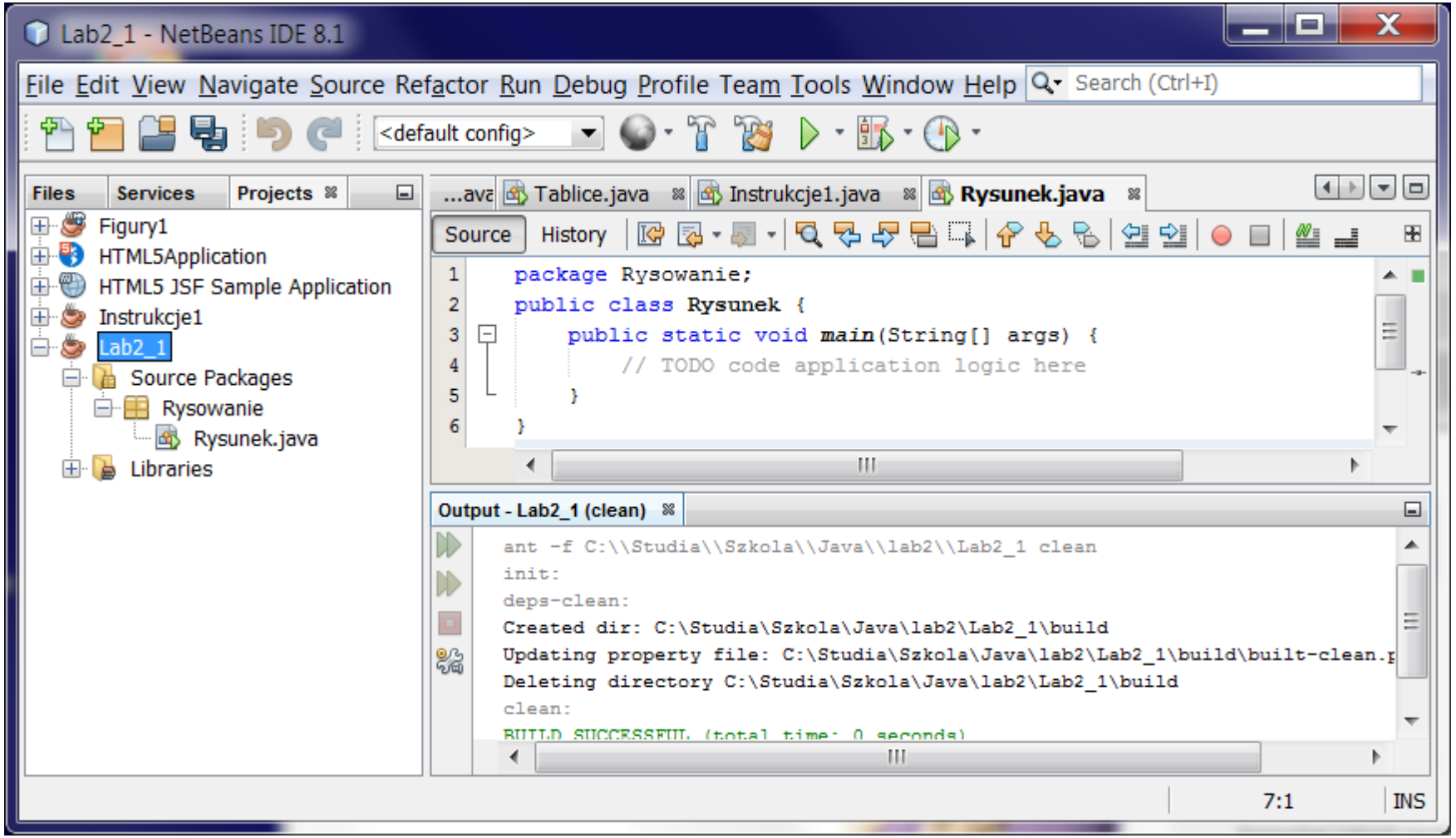
Zadanie 4.1. Należy dokonać analizy prezentowanego programu, który wykonuje czynności programu z zad2. z lab1. Został podzielony na trzy klasy- dwie z nich wykonują swoje działania niezależnie tzn. oddzielono wprowadzanie danych od tworzenie szablonu rysunku. Trzecia klasa integruje wynik wprowadzania danych z utworzeniem szablonu rysunku i rysuje na ekranie w trybie graficznym i konsolowym trójkąt. **Należy podobnie zmodyfikować własny program wykonany jako zad 2 podczas lab1, rysujący prostokąt lub kwadrat.**



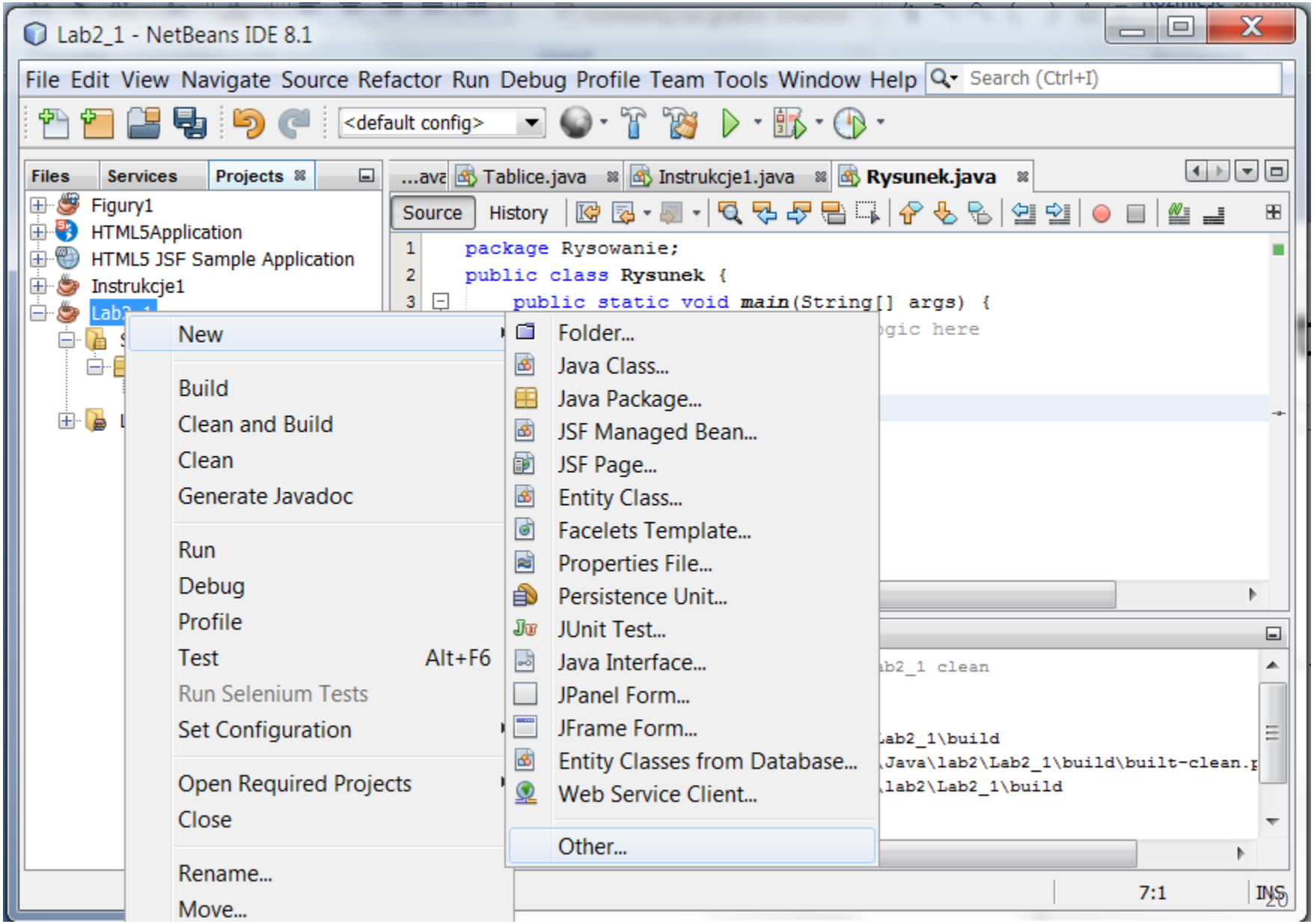
Wykonanie nowego projektu: w formularzu **New Project** należy wybrać **Java/Java Application** i następnie kliknąć na **Next**. Należy podać nazwę projektu (Project Name), położenia projektu (pole **Project Location**) oraz utworzyć pakiet **Rysowanie** oraz klasę główną **Rysunek** w tym pakiecie (pole **Create Main Class**)



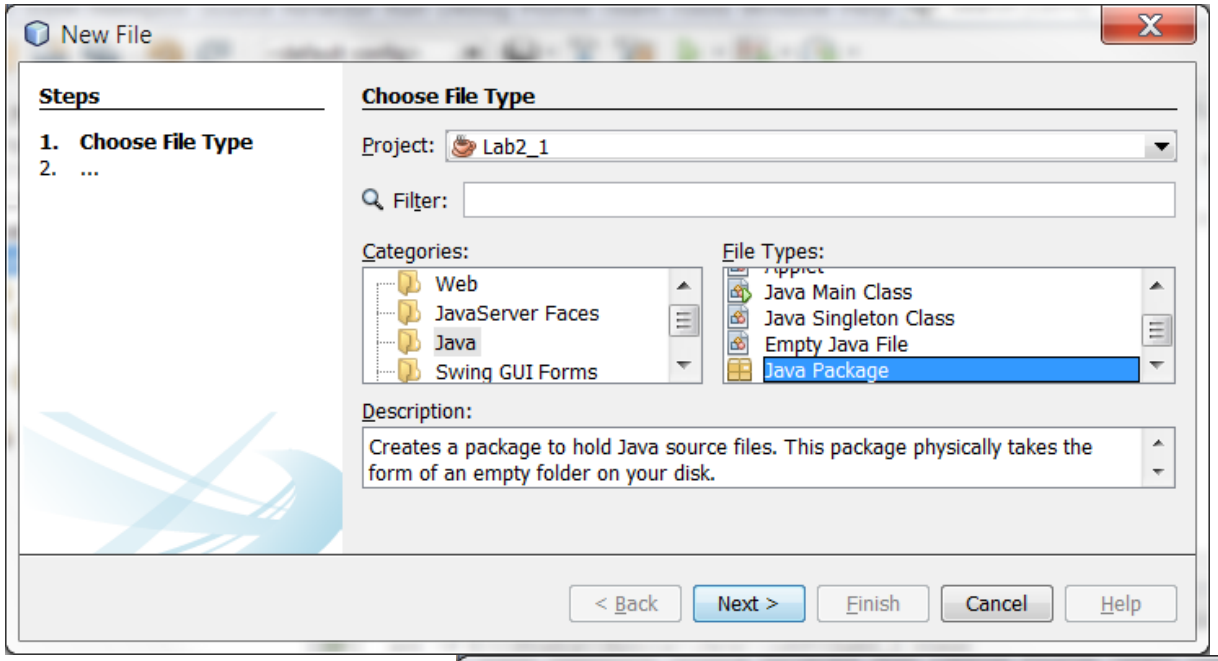
Rezultat – utworzony nowy projekt wraz z klasą **Rysunek** w pakiecie **Rysowanie**. Klasa będzie jednocześnie programem (funkcja main), odpowiedzialnym za narysowanie trójkąta.



Wykonanie kolejnego pakietu **Wprowadzanie_danych** do przechowania klasy typu **Dane**, służącej do wprowadzania danych

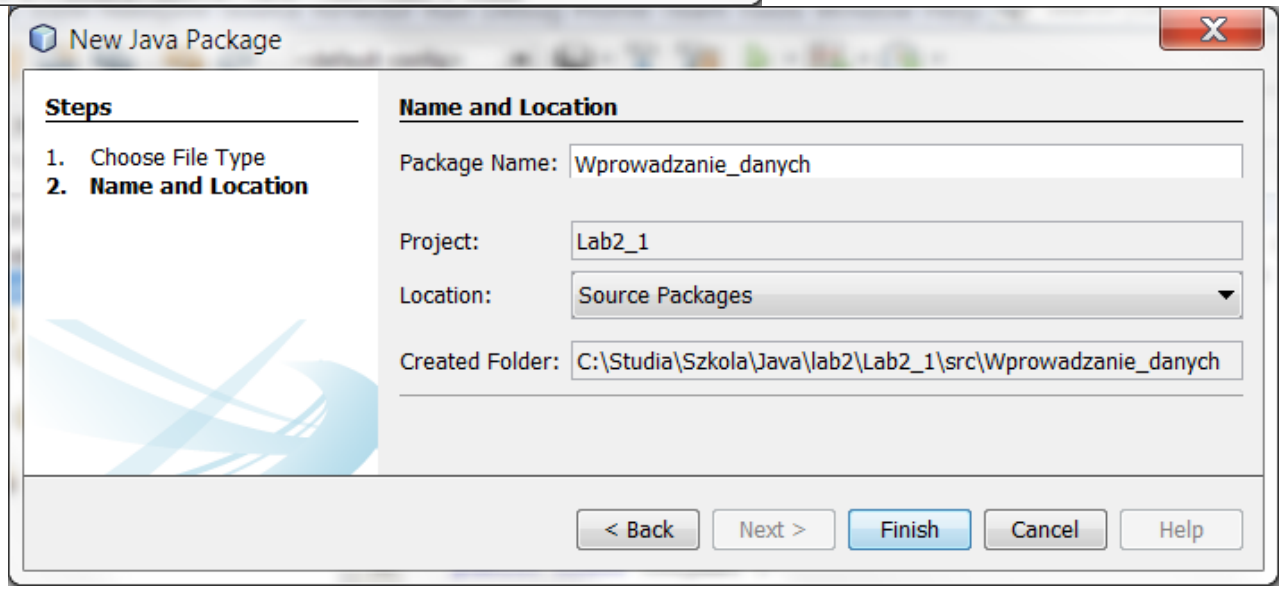


Wykonanie kolejnego pakietu **Wprowadzanie_danych** do przechowania klasy typu **Dane**, służącej do wprowadzania danych (kliknięcie na projekt **New/Other**)

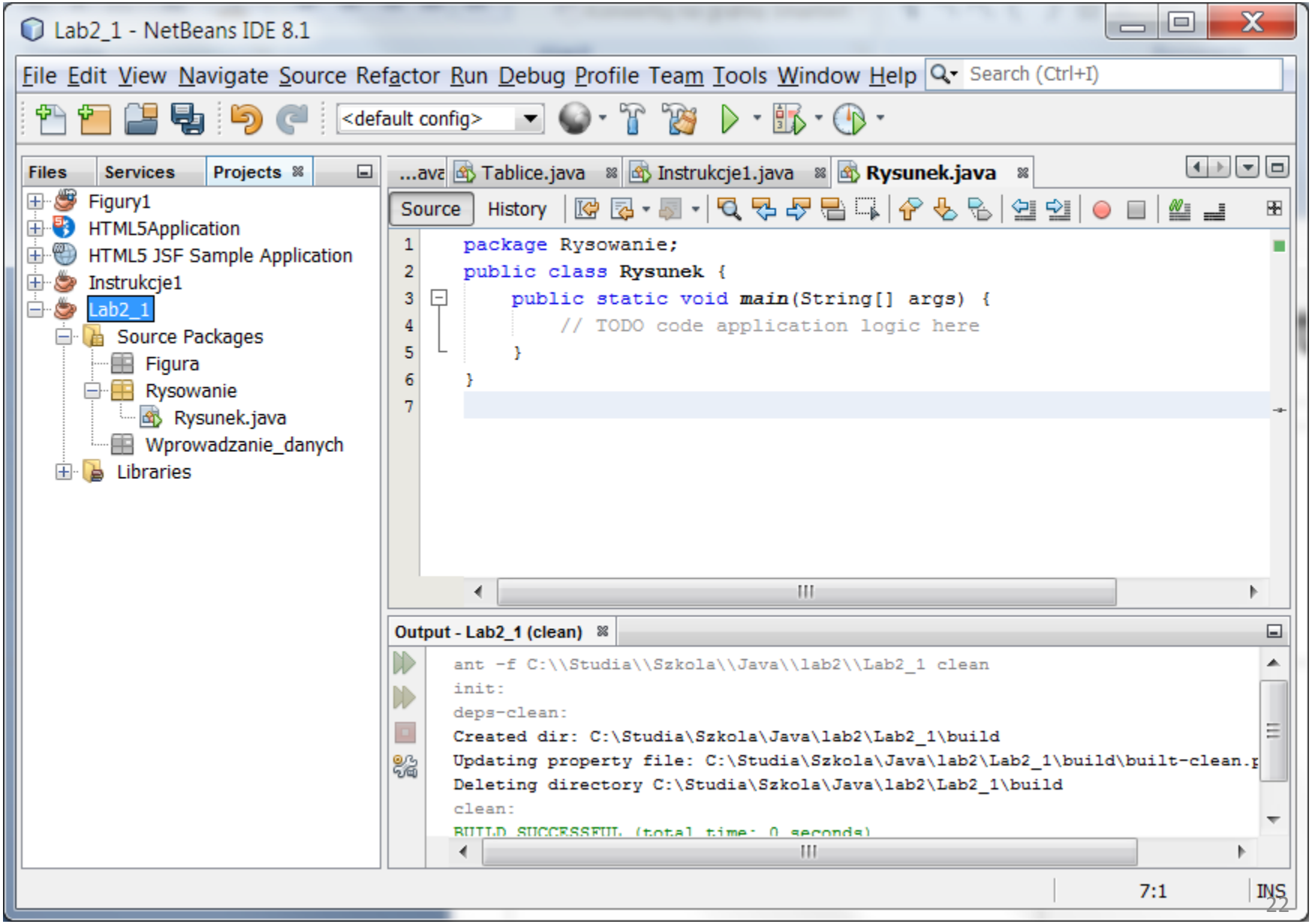


Wybór **Java/Java Package/ Next**

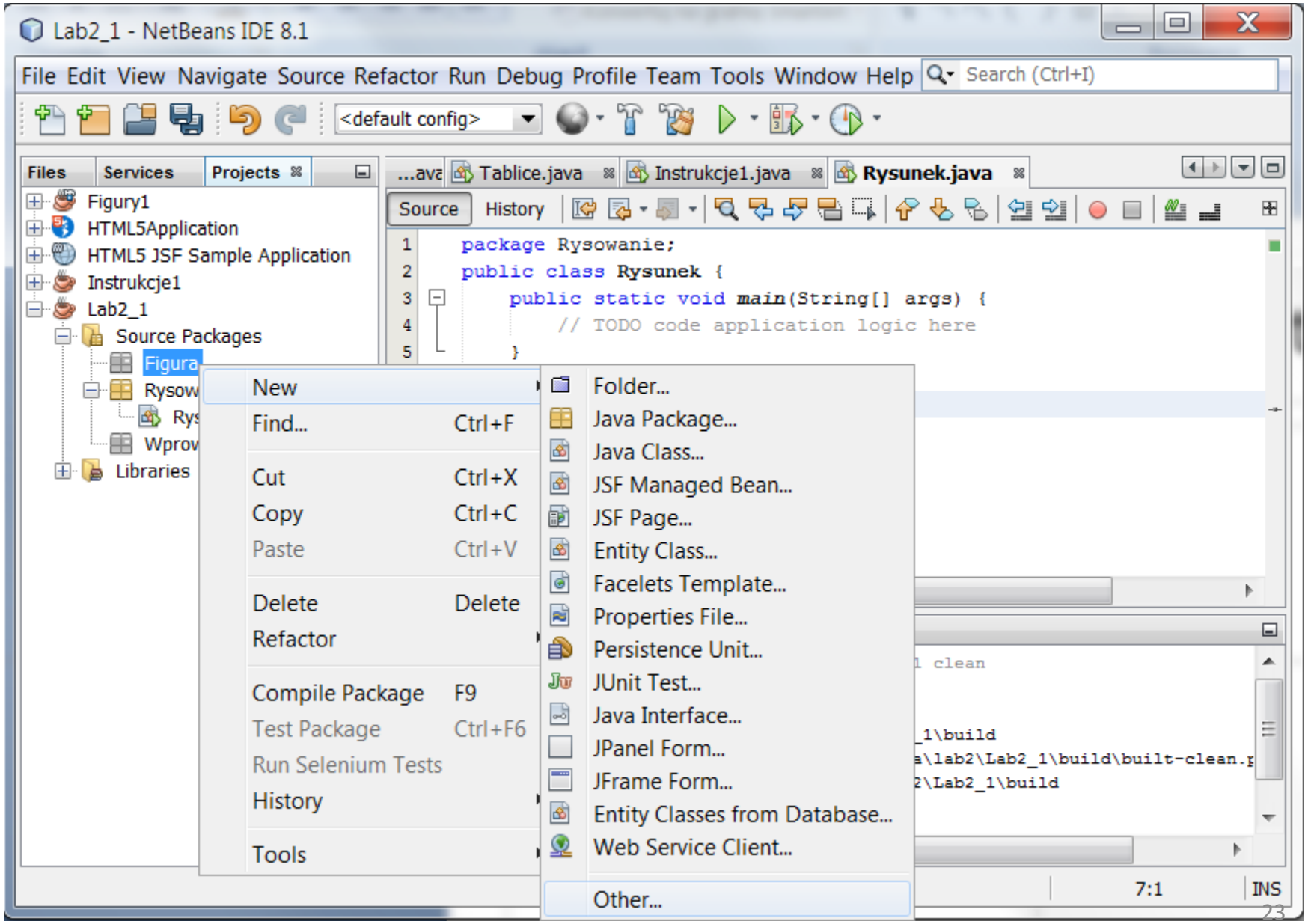
Utworzenie nowego pakietu o nazwie podanej w polu **Package Name**

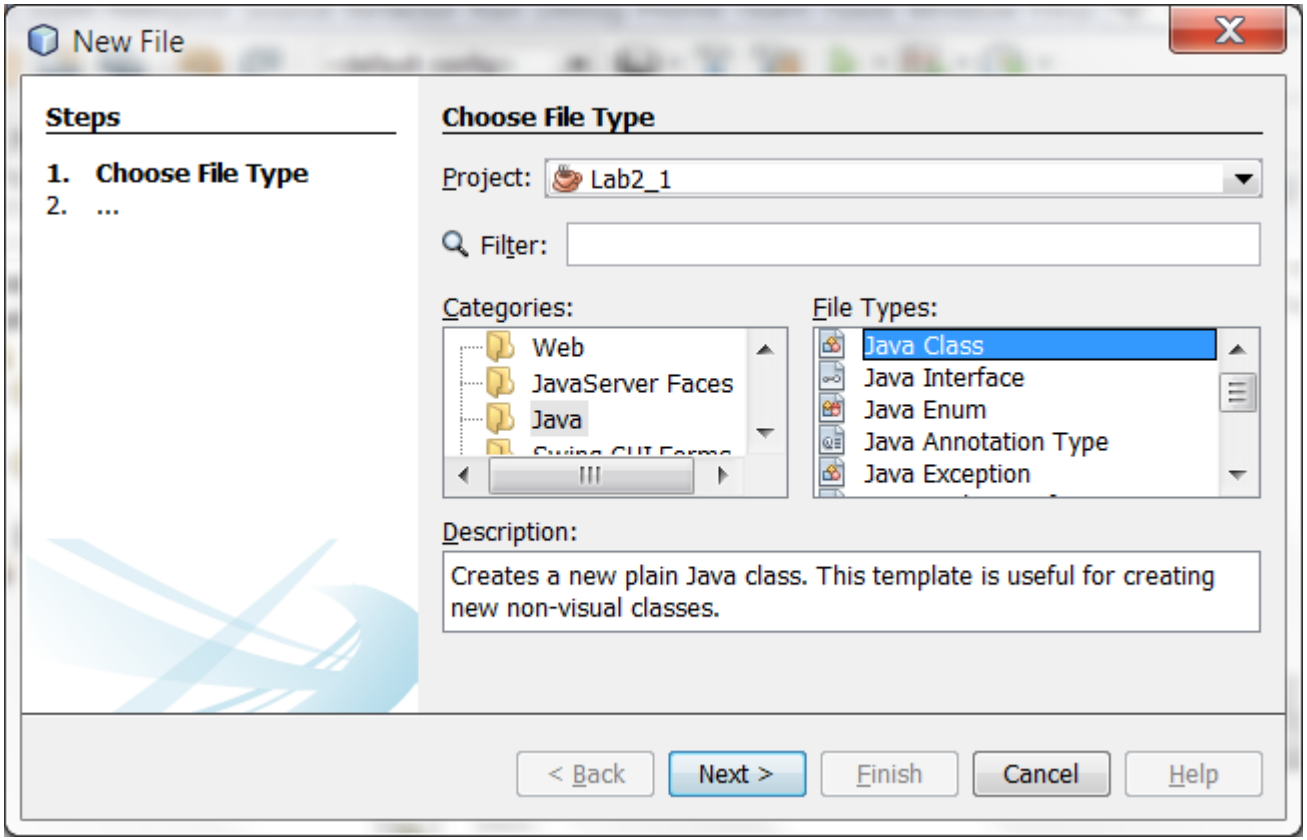


Utworzenie w podobny sposób pakietu **Figura**



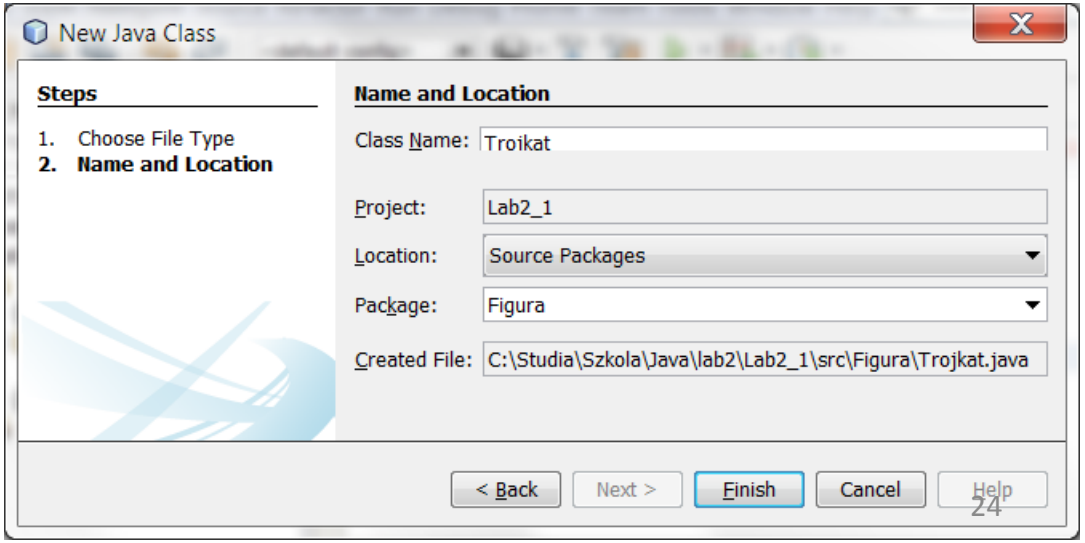
Dodanie klasy w pakiecie **Figura**



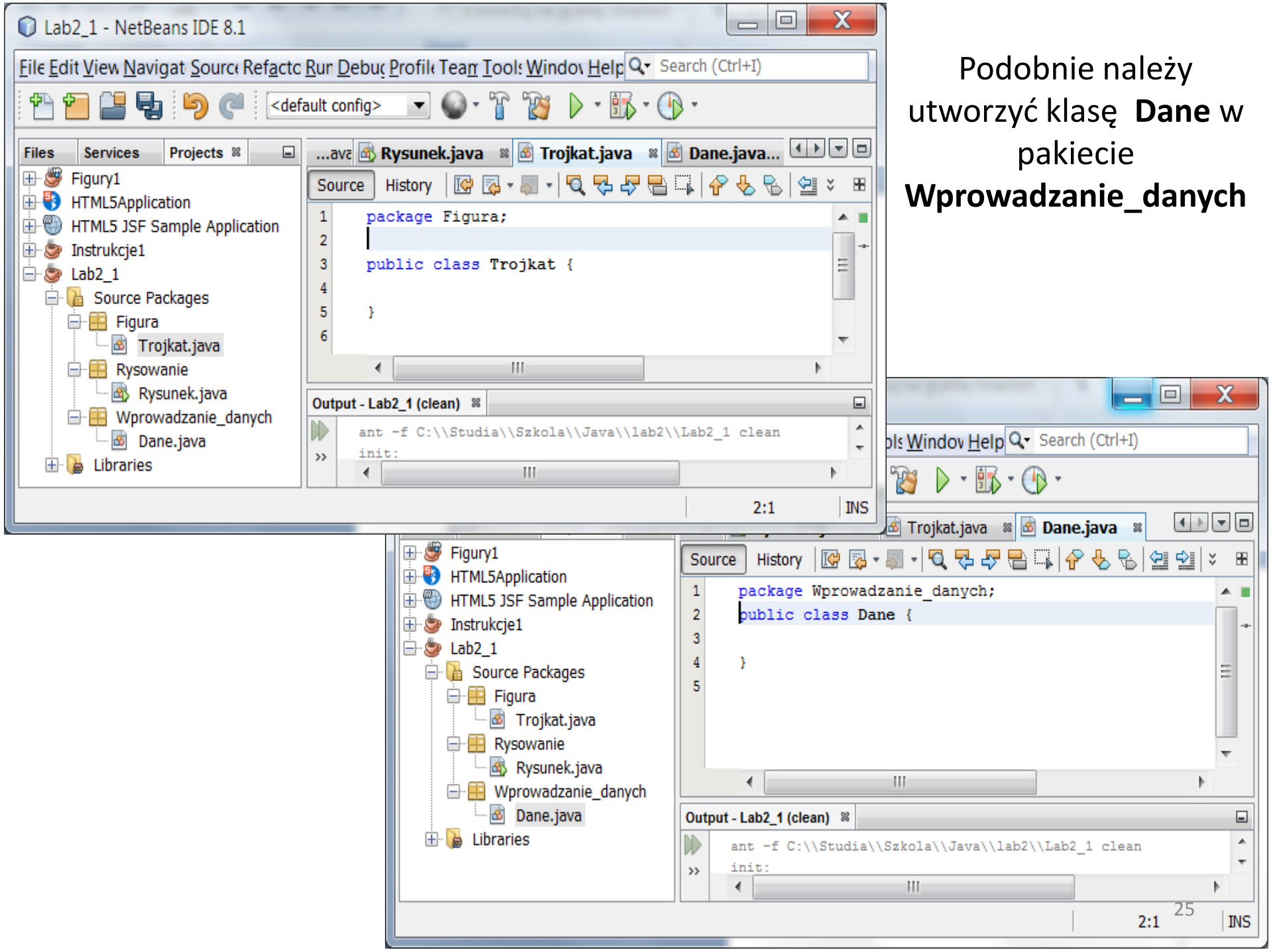


Wybór pozycji
Java/Java Class

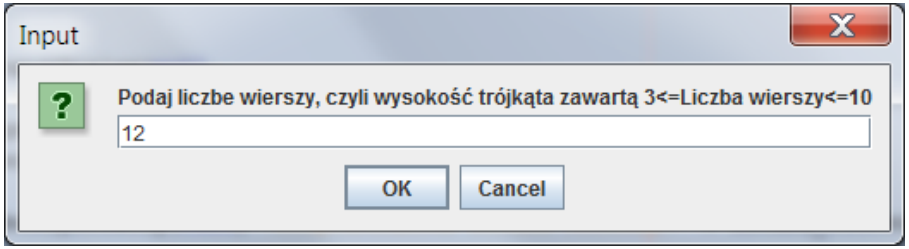
Nadanie nazwy
nowej klasie **Trojkat**
w polu **Class Name**

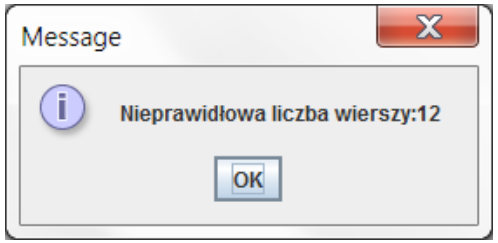


Podobnie należy utworzyć klasę **Dane** w pakiecie **Wprowadzanie_danych**

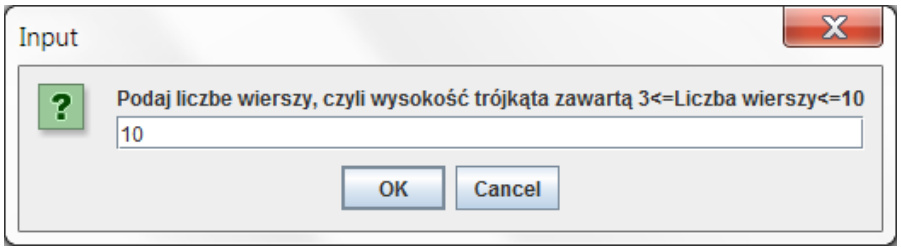


Na podstawie zadania 2 z lab1 należy podzielić logicznie kod na trzy części, czyli trzy klasy:
1) 1-a część: wprowadzanie danych liczba_wierszy oraz wypełnienie s – klasa Dane

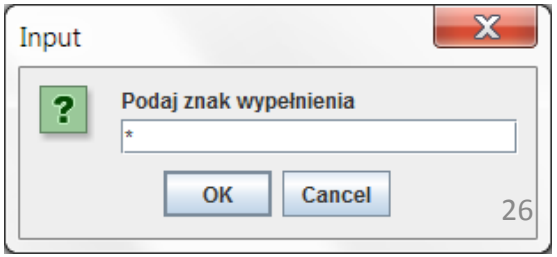





```
package instrukcje1;
import javax.swing.JOptionPane;

public class Instrukcje1 {
public static void main(String[] args) {
//wprowadzanie danych
      int i, j, liczba_wierszy, liczba_spacji, liczba_znakow;
     String s, rysunek = "";
     char z;
     boolean warunek;
     do {
        s = JOptionPane.showInputDialog(null,
         "Podaj liczbe wierszy, czyli wysokość trójkąta zawartą 3<=Liczba wierszy<=10");
        liczba_wierszy = Integer.parseInt(s);
        warunek = !(liczba_wierszy >= 3 && liczba_wierszy <= 10);
```




```
      if (warunek) {
          JOptionPane.showMessageDialog(null,
                "Nieprawidłowa liczba wierszy:" + liczba_wierszy);
        }
      } while (warunek);
   s = JOptionPane.showInputDialog(null, "Podaj znak wypełnienia");
```

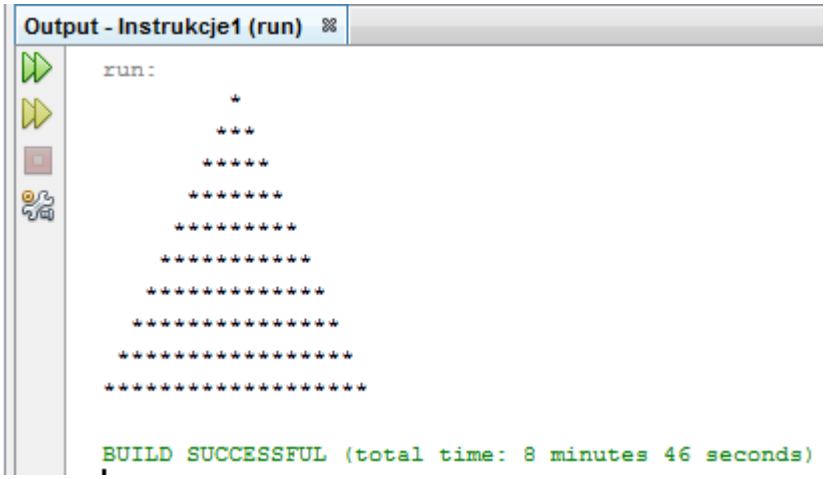


2) 2-a część: utworzenie szablonu figury – klasa Trojkat

```
//przygotowanie szablonu rysunku
      rysunek = "";
      for (j = 0; j < liczba_wierszy; j++) {
         liczba_spacji = liczba_wierszy - j - 1;
         for (i = 0; i < liczba_spacji; i++) {
            rysunek += " ";
         }
         liczba_znakow = 2 * j + 1;
         for (i = 0; i < liczba_znakow; i++) {
            rysunek += s;
         }
         rysunek += "\r\n";
      }
```

3) 3-a część: wyświetlenie szablonu figury – klasa Rysunek

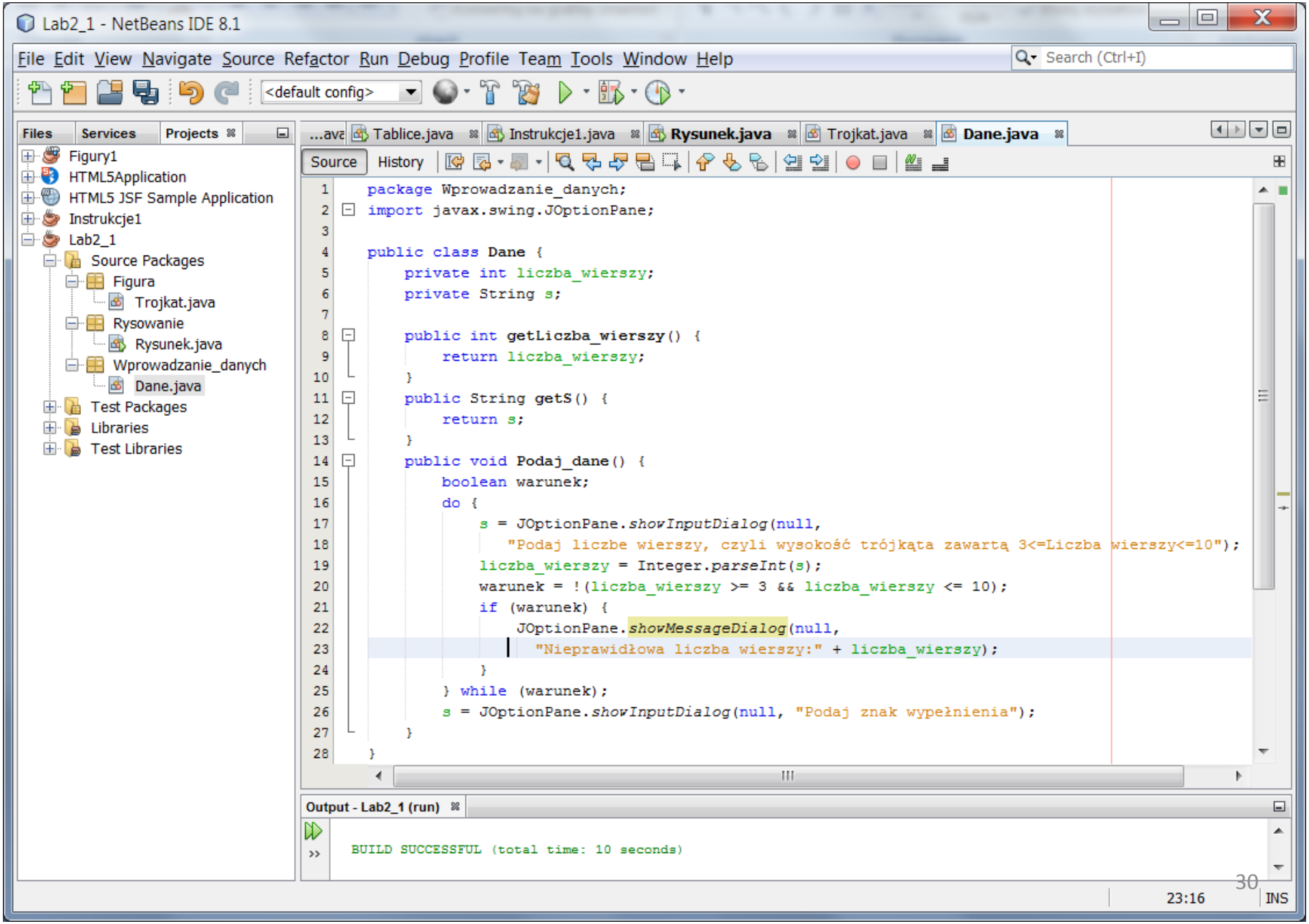
```
//rysowanie szablonu
        JOptionPane.showMessageDialog(null, rysunek);
        System.out.println(rysunek);
        System.exit(0);
    }
}
```



1-a część: Kod klasy **Dane** (1)

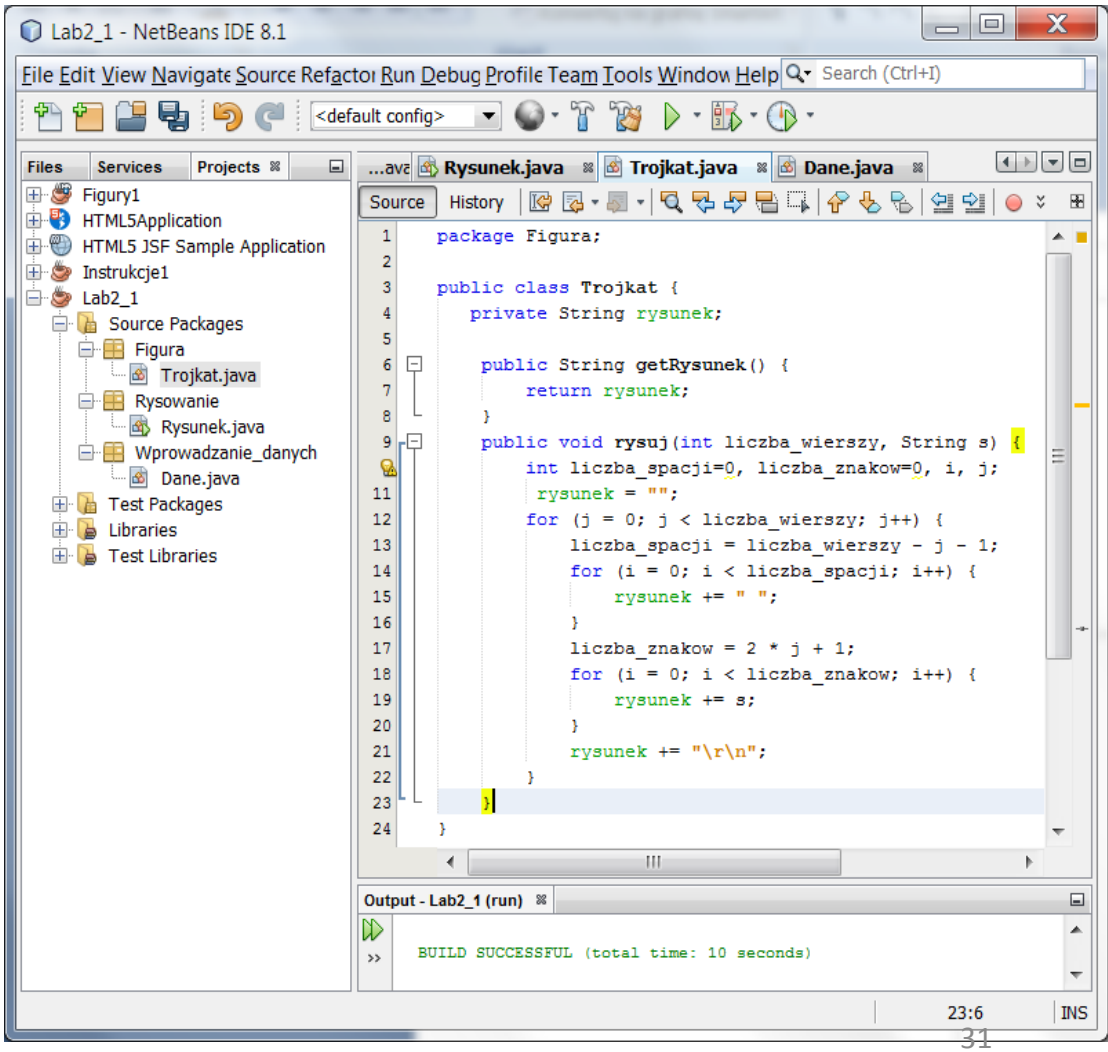
```
package Wprowadzanie_danych;

import javax.swing.JOptionPane;

public class Dane {
   private int liczba_wierszy;
   private String s;

   public int getLiczba_wierszy() {
      return liczba_wierszy;
   }
   public String getS() {
      return s;
   }
   public void Podaj_dane() {
      boolean warunek;
      do {
         s = JOptionPane.showInputDialog(null,
           "Podaj liczbe wierszy, czyli wysokość trójkąta zawartą 3<=Liczba wierszy<=10");
         liczba_wierszy = Integer.parseInt(s);
         warunek = !(liczba_wierszy >= 3 && liczba_wierszy <= 10);
         if (warunek) {
            JOptionPane.showMessageDialog(null,
              "Nieprawidłowa liczba wierszy:" + liczba_wierszy);
         }
      } while (warunek);
      s = JOptionPane.showInputDialog(null, "Podaj znak wypełnienia");
   }
}
```

Kod klasy **Dane** (2)



2-a część - Kod klasy **Trojkat**

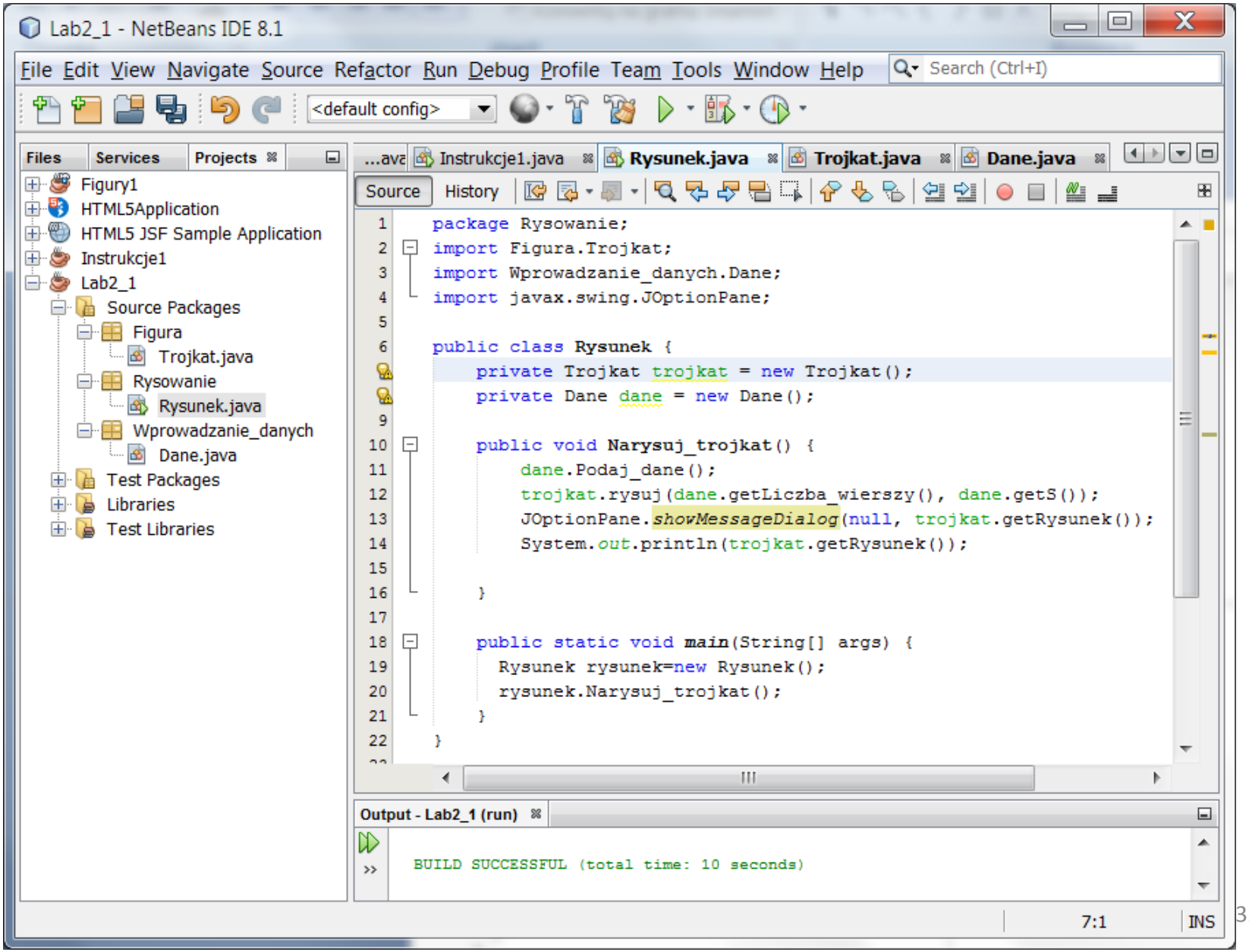
```
package Figura;

public class Trojkat {
   private String rysunek;

   public String getRysunek() {
      return rysunek;
   }
   public void rysuj(int liczba_wierszy, String s) {
      int liczba_spacji=0, liczba_znakow=0, i, j;
       rysunek = "";
      for (j = 0; j < liczba_wierszy; j++) {
         liczba_spacji = liczba_wierszy - j - 1;
         for (i = 0; i < liczba_spacji; i++) {
            rysunek += " ";
         }
         liczba_znakow = 2 * j + 1;
         for (i = 0; i < liczba_znakow; i++) {
            rysunek += s;
         }
         rysunek += "\r\n";
      }
   }
}
```



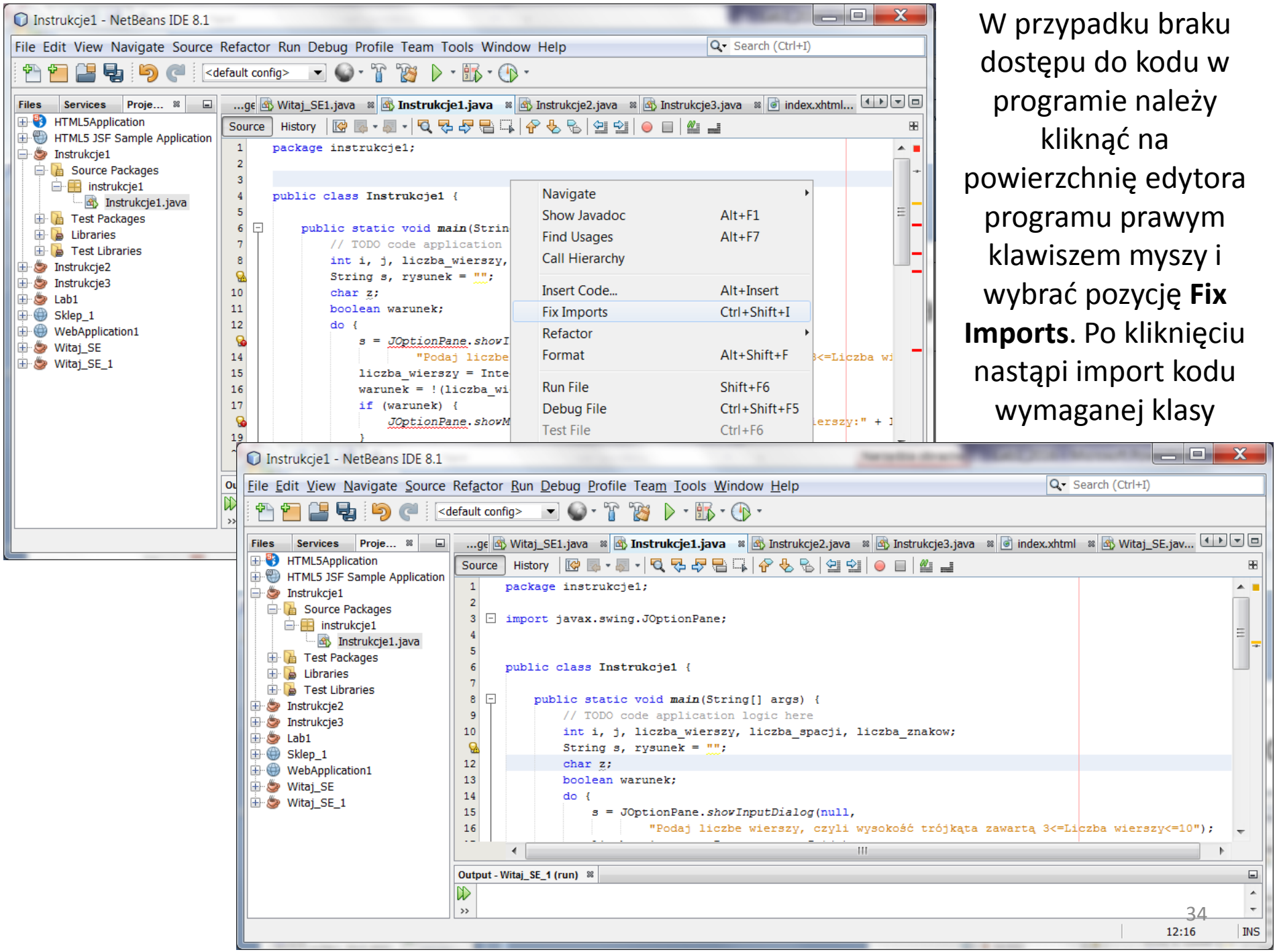
3-a część – kod klasy **Rysunek** (1)

```
package Rysowanie;

import Figura.Trojkat;
import Wprowadzanie_danych.Dane;
import javax.swing.JOptionPane;


public class Rysunek {
   private Trojkat trojkat = new Trojkat();
   private Dane dane = new Dane();

   public void Narysuj_trojkat() {
      dane.Podaj_dane();
      trojkat.rysuj(dane.getLiczba_wierszy(), dane.getS());
      JOptionPane.showMessageDialog(null, trojkat.getRysunek());
      System.out.println(trojkat.getRysunek());
   }

   public static void main(String[] args) {
     Rysunek rysunek=new Rysunek();
     rysunek.Narysuj_trojkat();
   }
}
```

Kod klasy **Rysunek** (2)



W przypadku braku dostępu do kodu w programie należy kliknąć na powierzchnię edytora programu prawym klawiszem myszy i wybrać pozycję **Fix Imports**. Po kliknięciu nastąpi import kodu wymaganej klasy



Zadanie 4.2 Modyfikacja kodu klasy Rysunek – składowe typu **static.** Należy wykonać kopię projektu zrealizowanego w zad1 i podobnie wprowadzić składowe typu **static.**

```
package Rysowanie;
import Figura.Trojkat;
import Wprowadzanie_danych.Dane;
import javax.swing.JOptionPane;

public class Rysunek {
   private static Trojkat trojkat = new Trojkat();
   private static Dane dane = new Dane();

   public static void Narysuj_trojkat() {
      dane.Podaj_dane();
      trojkat.rysuj(dane.getLiczba_wierszy(), dane.getS());
      JOptionPane.showMessageDialog(null, trojkat.getRysunek());
      System.out.println(trojkat.getRysunek());
   }
   public static void main(String[] args) {
    Narysuj_trojkat();              // teraz można wywołać metodę typu static
                                    // we własnej metodzie main klasy , która zawiera metodę static
    Rysunek.Narysuj_trojkat();      // tak należy wywołać metodę statyczną w innej klasie –
                                    //bez tworzenie obiektu operatorem new
   }
```

Modyfikacja kodu klasy **Rysunek** – składowe typu **static**