

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.

Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name: Lab2_punkt

Project Location: C:\Studia\Szkola\Java\lab2

Project Folder: C:\Studia\Szkola\Java\lab2\Lab2_punkt

Use Dedicated Folder for Storing Libraries

Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

Create Main Class punkt_

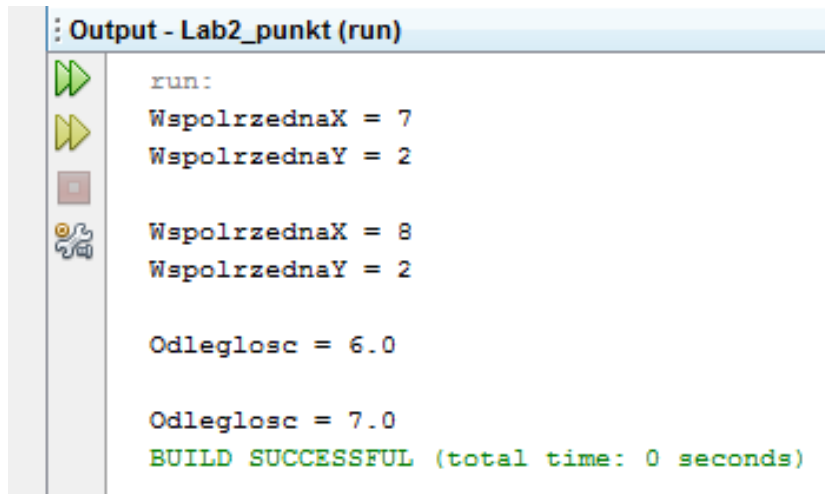
< Back Next > **Finish** Cancel Help

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));  
    }  
}
```



```
Output - Lab2_punkt (run)  
run:  
WspolrzednaX = 7  
WspolrzednaY = 2  
  
WspolrzednaX = 8  
WspolrzednaY = 2  
  
Odleglosc = 6.0  
  
Odleglosc = 7.0  
BUILD SUCCESSFUL (total time: 0 seconds)
```

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_**)

Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name: Lab2_punkt1

Project Location: C:\Studia\Szkola\Java\lab2

Project Folder: C:\Studia\Szkola\Java\lab2\Lab2_punkt1

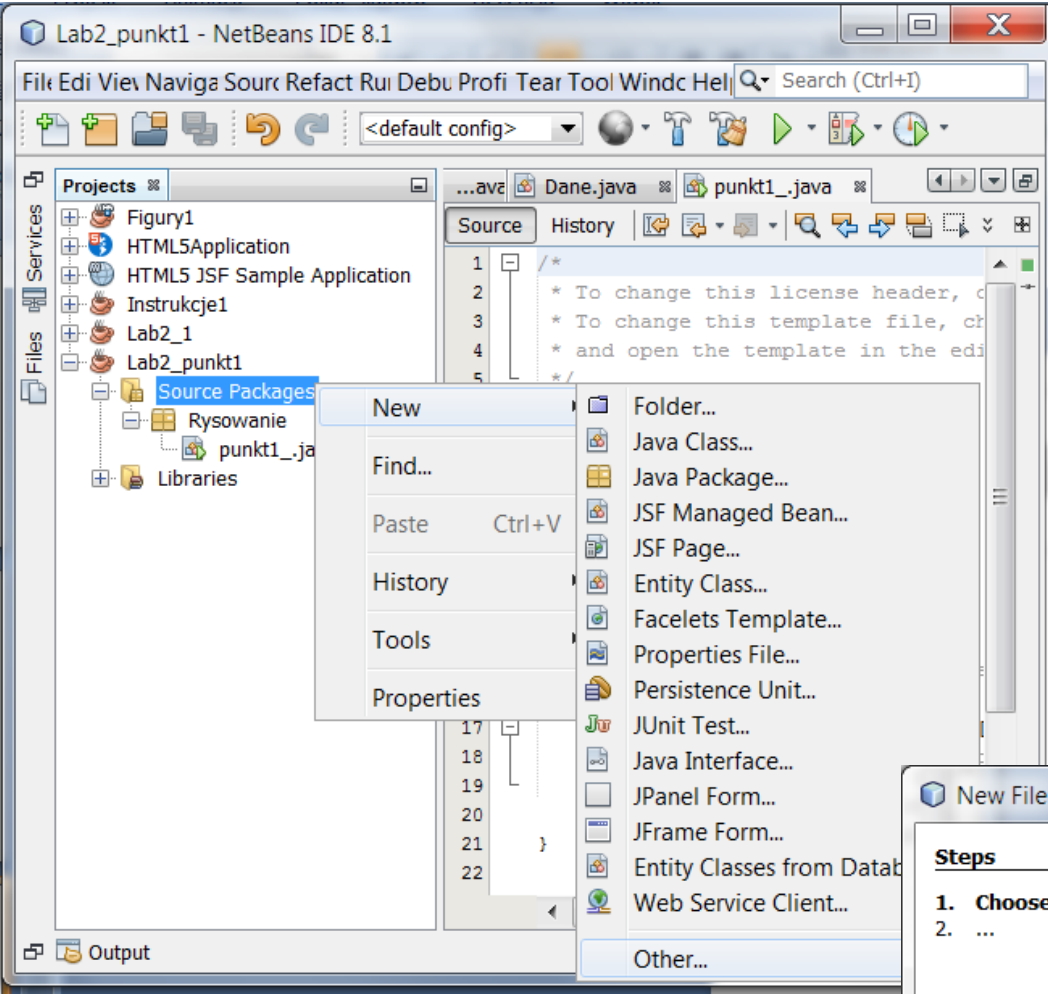
Use Dedicated Folder for Storing Libraries

Libraries Folder:

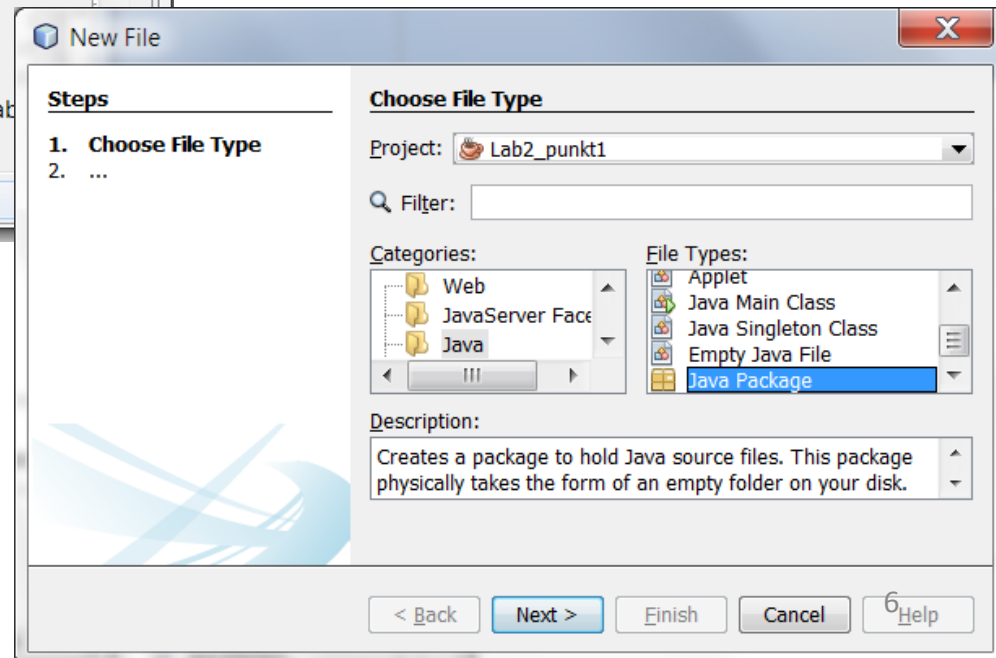
Different users and projects can share the same compilation libraries (see Help for details).

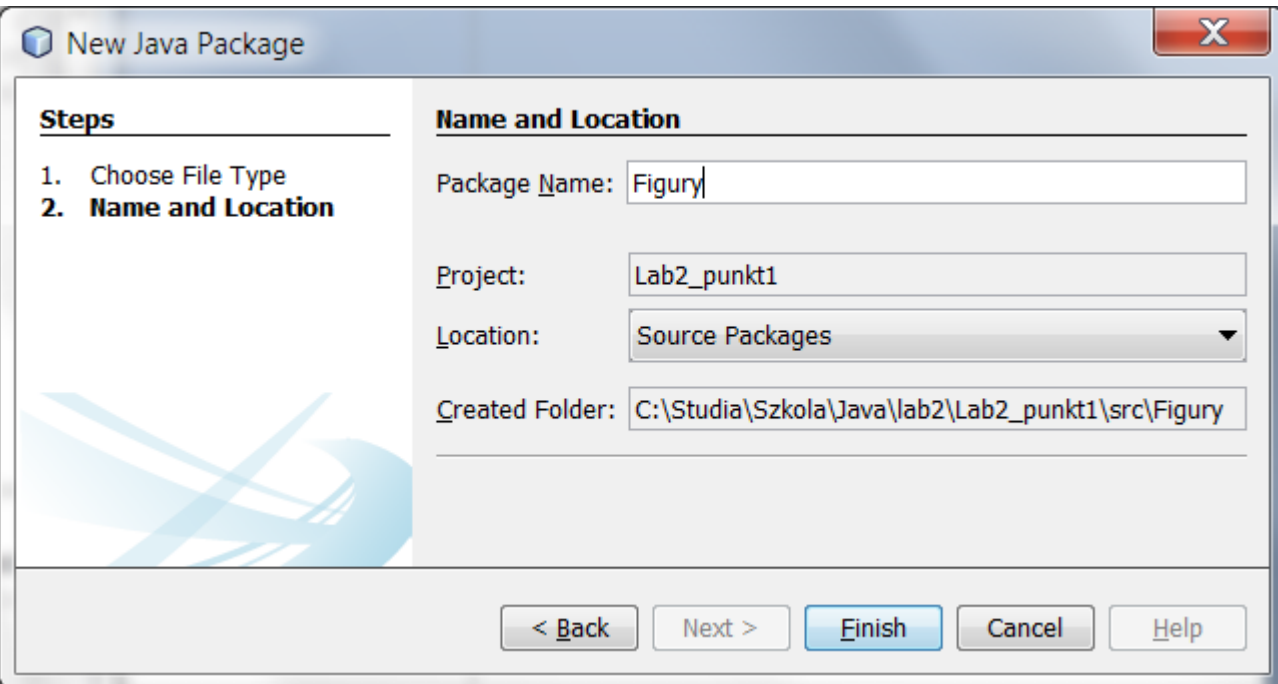
Create Main Class Rysowanie.punkt1_

< Back Next > **Finish** Cancel Help

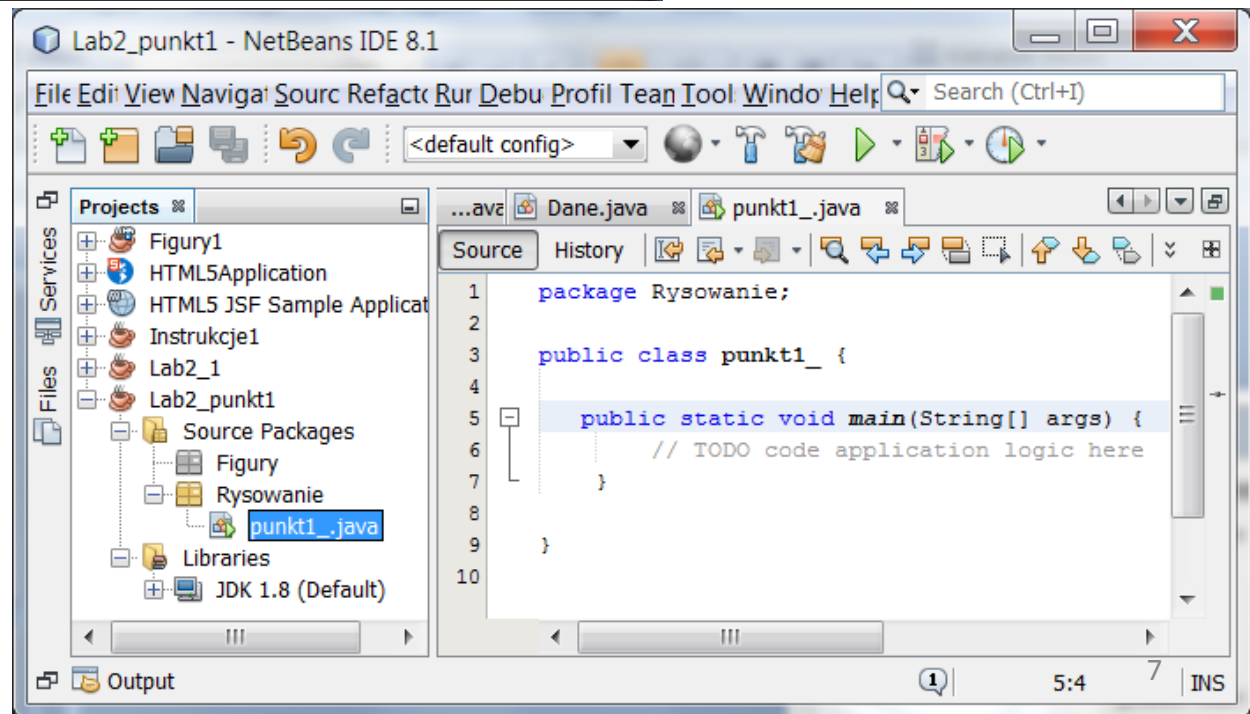


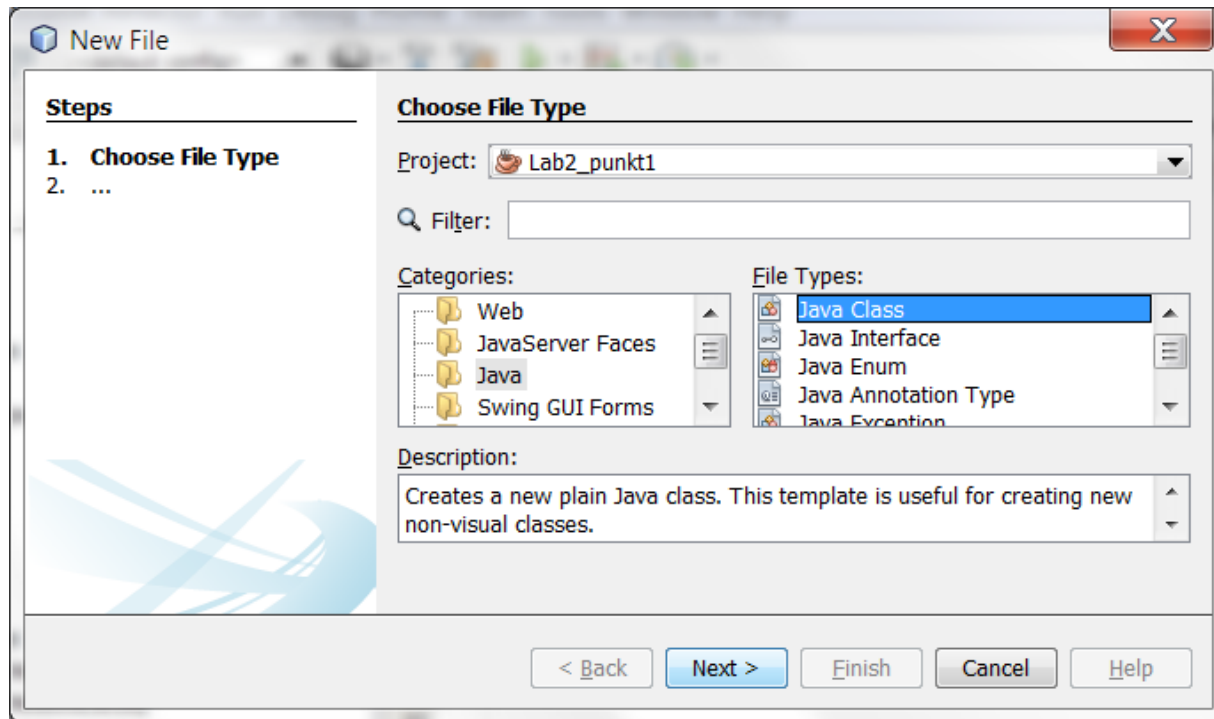
Należy wykonać nowy pakiet w utworzonym projekcie: po kliknięciu prawym klawiszem na nazwę projektu w oknie **Projects** wybrać : **New/Other/Java/Java Packages/Next**





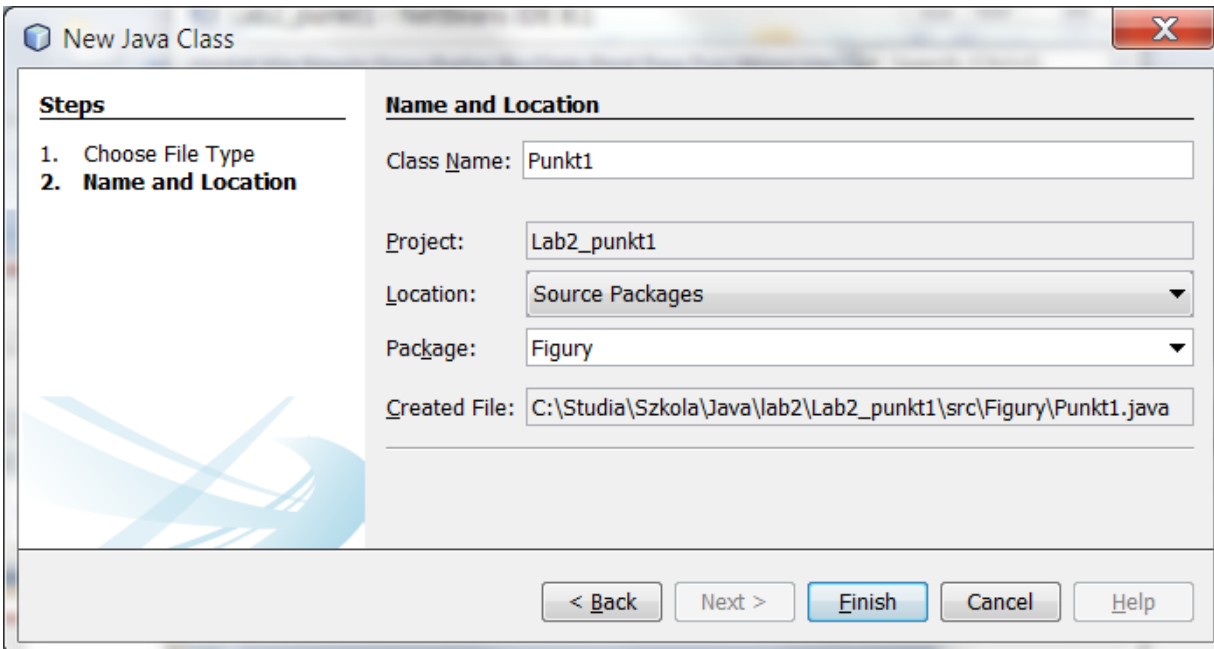
i nadać nazwę nowemu pakietowi: w polu **Package Name**. Poniżej pokazano wynik dotychczasowych działań.

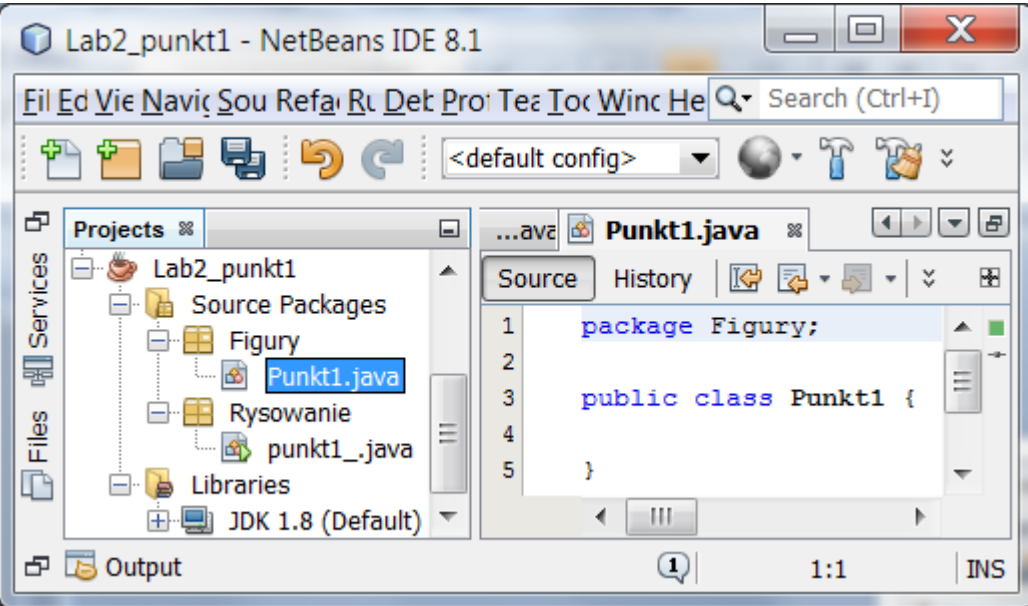




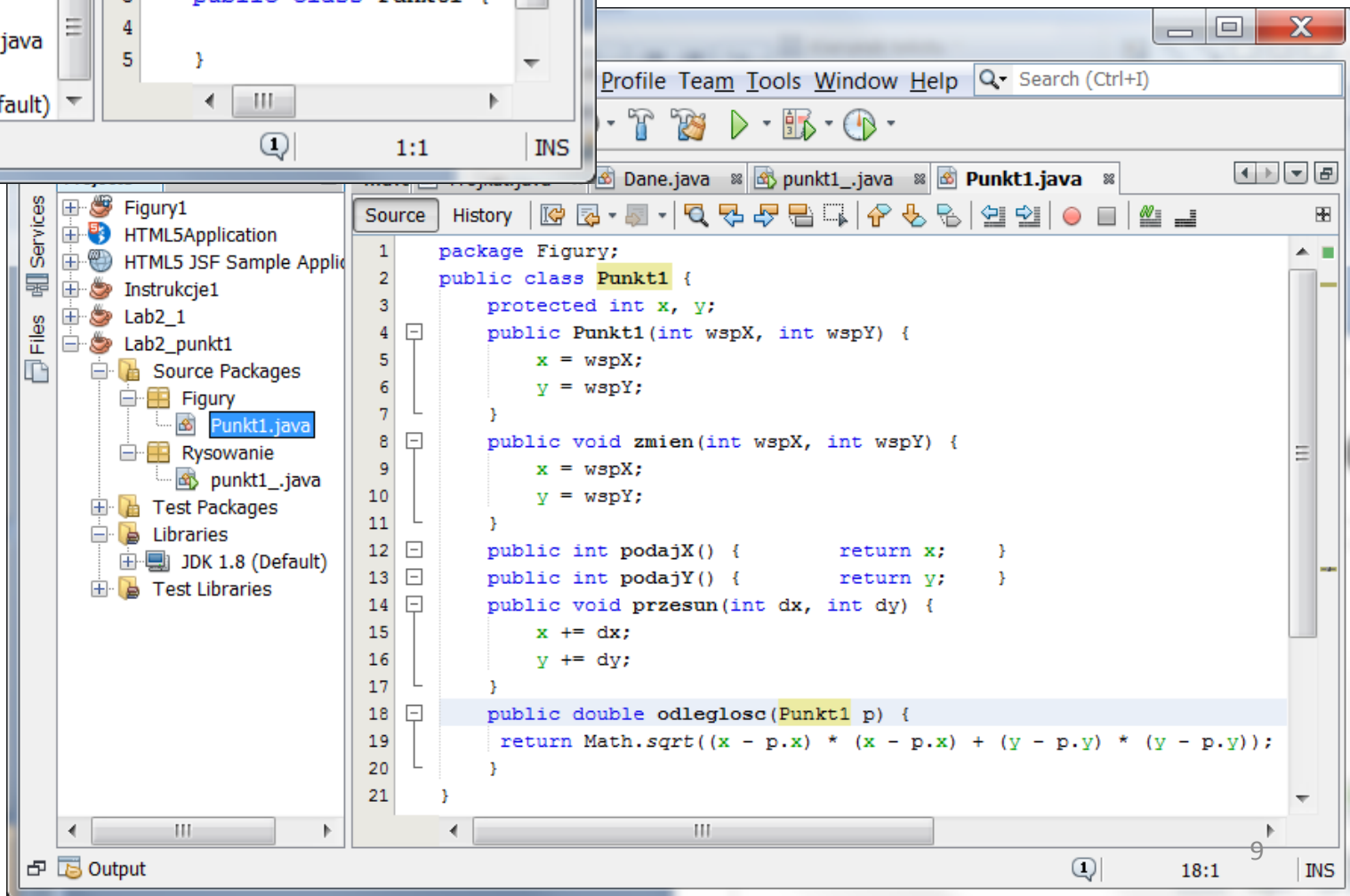
W nowym pakiecie **Figury** należy dodać nową klasę **Punkt1** w wyniku następujących działań: po kliknięciu w oknie **Projects** na nazwę pakietu **Figury**, wybrać:

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 wcześniej (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..

The screenshot displays the NetBeans IDE 8.1 interface. The main editor window shows the source code for `punkt1_.java` in the `Rysowanie` package. The code includes an import for `Figury.Punkt1` and a `main` method that creates two `Punkt1` objects, prints their coordinates, and calculates distances.

```
1 package Rysowanie;
2 import Figury.Punkt1;
3
4 public class punkt1_ {
5     public static void main(String[] args) {
6         Punkt1 p1 = new Punkt1(7, 2);
7         System.out.println("WspolrzednaX = " + p1.podajX());
8         System.out.println("WspolrzednaY = " + p1.podajY());
9         Punkt1 p2 = new Punkt1(8, 2);
10        System.out.println("\nWspolrzednaX = " + p2.podajX());
11        System.out.println("WspolrzednaY = " + p2.podajY());
12        p2.zmien(1, 2);
13        System.out.println("\nOdleglosc = " + p2.odleglosc(p1));
14        p1.przesun(1, 0);
15        System.out.println("\nOdleglosc = " + p1.odleglosc(p2));
16    }
17 }
```

The `Output - Lab2_punkt1 (run)` window on the right shows the following output:

```
run:
WspolrzednaX = 7
WspolrzednaY = 2

WspolrzednaX = 8
WspolrzednaY = 2

Odleglosc = 6.0

Odleglosc = 7.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

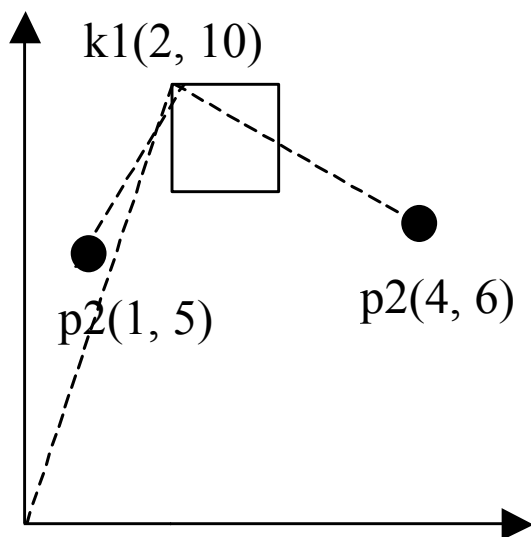
Zadanie 2 – dziedziczenie metod w klasie pochodnej – część 1. Przedstawiony program zawiera klasy: **Punkt**, pochodną klasę typu **Kwadrat** oraz klasę **punkt_**, która pozwala uruchomić program. **Należy wykonać podobny program, w którym jest klasa Punkt1, a klasą pochodną powinien być Prostokąt - 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, PRZEDFINIOWANIE 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 odleglosc(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 podajDI()
    {
        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

```
C:\Program Files\Xinox Soft...  
WspolrzednaX = 7  
WspolrzednaY = 2  
Dlugosc boku = 5  
  
\nWspolrzednaX = 8  
WspolrzednaY = 2  
  
\nOdleglosc = 6.0  
Odleglosc = 7.0  
  
Powierzchnia = 25  
Press any key to continue...
```

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 konstruktor dziedziczonej klasy *Punkt*. Konstruktory domyślne nie trzeba wywoływać (są wywoływane niejawnie), pozostałe konstruktory, jeśli nie ma domyślnych, trzeba zawsze wywołać za pomocą słowa `super`, jako pierwsze wywołanie w konstruktorze 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 odleglosc(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**, 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 **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 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());  
    }  
}
```

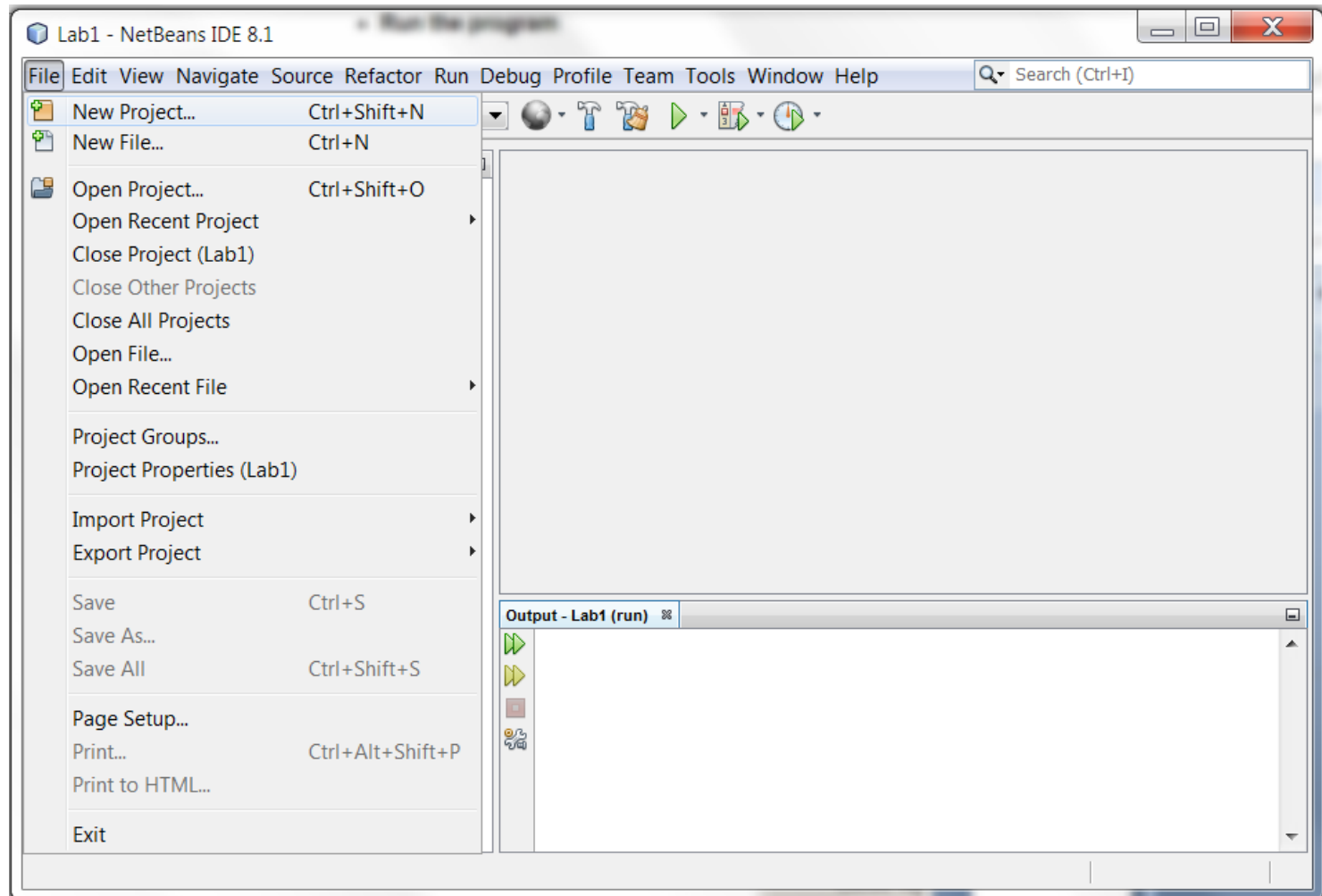
Te metody są
wywołane dzięki
dziedziczeniu
składowych od klasy
Punkt

```
C:\Program Files\Xinox Soft...  
WspolrzednaX = 7  
WspolrzednaY = 2  
Dlugosc boku = 5  
WspolrzednaX = 8  
WspolrzednaY = 2  
Odleglosc = 6.0  
Odleglosc = 7.0  
Powierzchnia = 25  
Press any key to continue...
```

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 konstruktor dziedziczonej klasy *Punkt*. Konstruktory domyślne nie trzeba wywoływać (są wywoływane niejawnie), pozostałe konstruktory, jeśli nie ma domyślnych, trzeba zawsze wywołać za pomocą słowa `super`, jako pierwsze wywołanie w konstruktorze 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**)

New Java Application

Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name:

Project Location:

Project Folder:

Use Dedicated Folder for Storing Libraries

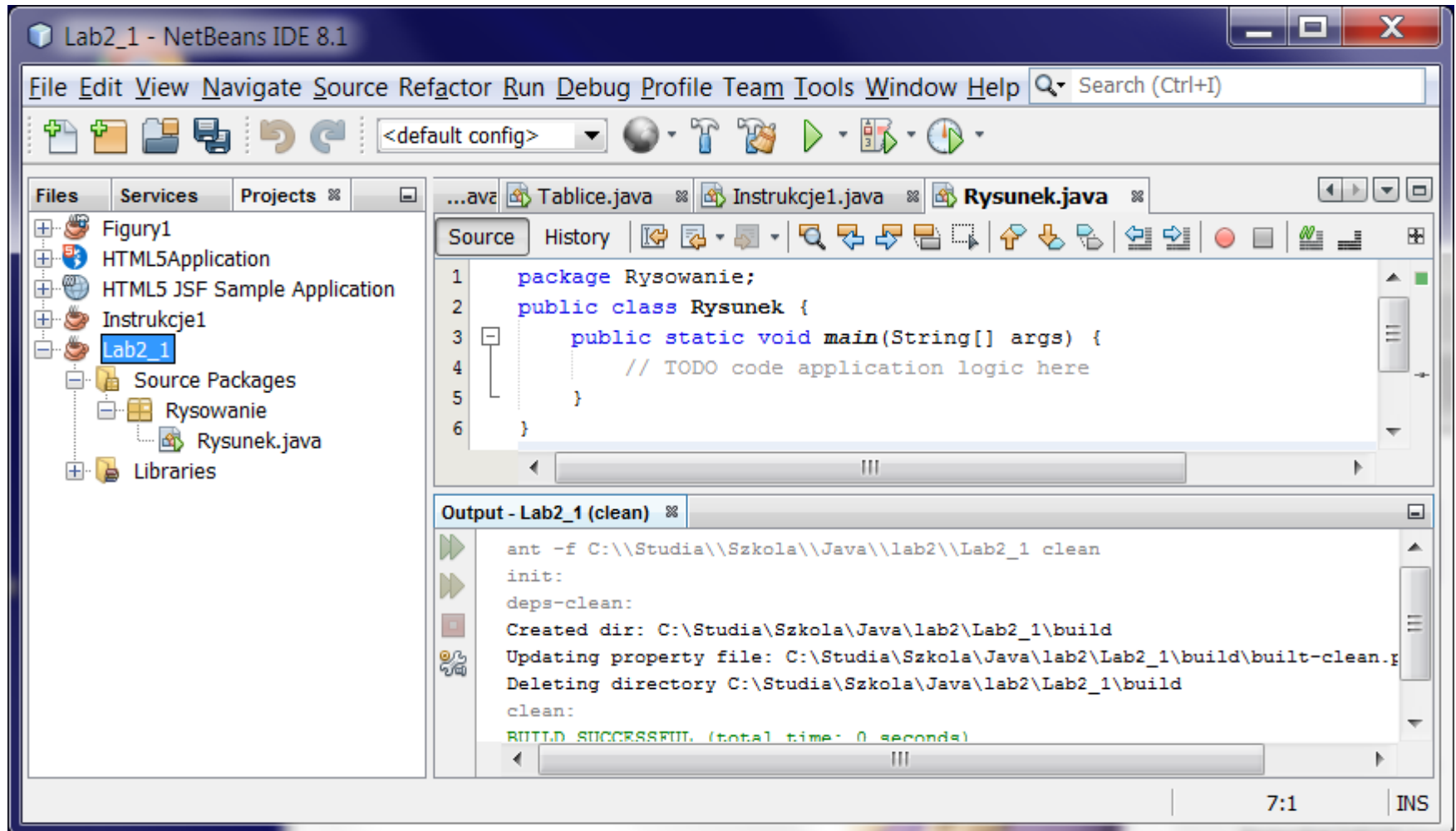
Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

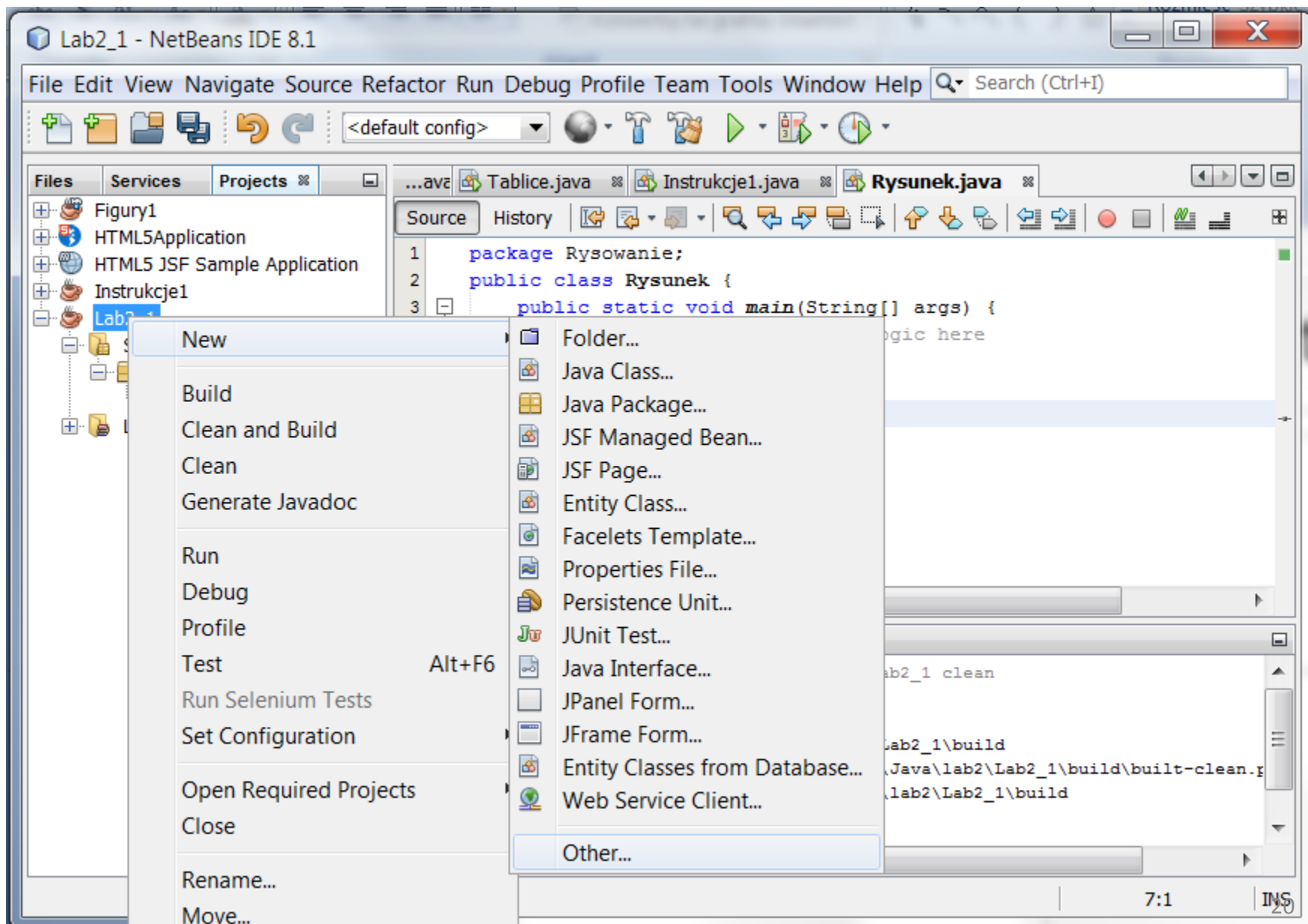
Create Main Class

< **Back** **Next** > **Finish** **Cancel** **Help**

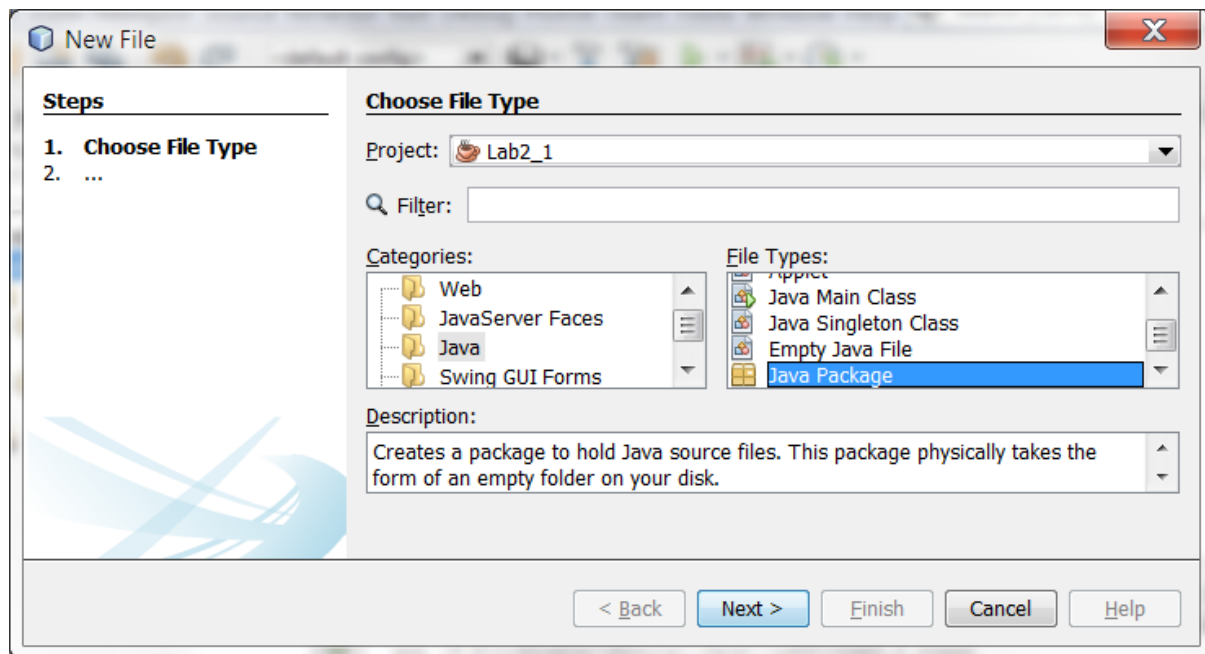
Rezultat – utworzony nowy projekt wraz z klasą **Rysunek** w pakiecie **Rysowanie**. Klasa będzie jednocześnie programem (funkcja main), odpowiedzialnym za narysowanie trójkąta.



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

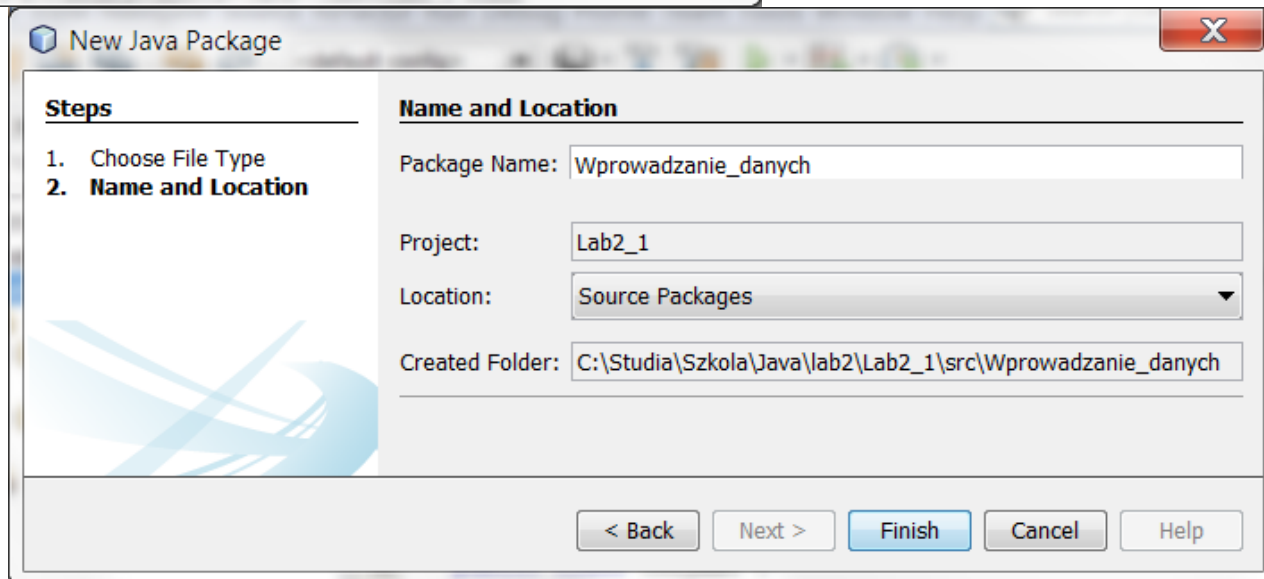


Wykonanie kolejnego pakietu **Wprowadzanie_danych** do przechowania klasy typu **Dane**, służącej do wprowadzania danych (kliknięcie na projekt **New/Other**)



Wybór **Java/Java Package/**
Next

Utworzenie
nowego pakietu
o nazwie
podanej w polu
Package Name



Utworzenie w podobny sposób pakietu **Figura**

The screenshot displays the NetBeans IDE 8.1 interface. The main window title is "Lab2_1 - NetBeans IDE 8.1". The menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, and Help. The toolbar contains various icons for file operations and execution. The left sidebar shows the "Projects" view with a tree structure: "Lab2_1" (selected) contains "Source Packages" (expanded) with sub-packages "Figura" (newly created), "Rysowanie", and "Wprowadzanie_danych". Other projects listed are "Figury1", "HTML5Application", "HTML5 JSF Sample Application", and "Instrukcje1". The main editor window shows the "Source" view of the file "Rysunek.java" in the "Rysowanie" package. The code is as follows:

```
1 package Rysowanie;
2 public class Rysunek {
3     public static void main(String[] args) {
4         // TODO code application logic here
5     }
6 }
7
```

The bottom panel shows the "Output - Lab2_1 (clean)" window with the following text:

```
ant -f C:\\Studia\\Szkola\\Java\\lab2\\Lab2_1 clean
init:
deps-clean:
Created dir: C:\\Studia\\Szkola\\Java\\lab2\\Lab2_1\\build
Updating property file: C:\\Studia\\Szkola\\Java\\lab2\\Lab2_1\\build\\built-clean.p
Deleting directory C:\\Studia\\Szkola\\Java\\lab2\\Lab2_1\\build
clean:
BUILD SUCCESSFUL (total time: 0 seconds)
```

The status bar at the bottom right shows "7:1" and "INS".

Dodanie klasy w pakiecie Figura

The screenshot shows the NetBeans IDE 8.1 interface. The 'Projects' view on the left shows a project named 'Lab2_1' with a sub-package 'Figura'. The 'Source' editor displays the code for 'Rysunek.java' in the 'Rysowanie' package. A context menu is open over the 'Figura' package, listing various options for creating new elements. The 'New' menu item is selected, and a sub-menu is visible with options like 'Folder...', 'Java Package...', 'Java Class...', etc.

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help Search (Ctrl+I)

Files Services Projects

Figury1
HTML5Application
HTML5 JSF Sample Application
Instrukcje1
Lab2_1
Source Packages
Figura
Rysow
Rys
Wprov
Libraries

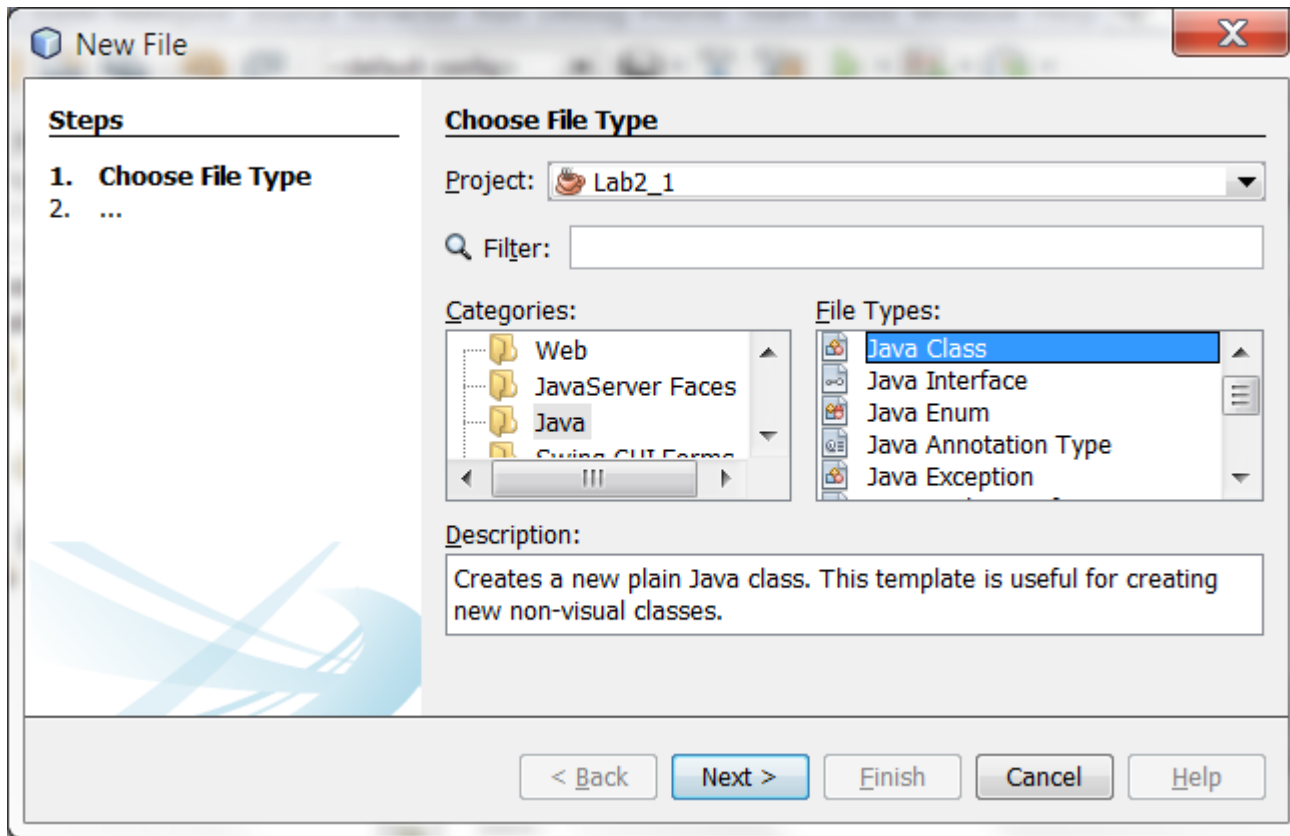
Tablice.java Instrukcje1.java Rysunek.java

```
1 package Rysowanie;  
2 public class Rysunek {  
3     public static void main(String[] args) {  
4         // TODO code application logic here  
5     }
```

New
Find... Ctrl+F
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Delete Delete
Refactor
Compile Package F9
Test Package Ctrl+F6
Run Selenium Tests
History
Tools

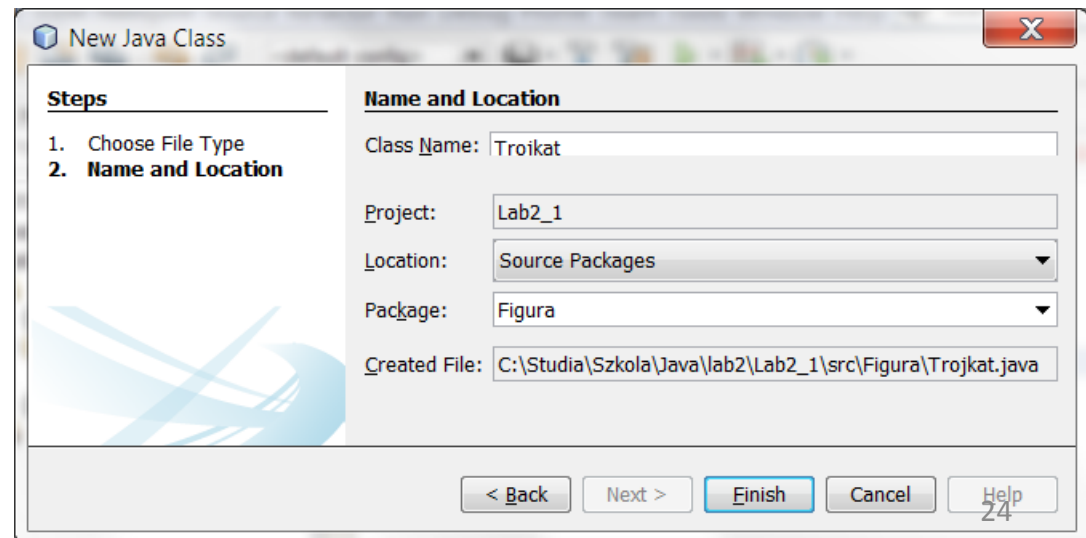
Folder...
Java Package...
Java Class...
JSF Managed Bean...
JSF Page...
Entity Class...
Facelets Template...
Properties File...
Persistence Unit...
JUnit Test...
Java Interface...
JPanel Form...
JFrame Form...
Entity Classes from Database...
Web Service Client...
Other...

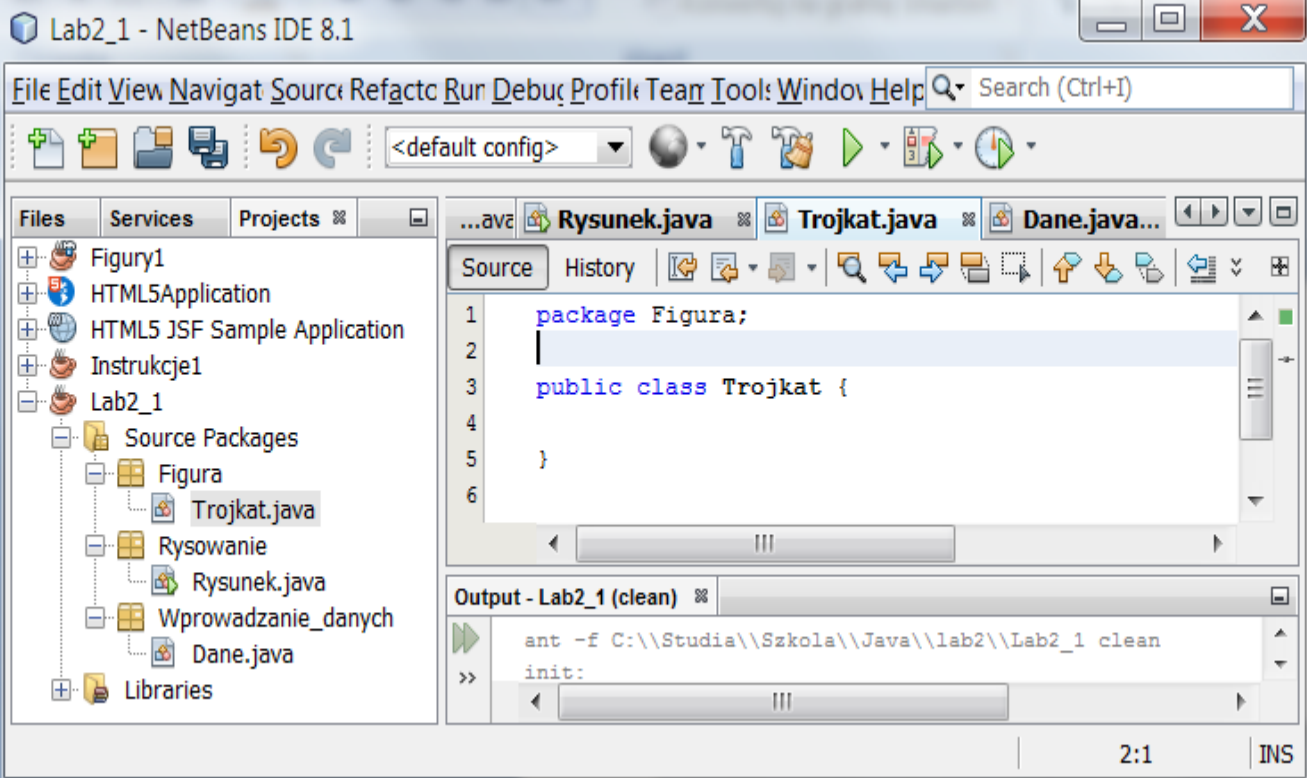
7:1 INS



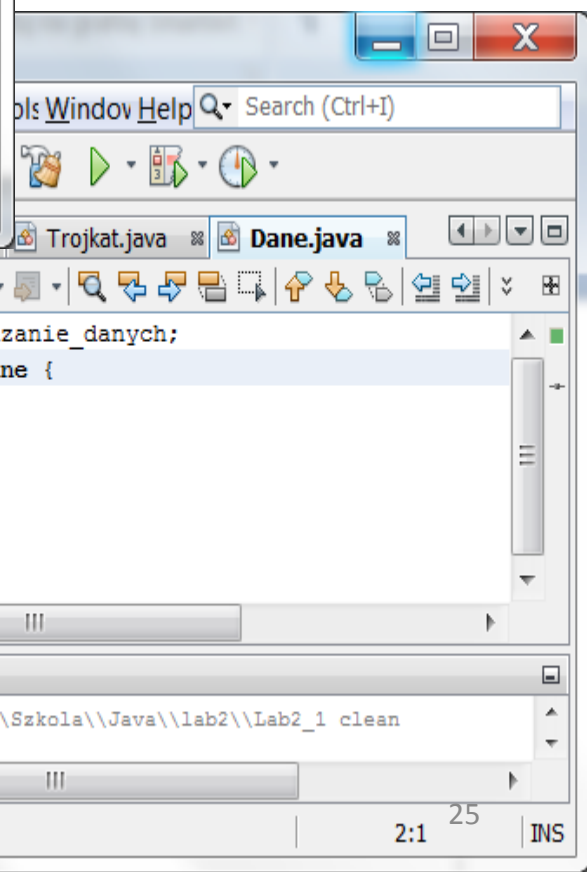
Wybór pozycji
Java/Java Class

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





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



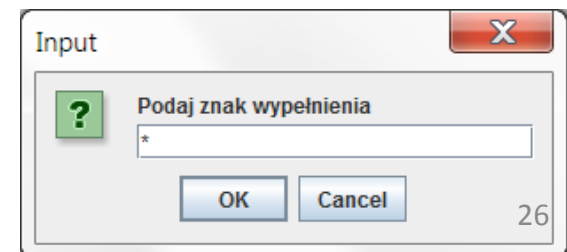
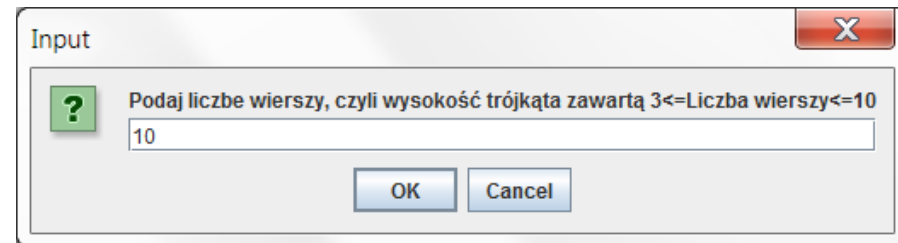
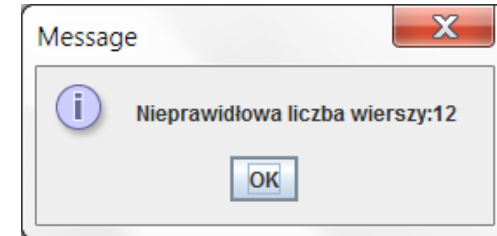
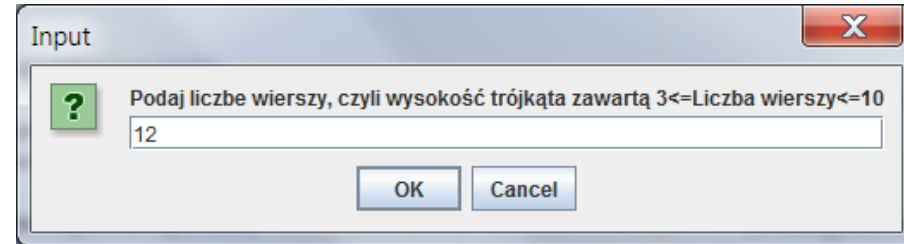
Na podstawie zadania 2 z lab1 należy podzielić logicznie kod na trzy części, czyli trzy klasy:

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

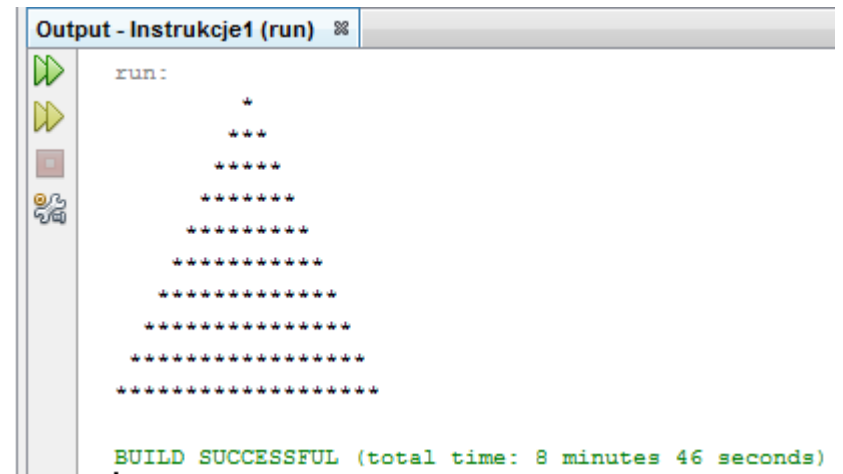
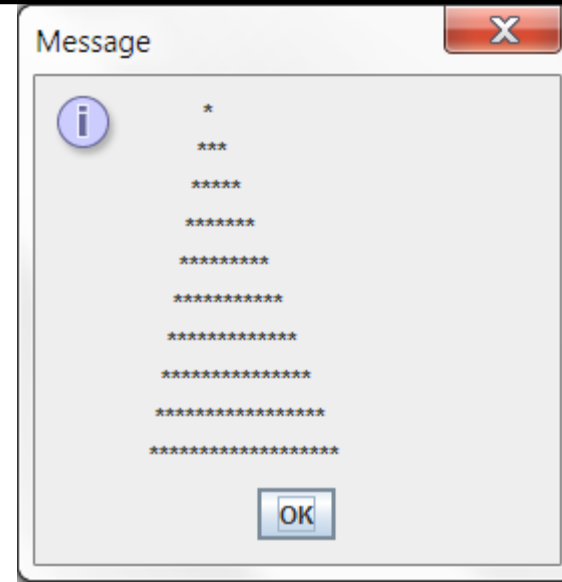
```
JOptionPane.showMessageDialog(null, rysunek);
```

```
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 liczbę 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)

The screenshot displays the NetBeans IDE 8.1 interface. The main editor window shows the source code for the `Dane.java` class. The code is as follows:

```
1 package Wprowadzanie_danych;
2 import javax.swing.JOptionPane;
3
4 public class Dane {
5     private int liczba_wierszy;
6     private String s;
7
8     public int getLiczba_wierszy() {
9         return liczba_wierszy;
10    }
11
12    public String getS() {
13        return s;
14    }
15
16    public void Podaj_dane() {
17        boolean warunek;
18        do {
19            s = JOptionPane.showInputDialog(null,
20                "Podaj liczbę wierszy, czyli wysokość trójkąta zawartą 3<=Liczba wierszy<=10");
21            liczba_wierszy = Integer.parseInt(s);
22            warunek = !(liczba_wierszy >= 3 && liczba_wierszy <= 10);
23            if (warunek) {
24                JOptionPane.showMessageDialog(null,
25                    "Nieprawidłowa liczba wierszy:" + liczba_wierszy);
26            }
27        } while (warunek);
28        s = JOptionPane.showInputDialog(null, "Podaj znak wypełnienia");
29    }
30 }
```

The Output window at the bottom shows the result of a successful build:

```
>> BUILD SUCCESSFUL (total time: 10 seconds)
```

The status bar at the bottom right indicates the time 23:16 and the page number 30.

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 = "";
```

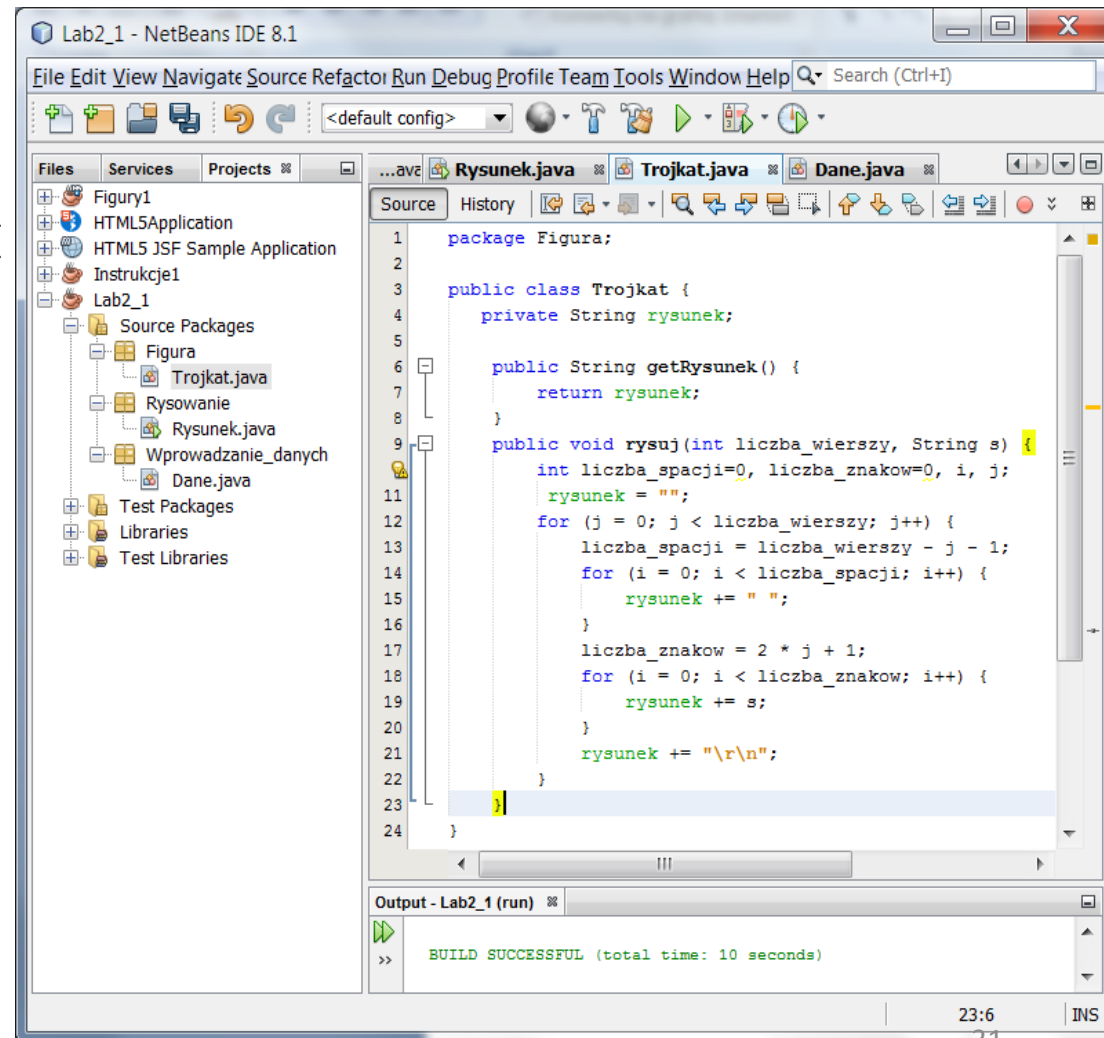
```
        for (j = 0; j < liczba_wierszy; j++) {  
            liczba_spacji = liczba_wierszy - j - 1;  
            for (i = 0; i < liczba_spacji; i++) {  
                rysunek += " ";
```

```
            }  
            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;

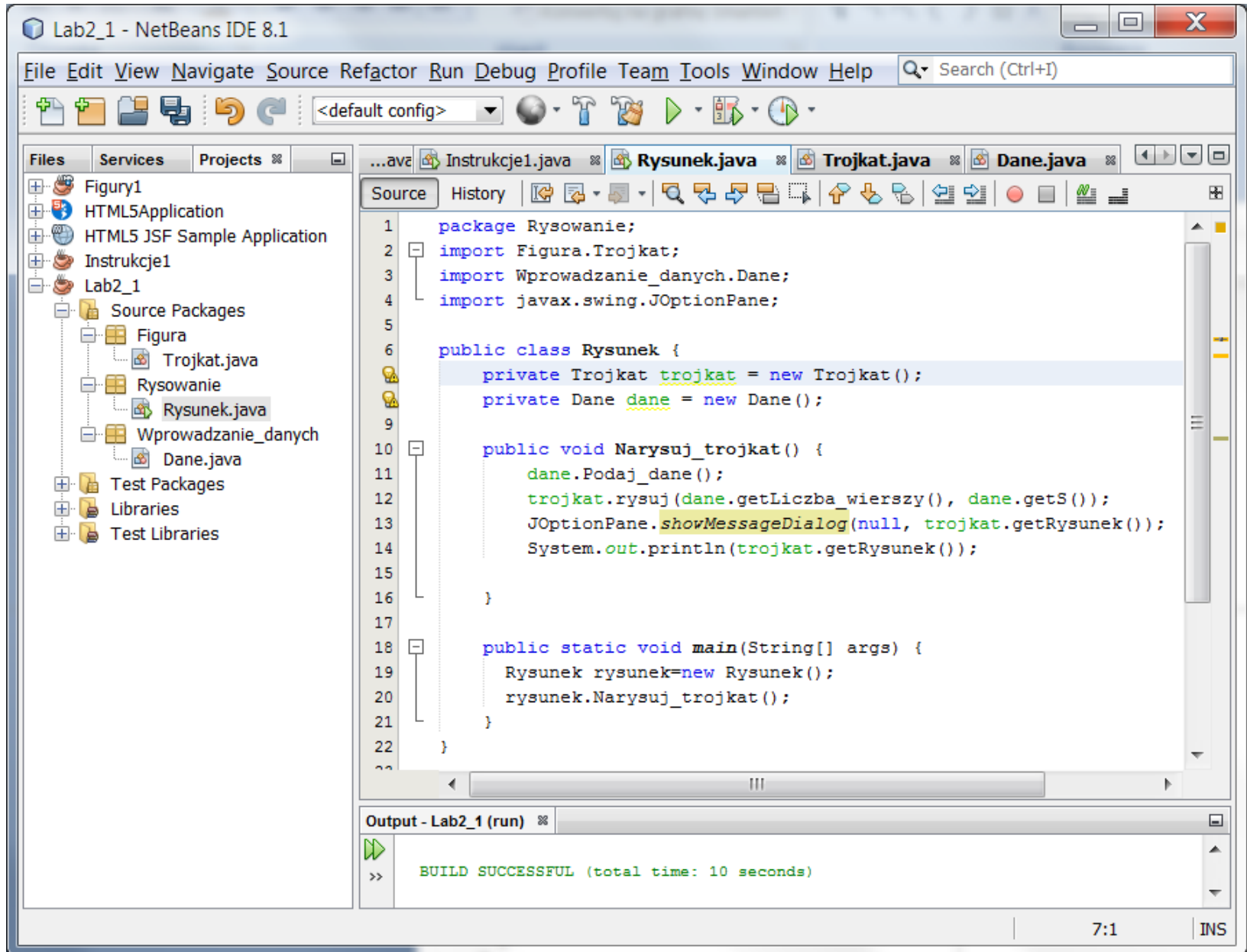
import Figura.Trojkat;
import Wprowadzanie_danych.Dane;
import javax.swing.JOptionPane;

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)



The screenshot displays the NetBeans IDE 8.1 interface. The main window shows the source code for the `Rysunek.java` file. The code is as follows:

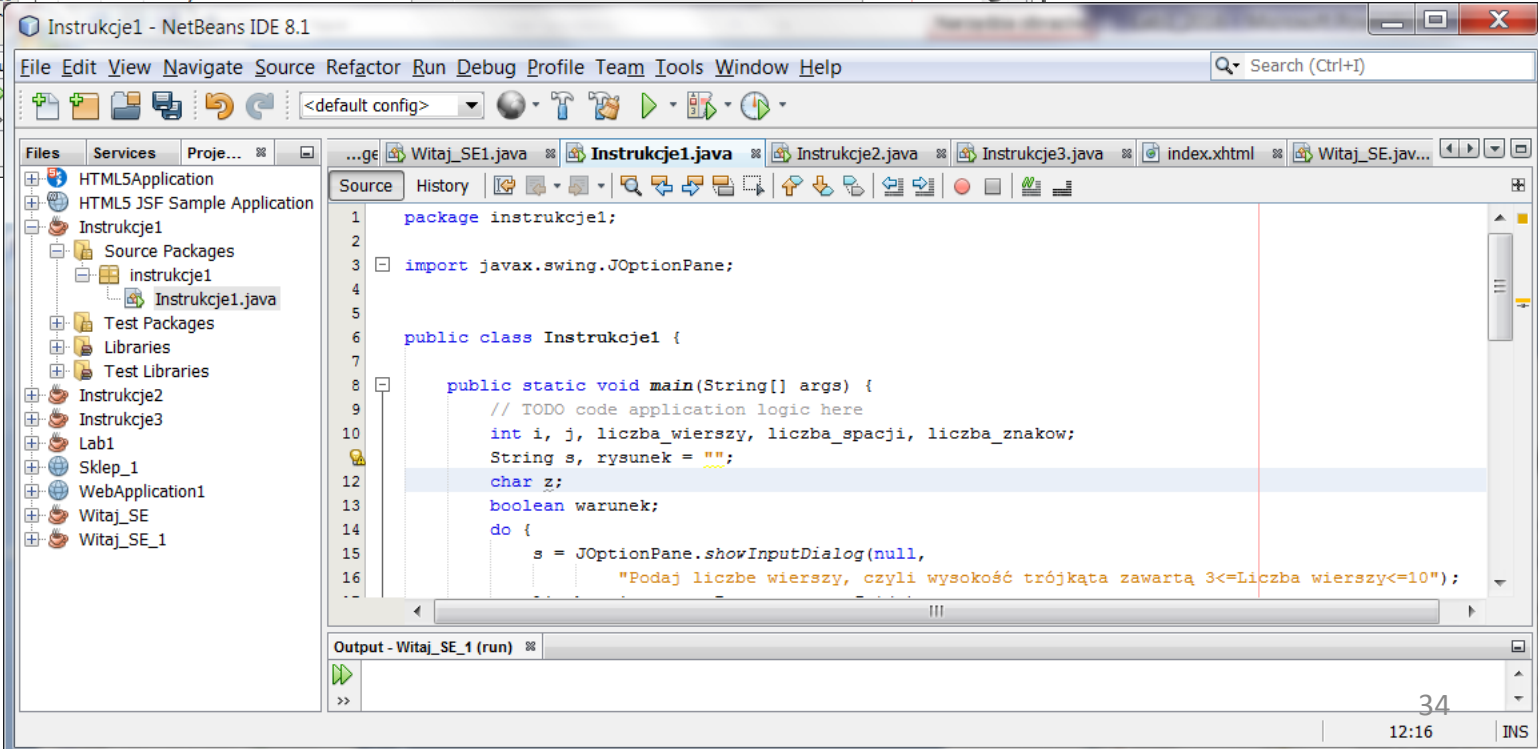
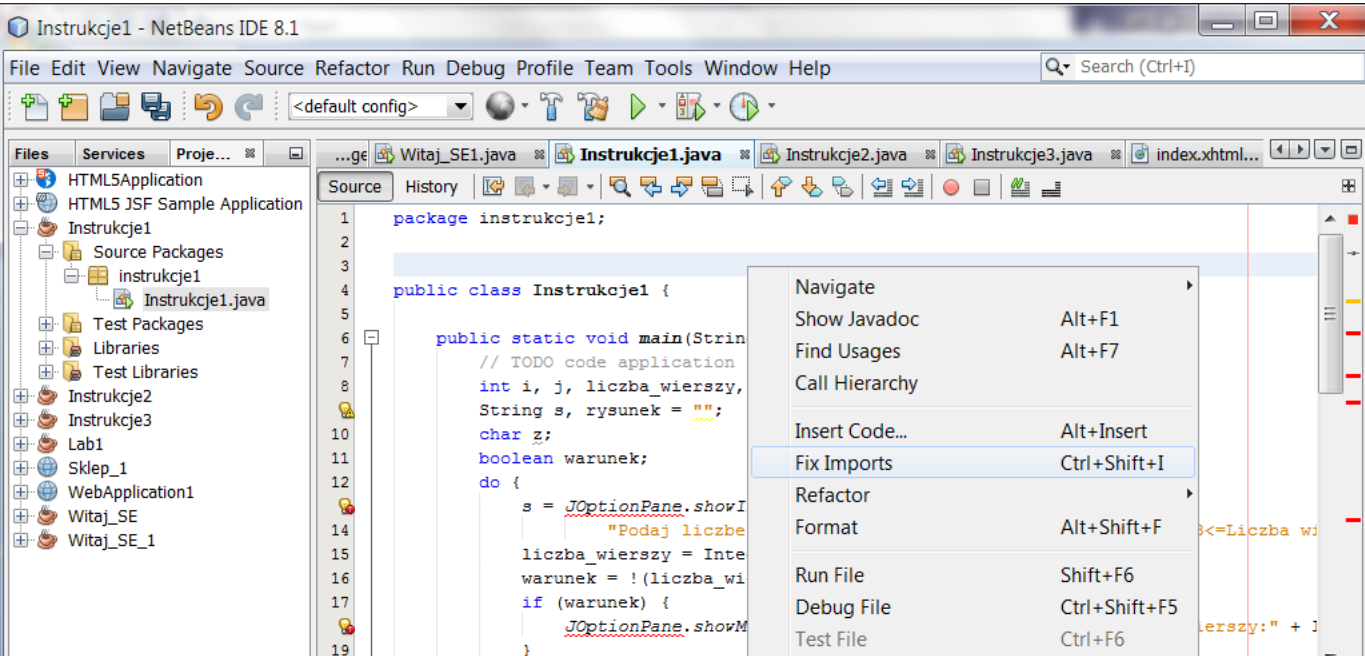
```
1 package Rysowanie;
2 import Figura.Trojkat;
3 import Wprowadzanie_danych.Dane;
4 import javax.swing.JOptionPane;
5
6 public class Rysunek {
7     private Trojkat trojkat = new Trojkat();
8     private Dane dane = new Dane();
9
10    public void Narysuj_trojkat() {
11        dane.Podaj_dane();
12        trojkat.rysuj(dane.getLiczba_wierszy(), dane.getS());
13        JOptionPane.showMessageDialog(null, trojkat.getRysunek());
14        System.out.println(trojkat.getRysunek());
15    }
16
17
18    public static void main(String[] args) {
19        Rysunek rysunek=new Rysunek();
20        rysunek.Narysuj_trojkat();
21    }
22 }
```

The IDE's output window at the bottom shows the result of a successful build:

```
>> BUILD SUCCESSFUL (total time: 10 seconds)
```

The status bar at the bottom right indicates the cursor is at line 7:1 in the `INS` mode.

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**

The screenshot displays the NetBeans IDE 8.1 interface. The main editor window shows the source code for the `Rysunek` class in the `Rysowanie` package. The code is as follows:

```
1 package Rysowanie;
2 import Figura.Trojkat;
3 import Wprowadzanie_danych.Dane;
4 import javax.swing.JOptionPane;
5
6 public class Rysunek {
7     private static Trojkat trojkat = new Trojkat();
8     private static Dane dane = new Dane();
9
10    public static void Narysuj_trojkat() {
11        dane.Podaj_dane();
12        trojkat.rysuj(dane.getLiczba_wierszy(), dane.getS());
13        JOptionPane.showMessageDialog(null, trojkat.getRysunek());
14        System.out.println(trojkat.getRysunek());
15    }
16
17
18    public static void main(String[] args) {
19        Narysuj_trojkat();
20        Rysunek.Narysuj_trojkat();
21    }
22 }
```

The IDE interface includes a menu bar (File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help), a toolbar with various icons, and a project explorer on the left showing the project structure. The project explorer shows the following structure:

- Figury1
- HTML5Application
- HTML5 JSF Sample Application
- Instrukcje1
- Lab2_1
 - Source Packages
 - Figura
 - Trojkat.java
 - Rysowanie
 - Rysunek.java
 - Wprowadzanie_danych
 - Dane.java
 - Test Packages
 - Libraries
 - Test Libraries

The Output window at the bottom shows the message: `BUILD SUCCESSFUL (total time: 19 seconds)`. The system tray at the bottom right shows the time 7:20 and the keyboard indicator INS.