

Tworzenie i korzystanie z plików JAR

Biblioteka JFreeChart

Czy są pliki JAR?

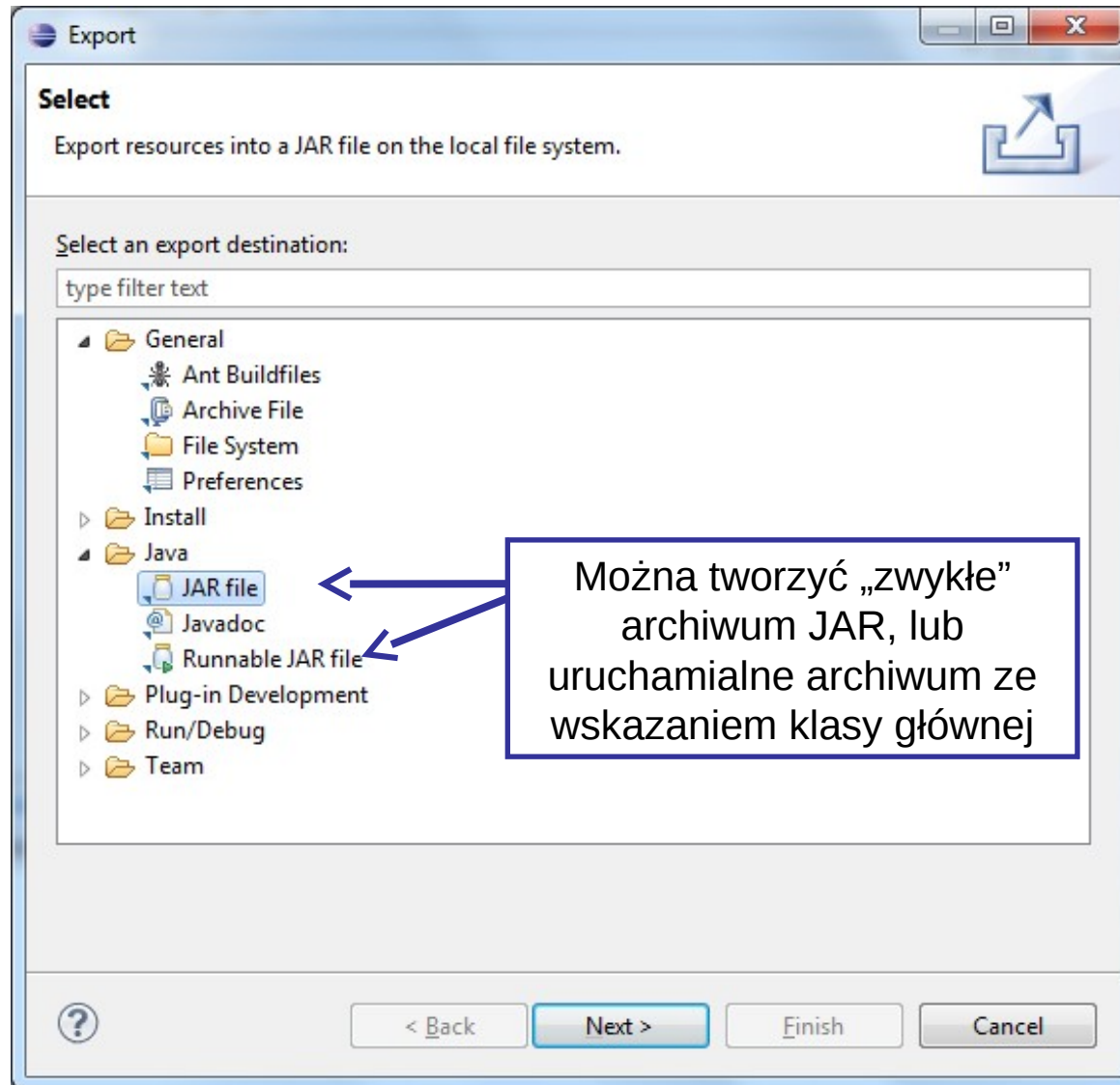
- JAR (ang. **J**ava **AR**chive) – archiwum ZIP używane do strukturalizacji i kompresji plików klas języka Java oraz powiązanych z nimi zasobów i metadanych.
- Archiwum JAR, o ile posiada wyszczególnioną klasę główną, może stanowić osobną aplikację

Plik manifestu

<http://docs.oracle.com/javase/tutorial/deployment/jar/manifestindex.html>

- Archiwum JAR powinno zawierać plik manifestu umieszczony w ścieżce META-INF/MANIFEST.MF, który informuje o sposobie użycia, przeznaczeniu archiwum, wskazuje klasę główną jeśli archiwum jest wykonywalne itp.
- Większość współczesnych IDE dla Javy pozwala na szybkie tworzenie plików JAR i generowanie plików manifestu

Tworzenie pliku JAR w Eclipse – poprzez eksport projektu/projektów



Tworzenie własnych bibliotek

- Eksportując wybrany projekt/projekty do archiwum JAR można utworzyć własną bibliotekę z klasami, którą można wykorzystać w innych projektach
- Przykład – wyeksportowanie projektu z przykładami z poprzedniego wykładu (oprócz klas zawierał również zasoby w postaci plików graficznych, które również będą dostępne w utworzonym archiwum)

JAR File Specification

Define which resources should be exported into the JAR.



Select the resources to export:

- ☒ Wyklad6
 - ☒ src
 - ☒ pl.edu.pw.fizyka.pojava.wyklad6
 - ☒ pl.edu.pw.fizyka.pojava.wyklad6.animacje
 - ☒ pl.edu.pw.fizyka.pojava.wyklad6.animacje.obrazki
 - ☒ pl.edu.pw.fizyka.pojava.wyklad6.grafika
 - ☒ pl.edu.pw.fizyka.pojava.wyklad6.grafika.obrazki
 - ☒ .settings

- ☒ obrazek0.jpg
- ☒ obrazek1.jpg
- ☒ obrazek2.jpg
- ☒ obrazek3.jpg
- ☒ obrazek4.jpg

- ☐ Export generated class files and resources
- ☒ Export all output folders for checked projects
- ☐ Export Java source files and resources
- ☐ Export refactorings for checked projects. [Select refactorings...](#)

Select the export destination:

JAR file: E:\JAVA\wyklad6_jar.jar

Browse...

Options:

- ☒ Compress the contents of the JAR file
- ☐ Add directory entries
- ☐ Overwrite existing files without warning



< Back

Next >

Finish

Cancel

Korzystanie z bibliotek

- Przy uruchamianiu z linii poleceń:

java -jar nazwa_pliku.jar pakiet.NazwaKlasy

Np. dla biblioteki utworzonej wcześniej

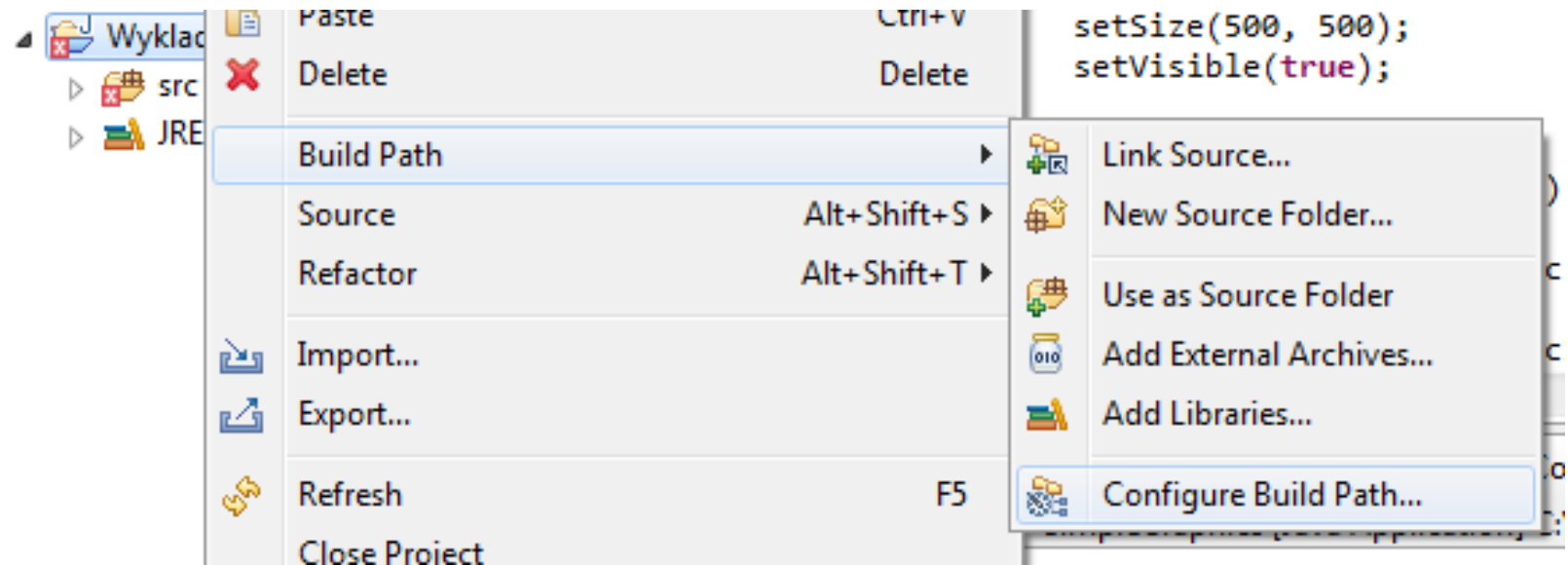
```
java -jar wyklad6_jar.jar  
    pl.edu.pw.fizyka.pojojava.wyklad6.grafika.SimpleGraphi  
cs
```

Dla bardziej złożonych projektów należy odpowiednio
ustawiać zmienną CLASSPATH... (

<http://docs.oracle.com/javase/tutorial/essential/environment/paths.html>)

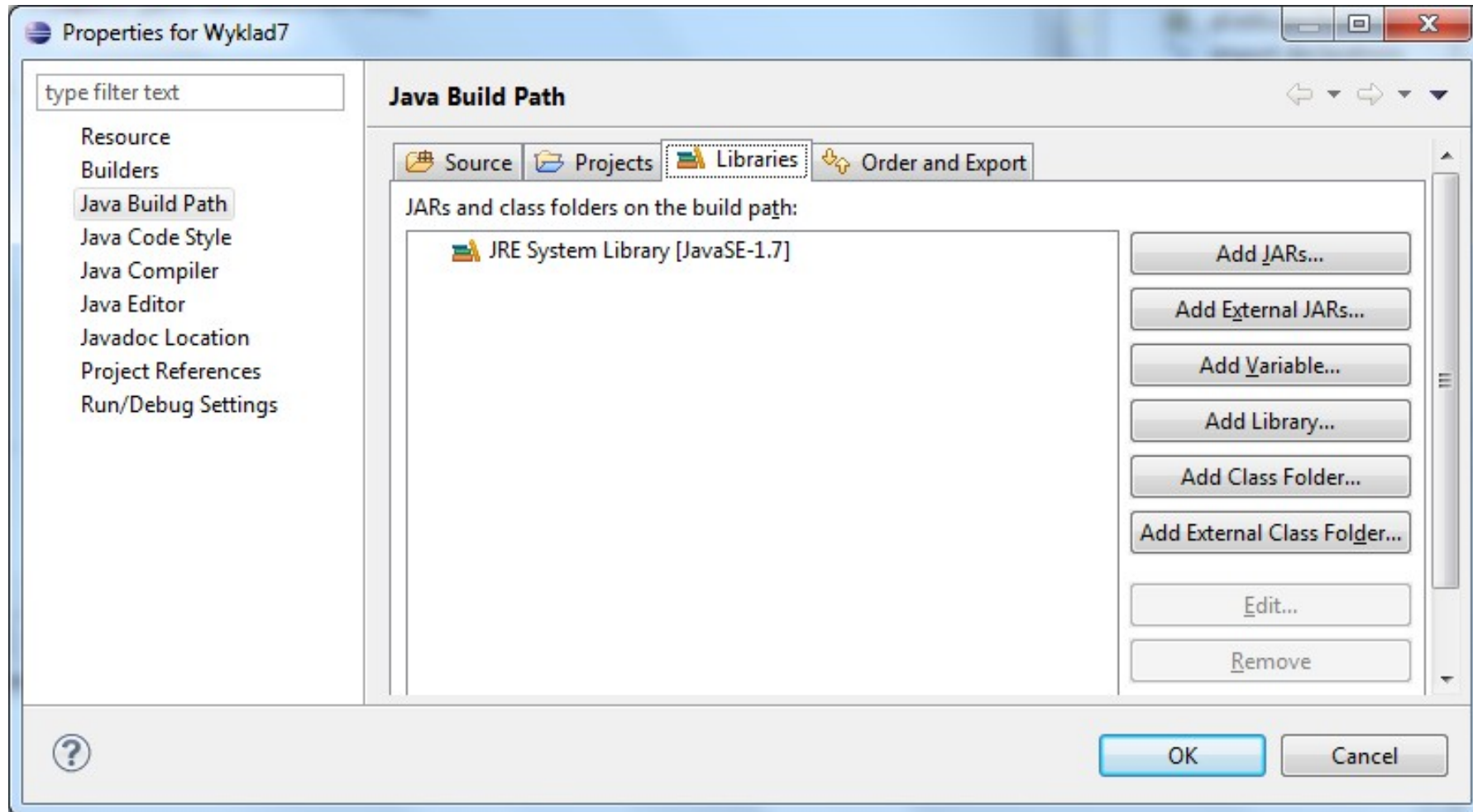
Korzystanie z bibliotek w Eclipse

- Najszybciej: Prawym klawiszem na nazwie projektu
->Build Path->Configure Build Path ...



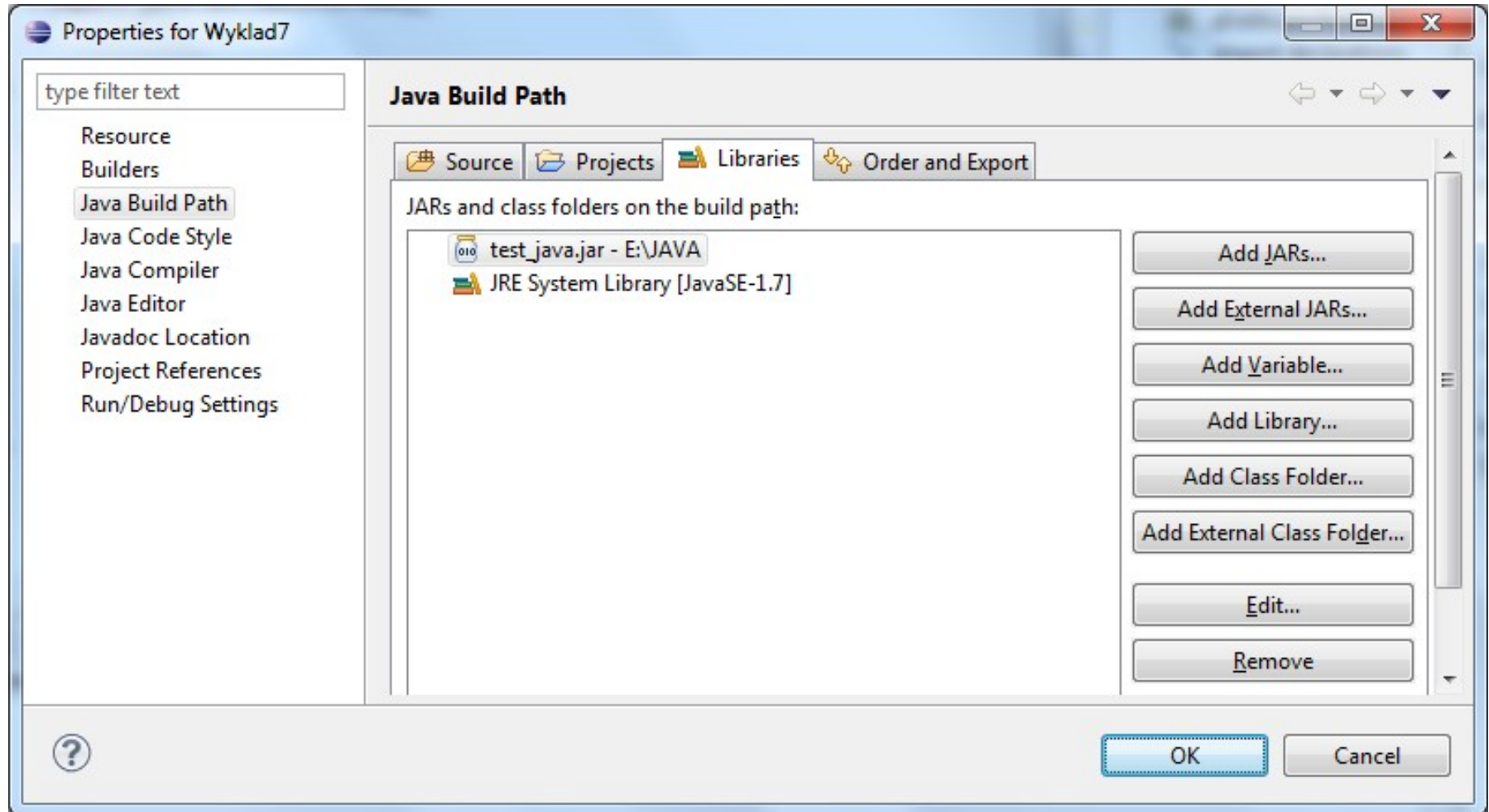
Korzystanie z bibliotek w Eclipse

- ... następnie w zakładce „Libraries” opcja „Add External JARs” i wskazanie lokalizacji biblioteki



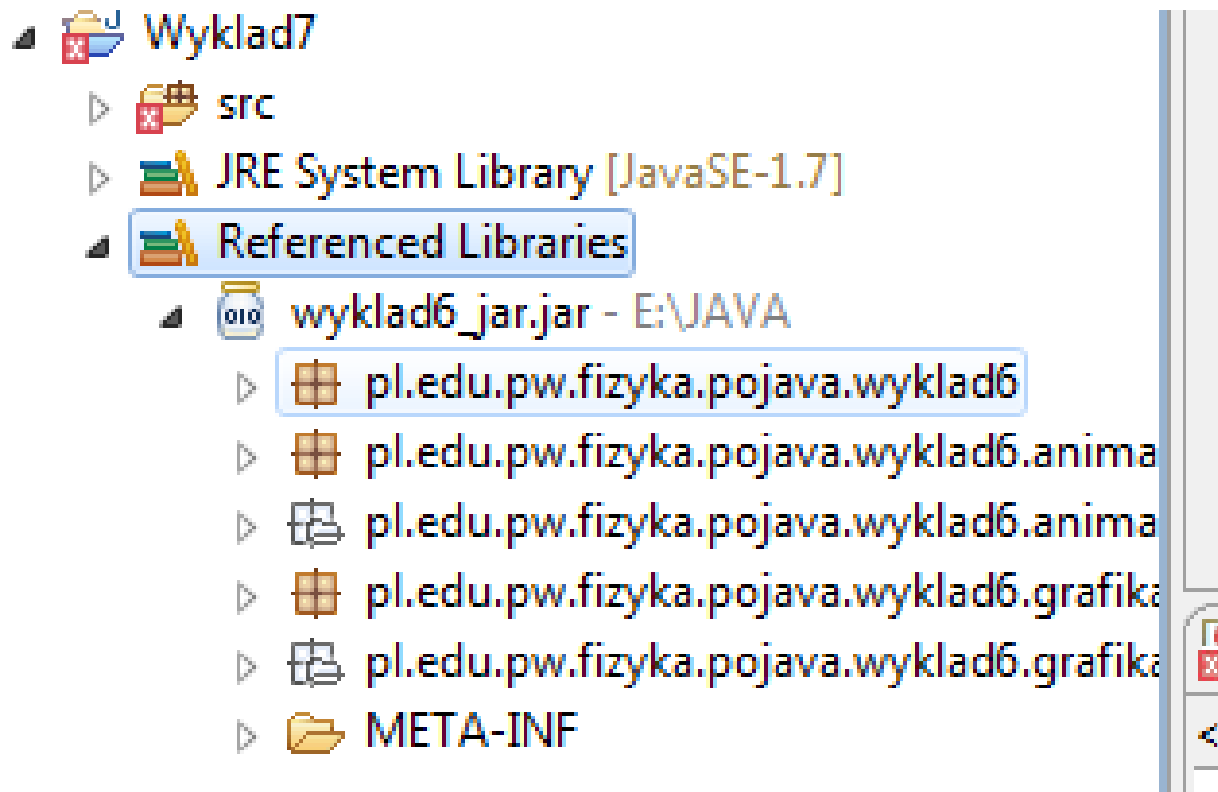
Korzystanie z bibliotek w Eclipse

- ... następnie w zakładce „Libraries” opcja „Add External JARs” i wskazanie lokalizacji biblioteki



Korzystanie z bibliotek w Eclipse

Po poprawnym zaimportowaniu w widoku Eksploratora Pakietów powinny się pojawić pakiety z dołączonej biblioteki:



Korzystanie z bibliotek w Eclipse

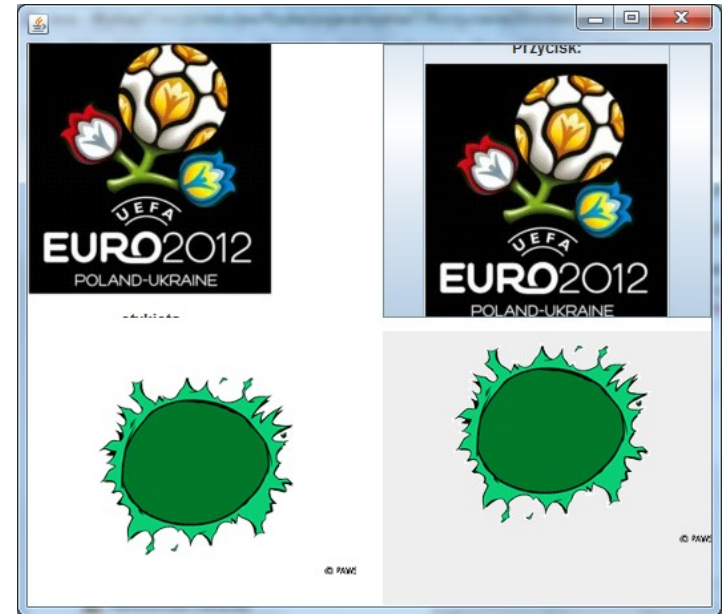
- Po dołączeniu biblioteki do projektu można do tworzonych klas importować klasy zawarte w bibliotece:

```
package pl.edu.pw.fizyka.pojava.wyklad7;  
  
import pl.edu.pw.fizyka.pojava.wyklad6.grafika.Logo;  
  
public class KorzystanieZBibliliteki {  
  
    public static void main(String[] args) {  
        Logo okno = new Logo();  
        okno.setVisible(true);  
    }  
  
}
```

- Po zaimportowaniu można korzystać z tych klas, metod, konstruktorów które były deklarowane jako publiczne

Korzystanie z bibliotek w Eclipse

```
package pl.edu.pw.fizyka.pojojava.wyklad7;  
  
import pl.edu.pw.fizyka.pojojava.wyklad6.grafika.Logo;  
  
public class KorzystanieZBiblilteki {  
  
    public static void main(String[] args) {  
        Logo okno = new Logo();  
        okno.setVisible(true);  
    }  
}
```

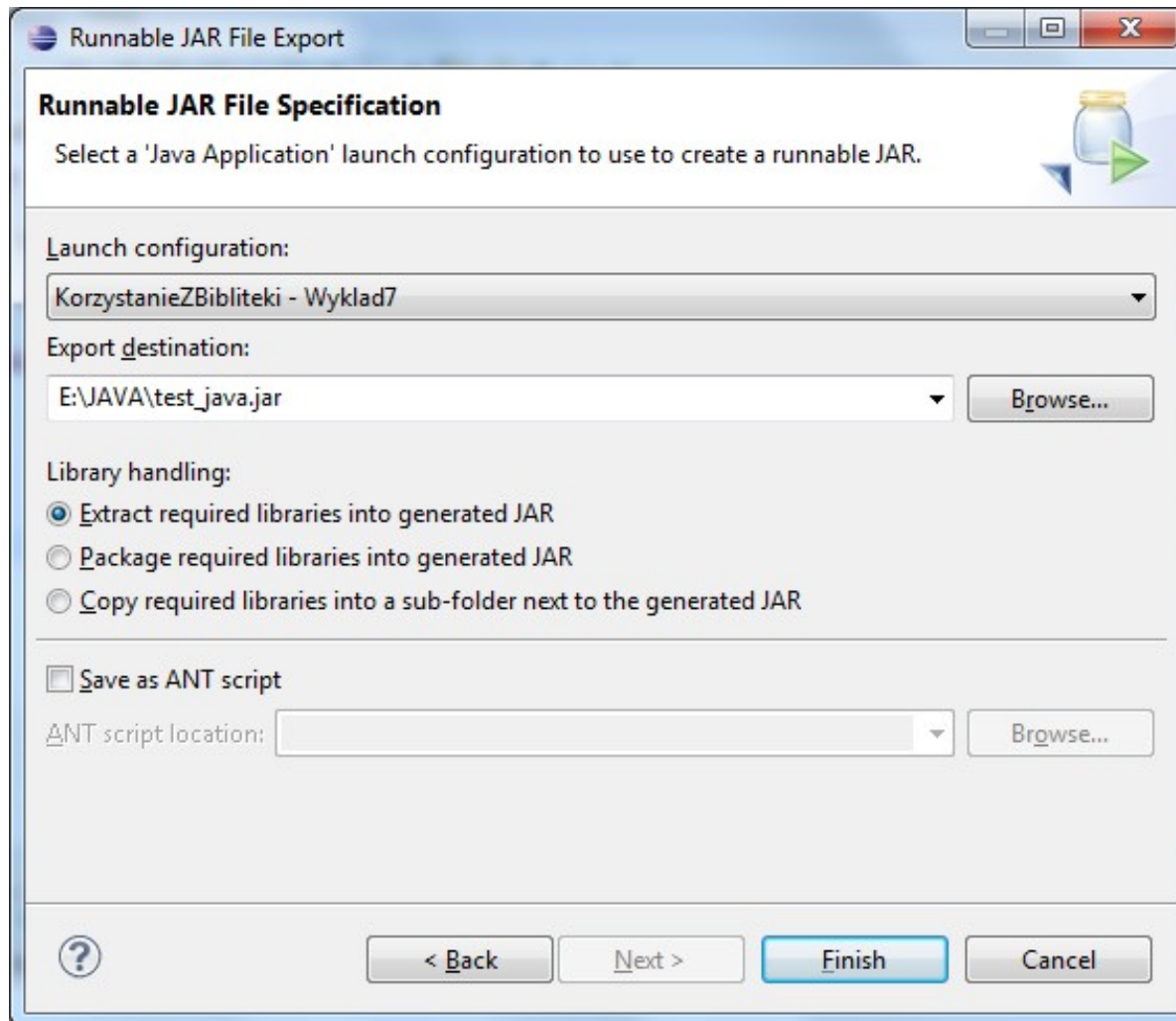


Obrazki bez problemów się wyświetlają ponieważ:

- zawarte są w zaimportowanej bibliotece
- w klasie Logo pobierane są metodą „getResource” – odwołującą się do zasobu z konkretnego pakietu/podpakietu

Tworzenie wykonywalnego pliku JAR

Obecnie w Eclipse możliwe jest szybkie utworzenie uruchamialnego pliku JAR, zawierającego w razie konieczności wszystkie używane biblioteki -> **Export -> Runnable JAR file**



Można wybrać jedną z konfiguracji uruchamiania, na podstawie której generowany jest odpowiedni plik manifestu

Można też wybrać sposób dodawania wymaganych bibliotek

Tworzenie wykonywalnego pliku JAR

- Tak utworzony plik JAR można uruchomić z linii poleceń:

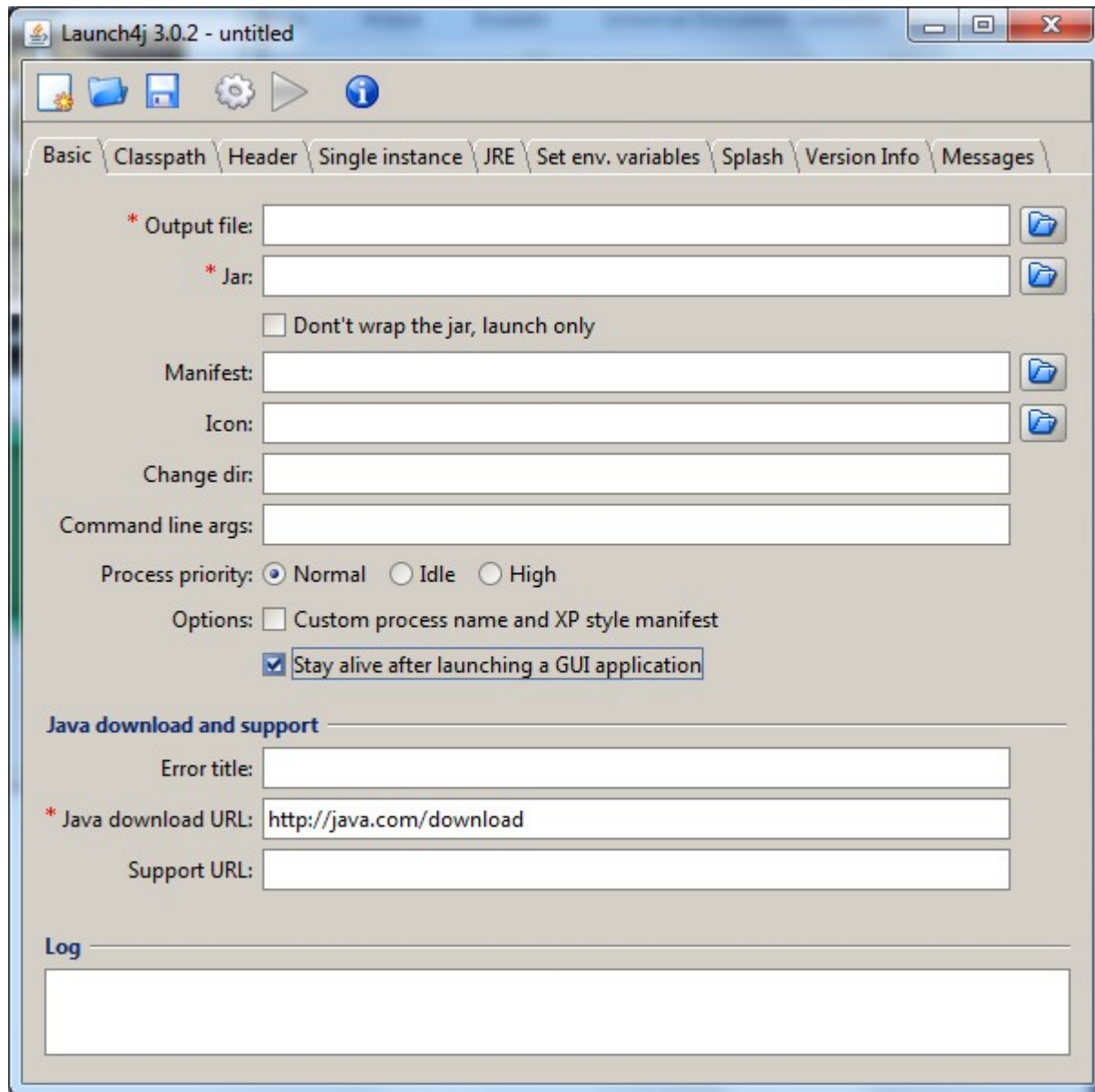
np.: **java -jar test_java.jar**

W zależności od konfiguracji systemu możliwe też może być uruchamianie aplikacji poprzez dwukrotne kliknięcie ikony pliku JAR lub poprzez menu kontekstowe

Tworzenie plików EXE

- Na bazie wykonywalnych plików JAR możliwe jest utworzenie plików wykonywalnych dla Windows, które nie tylko pozwalałyby na uruchomienie aplikacji, ale też sprawdzałyby czy zainstalowana jest odpowiednia wirtualna maszyna Javy, w razie konieczności proponowałyby ściągnięcie Javy
- Możliwe jest nawet stworzenie instalatora (np. programem Nullsoft Scriptable Install System - <http://nsis.sourceforge.net/>)

Tworzenie plików EXE - Launch4J Executable Wrapper <http://sourceforge.net/projects/launch4j/>



Tworzenie plików EXE - Launch4J Executable Wrapper <http://sourceforge.net/projects/launch4j/>

Program posiada sporo opcji i możliwości konfigurowania uruchomienia, ale do utworzenia pliku EXE z pliku wykonywalnego JAR wystarczy:

1. zdefiniowanie ścieżki pliku wyjściowego EXE
2. podanie ścieżki do pliku JAR ()
3. podanie minimalnej wersji JRE (np. 1.0.0 lub 1.7.0) – w zakładce JRE
4. zapisanie konfiguracji
5. Kliknięcie ikony Build Wrapper

Podanie nazw plików JAR i EXE

The screenshot shows the Launch4j 3.0.2 - untitled window. The 'Basic' tab is selected, showing fields for 'Output file' (E:\JAVA\plik.exe), 'Jar' (E:\JAVA\test_java.jar), 'Manifest', 'Icon', 'Change dir', and 'Command line args'. There are also radio buttons for 'Process priority' (Normal, Idle, High) and checkboxes for 'Options' (Custom process name and XP style manifest, Stay alive after launching a GUI application). The 'Java download and support' section includes fields for 'Error title', 'Java download URL' (http://java.com/download), and 'Support URL'.

Launch4j 3.0.2 - untitled

Basic Classpath Header Single instance JRE Set env. variables Splash Version Info Messages

* Output file: E:\JAVA\plik.exe

* Jar: E:\JAVA\test_java.jar

☐ Don't wrap the jar, launch only

Manifest:

Icon:

Change dir:

Command line args:

Process priority: ☒ Normal ☐ Idle ☐ High

Options: ☐ Custom process name and XP style manifest
☒ Stay alive after launching a GUI application

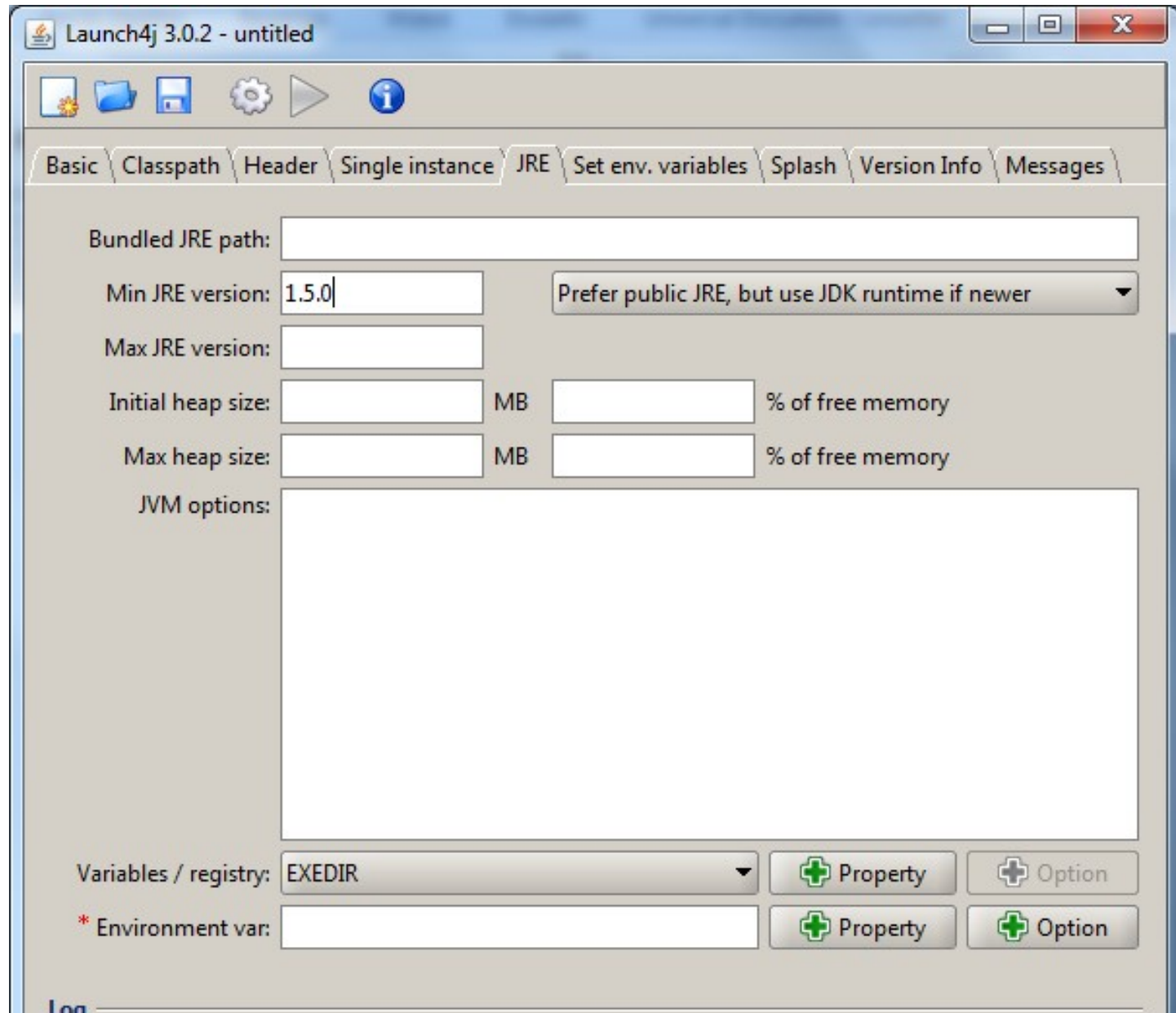
Java download and support

Error title:

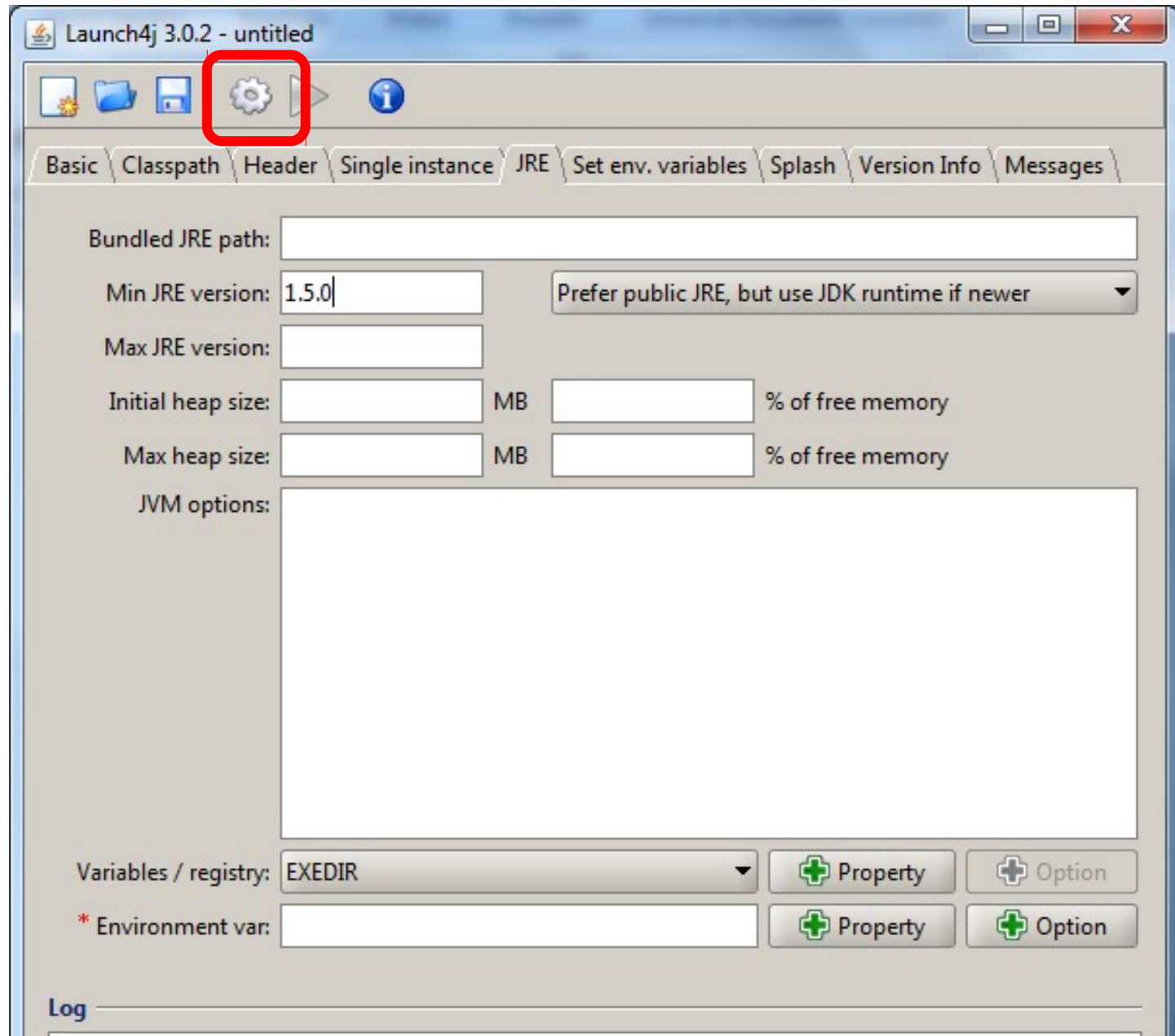
* Java download URL: http://java.com/download

Support URL:

Ustawienie minimalnej wersji JRE



... po czym można klikać ikonę BuildWrapper



Initial heap size: MB % of free memory

Max heap size: MB % of free memory

JVM options:

Variables / registry: EXEDIR

* Environment var:

Log

Wrapping
Successfully created E:\JAVA\plik.exe

Wynik udanego tworzenia pliku EXE

Wykorzystanie innych bibliotek

- Istnieje bardzo dużo różnych bibliotek stworzonych w Javie – część jest odpłatna, ale jest też ogromny wybór darmowych bibliotek o rozmaitej funkcjonalności.
- Na stronie <http://java-source.net/> zebrane są biblioteki open-source dla Javy

Biblioteka JFreeChart -

<http://www.jfree.org/jfreechart/>



JFreeChart

[HOME](#)[JFREECHART](#)[SAMPLES](#)[DOWNLOAD](#)[API DOCS](#)[DEVELOPER GUIDE](#)[SUPPORT](#)[FAQ](#)

Welcome To JFreeChart!

JFreeChart is a free 100% Java chart library that makes it easy for developers to display professional quality charts in their applications. JFreeChart's extensive feature set includes:

- a consistent and well-documented API, supporting a wide range of chart types;
- a flexible design that is easy to extend, and targets both server-side and client-side applications;
- support for many output types, including Swing components, image files (including PNG and JPEG), and vector graphics file formats (including PDF, EPS and SVG);
- JFreeChart is "open source" or, more specifically, **free software**. It is distributed under the terms of the **GNU Lesser General Public Licence** (LGPL), which permits use in proprietary applications.

For a closer look at JFreeChart, please try our **JFreeChart Demo (web start)** or browse the **Samples** page.

Latest News

1 Jan 2013

Happy New Year from the JFree team!

The Project

The JFreeChart project was founded twelve years ago, in February 2000, by David



[Follow @jfreechart](#)

Links

[JFreeChart Forum](#)

[Project Page at SourceForge](#)

[JFreeChart Developer Guide \(\\$\)](#)

[API Documentation](#)

Biblioteka JFreeChart

- Aby tworzyć proste wykresy do projektu należy dodać biblioteki JFreeChart i JCommon
- W archiwum z JFreeChart w katalogu lib jest więcej plików jar, m.in. biblioteka IText pozwalająca na tworzenie plików PDF

Wykres PieChart – PieChartDemo.java

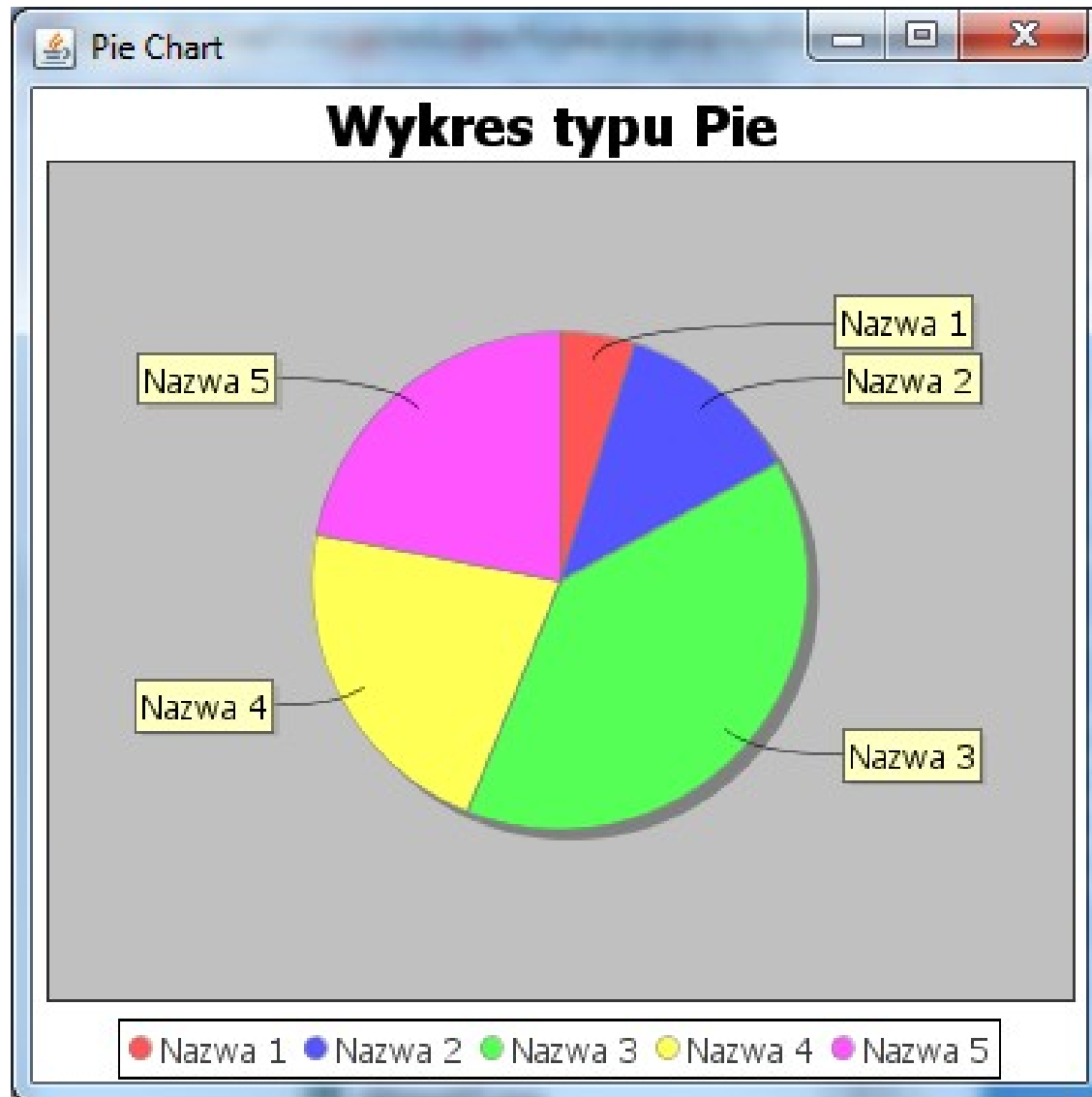
```
DefaultPieDataset dane = new DefaultPieDataset();

dane.setValue("Nazwa 1", 10); //wartosci
dane.setValue("Nazwa 2", 25);
dane.setValue("Nazwa 3", 80);
dane.setValue("Nazwa 4", 45);
dane.setValue("Nazwa 5", 45);

//Tworzymy wykres JFreeChart typu PieChart
JFreeChart chart = ChartFactory.createPieChart
    ("Wykres typu Pie ", // Tytuł wykresu
    dane, // dane typu PieDataset
    true, // legenda
    true, // tooltips
    false // Configure chart to generate URLs?
);

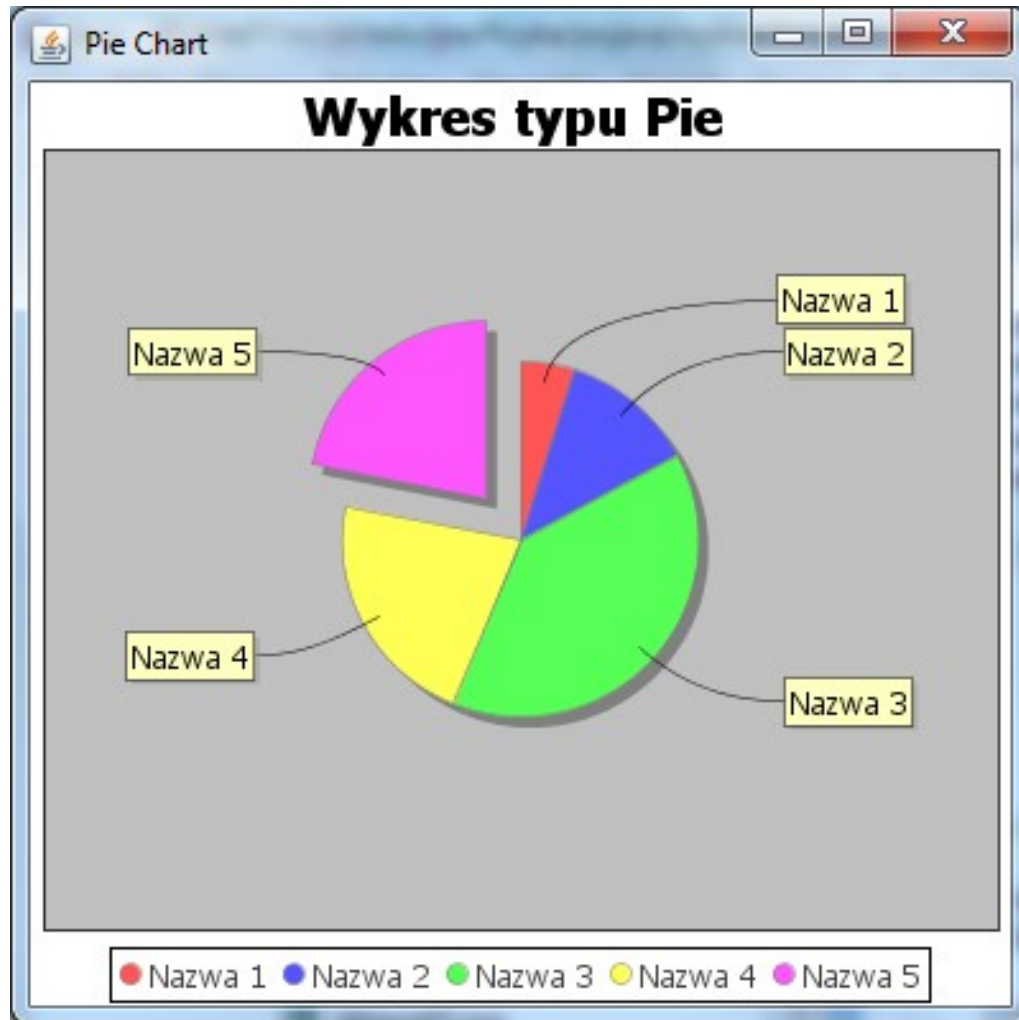
ChartFrame frame=new ChartFrame("Pie Chart",chart);
frame.setVisible(true);
frame.setSize(400,400);
```

Wykres PieChart – PieChartDemo.java



Wykres PieChart – PieChartDemo.java

```
PiePlot plot = (PiePlot) chart.getPlot();  
plot.setExplodePercent("Section A", 0.30); // wycinek z  
wykresu
```

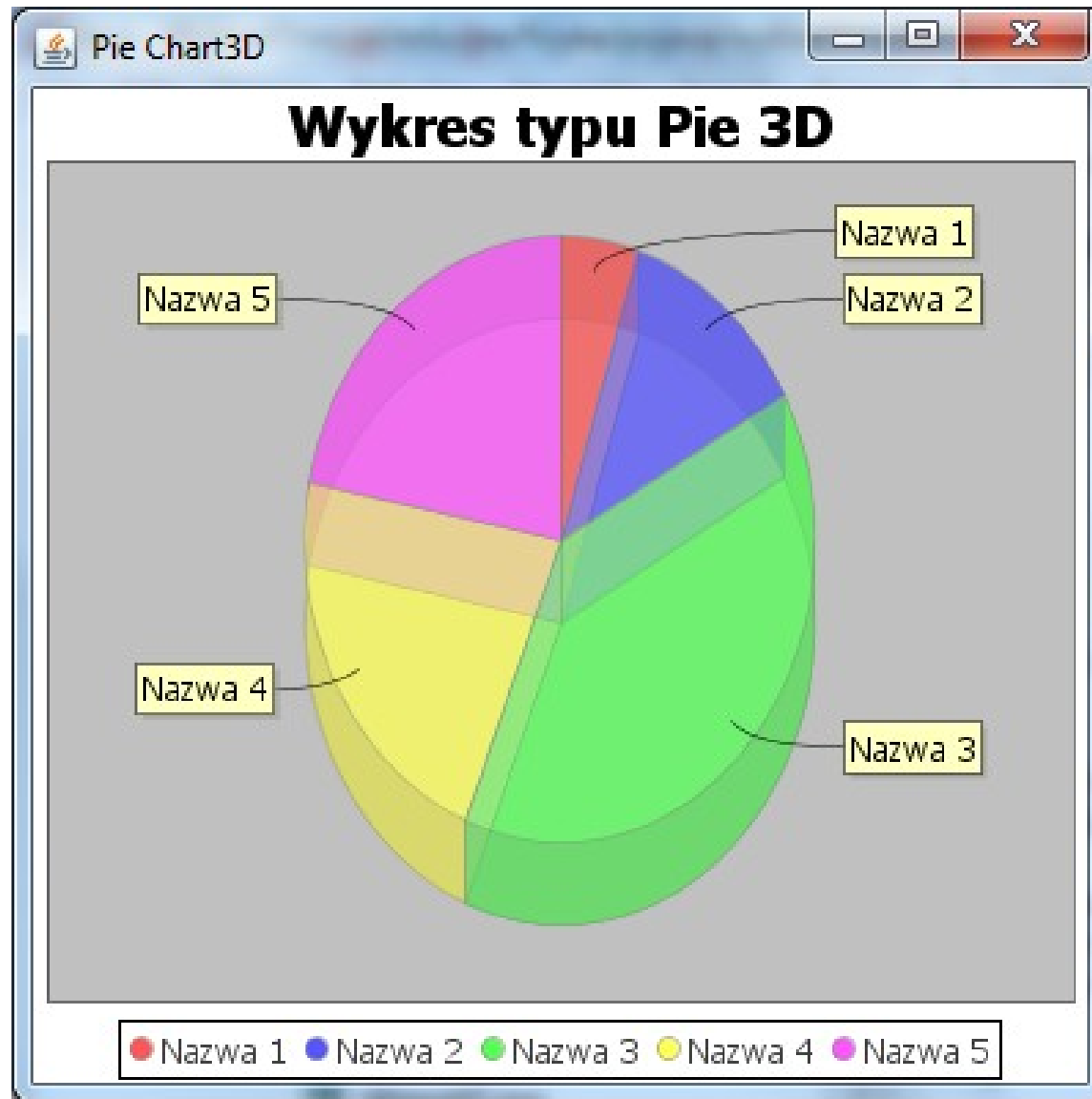


PieChartDemo3D.java

```
//Tworzymy wykres JFreeChart typu PieChart3D - podobnie jak
PieChart
JFreeChart chart = ChartFactory.createPieChart3D
    ("Wykres typu Pie ", // Tytuł wykresu
        dane, // dane typu PieDataset
        true, // legenda
        true, // tooltips
        false // Configure chart to generate URLs?
    );

// modyfikowanie wykresu:
PiePlot plot = (PiePlot) chart.getPlot();
plot.setForegroundAlpha(0.5f); // dodawanie przezroczystosci
```

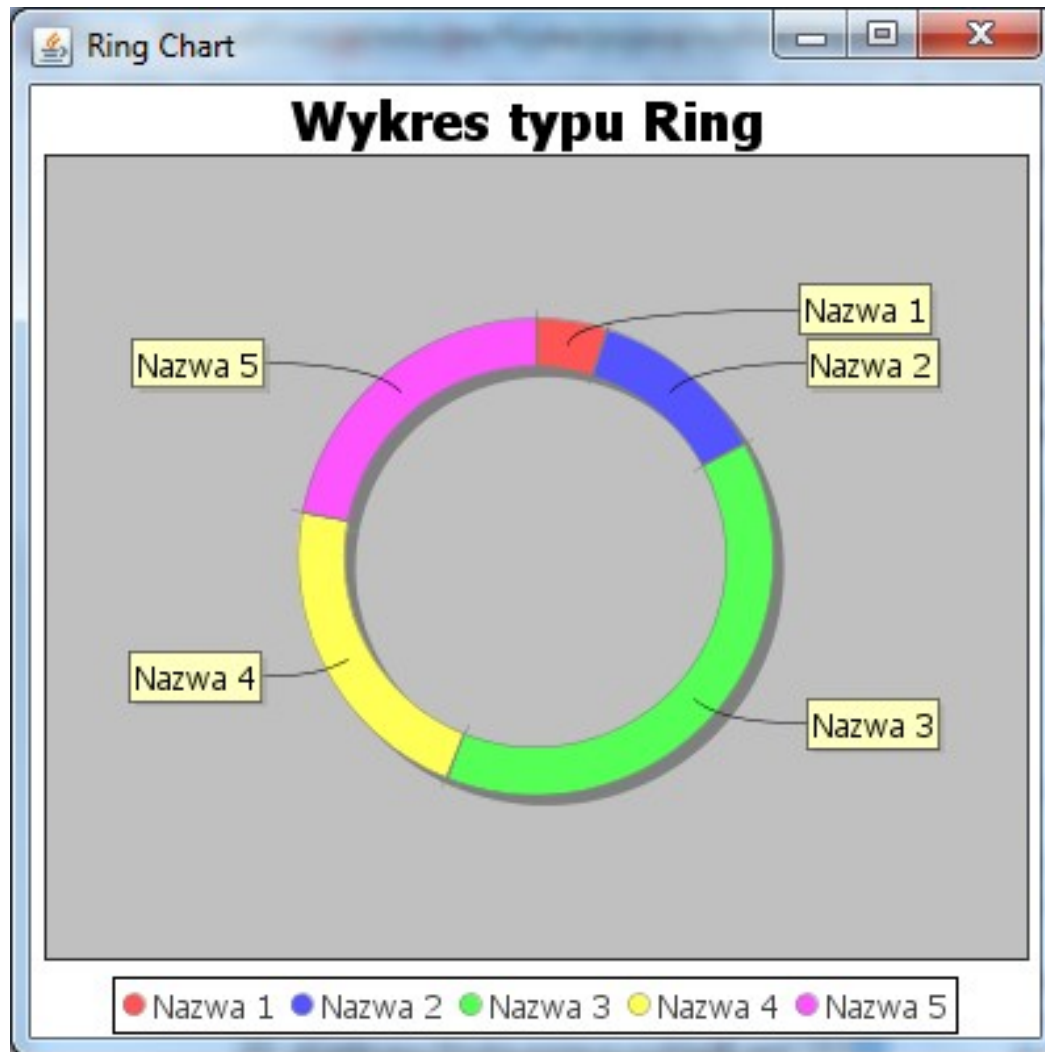
PieChartDemo3D.java



RingChartDemo.java

//jedyna różnica w tworzeniu:

```
JFreeChart chart = ChartFactory.createRingChart  
("Wykres typu Pie ", ...
```

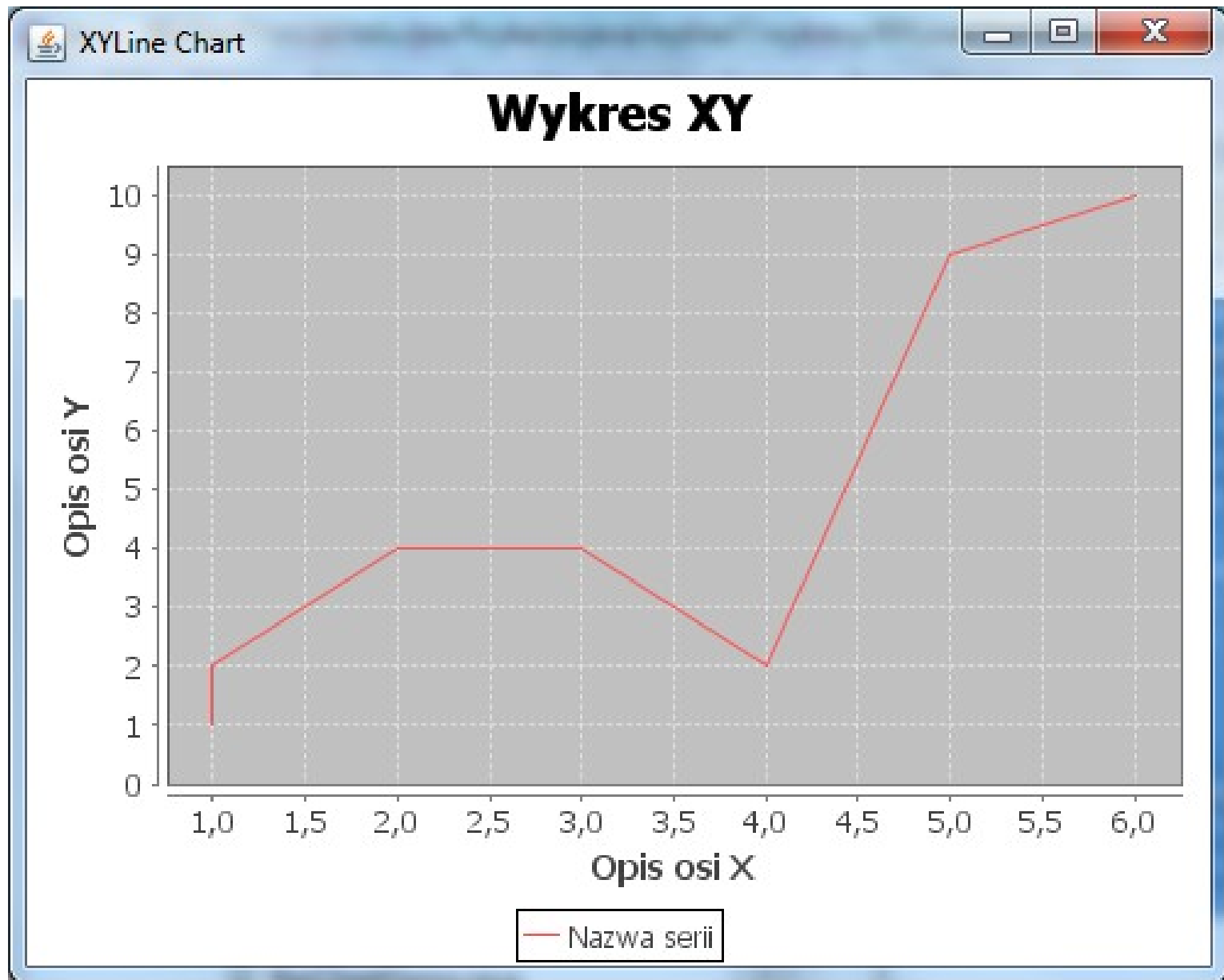


XYLineChartDemo.java

```
XYSeries series = new XYSeries("Nazwa serii");
series.add(1, 1);
series.add(1, 2);
series.add(2, 4);
...
series.add(6, 10);

XYSeriesCollection dataset = new XYSeriesCollection();
dataset.addSeries(series);
//Tworzymy wykres XY
JFreeChart chart = ChartFactory.createXYLineChart(
    "Wykres XY", //Tytuł
    "Opis osi X", // opisy osi
    "Opis osi Y",
    dataset, // Dane
    PlotOrientation.VERTICAL, // Orjentacja
    wykresu
    true, // legenda
    true, // tooltips
    false
);
```


XYLineChartDemo.java



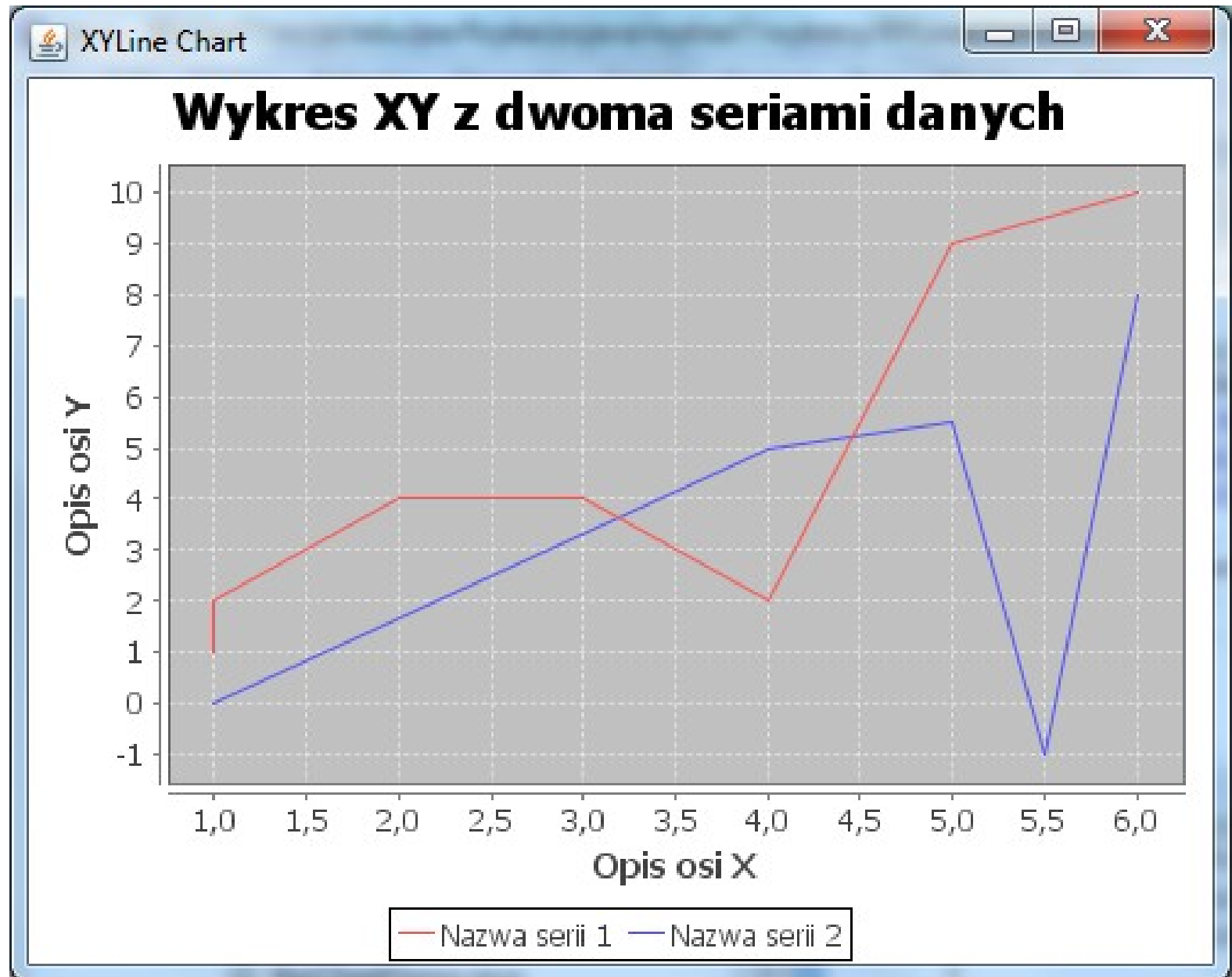
XYLineChartDemo2.java – dwie serie

```
//Tworzenie poszczególnych serii
XYSeries series = new XYSeries("Nazwa serii 1");
series.add(1, 1);
...
series.add(6, 10);
XYSeries series2 = new XYSeries("Nazwa serii 2");
series2.add(1,0);
...
series2.add(6, 8);

//Tworzenie kolekcji serii
XYSeriesCollection dataset = new XYSeriesCollection();
//dodawanie kolejnych serii do kolekcji
dataset.addSeries(series);
dataset.addSeries(series2);

// oczywiscie serie mozna usuwac:
//dataset.removeSeries(series2); // stosujac nazwe serii
//dataset.removeSeries(0); // stosujac numer serii
//dataset.removeAllSeries(); // lub usunac wszystkie serie;
```

XYLineChartDemo2.java – dwie serie



BarChartDemo.java

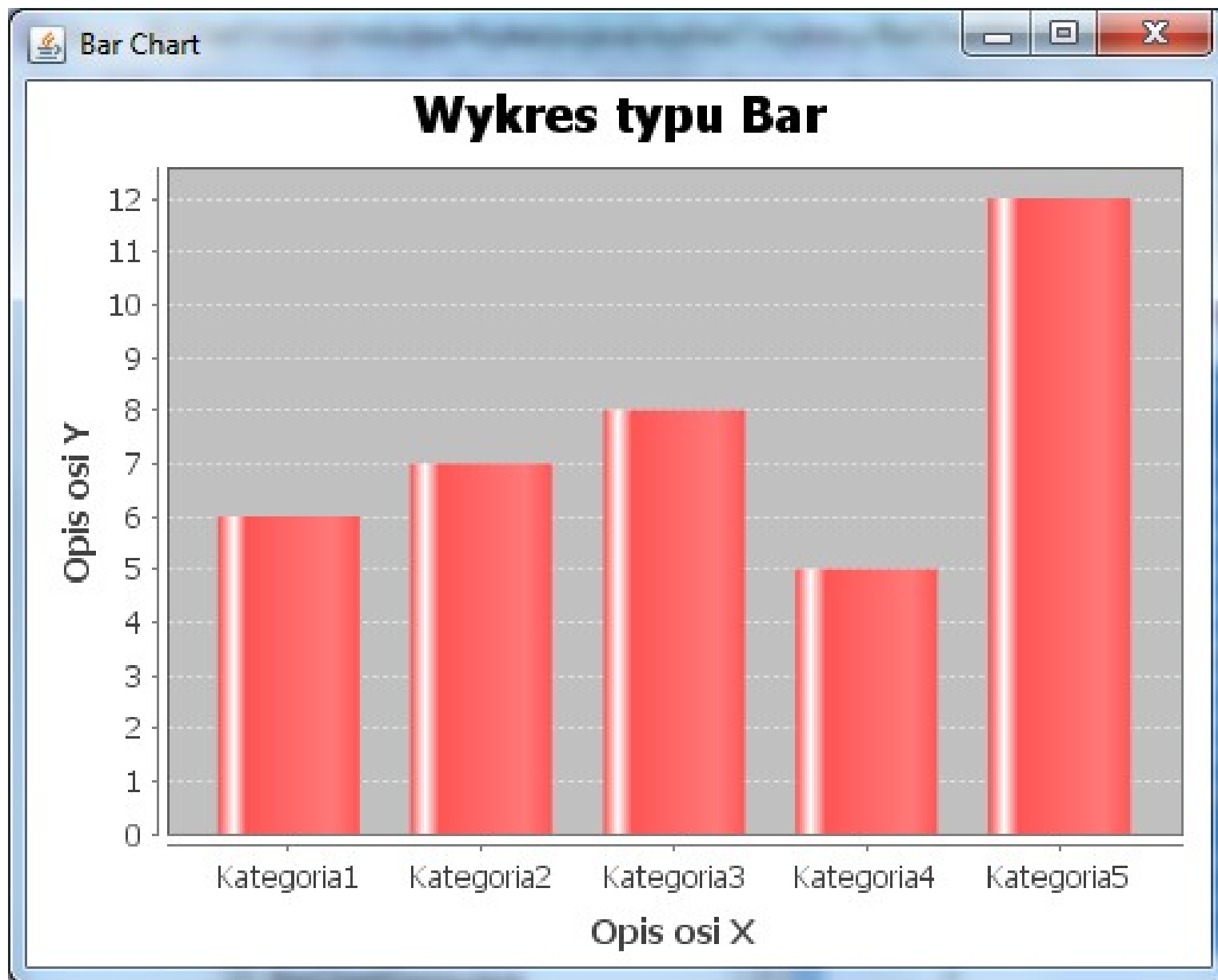
```
DefaultCategoryDataset dataset = new
DefaultCategoryDataset();

dataset.setValue(6, "Seria1", "Kategoria1");
dataset.setValue(7, "Seria1", "Kategoria2");
dataset.setValue(8, "Seria1", "Kategoria3");
dataset.setValue(5, "Seria1", "Kategoria4");
dataset.setValue(12, "Seria1", "Kategoria5");
// Tworzy wykres typu Bar - słupkowy

JFreeChart chart = ChartFactory.createBarChart("Wykres typu
Bar",
"Opis osi X", "Opis osi Y", dataset,
PlotOrientation.VERTICAL,
false, true, false);
//parametry podobnie jak w poprzednich przykładach

ChartFrame frame1=new ChartFrame("Bar Chart",chart);
frame1.setVisible(true);
frame1.setSize(500,400);
```

BarChartDemo.java



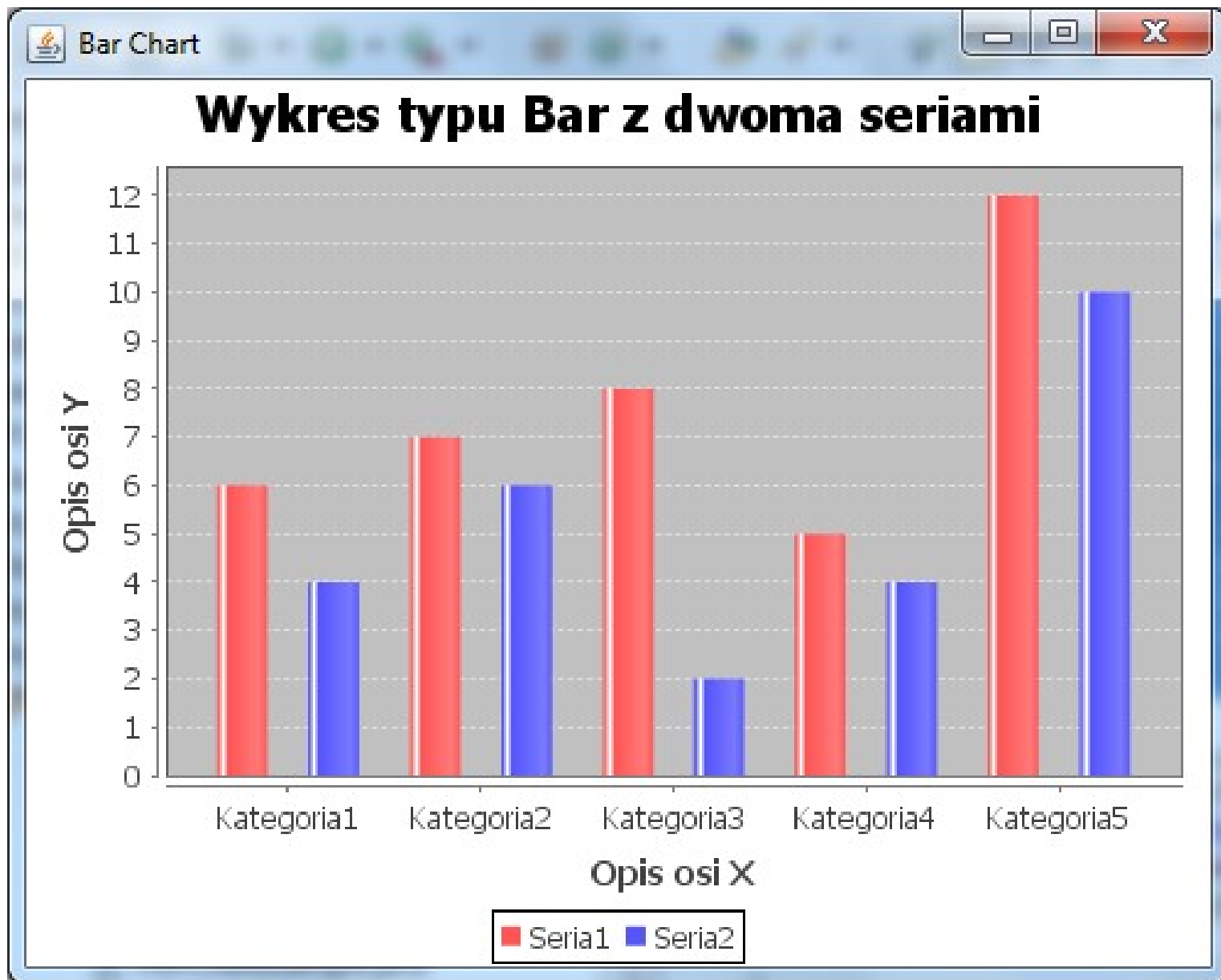
BarChartDemo2.java – dwie serie

```
DefaultCategoryDataset dataset = new  
DefaultCategoryDataset();
```

```
dataset.setValue(6, "Seria1", "Kategoria1");  
dataset.setValue(7, "Seria1", "Kategoria2");  
dataset.setValue(8, "Seria1", "Kategoria3");  
dataset.setValue(5, "Seria1", "Kategoria4");  
dataset.setValue(12, "Seria1", "Kategoria5");
```

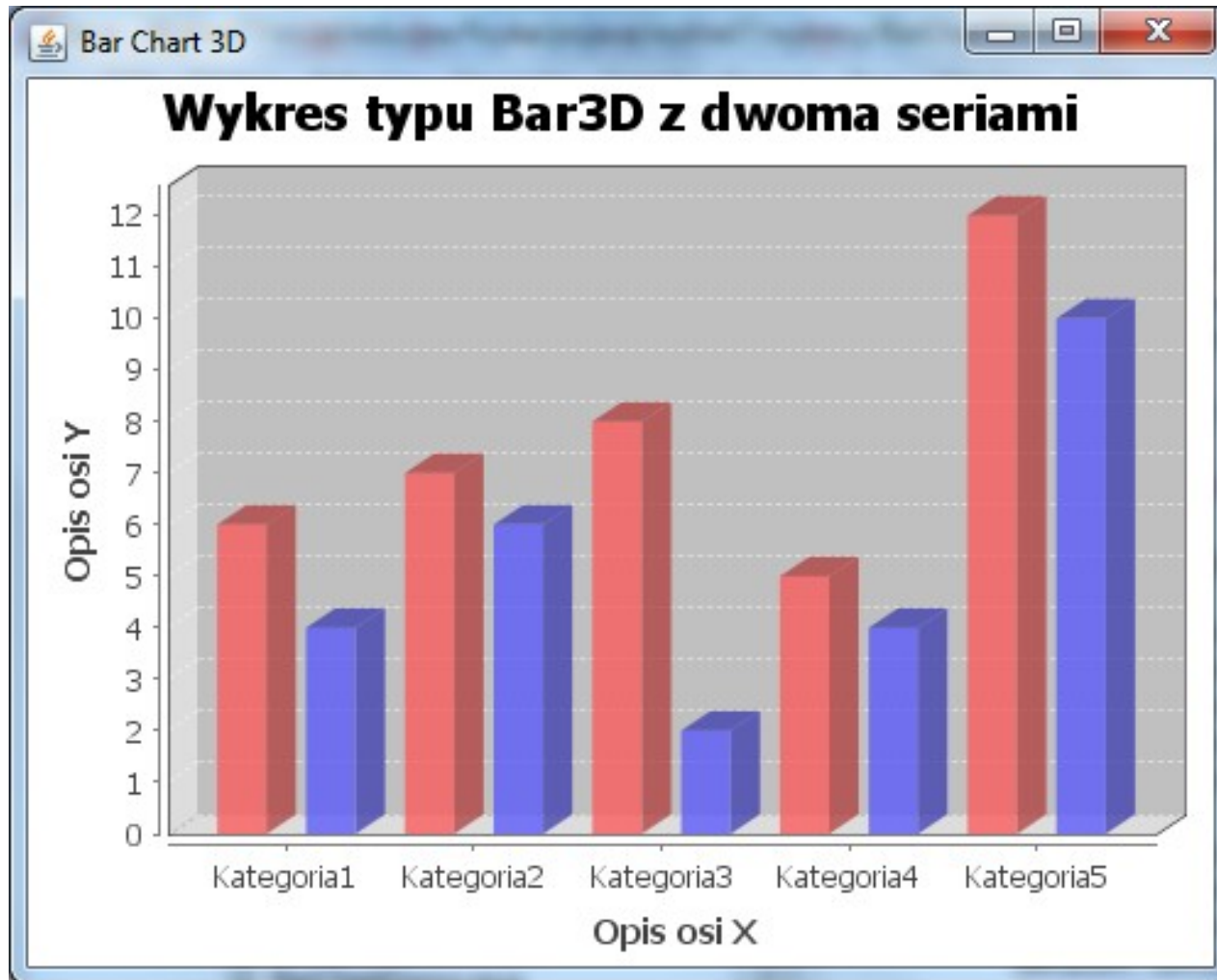
```
dataset.setValue(4, "Seria2", "Kategoria1");  
dataset.setValue(6, "Seria2", "Kategoria2");  
dataset.setValue(2, "Seria2", "Kategoria3");  
dataset.setValue(4, "Seria2", "Kategoria4");  
dataset.setValue(10, "Seria2", "Kategoria5");
```

BarChartDemo2.java – dwie serie



BarChart3DDemo.java, jedyna różnica:

```
JFreeChart chart = ChartFactory.createBarChart3D("Wykres  
typu Bar3D z dwoma seriami", "Opis osi X", "Opis osi Y",  
dataset, PlotOrientation.VERTICAL, false, true, false);
```



TimeSeriesChartDemo.java

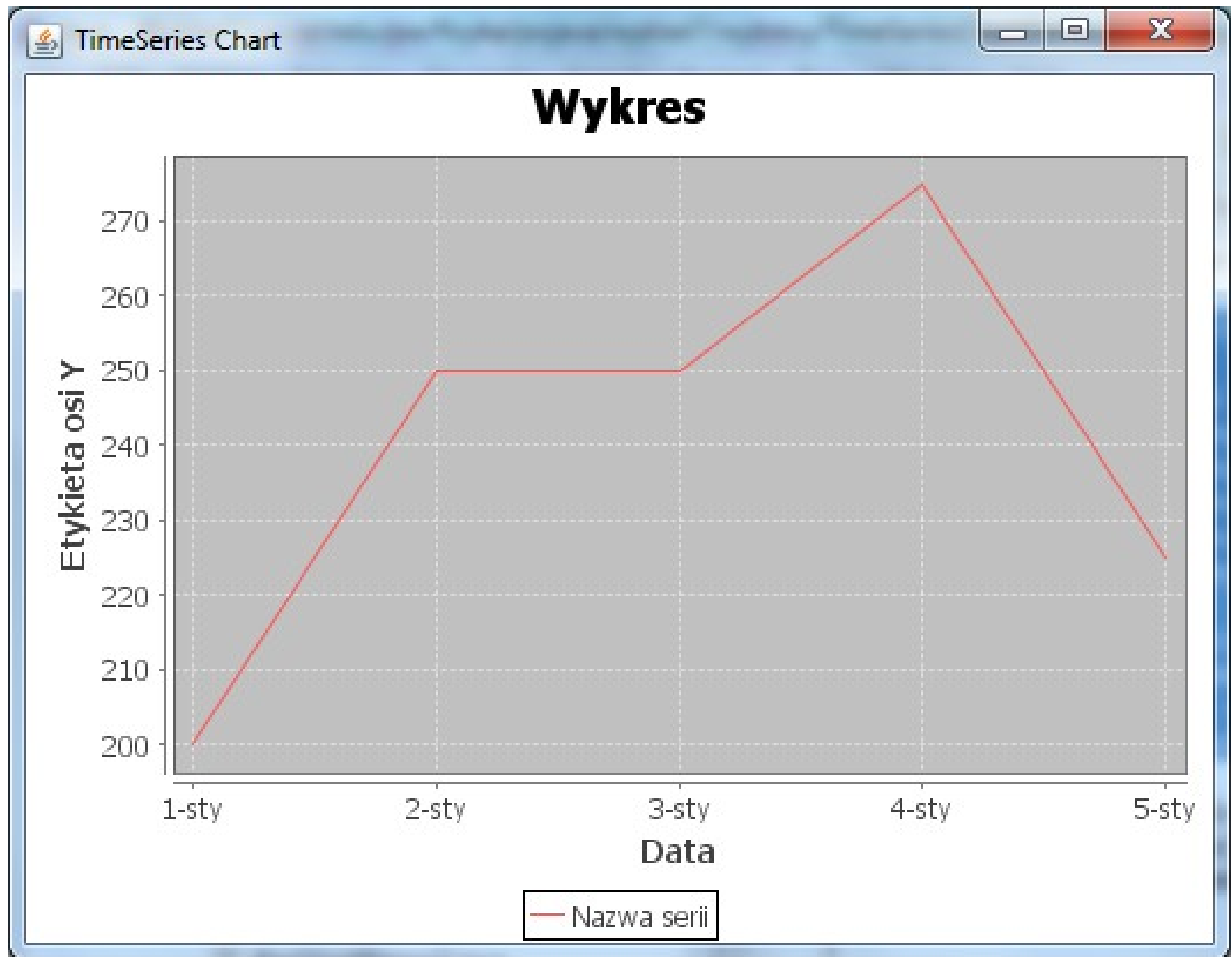
```
//tworzenie kolekcji serii
TimeSeriesCollection dataset = new
TimeSeriesCollection();

//tworzenie serii
TimeSeries seria1 = new TimeSeries("Nazwa serii");
seria1.add(new Day(1, 1, 2013), 200);
seria1.add(new Day(2, 1, 2013), 250);
seria1.add(new Day(3, 1, 2013), 250);
seria1.add(new Day(4, 1, 2013), 275);
seria1.add(new Day(5, 1, 2013), 225);

//dodawanie serii
dataset.addSeries(seria1);

//Tworzenie wykresu typu TimeSeries
JFreeChart chart = ChartFactory.createTimeSeriesChart
("Wykres", "Data", "Etykieta osi
Y", dataset, true, true, false);
```

TimeSeriesChartDemo.java



TimeSeriesChartDemo2.java – dwie serie

```
TimeSeriesCollection dataset = new TimeSeriesCollection();
```

```
TimeSeries seria1 = new TimeSeries("Nazwa serii");
```

```
seria1.add(new Day(1, 1, 2013), 200);
```

```
seria1.add(new Day(2, 1, 2013), 250);
```

```
...
```

```
seria1.add(new Day(5, 1, 2013), 225);
```

```
TimeSeries seria2 = new TimeSeries("Nazwa serii2");
```

```
seria2.add(new Year(2012), 200);
```

```
seria2.add(new Year(2013), 250);
```

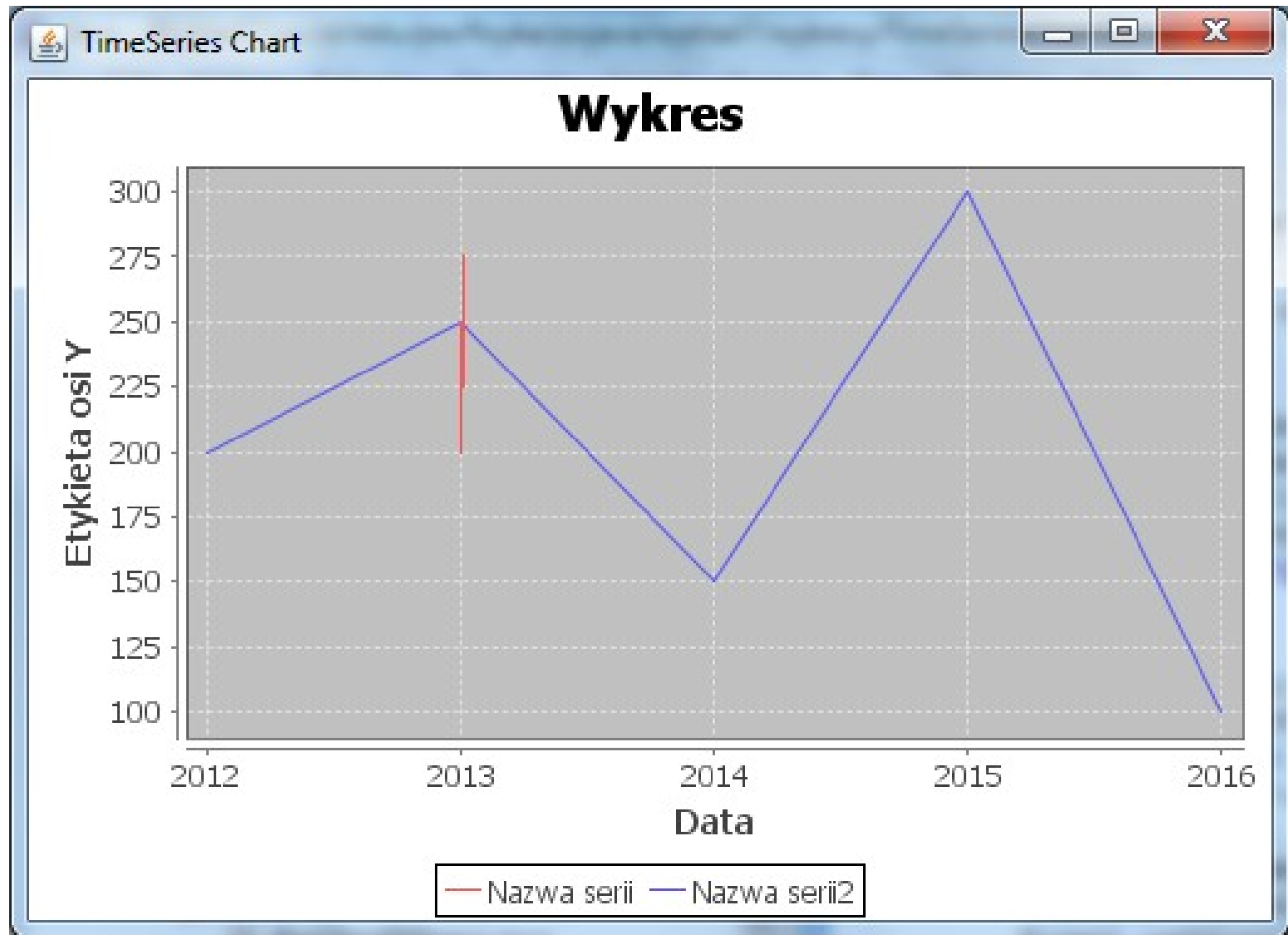
```
..
```

```
seria2.add(new Year(2016), 100);
```

```
dataset.addSeries(seria1);
```

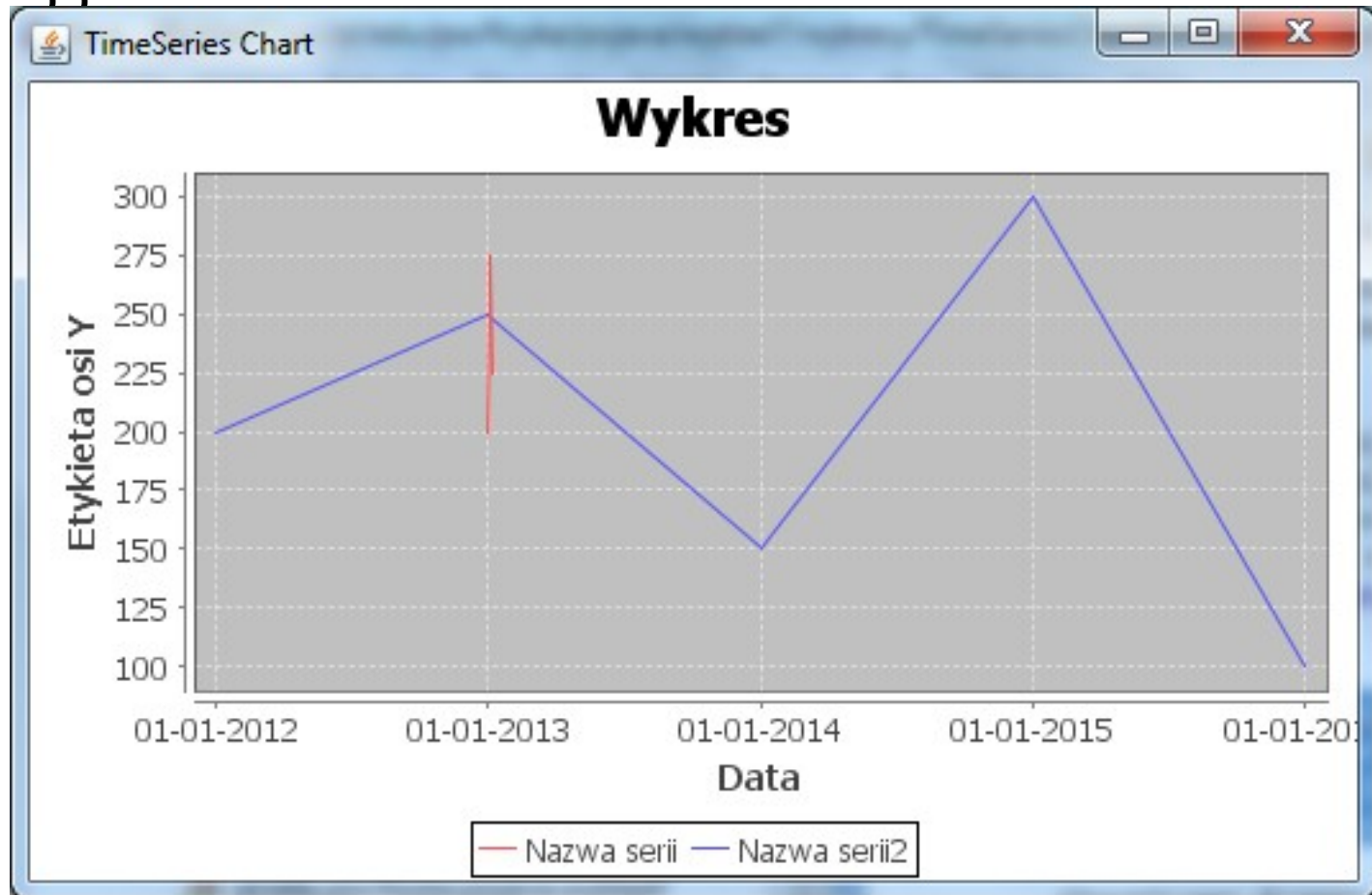
```
dataset.addSeries(seria2);
```

TimeSeriesChartDemo2.java – dwie serie



Wymuszanie określonego formatowania osi czasu

```
XYPlot plot = chart.getXYPlot();  
DateAxis axis = (DateAxis) plot.getDomainAxis();  
axis.setDateFormatOverride(new SimpleDateFormat("dd-MM-  
yyyy"));
```



WyswietlanieWykresow.java

Kilka przykładów wyświetlania wykresów

```
// szybkie wyswietlanie wykresu przy pomocy klasy
ChartFrame
ChartFrame frame = new ChartFrame("Szybkie wyswietlanie wykresu -
klasa ChartFrame", lineGraph);
frame.pack();
frame.setVisible(true);

// Zapisywanie wykresu do pliku JPG:
try {
    ChartUtilities.saveChartAsJPEG(new File("LineGraph.jpg"),
                                   lineGraph, 800, 600);
} catch (Exception e) {
    System.out.println("Problem z zapisem wykresu do pliku");
}
```

WyswietlanieWykresow.java

```
// WYSWIETLANIE WYKRESOW W OKNIE SWING
// Tworzenie okienka Swing:
JFrame jframe = new JFrame("Przeskaluj okno oraz
kliknij prawym klawiszem myszy na obu
wykresach...");

jframe.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
jframe.getContentPane().setLayout(new
GridLayout( 2,2));
jframe.setSize(640,480);
// Dodawanie wykresu jako obraz - klasa BufferedImage
BufferedImage image =
    lineGraph.createBufferedImage(300,200);
JLabel lblChart = new JLabel();
lblChart.setIcon(new ImageIcon(image));
jframe.getContentPane().add(lblChart);
jframe.getContentPane().add(new JLabel("<<< wykres
odany jako obraz (Image)"));
// Dodawanie wykresu przy pomocy klasy ChartPanel
ChartPanel chartPanel = new ChartPanel(lineGraph);
jframe.getContentPane().add(chartPanel);
jframe.getContentPane().add(new JLabel("<<< wykres"));
```

PDFChartExample.java


- Przykład pokazujący jak zapisywać wykresy do pliku PDF – wymagane dołączenie do projektu biblioteki IText


Na stronie biblioteki iText jest znacznie więcej przykładów jej użycia: <http://itextpdf.com/book/examples.php>

Web

itextpdf.com/book/examples.php

★

 Szukaj w Google






About us | News | Contact

iText®

Licenses

Support

  1.4k

 Search Go

You are here: [Home](#) > [Support](#) > [Book](#) > [Examples](#)

Chapters

Chapter 1
Introducing PDF and iText

Chapter 2
Using iText's basic building blocks

Chapter 3
Adding content at absolute positions

Chapter 4
Organizing content in tables

Chapter 5
Table, cell, and page events


Chapter 6
Working with existing PDFs


Chapter 7
Making documents interactive


List of the examples used in the book


For a quick look at one example and the resources that are needed, please use the list below.


Extra: helper classes


 **DatabaseConnection**
Keywords: Database connection, Hypersonic SQL, mySql


 **HsqldbConnection**
Keywords: Database connection, Hypersonic SQL


 **CreateHsqldbTables**
Keywords: Database connection, Database tables, Hypersonic SQL


 **MySqlConnection**
Keywords: Database connection, mySql

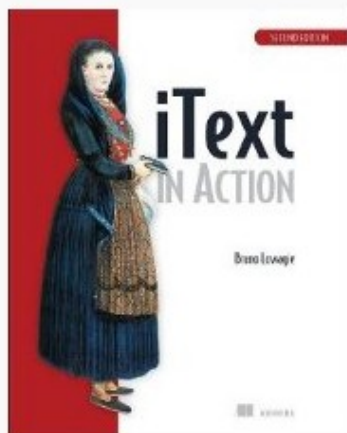
 **Movie**
Keywords: Plain Old Java Object

 **Director**
Keywords: Plain Old Java Object


 **Country**
Keywords: Plain Old Java Object


 **Entry**
Keywords: Plain Old Java Object


 **Category**





Buy it!


 [Manning.com](#)


 [Amazon.com](#)

 [Amazon.co.uk](#)

 [Amazon.de](#)

 [Amazon.at](#)

 [Amazon.fr](#)

 [Amazon.co.in](#)