PROGRAMOWANIE W JĘZYKU JAVA

Prowadzący: dr hab. inż. **Jan Prokop**, prof. PRz, e-mail: *jprokop@prz.edu.pl*,
Politechnika Rzeszowska, Wydział Elektrotechniki i Informatyki

LABORATORIUM

ĆWICZENIE nr 1

Temat: Podstawy JAVA SE

1. Java - wybrane zasoby sieci WWW

```
https://www.oracle.com/java/
https://www.java.com/pl/
http://www.javaworld.com/
http://www.java.pl/
```

Strona przedmiotu **`http://java.prz.edu.pl/`**

2. Java SE Development Kit – instalacja, konfiguracja

`http://www.oracle.com/technetwork/java/javase/downloads/`

Instalacja tylko środowiska Java - `jre-x-windows-i586.exe`
Instalacja pakietu JDK - `jdk-x-windows-i586.exe`
Instalacja pakietu JDK z NetBeans IDE - `jdk-x-nb-x-win.exe`
Instalacja dokumentacji - `jdk-x-doc.zip`

Java Control Panel - `Panel sterowania ⇒ Programy ⇒ Java`

Narzędzia - `C:\Program Files\Java\jdk1.x.0\bin ↵`

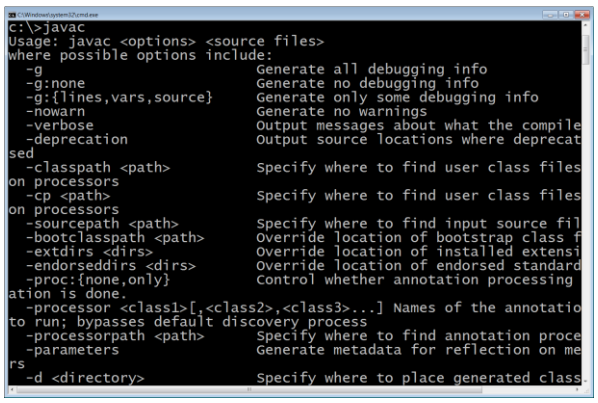
```
>set path=%path%;path_catalog_jdk_bin ↵
```

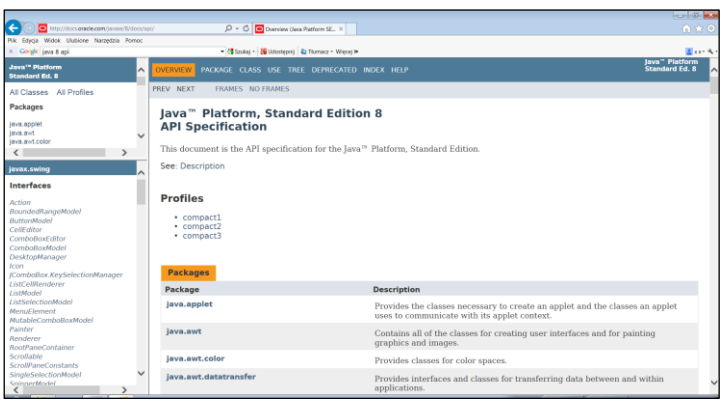
Sprawdzenie działania i opcji narzędzi:

```
>javac ↵
>java ↵    ( >javaw ↵ )
>java -version ↵
```

Przegląd dokumentacji API:

`http://docs.oracle.com/javase/8/docs/api/`





3. Przykłady kompilacji i uruchomienia aplikacji konsolowych

3.1. Plik `1Program.java`

```
/* Pierwszy program
public class PierwszyProgram {
   public static void main(String[] args) // {
      System.out.println("Java - pierwszy program !);
   }
}
```

Skompilować (usunąć błędy !!!) i uruchomić program z poziomu konsoli DOS

```
>javac PierwszyProgram.java ↵

>java PierwszyProgram ↵
```

Sprawdzić działanie opcji kompilatora: `>javac -verbose PierwszyProgram.java ↵`

3.2. Plik `DrugiProgram.java` – przekazywanie parametrów do aplikacji – usunąć błąd wykonania !

```
public class DrugiProgram {
    public static void main(String[] args) {
        if (args.length == 0) {
           System.out.println("No arguments");
           System.exit(0);
        }
        for (int i = 0; i < args.length + 1; i++) {
            System.out.println("args[" + i + "]:" + args[i]);
        }
    }
}
```

```
>java DrugiProgram Jeden Dwa Trzy ↵
```

3.3. Aplikacja `Quiz.java` – czytanie z pliku `questions.txt` – program konsolowy interaktywny

`Quiz.java`

```
import java.io.*;
import java.util.*;
public class Quiz {
   public static void main(String[] args) {
        int sum = 0;
        try {
            Scanner file = new Scanner(new File("questions.txt"));
            Scanner user = new Scanner(System.in);
            while (file.hasNext()) {
                for (int i = 0; i < 4; i++) {
                    System.out.println(file.nextLine());
                }
                String ok = file.nextLine();
                System.out.println("What is the correct ? ");
                String ans = user.next();
                ans = ans.toUpperCase();
                if (ans.length() > 1) {
                    ans = ans.substring(0, 1);
```

`questions.txt`

```
Question 1 ?
A - Answer 1
B - Answer 2
C - Answer 3
A
Question 2 ?
A - Answer 1
B - Answer 2
C - Answer 3
C
Question 3 ?
A - Answer 1
B - Answer 2
C - Answer 3
B
```

```
                }
                if (ans.equals(ok)) {
                    sum++;
                    System.out.println("OK !\n");
                } else {
                    System.out.println("NO !\n");
                }
            }
            System.out.println("\nResult:" + sum + " of 3");
        } catch (FileNotFoundException e) {
            System.out.println("No questions !");
        }
    }
}
```

```
>java Quiz ↵
```

3.4. Plik `MyFile.java` **– aplikacja konsolowa zapisująca do pliku tekstowego** `Myfile.txt` **– usunąć błędy !**

```
import java.io.File;
import java.io.IOException;
public class MyFile {
      public void main(String[] args) {
        try {
            File file = new File("./Myfile.txt");
            if (file.createNewFile()) {
                System.out.println("Success!");
            } else {
                System.out.println("Error, file already exists.");
            }
            FileWriter writer = new FileWriter(file);
            writer.write("My file test data");
            writer.close();
        } catch (IOException ioe) {
            ioe.printStackTrace();
        }
    }
}
```

```
>java MyFile ↵
```

4. Przykłady kompilacji i uruchomienia aplikacji okienkowych

4.1. Biblioteka AWT `java.awt.*` - Program `AWTDemo.java`

W oparciu o dokumentację **Java Standard Edition 8 API Specification** uruchomić program:

```
import java.awt.event.*;

public class AWTDemo extends Frame {
   public AWTDemo(){
      super("AWT Demo");
      prepareGUI();
   }
   public static void main(String[] args){
      AWTDemo  awtDemo = new AWTDemo();
```

```
      awtDemo.setVisible(true);
   }
   private void prepareGUI(){
      setSize(800,800);
      addWindowListener(new WindowAdapter() {
         public void windowClosing(WindowEvent windowEvent){
            System.exit(0);
         }
      });
   }
   public void paint(Graphics g) {
      g.setColor(Color.RED);
      Font font = new Font("Serif", Font.BOLD, 54);
      g.setFont(font);
      g.drawString("AWT Demo", 50, 150);
      g.setColor(Color.BLUE);
      g.drawRect(100,200,150,150);
      g.setColor(Color.GREEN);
      g.fillRect(100,200,400,400);

   }
}
```

4.2. Biblioteka Swing `javax.swing.*` - Program `SwingProgram.java`

W oparciu o dokumentację **Java Standard Edition 8 API Specification** uruchomić program (usunąć błędy !!!):

```
package jp;
import java.awt.*;

public class SwingProgram extends Jframe {
   public TrzeciProgram() {
      super("SwingProgram");
      String input = JoptionPane.showInputDialog(“Please enter your name”);
      String name = "Hello " + input + " ! ";
      Jlabel label = new Label(name, Jlabel.CENTER);
      label.setFont(new Font("Dialog", Font.BOLD, 50));
      label.setBackground(Color.yellow);
      label.setForeground(Color.blue);
      label.setOpaque(true);
      setDefaultCloseOperation(Jframe.EXIT_ON_CLOSE);
      add(label);
      setSize(600,400);
      //setVisible(true);
   }
   public static void main(String[] args) {
      new SwingProgram();
   }
}
```

Zmodyfikować powyższą aplikację tak aby możliwe było przekazywanie z linii poleceń łańcucha tekstu wyświetlanego w oknie aplikacji oraz wielkości czcionki (zastosować metodę `Integer.parseInt( )`).

4.3. Biblioteka JavaFX

`https://docs.oracle.com/javase/8/javafx/api/`

W oparciu o **JavaFX 8 API** uruchomić programy JavaFX

Plik `FXProgram.java`

```
import javafx.scene.*;
import javafx.scene.control.*;
import javafx.scene.image.*;
import javafx.scene.layout.*;

public class FXProgram extends Application {
  public static void main(String[] args) {
    Application.launch(args);
  }
  @Override
  public void start(Stage stage) {
    Image image = new Image("myImage.jpg");
    ImageView imageView = new ImageView(image);
    DatePicker datePicker = new DatePicker();
    VBox root = new VBox(10, datePicker, imageView);
    Scene scene = new Scene(root);
    stage.setScene(scene);
    stage.setTitle("MyTitle");
    //stage.show();
  }
}
```

Plik `PieChartSample.java`

```
package jp.prz.edu.pl;
import javafx.application.Application;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.scene.Scene;
import javafx.stage.Stage;
import javafx.scene.chart.*;
import javafx.scene.Group;
public class PieChartSample extends Application {
    @Override
    public void starts(Stage stage) {
        Scene scene = new Scene(new Group());
        stage.setTitle("Języki programowania");
        stage.setWidth(500);
        stage.setHeight(500);
        ObservableList<PieChart.Data> pieChartData
                = FXCollections.observableArrayList(
                      new PieChart.Data("Java", 21),
                      new PieChart.Data("C", 14),
                      new PieChart.Data("C++", 6),
                      new PieChart.Data("C#", 3.8),
                      new PieChart.Data("PHP", 3),
                      new PieChart.Data("Python",3.2));
        final PieChart chart = new  PieChart(pieChartData);
        chart.setTitle("Języki programowania");
        ((Group) scene.getRoot()).getChildren().add(chart);
        stage.setScene(scene);
        //stage.show();
    }
    public static void main(String[] args) {
        launch(args);
    }
}
```

5. Sposoby uruchamiania aplikacji Java

5.1. Uruchamianie aplikacji z linii poleceń

```
>java PierwszyProgram ↵
```

5.2. Uruchamianie za skryptów `*.bat` (Windows) lub `*.sh` (UNIX)

`PierwszyProgram.bat`

```
@echo off
javac PierwszyProgram.java
java PierwszyProgram
pause
```

```
>PierwszyProgram.bat ↵
```

5.3 . Uruchamianie aplikacji z archiwów Javy - pliki JAR

`>jar ↵` Sprawdzić opcje narzędzia `jar`

- **Tworzenie nowego archiwum**
  `>jar cvf nazwa_archiwum.jar plik1.class plik2.class ↵`
  `>jar cvf nazwa_archiwum.jar * ↵`
  `>jar cvf nazwa_archiwum.jar .class nazwa_podkatalogu/ .gif ↵`
- **Wyświetlenie listy plików z archiwum**
  `>jar tf nazwa_archiwum.jar ↵`
- **Tworzenie rozpakowanej kopii archiwum**
  `>jar xf nazwa_archiwum.jar ↵`
- **Uruchomienie aplikacji z pliku archiwum**
  `>java –jar app.jar ↵`

Kroki kompilacji i uruchomienia aplikacji z pliku archiwum jar

```
> javac *.java ↵
> echo Main-Class: MyClass >manifest.txt ↵
> jar cvfm MyClass.jar manifest.txt *.class ↵
> jar cvfe MyClass.jar MyClass *.class ↵
> MyClass.jar ↵
> java -jar MyClass.jar ↵
```

5.4. Java Web Start – uruchamianie aplikacji

Plik `launch.jnlp`

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<jnlp href="launch.jnlp" spec="1.0+">
    <information>
        <title>MyClass</title>
        <vendor>jp</vendor>
        <homepage href=""/>
        <description>MyClass</description>
        <description kind="short">MyClass</description>
    </information>
    <update check="always"/>
    <resources>
        <j2se version="1.8+"/>
        <jar href="MyClass.jar" main="true"/>
```

```
    </resources>
    <application-desc main-class="myclass.MyClass">
    </application-desc>
</jnlp>
```

- **Uruchomienie z konsoli -** `javaws`

```
> javaws launch.jnlp ↵
```

- **Uruchomienie z poziomu Java Cache Viewer**

```
> javaws –viewer ↵
```

- **Uruchomienie z poziomu skrótu na pulpicie**

Kliknąć prawym klawiszem myszki na zaznaczonej aplikacji w `Java Cache Viewer` i wybrać opcję `Install Shortcuts,` potem kliknąć na skrócie

- **Uruchomienie z poziomu przeglądarki**

Plik `launch.html`

```
<!DOCTYPE html>
<html>
  <head>
        <title>Test page for launching the application via JNLP</title>
  </head>
  <body>
     <h3>Test page for launching the application via JNLP</h3>
     <script src="http://java.com/js/deployJava.js"></script>
     <script>
        deployJava.createWebStartLaunchButton("launch.jnlp")
     </script>
     <!--Or use the following link element to launch with the application-->
     <!-- <a href="launch.jnlp">Launch the application</a> -->
   </body>
</html>
```

5.5. JavaFX Package – `javafxpackager`

```
> javac –d . HelloWorld.java ↵
> javafxpackager -createjar -appclass jp.HelloWorldMain -srcdir . –outdir
  out -outfile  helloworld.jar –v ↵
> javaw –jar out/ helloworld.jar ↵
```

Double-click `helloworld.jar`

6. Tworzenie dokumentacji programu za pomocą narzędzia *javadoc*

Utworzyć dokumentację programów za pomocą narzędzia `javadoc`

```
/**
* @author Jan Prokop
* @version 1.0
*/
```

```
>javadoc ↵
```

7. Przykłady tworzenia i uruchomienia apletów – technologia PRZESTARZAŁA !!!

- **Plik** `PierwszyAplet.java`

```
package jp;
import java.awt.*;
import javax.swing.*;
public class PierwszyAplet extends JApplet {
   String input;
   public void init() {
      input = JOptionPane.showInputDialog("Please enter your name" );
   }
   public void paint(Graphics g) {
      super.paint(g);
      g.drawString("Hello \"" + input + "\"",50,50);
      g.drawRect(100,100,150,150);
      g.setColor(Color.red);
      g.fillRect(300,400,400,400);
   }
}
```

- **Plik** `PierwszyAplet.html`

```
<html>
<head><title>Strona z pierwszym apletem</title></head>
<body>
<applet code = "jp/PierwszyAplet.class" width = "500" height = "300"><applet>
</body>
</html>
```

Uruchomić `PierwszyAplet.html` za pomocą `appletviewera` !

```
> appletviewer PierwszyAplet.html ↵
```

Umieścić na stronie HTML dwa różne aplety i uruchomić stronę za pomocą `appletviewera`.

Umieszczenie pliku archiwum Javy na stronie HTML

```
<applet code="Plik_w_archiwum.class" archive = "plik_archiwum.jar" width="700"
height="200">
```

7.1. Cykl życia apletu

Uruchomić aplet `Life.java` za pomocą `appletviewera` oraz obserwować działanie apletu w konsoli **DOS** oraz w **Java Console !!!**

- **Kodu apletu** `Life.java`

```
import java.awt.*;
import java.applet.*;
public class Life extends JApplet {
  public Life() {System.out.println("Konstruktor uruchomiony ... \n");}
  public void init() {System.out.println("To jest metoda init ... \n");}
  public void start() {System.out.println("Start apletu ... \n");}
  public void paint(Graphics args) {
      args.drawString("Mój aplet",0,50);
      System.out.println("Wywołanie metody paint ... \n");
  }
  public void stop() {System.out.println("Aplet zatrzymany ... \n");}
  public void destroy() {System.out.println("Zwolnione zasoby apletu ... \n");}
}
```

7.2. Przekazywanie parametrów z pliku HTML do apletu

- **Plik** `Parametry.html`

```
<html>
<head><title>Strona z apletem z parametrami</title></head>
<body>
<applet code="Parametry.class" width="700" height="200" alt="Wymagana JVM">
<param name="Napis" value="Mój aplet z parametrami !">
<param name="Czcionka" value="Helvetica">
</applet>
</body>
</html>
```

- **Plik** `Parametry.java`

```
import java.awt.*;
import javax.swing.*;
import java.applet.*;
public class Parametry extends JApplet {
  String Tekst;
  String NazwaCzcionki;
  Font NowaCzcionka;
  public void init() {
     Tekst=getParameter("Napis");
     NazwaCzcionki=getParameter("Czcionka");
     NowaCzcionka=new Font(NazwaCzcionki,Font.BOLD,50);
  }
  public void paint(Graphics g) {
     g.setFont(NowaCzcionka);
     g.drawString(Tekst,30,100);
  }
}
```

Zmodyfikować powyższy aplet tak aby możliwe było przekazywanie z pliku HTML wielkości czcionki

7.3. Aplet uruchamiany jako aplikacja

Uruchomić poniższy aplet jako aplikację oraz uruchomić go w przeglądarce internetowej

```
import java.awt.*;
import javax.swing.*;
public class ApAp extends JApplet {
   public void paint (Graphics g) {
      g.drawString("Welcome to Java!!", 100, 150);
   }
   public static void main (String args[]) {
      JFrame f = new JFrame ("ApAp");
      ApAp apap = new ApAp();
      f.setSize (600, 600);
      f.add ("Center", apap);
      f.setVisible(true);
   }
}
```

8. Java - narzędzia

- **Nakładki na pakiet JDK**

jGRASP - http://www.jgrasp.org

JCreator - http://www.jcreator.com

TextPad - http://www.textpad.com

DrJava - http://drjava.org

- **IDE**

NetBeans IDE - http://www.netbeans.org

Eclipse - http://www.eclipse.org/downloads/

InteliJ IDEA - http://www.jetbrains.com/idea/

JDeveloper - http://www.oracle.com/technology/products/jdev

Android Studio - http://developer.android.com/sdk/index.html

- **Inne**

Ant - http://ant.apache.org/

JUnit - http://www.junit.org/

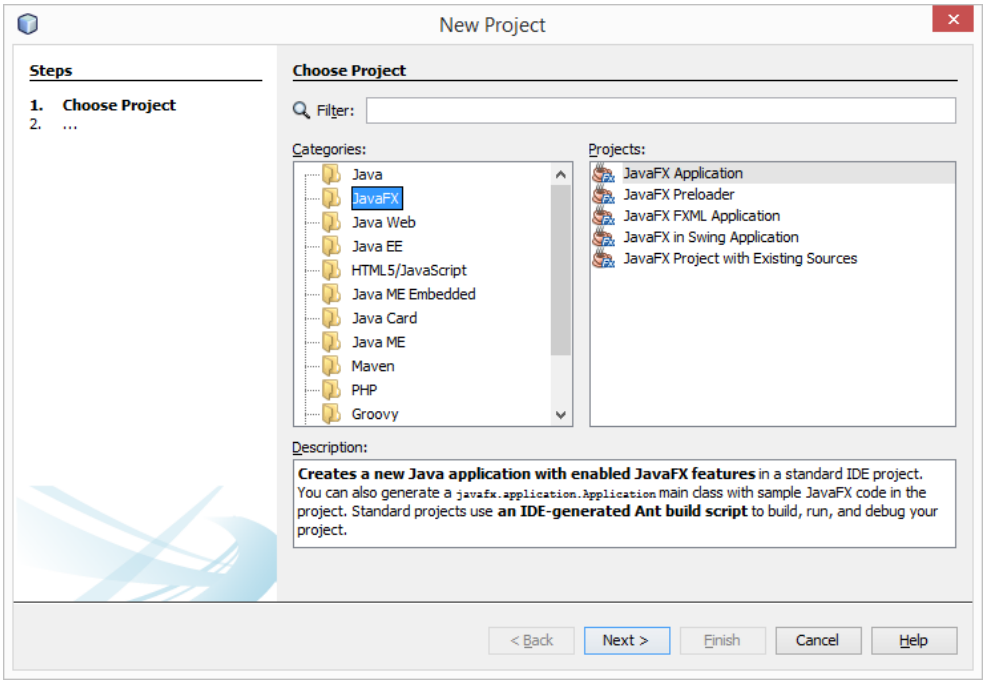
DbUnit - http://www.dbunit.org/

Log4J - http://logging.apache.org/log4j/

Tomcat - http://tomcat.apache.org/

9. Budowa aplikacji w środowisku NetBeans IDE

File →New Project → JavaFX → JavaFX Application ↵



Run → Clean and Build Project
Uruchomić aplikację:

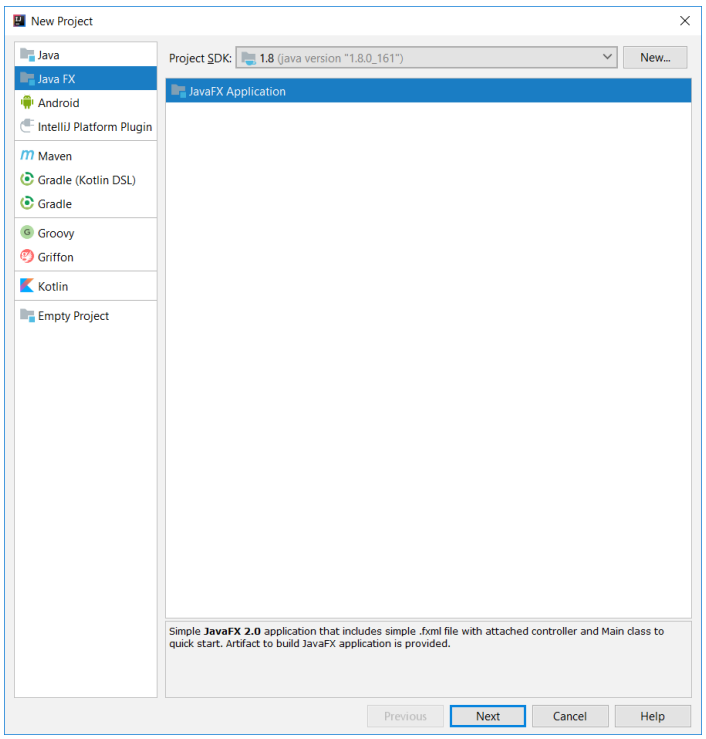
1. Z pliku JAR
2. Z pliku JNLP
3. Z pliku EXE po uzupełnieniu pliku `build.xml`:

`build.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="HelloWorld" default="default" basedir="."
                                xmlns:fx="javafx:com.sun.javafx.tools.ant">
    <description>Builds, tests, and runs the project </description>
    <import file="nbproject/build-impl.xml"/>
    <target name="-post-jfx-deploy">
        <fx:deploy width="${javafx.run.width}" height="${javafx.run.height}"
                   nativeBundles="all" outdir="${basedir}/${dist.dir}"
                                          outfile="${application.title}">
            <fx:application name="${application.title}"
                            mainClass="${javafx.main.class}"/>
            <fx:resources>
                <fx:fileset dir="${basedir}/${dist.dir}" includes="*.jar"/>
            </fx:resources>
            <fx:info title="${application.title}"
                                           vendor="${application.vendor}"/>
        </fx:deploy>
    </target>
</project>
```

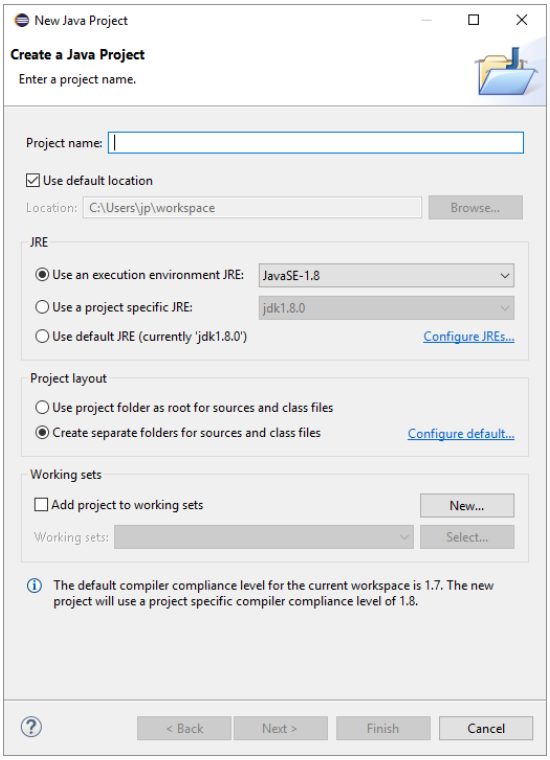
10. Budowa aplikacji w środowisku IntelliJ IDEA

http://www.jetbrains.com/idea/



11. Budowa aplikacji w środowisku Eclipse IDE for Java Developers

`http://eclipse.org/`



12. Zadania

Podaje prowadzący w każdej grupie laboratoryjnej