

# PROGRAMOWANIE W JĘZYKU JAVA

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

## LABORATORIUM 6

### Temat: Java – MySQL, XML, JSON, Web Services

#### 1. Obsługa bazy danych MySQL

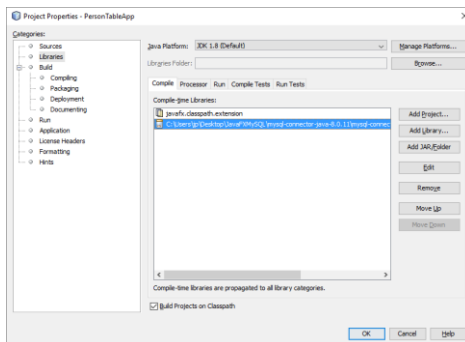
##### 1.1. MySQL – połączenie z bazą danych

##### Konfiguracja środowiska - dodanie driver-a bazy MySQL

- Pobrać sterownik JDBC `mysql-connector-java-x.x.x-bin.jar` (<https://dev.mysql.com/downloads/connector/j/>) i skopiować do katalogu `C:\Program Files\javajdk1.x.x_x\jre\lib\ext`
- Ustawić zmienną środowiskową `classpath`: `.;C:\Program Files\java\jdk1.x.x_x\jre\lib\ext\mysql-connector-java-x.x.x-bin.jar`

##### NetBeans – dodanie driver-a bazy MySQL

- Pobrać sterownik JDBC `mysql-connector-java-x.x.x-bin.jar` z adresu: <https://dev.mysql.com/downloads/connector/j/> (Platform Independent, ZIP Archive i rozpakować !)
- W oknie Projects kliknąć na nazwie projektu prawym klawiszem i wybrać Properties
- W oknie Project Properties wybrać Libraries i kliknąć Add JAR/Folder



- W otwartym okienku wybrać plik `'mysql-connector-java-8.0.11-bin.jar'` i kliknąć Open
- Po dodaniu do zakładki Compile w oknie Project Properties pliku `*.jar` kliknąć klawisz OK
- Sprawdzić połączenie z bazą realizując następujący kod:

- Plik `JDBCCConnection.java`

```
package jp;
import java.sql.*;
public class JDBCCConnection {
    private final static String DATABASE = "jdbc:mysql";
    private final static String HOSTNAME = "localhost";
    private final static String PORT = "3306";
    private final static String DBNAME = "jpdatabase";
    private final static String DBDRIVER = "com.mysql.cj.jdbc.Driver";
    private final static String USER = "root";
    private final static String PASSWORD = "";
    static Connection connection;
```

```

    public static void main(String[] args) throws SQLException {
        String url = DATABASE + "://" + HOSTNAME + ":" + PORT + "/" + DBNAME
            +
            "?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC";
        try {
            Class.forName(DBDRIVER).newInstance();
            connection = DriverManager.getConnection(url, USER, PASSWORD);
            System.out.println("Database connection successfully established !");

        } catch (ClassNotFoundException | IllegalAccessException |
            InstantiationException | SQLException e) {
            System.out.println("Cannot connect to database server: " + e.toString());
        }
    }
}

```

```

run:
Database connection successfully established !
BUILD SUCCESSFUL (total time: 0 seconds)

```

## 1.2. MySQL - Klient aplikacyjny

### 1.2.1. MySQL – Aplikacja Swing

- Plik JDBCMySQLApplication.java

```

import java.sql.*;
import javax.swing.*;
public class JDBCMySQLApplication extends JFrame {
    String url = "jdbc:mysql://localhost/myDatabaseName";
    String user = "root";
    String password = "";
    String text = "";
    public JDBCMySQLApplication() {
        super("JDBCMySQLApplication");
        try {
            Class.forName("com.mysql.jdbc.Driver");// load database driver class
        } catch (ClassNotFoundException classNotFound) {
            JOptionPane.showMessageDialog(null,
                classNotFound.getMessage(), "Driver Not Found",
                JOptionPane.ERROR_MESSAGE);
            System.exit(1);
        }
        try {
            // connect to database
            Connection connection = DriverManager.getConnection(url, user, password);
            // create Statement to query database
            Statement statement = connection.createStatement();
            String query = "DROP TABLE myDatabaseTable";
            int result = statement.executeUpdate(query);
            query = "CREATE TABLE myDatabaseTable (id INT NOT NULL " +
                " AUTO_INCREMENT PRIMARY KEY, first_name " +
                " VARCHAR(20), last_name VARCHAR(20), mail VARCHAR(20));";
            result = statement.executeUpdate(query);
            query = "INSERT INTO myDatabaseTable (first_name, last_name, mail) " +
                " + \"VALUES('Andrzej', 'Nowak', 'an@prz.edu.pl')\"";
            result = statement.executeUpdate(query);
            query = "INSERT INTO myDatabaseTable (first_name, last_name, mail) " +
                " + \"VALUES('Jan', 'Kowalski', 'jk@prz.edu.pl')\"";
            result = statement.executeUpdate(query);
            query = "UPDATE myDatabaseTable " +
                " + \"SET first_name='Jan',last_name='Nowak' " +
                " + \"WHERE id=2\";
            result = statement.executeUpdate(query);
            query = "SELECT * FROM myDatabaseTable ORDER BY id";
            // query database

```

```

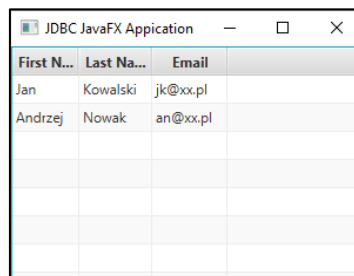
        ResultSet rs = statement.executeQuery(query);
        text = text + "ID\t" + "Imię\t" + "Nazwisko\t" + "e-mail\n";
        // process query results
        while (rs.next()) {
            int id = rs.getInt("id");
            String str1 = rs.getString("first_name");
            String str2 = rs.getString("last_name");
            String str3 = rs.getString("mail");
            text = text + id + "\t" + str1 + "\t" + str2 + "\t" + str3 + "\n";
        }

        // close statement and connection
        statement.close();
        connection.close();
    } catch (SQLException sqlException) {
        JOptionPane.showMessageDialog(null,
            sqlException.getMessage(), "Database Error",
            JOptionPane.ERROR_MESSAGE);
        System.exit(1);
    }
    JTextArea textArea = new JTextArea(text);
    textArea.setEditable(false);
    add(new JScrollPane(textArea));
    setSize(400, 300);
    setVisible(true);
}

public static void main(String[] args) {
    JDBCMySQLApplication frame = new JDBCMySQLApplication();
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
}

```

### 1.2.2. MySQL – Aplikacja JavaFX



First N...	Last Na...	Email
Jan	Kowalski	jk@xx.pl
Andrzej	Nowak	an@xx.pl

- Plik PersonTableApp.java

```

package jp;
import javafx.application.Application;
import javafx.scene.control.TableView;
import javafx.scene.control.TableColumn;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.BorderPane;
import javafx.scene.Scene;
import javafx.stage.Stage;
public class PersonTableApp extends Application {
    private PersonDataAccessor dataAccessor;
    @Override
    public void start(Stage primaryStage) throws Exception {
        primaryStage.setTitle("JDBC JavaFX Application");
        String databaseURL;
        databaseURL = "jdbc:mysql://localhost:3306/databaseName"
            +
            "?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serv
            erTimezone=UTC";
        String user = "root";
    }
}

```

```

        String password = "";
        dataAccessor = new PersonDataAccessor(databaseURL, user, password);
        TableView<Person> personTable = new TableView<>();
        TableColumn<Person, String> firstNameCol = new TableColumn<>("First Name");
        firstNameCol.setCellValueFactory(new PropertyValueFactory<>("firstName"));
        TableColumn<Person, String> lastNameCol = new TableColumn<>("Last Name");
        lastNameCol.setCellValueFactory(new PropertyValueFactory<>("lastName"));
        TableColumn<Person, String> emailCol = new TableColumn<>("Email");
        emailCol.setCellValueFactory(new PropertyValueFactory<>("email"));
        personTable.getColumns().addAll(firstNameCol, lastNameCol, emailCol);
        personTable.getItems().addAll(dataAccessor.getPersonList());
        BorderPane root = new BorderPane();
        root.setCenter(personTable);
        Scene scene = new Scene(root, 300, 200);
        primaryStage.setScene(scene);
        primaryStage.show();
    }
    @Override
    public void stop() throws Exception {
        if (dataAccessor != null) {
            dataAccessor.shutdown();
        }
    }
    public static void main(String[] args) {
        launch(args);
    }
}

```

- Plik PersonDataAccessor.java

```

package jp;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.ResultSet;
import java.util.List;
import java.util.ArrayList;

public class PersonDataAccessor {
    Connection connection;
    public PersonDataAccessor(String dbURL, String user, String password) throws
SQLException, ClassNotFoundException, InstantiationException {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver").newInstance();
            connection = DriverManager.getConnection(dbURL, user, password);
            System.out.println("Database connection successfully established !");
        } catch (ClassNotFoundException | IllegalAccessException |
InstantiationException | SQLException e) {
            System.out.println("Cannot connect to database server: " + e.toString());
        }
    }
    public void shutdown() throws SQLException {
        if (connection != null) {
            System.out.println("Connection close ! ");
            connection.close();
        }
    }
    public List<Person> getPersonList() throws SQLException {
        try {
            Statement stmtnt = connection.createStatement();
            ResultSet rs = stmtnt.executeQuery("select * from tableName");
            List<Person> personList = new ArrayList<>();
            while (rs.next()) {
                String firstName = rs.getString("first_name");
            }
        }
    }
}

```

```

        String lastName = rs.getString("last_name");
        String email = rs.getString("email_address");
        Person person = new Person(firstName, lastName, email);
        personList.add(person);
    }
    return personList;
}
}
}

```

- Plik Person.java

```

package jp;

import javafx.beans.property.StringProperty;
import javafx.beans.property.SimpleStringProperty;
public class Person {
    private final StringProperty firstName = new SimpleStringProperty(this,
"firstName");
    public StringProperty firstNameProperty() {
        return firstName;
    }
    public final String getFirstName() {
        return firstNameProperty().get();
    }
    public final void setFirstName(String firstName) {
        firstNameProperty().set(firstName);
    }
    private final StringProperty lastName = new SimpleStringProperty(this,
"lastName");
    public StringProperty lastNameProperty() {
        return lastName;
    }
    public final String getLastName() {
        return lastNameProperty().get();
    }
    public final void setLastName(String lastName) {
        lastNameProperty().set(lastName);
    }
    private final StringProperty email = new SimpleStringProperty(this, "email");
    public StringProperty emailProperty() {
        return email;
    }
    public final String getEmail() {
        return emailProperty().get();
    }
    public final void setEmail(String email) {
        emailProperty().set(email);
    }
    public Person() {
    }
    public Person(String firstName, String lastName, String email) {
        setFirstName(firstName);
        setLastName(lastName);
        setEmail(email);
    }
}

```

### 1.3. MySQL – Klient Web

#### 1.3.1. MySQL – Servlet

```

import java.io.*;
import javax.servlet.*;
import javax.servlet.annotation.WebServlet;

```

```
import javax.servlet.http.*;
import java.sql.*;
@WebServlet(name="JDBCMySQLServlet", urlPatterns={"/JDBCMySQLServlet"})
public class JDBCMySQLServlet extends HttpServlet {
    Connection connection = null;
    Statement statement = null;
    ResultSet rs = null;
    ResultSetMetaData rmd = null;
    String query = null;
    private void createConnection(PrintWriter out) {
        try {
            Class.forName("com.mysql.jdbc.Driver").newInstance();
            System.out.println("Driver Registration Successful !");
        } catch (InstantiationException ie) {
            out.println("Class instantiation exception: " + ie);
        } catch (ClassNotFoundException nf) {
            out.println("Class not found exception: " + nf);
        } catch (IllegalAccessException iae) {
            out.println("Illegal access exception: " + iae);
        }
    }
    try {
        String url = "jdbc:mysql://localhost/myDatabaseName?" +
            "user=root&password=";
        connection = DriverManager.getConnection(url);
        System.out.println("Connection successful !");
    } catch (SQLException sql) {
        out.println("Caught SQL exception " + sql);
    }
}
} // End createConnection()
private void runQuery(PrintWriter out) throws SQLException {
    statement = connection.createStatement();
    query = "INSERT INTO myDatabaseTable (first_name, last_name, mail)"
        + "VALUES('Andrzej', 'Nowak', 'an@prz.edu.pl')";
    int result = statement.executeUpdate(query);
    query = "DELETE FROM myDatabaseTable WHERE id=4";
    result = statement.executeUpdate(query);
    query = "UPDATE myDatabaseTable "
        + "SET first_name='Olaf',last_name='xxx' "
        + "WHERE id=2";
    result = statement.executeUpdate(query);
    query = "SELECT * FROM myDatabaseTable ORDER BY id";
    rs = statement.executeQuery(query);
    out.println("<table border = '1'>");
    out.println("<tr><td>ID</td>" + "<td>Imię</td>" +
        "<td>Nazwisko</td>" + "<td>e-mail</td></tr>");
    while (rs.next()) {
        int id = rs.getInt("id");
        String str1 = rs.getString("first_name");
        String str2 = rs.getString("last_name");
        String str3 = rs.getString("mail");
        out.println("<tr><td>" + id + "</td>" + "<td>" + str1 + "</td>" +
            "<td>" + str2 + "</td>" + "<td>" + str3 + "</td></tr>");
    }
    out.println("</table>");
} //end runQuery()
protected void processRequest(HttpServletRequest request,
    HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    try {
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet Conection Example</title>");
        out.println("</head>");
    }
```

```

        out.println("<body>");
        out.println("<h1>Test Database Connectivity</h1>");
        createConnection(out);
        try {
            runQuery(out);
        } catch (SQLException sqe) {
            out.println("Caught SQLException from runQuery(): " +
                        sqe);
        }
        out.println("</body>");
        out.println("</html>");

    } finally {
        out.close();
    }
}
@Override
public void destroy() {
}
@Override
protected void doGet(HttpServletRequest request,
                      HttpServletResponse response)
    throws ServletException, IOException {
    processRequest(request, response);
}
@Override
protected void doPost(HttpServletRequest request,
                      HttpServletResponse response)
    throws ServletException, IOException {
    processRequest(request, response);
}
@Override
public String getServletInfo() {
    return "Short description";
}
}

```

### 1.3.2. MySQL – Klient strona JSP

```

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <%@ page import="java.io.*" %>
        <%@ page import="javax.servlet.*" %>
        <%@ page import="javax.servlet.http.*" %>
        <%@ page import="java.sql.*" %>
        <%
            Class.forName("com.mysql.jdbc.Driver");
            String url = "jdbc:mysql://localhost/myDatabaseName";
            Connection connection = DriverManager.getConnection(url, "root",
                                                                "");

            Statement statement = connection.createStatement();
            String query = "SELECT * FROM myDatabaseTable ORDER BY id";
            ResultSet rs = statement.executeQuery(query);

        %>
    </body>
</html>

```

```
<h3>Data Retrieved:</h3>
<table border='1'>
  <tr><th>id</th><th>Imię</th><th>Nazwisko</th><th>E-mail</th></tr>
  <% while (rs.next()) {%>
    <tr>
      <td><%=rs.getString(1)%></td>
      <td><%=rs.getString(2)%></td>
      <td><%=rs.getString(3)%></td>
      <td><%=rs.getString(4)%></td>
    </tr>
  <% }%>
</table>
</body>
</html>
<% rs.close();
    statement.close();
    connection.close();%>
</body>
</html>
```

#### 1.4. MySQL - Zadania

- Napisać aplikację pobierającą dane z tabeli bazy danych MySQL i wyświetlającą je w postaci tabeli obiektu klasy `JTable`.
- Napisać aplikację JavaFX, w której użytkownik żąda wyświetlenia danych z tabeli bazy danych MySQL i wyświetlenia ich w postaci tabeli. Aplikacja powinna obsługiwać operacje CRUD.
- Napisać aplikację (serwlet) pobierającą dane z tabeli bazy danych MySQL i zapisującą je do pliku XML oraz wyświetlającą uaktualnione dane z pliku XML w aplikacji (na stronie HTML).
- Napisać serwlet pobierający dane z tabeli bazy danych MySQL, zapisujący je w pliku XML i realizujący transformację XSLT, która generuje plik XHTML.
- Napisać aplikację (serwlet) pobierającą dane z pliku XML i zapisującą je do tabeli bazy danych MySQL oraz wyświetlającą uaktualnione dane z tabeli bazy na stronie HTML.
- Napisać stronę formularza z którego wysłane dane (metodą POST) aktualizują tabelę bazy MySQL, a w odpowiedzi użytkownik otrzymuje stronę z uaktualnioną tabelą HTML.
- Napisać aplikację wg wzorca MVC, w której użytkownik z interfejsu jak w formularzu wprowadza i wysyła dane, które uaktualniają tabelę MySQL (model) oraz stronę JSP, która wyświetla te dane (widok) oraz serwlet, który zarządza operacjami na serwerze (sterownik).



## 2. Java – XML - JSON

### 2.1. SAX (<http://www.saxproject.org/>)

#### 2.1.1. Parsowanie i obsługa zdarzeń parsera

##### ● Plik źródłowy aplikacji - SAXSimpleDemo.java

```
import javax.xml.parsers.*;
import org.xml.sax.*;
import org.xml.sax.helpers.*;
import java.io.*;

public class SAXSimpleDemo extends DefaultHandler {
    public static void main(String[] arg) {
        try {
            SAXParserFactory factory = SAXParserFactory.newInstance();
            SAXParser parser = factory.newSAXParser();
            parser.parse(new File("SAXSimpleDemo.xml"), new SAXSimpleDemo());
        } catch (Exception e) { e.printStackTrace(); }
    }

    public void startDocument() throws SAXException {
        System.out.println("Start document");
    }

    public void startElement(String uri, String localName, String qName,
                             Attributes attrs) throws SAXException {
        System.out.println("Start element: " + qName);
        for(int i=0; i<attrs.getLength(); i++){
            System.out.println("Attributes: " + attrs.getQName(i) +
                               "=" + attrs.getValue(i));
        }
    }

    public void characters(char[] ch, int start, int length) throws SAXException {
        System.out.println("Characters: " + new String(ch, start, length));
    }

    public void endElement(String uri, String localName, String qName) throws
        SAXException {
        System.out.println("End element: " + qName);
    }

    public void endDocument() throws SAXException {
        System.out.println("End document");
    }
}
```

##### ● Plik danych XML - SAXSimpleDemo.xml

```
<?xml version="1.0"?>
<personnel>
    <person id="001" pesel="123456789">
        <name>Jan Prokop</name>
        <address>Rzeszów</address>
        <phone>017123456</phone>
        <email>jp@prz.edu.pl</email>
        <www>http://jp.prz.rzeszow.pl</www>
        <foto>jp.jpeg</foto>
    </person>
    <person id="002" pesel="23456789">
        <name>Jan Kowalski</name>
        <address>Kraków</address>
        <phone>017123456</phone>
        <email>jk@prz.edu.pl</email>
        <www>http://jk.prz.rzeszow.pl</www>
        <foto>jk.jpeg</foto>
    </person>
</personnel>
```

##### ● Zadania

1. Zmodyfikować kod tak aby obsłużyć zdarzenia parsera w klasie:
  - nazwanej wewnętrznej
  - anonimowej wewnętrznej
2. Napisać aplet, który czyta dane z pliku XML i wyświetla je na swoim panelu
3. Korzystając z interfejsu **SAX** sparsować plik `SAXSimpleDemo.xml`, zapisać dane do tablicy i wyświetlić w postaci obiektu klasy `JTable`
4. Sparsować plik `SAXSimpleDemo.xml` i wyświetlić dane w postaci obiektu klasy `JTree` w aplecie i aplikacji
5. Dodać do pliku XML DTD (jak w punkcie 1.3) oraz zmodyfikować aplikację tak aby obsługiwała błędy walidacji

## 2.1.2. SAX – budowa interfejsu użytkownika z danych w pliku XML

### • Plik źródłowy – XMLToUIMenu.java

```
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import javax.swing.*;
import org.xml.sax.*;
import org.xml.sax.helpers.*;
import javax.xml.parsers.*;

public class XMLToUIMenu {
    public XMLToUIMenu(JFrame frame) {
        MyXMLComponent xmlComponent = new MyXMLComponent();
        frame.add(xmlComponent.buildMenuPanel("XMLToUIMenu.xml"), BorderLayout.CENTER);
        frame.setVisible(true);
    }

    public static void main(String args[]) {
        JFrame frame = new JFrame("XMLToUIMenu");
        frame.setSize(300, 200);
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        new XMLToUIMenu(frame);
    }
}

class MyXMLComponent extends DefaultHandler implements ActionListener {
    private JPanel panel = new JPanel();
    public MyXMLComponent() {
        super();
    }

    public JComponent buildMenuPanel(String xmlFile) {
        try {
            SAXParserFactory factory = SAXParserFactory.newInstance();
            SAXParser parser = factory.newSAXParser();
            parser.parse(new InputSource(new FileInputStream(xmlFile)), this);
        } catch (Exception e) {
            System.out.println(e);
        }
        return panel;
    }

    public void startElement(String uri, String localName, String qName, Attributes attributes) {
        if (qName.equals("button")) {
            String s = attributes.getValue("label");
            JButton b = new JButton(s);
            b.addActionListener(this);
            panel.add(b);
        }
    }

    public void actionPerformed(ActionEvent e) {
        System.out.println("You clicked " + e.getActionCommand());
    }
}
```

### • Plik danych XML - XMLToUIMenu.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<menu>
  <button label="Button 1"/>
  <button label="Button 2"/>
  <button label="Button 3"/>
  <button label="Button 4"/>
</menu>
```

#### Zadanie

Korzystając z interfejsu SAX sparsować plik XMLToUIMenu.xml i zbudować na jego bazie menu aplikacji

### 2.1.3. SAX - walidacja dokumentu XML z DTD

- **Plik danych XML z DTD z błędami - saxerror.xml**

```
<?xml version="1.0"?>
<!DOCTYPE person [
  <!ELEMENT person (name,title,phone,email)>
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT title (#PCDATA)>
  <!ELEMENT phone (#PCDATA)>
  <!ELEMENT email (#PCDATA)>
]>
<person>
<name>Prokop</name>
<title>dr</title>
<phone>12345</phone>
<email>jprokop@prz.rzeszow.pl</email>
<fax>12345</fax>
</person>
```

- **Plik źródłowy aplikacji - SAXError.java**

```
import javax.xml.parsers.*;
import org.xml.sax.*;
import org.xml.sax.helpers.*;
import java.io.*;

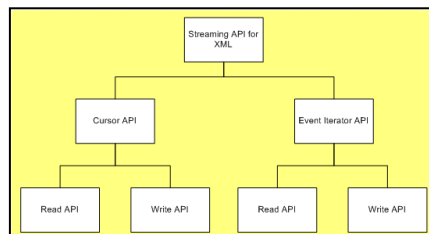
public class SAXError extends DefaultHandler {
    public static void main(String[] args) {
        try {
            SAXParserFactory spfactory = SAXParserFactory.newInstance();
            spfactory.setValidating(true);
            SAXParser parser = spfactory.newSAXParser();
            parser.parse(new File("saxerror.xml"), new SAXError());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void error(SAXParseException e){
        System.out.println("Error in line: " + e.getLineNumber() + " " + e.getMessage());
    }

    public void fatalError(SAXParseException e){
        System.out.println("FatalError in line: "+e.getLineNumber()+" "+e.getMessage());
    }

    public void warning(SAXParseException e){
        System.out.println("Warning in line: "+e.getLineNumber()+" "+ e.getMessage());
    }
}
```

## 2.2. StAX



### 2.2.1. StAX – CursorApi

- **Read API**

```
import java.io.*;
import javax.xml.stream.*;
import javax.xml.stream.events.*;

public class ReadingUsingCursorApi {
    public void ReadXML() {
```

```

    try {
        XMLInputFactory inputFactory = XMLInputFactory.newInstance();
        FileInputStream in = new FileInputStream("ReadingUsingCursorApi.xml");
        XMLStreamReader xmlReader = inputFactory.createXMLStreamReader(in);
        while (xmlReader.hasNext()) {
            Integer eventType = xmlReader.next();
            if (eventType.equals(XMLEvent.START_ELEMENT)) {
                System.out.print("<" + xmlReader.getName() + ">");
            } else if (eventType.equals(XMLEvent.CHARACTERS)) {
                System.out.print(" " + xmlReader.getText() + " ");
            } else if (eventType.equals(XMLEvent.ATTRIBUTE)) {
                System.out.print(" " + xmlReader.getName() + " ");
            } else if (eventType.equals(XMLEvent.END_ELEMENT)) {
                System.out.print("</ " + xmlReader.getName() + ">");
            }
        }
        xmlReader.close();
    } catch (Exception exception) {
        exception.printStackTrace();
    }
}

public static void main(String args[]) {
    new ReadingUsingCursorApi().ReadXML();
}
}

```

- **Write API**

```

import javax.xml.stream.XMLOutputFactory;
import javax.xml.stream.XMLStreamWriter;

public class XMLStreamWriterDemo {
    public static void main(String[] args) throws Exception {

        XMLOutputFactory factory = XMLOutputFactory.newInstance();
        XMLStreamWriter writer = factory.createXMLStreamWriter(System.out);
        writer.writeStartDocument("1.0");
        writer.writeStartElement("catalog");
        writer.writeStartElement("book");
        writer.writeAttribute("isbn", "123");
        writer.writeStartElement("Author");
        writer.writeCharacters("John Author");
        writer.writeEndElement();
        writer.writeStartElement("title");
        writer.writeCharacters("My Title");
        writer.writeEndElement();
        writer.writeStartElement("price");
        writer.writeCharacters("123.34");
        writer.writeEndElement();
        writer.writeEndDocument();
        writer.flush();
        writer.close();
    }
}

```

### 2.2.2. StAX – IteratorApi

- **Read API**

```

import java.io.*;
import java.util.*;
import javax.xml.namespace.*;
import javax.xml.stream.*;
import javax.xml.stream.events.*;

public class ReadingUsingEventIteratorApi {

```

```

private XMLInputFactory inputFactory = null;
private XMLEventReader xmlEventReader = null;

public ReadingUsingEventIteratorApi() {
    inputFactory = XMLInputFactory.newInstance();
}

public void readXML() throws Exception {
    xmlEventReader = inputFactory.createXMLEventReader(
        new FileInputStream("ReadingUsingCursorApi.xml"));
    while (xmlEventReader.hasNext()) {
        XMLEvent event = xmlEventReader.nextEvent();
        if (event.isStartElement()) {
            StartElement element = (StartElement) event;
            System.out.println("Start Element: " + element.getName());
            Iterator iterator = element.getAttributes();
            while (iterator.hasNext()) {
                Attribute attribute = (Attribute) iterator.next();
                QName name = attribute.getName();
                String value = attribute.getValue();
                System.out.println("Attribute name/value: " + name + "/" +
                                   value);
            }
        }
        if (event.isCharacters()) {
            Characters characters = (Characters) event;
            System.out.println("Text: " + characters.getData());
        }
        if (event.isEndElement()) {
            EndElement element = (EndElement) event;
            System.out.println("End element:" + element.getName());
        }
    }
    xmlEventReader.close();
}

public static void main(String args[]) {
    try {
        new ReadingUsingEventIteratorApi().readXML();
    } catch (Exception exception) {
        exception.printStackTrace();
    }
}
}

```

- **Write API**

```

import java.io.*;
import javax.xml.stream.*;
import javax.xml.stream.events.*;

public class XMLEventWriterDemo {

    public static void main(String[] args) {

        XMLOutputFactory factory = XMLOutputFactory.newInstance();
        XMLEventFactory eventFactory = XMLEventFactory.newInstance();

        try {
            XMLEventWriter writer = factory.createXMLEventWriter(
                new FileWriter("output.xml"));

            XMLEvent event = eventFactory.createStartDocument();
            writer.add(event);

```

```

        event = eventFactory.createStartElement(
            "jp", "http://jp.xmlns", "root");
        writer.add(event);

        event = eventFactory.createNamespace(
            "jp", "http://jp.xmlns");
        writer.add(event);

        event = eventFactory.createAttribute("attr", "attrValue");
        writer.add(event);

        event = eventFactory.createEndElement(
            "jp", "http://jp.xmlns", "root");
        writer.add(event);

        writer.flush();
        writer.close();
    } catch (XMLStreamException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
}

```

### 2.3. DOM (<http://www.w3.org/DOM/>)

• <b>Plik danych XML - addressbook.xml</b>	• <b>Zadania</b>
<pre> &lt;?xml version="1.0"?&gt; &lt;addressbook&gt;   &lt;title&gt;Address book&lt;/title&gt;   &lt;person id="1"&gt;     &lt;name&gt;Jan Kowalski&lt;/name&gt;     &lt;email&gt;jk@rzeshow.pl&lt;/email&gt;   &lt;/person&gt;   &lt;person id="2"&gt;     &lt;name&gt;Jan Nowak&lt;/name&gt;     &lt;email&gt;jn@rzeshow.pl&lt;/email&gt;   &lt;/person&gt; &lt;/addressbook&gt; </pre>	<ol style="list-style-type: none"> <li>1. Przerobić aplikację z punktu 2.1 na aplet, który wyświetla dane z pliku XML na swoim panelu</li> <li>2. Zmodyfikować aplikację, tak aby czytała plik addressbook.xml, dodawała kilka rekordów odpowiadających znacznikom &lt;person&gt;...&lt;/person&gt; i wyświetlała wynik</li> <li>3. Korzystając z interfejsu <b>DOM</b> sparsować plik addressbook.xml, zapisać dane do tablicy i wyświetlić w postaci obiektu klasy JTable</li> <li>4. Sparsować plik addressbook.xml i wyświetlić dane w postaci obiektu klasy JTree w aplicie i aplikacji</li> </ol>

#### 2.3.1. DOM - Dostęp do elementów drzewa DOM przez węzły

- **Plik źródłowy aplikacji - DOMgetDocumentNodes.java**

```

import javax.xml.parsers.*;
import org.w3c.dom.*;
public class DOMgetDocumentNodes {
    public static void main(String[] args) throws Exception {
        Document document = DocumentBuilderFactory.newInstance()
            .newDocumentBuilder()
            .parse("addressbook.xml");

        int i = 0;
        Node rootNode = (Node) (document.getDocumentElement());
        Node firstChild = rootNode.getFirstChild();
        while (firstChild != null) {
            if (!firstChild.getNodeName().equals("person")) {
                firstChild = firstChild.getNextSibling();
                continue;
            }
            i++;
            Node nameNode = firstChild.getFirstChild();
            while (nameNode != null) {
                if (!nameNode.getNodeName().equals("name")) {

```

```

        nameNode = nameNode.getNextSibling();
        continue;
    }
    Node node = nameNode.getFirstChild();
    if (node != null) {
        System.out.println(i + " Name: " + node.getNodeValue());
    }
    break;
}
firstChild = firstChild.getNextSibling();
}
}
}

```

### 2.3.2. DOM - Dostęp do elementów drzewa DOM przez nazwę elementu

- **Plik źródłowy aplikacji - AddressBookList.java**

```

import javax.xml.parsers.*;
import org.w3c.dom.*;
import java.io.*;
public class AddressBookList {
    public static void main(String[] args) {
        try {
            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();
            Document document = builder.parse(new File("addressbook.xml"));
            Element root = document.getDocumentElement();
            System.out.println("root Element Name: " + root.getTagName());
            Element titleElement = (Element)root.getElementsByTagName("title").item(0);
            String title = titleElement.getFirstChild().getNodeValue();
            System.out.println("Title: " + title);
            System.out.println("***** Elementy potomne *****");
            NodeList list = root.getElementsByTagName("person");
            for (int i=0; i < list.getLength() ; i++) {
                Element element = (Element)list.item(i);
                String id = element.getAttribute("id");
                NodeList nameList = element.getElementsByTagName("name");
                Element nameElement = (Element)nameList.item(0);
                String name = nameElement.getFirstChild().getNodeValue();
                NodeList emailList = element.getElementsByTagName("email");
                Element emailElement = (Element)emailList.item(0);
                String email = emailElement.getFirstChild().getNodeValue();
                System.out.println("ID: " + id + " " +
                    "Name: " + name + " " + "Email: " + email);
            }
        } catch (Exception e) { e.printStackTrace(); }
    }
}

```

### 2.3.3. Tworzenie drzewa DOM i zapis do pliku XML - CreateNewDocumentDOM.java

```

import org.w3c.dom.*;
import javax.xml.parsers.*;
import javax.xml.transform.dom.*;
import javax.xml.transform.*;
import javax.xml.transform.stream.*;
import java.io.*;
public class CreateNewDocumentDOM {
    public static void main(String[] args) {
        try {
            DocumentBuilderFactory dbfactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = dbfactory.newDocumentBuilder();
            Document document = builder.newDocument();
            Element rootElement = document.createElement("addressbook");

```

```

        rootElement.setAttribute("Att_Name", "Att_Value");
        document.appendChild(rootElement);
        Element childElement1 = document.createElement("person");
        childElement1.appendChild(document.createTextNode("JP"));
        rootElement.appendChild(childElement1);
        Element childElement2 = document.createElement("person");
        childElement2.appendChild(document.createTextNode("JN"));
        rootElement.appendChild(childElement2);
        Comment comment = document.createComment("Comment");
        rootElement.appendChild(comment);
        DOMSource source = new DOMSource(document);
        FileOutputStream fileoutput = new FileOutputStream(new
                                                                File("newDocument.xml"));

        StreamResult result = new StreamResult(fileoutput);
        TransformerFactory tfactory = TransformerFactory.newInstance();
        Transformer transformer = tfactory.newTransformer();
        transformer.transform(source, result);
        System.out.println("Save: newDocument.xml");
    } catch (Exception e) { e.printStackTrace(); }
}
}

```

### 2.3.4. Zastosowanie języka XPath - XPathQueryXML.java

```

import javax.xml.parsers.*;
import javax.xml.xpath.*;
import org.w3c.dom.*;
public class XPathQueryXML {
    public static void main(String[] args) throws Exception {
        try {
            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();
            Document document = builder.parse("XPathQueryXML.xml");
            XPathFactory xFactory = XPathFactory.newInstance();
            XPath xpath = xFactory.newXPath();
            XPathExpression expr = xpath.compile("//person[firstname='Jan']/lastname/text()");
            //XPathExpression expr = xpath.compile("/people/person[2]/firstname/text()");
            //XPathExpression expr = xpath.compile("/people/person[3]/@id");
            Object result = expr.evaluate(document, XPathConstants.NODESET);
            NodeList nodes = (NodeList) result;
            for (int i=0; i<nodes.getLength();i++){
                System.out.println("Node value: " + nodes.item(i).getNodeValue());
            }
            expr = xpath.compile("count(//person[firstname='Jan'])");
            Double number = (Double) expr.evaluate(document, XPathConstants.NUMBER);
            System.out.println("Number of objects: " + number);
            expr = xpath.compile("count(//person[firstname='Jan']) >=2");
            Boolean check = (Boolean)expr.evaluate(document, XPathConstants.BOOLEAN);
            System.out.println("2 people with name Jan: " + check);
        }
        catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

## 2.4. XML - Transformacje XSLT

### • Plik danych lista.xml

```

<?xml version="1.0"?>
<lista>
    <pracownik>
        <nazwisko>Jan Prokop</nazwisko>
        <uczelnia>Politechnika Rzeszowska</uczelnia>
        <wydzial>Wydział Elektrotechniki i Informatyki</wydzial>
    
```



```

        <adres>ul. W. Pola 2, 35-959 Rzeszów</adres>
        <telefon>(0-prefix-17) 8651384</telefon>
        <mail>jprokop@prz.rzeszow.pl</mail>
        <wzrost>175</wzrost>
    </pracownik>
    <pracownik>
        <nazwisko>Stanisław Nowak</nazwisko>
        <uczelnia>Uniwersytet Warszawski</uczelnia>
        <wydzial>Wydział Prawa</wydzial>
        <adres>ul. Rzeszowska 2, 35-123 Warszawa</adres>
        <telefon>(0-prefix-02) 5555555</telefon>
        <mail>jnowak@univ.warszawa.pl</mail>
        <wzrost>182</wzrost>
    </pracownik>
</lista>

```

- **Plik szablonu** lista.xml

```

<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html><head></head>
<body>
    <h2>Pracownicy</h2>
    <table border="1">
        <tr bgcolor="#00ff00">
            <th>Imię i Nazwisko</th>
            <th>Uczelnia</th>
        </tr>
        <xsl:for-each select="lista/pracownik">
            <tr>
                <td><xsl:value-of select="nazwisko"/></td>
                <td><xsl:value-of select="uczelnia"/></td>
            </tr>
        </xsl:for-each>
    </table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>

```

- **Konwersja pliku XML na plik HTML - UtworzHTML.java**

```

import javax.xml.transform.*;
import javax.xml.transform.stream.*;
public class lista {
    public static void main(String[] args)    {
        try {
            TransformerFactory tf = TransformerFactory.newInstance();
            Transformer transformer = tf.newTransformer(new
                StreamSource("lista.xml"));
            transformer.transform(new StreamSource("lista.xml"), new
                StreamResult("lista.html"));
        }
        catch (TransformerConfigurationException tce) {
            tce.printStackTrace();
        }
        catch (TransformerException te) {
            te.printStackTrace();
        }
    }
}

```

## 2.5. Java EE - JSON

### JSON - plik

```
{
  "name": "My Name",
  "email": "My email",
  "phone": "172017"
}
```

### 2.5.1. Object Model

```
package jp;
import java.io.IOException;
import java.io.InputStream;
import java.net.URL;
import javax.json.Json;
import javax.json.JsonObject;
import javax.json.JsonReader;
public class JsonDemo {
    public static void main(String[] args) {
        try {
            URL url = new URL("http://java.prz.edu.pl/json.txt");
            InputStream in = url.openStream();
            JsonReader reader = Json.createReader(in);
            JsonObject jobj = reader.readObject();
            System.out.println("Name: " + jobj.getJsonString("name"));
            System.out.println("Email: " + jobj.getJsonString("email"));
            System.out.println("Phone: " + jobj.getJsonString("phone"));
        } catch (IOException ex) {
            System.out.println(ex);
        }
    }
}
```

### 2.5.2. Streaming Model

```
package jp;
import java.io.IOException;
import java.io.InputStream;
import java.net.URL;
import javax.json.Json;
import javax.json.stream.JsonParser;
import javax.json.stream.JsonParser.Event;

public class JsonDemo {
    public static void main(String[] args) {
        try {
            URL url = new URL("http://java.prz.edu.pl/json.txt ");
            InputStream in = url.openStream();
            JsonParser parser = Json.createParser(in);
            while (parser.hasNext()) {
                Event event = parser.next();
                if (event == Event.KEY_NAME) {
```

```
        switch (parser.getString()) {
            case "name":
                parser.next();
                System.out.println("Name: " + parser.getString());
                break;
            case "email":
                parser.next();
                System.out.println("Email: " + parser.getString());
                break;
            case "phone":
                parser.next();
                System.out.println("Phone: " + parser.getString());
                break;
        }
    }
}
catch (IOException ex) {
    System.out.println(ex);
}
}
```

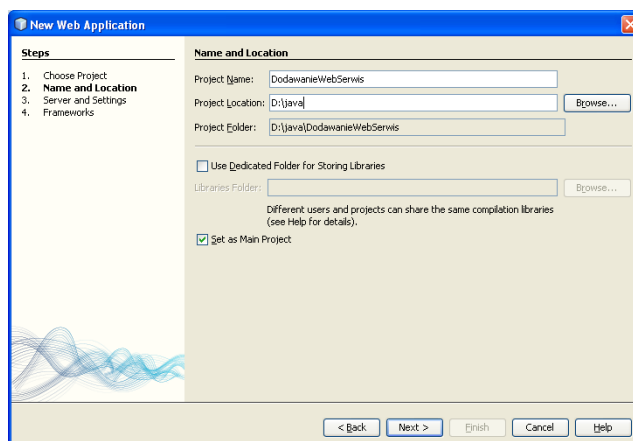
**Zadanie** Napisać serwlet generujący plik JSON-a z tablicy Java

## 3. Java usługi

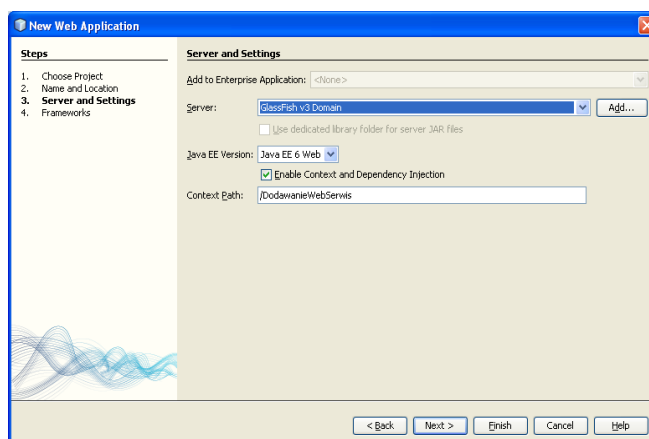
### 3.1. Java WebServices

#### 3.1.1. Utworzenie usługi sieciowej zwracającej wynik dodawania dwóch liczb

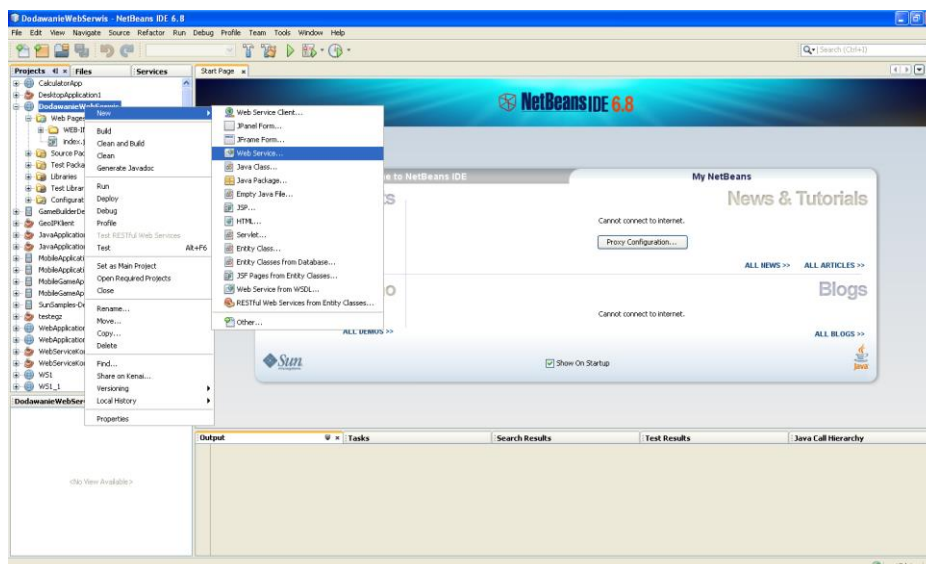
- Z menu środowiska Netbeans wybieramy pozycję: **File** → **New Project...**
- W nowym oknie wybieramy kategorię **JavaWeb** i rodzaj projektu **Web Application**. Klikamy na przycisk **Next >**
- Nadajemy nazwę tworzonego projektu i klikamy na przycisk **Next >**



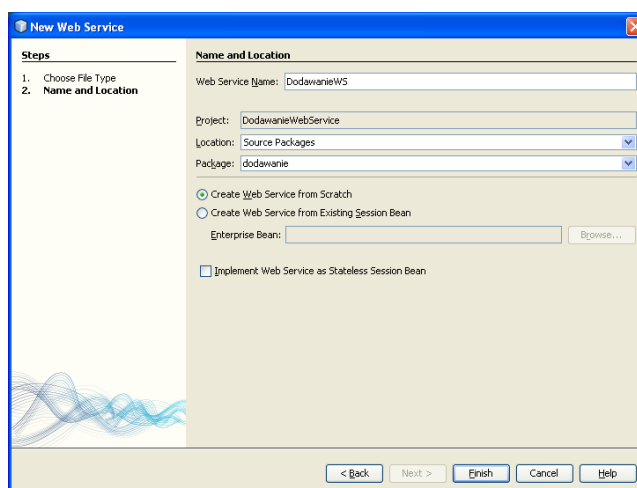
- Wybieramy serwer GlassFish oraz pakiet Java EE. Klikamy na przycisk **Next >**



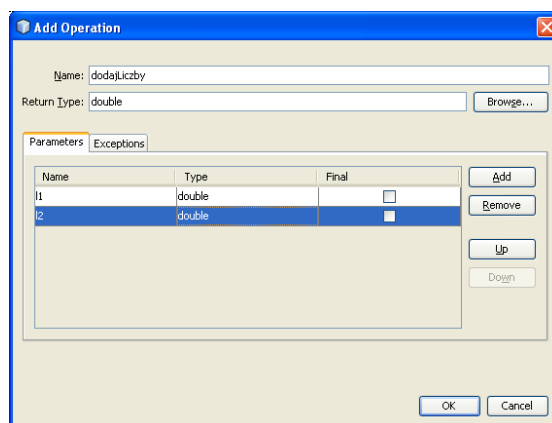
- W ostatnim kroku tworzenia projektu istnieje możliwość wyboru dodatkowych frameworków. W naszym przypadku pozostawiamy wszystkie pola odznaczone i klikamy na przycisk **Finish**
- Po utworzeniu tworzenia projektu zamykamy (automatycznie otwarty) plik **index.jsp**. Następnie klikamy prawym przyciskiem myszy na nazwie projektu i wybieramy **New** → **Web Service...**



- g) W nowym oknie nadajemy nazwę tworzonej usługi oraz wybieramy pakiet w którym zostanie umieszczony plik \*.java z wygenerowanym podstawowym kodem usługi. Klikamy na przycisk *Finish*.



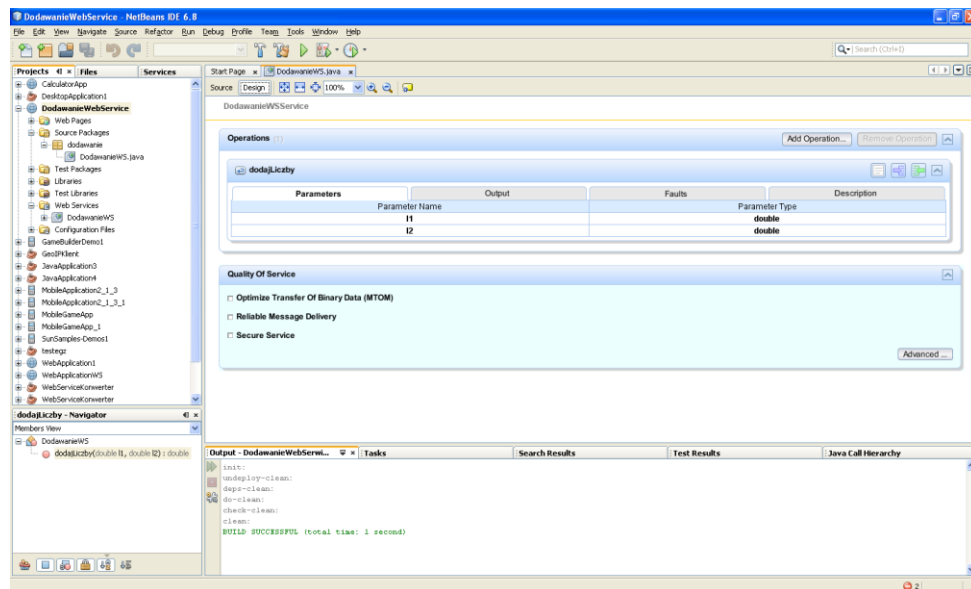
- h) Po utworzeniu nowej usługi dodajemy do niej nową operację. Należy przełączyć się do trybu *Design* lub z menu kontekstowego pod prawym klawiszem w okienku *Projects* – *Web Services* – *DodawanieWS* i wybierać *Add Operation...*
- i) W nowym oknie wprowadzamy nazwę tworzonej operacji, typ argumentu zwracanego oraz typy i nazwy argumentów pobieranych przez operację. W naszym przypadku pobieramy wartości dwóch liczb typu *double* oraz zwracamy wynik ich dodawania, również typu *double*.



- j) Po kliknięciu na przycisk *OK* zostaje wygenerowany kod metody:

```
@WebMethod(operationName = "dodajLiczby")
public double dodajLiczby(@WebParam(name = "l1")
double l1, @WebParam(name = "l2")
double l2) {
    //TODO write your implementation code here:
    return 0.0;
}
```

- k) Przy tworzeniu i przeglądaniu dotychczasowych operacji można się posłużyć również wbudowanym designerem. Przechodzimy do niego klikając na przycisk *Design*. Następnie wybieramy opcję *Add Operation...*



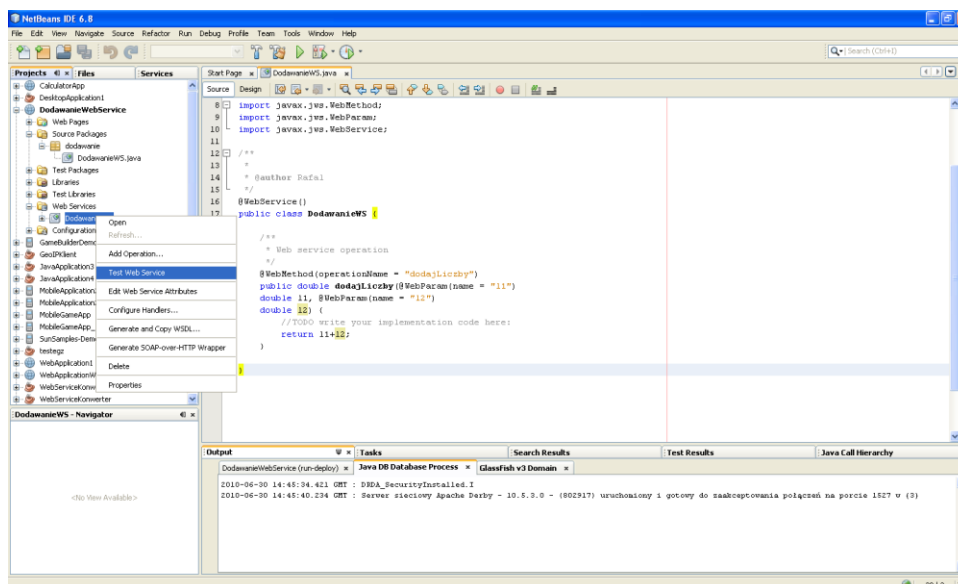
- l) Metoda zwraca w każdym przypadku wartość *0.0*, należy zastąpić kod tak, aby wartością zwracaną był wynik dodawania dwóch liczb.

```
@WebMethod(operationName = "dodajLiczby")
public double dodajLiczby(@WebParam(name = "l1")
double l1, @WebParam(name = "l2")
double l2) {
    return l1+l2;
}
```

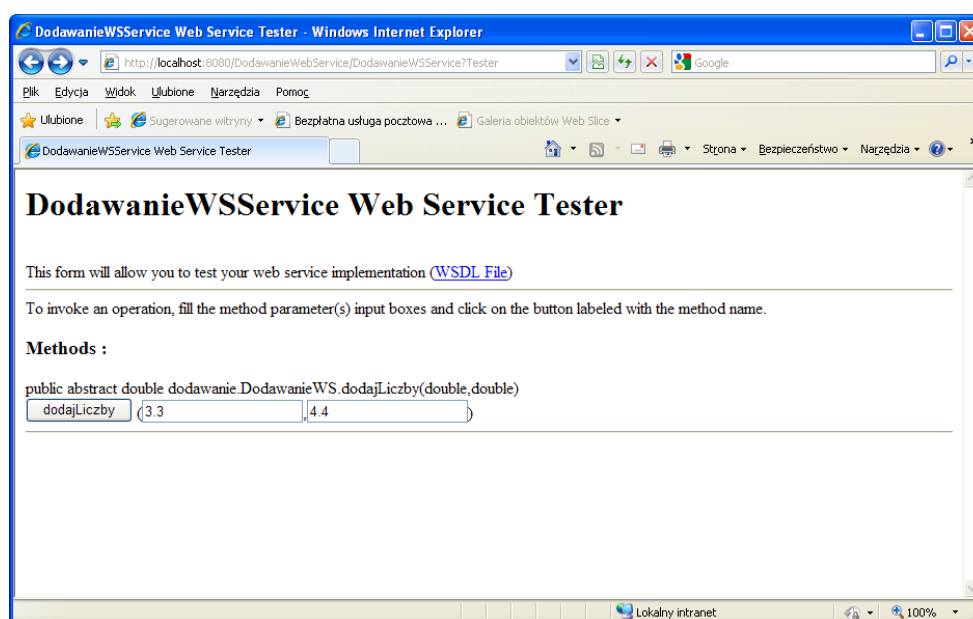
Usuwamy kod wygenerowany automatycznie:

```
/** This is a sample web service operation */
@WebMethod(operationName = "hello")
public String hello(@WebParam(name = "name") String txt) {
    return "Hello " + txt + " !";
}
```

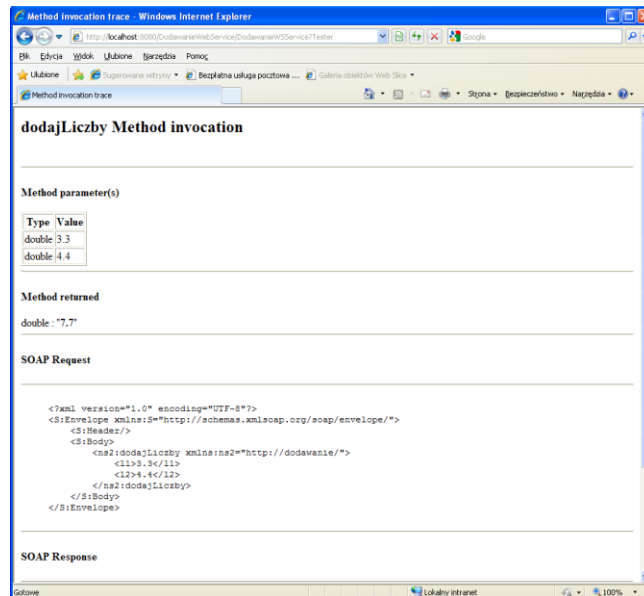
- m) Pozostaje nam przetestować działanie naszej usługi. Klikamy prawym przyciskiem myszy na nazwie projektu i wybieramy opcję *Deploy*. Dzięki temu umieszczamy naszą aplikację webową na serwerze.
- n) Następnie rozwijamy drzewko projektu. Z kategorii *Web Services* klikamy prawym przyciskiem myszy na nazwę usługi i wybieramy *Test Web Service*



- o) Zostajemy przekierowani do przeglądarki gdzie mamy możliwość sprawdzenia działania usługi (jak również przejrzenia pliku WSDL). Na wygenerowanej stronie podajemy dwie liczby i klikamy na przycisk *dodajLiczby*, który powoduje uruchomienie stworzonej metody.



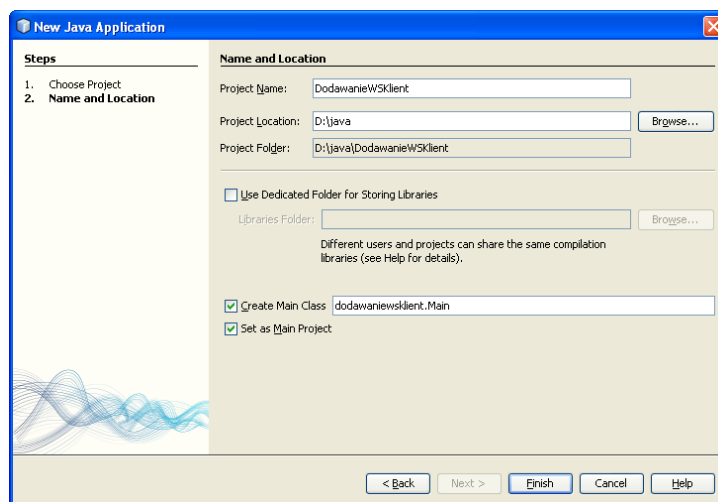
- p) Po przeładowaniu strony otrzymujemy zestawienie w którym można wyróżnić parametry wywołania metody, wartość zwróconą w wyniku jej działania oraz treść komunikatów żądania i odpowiedzi protokołu SOAP.



### 3.1.2. Korzystanie z usługi sieciowej – aplikacja klienta

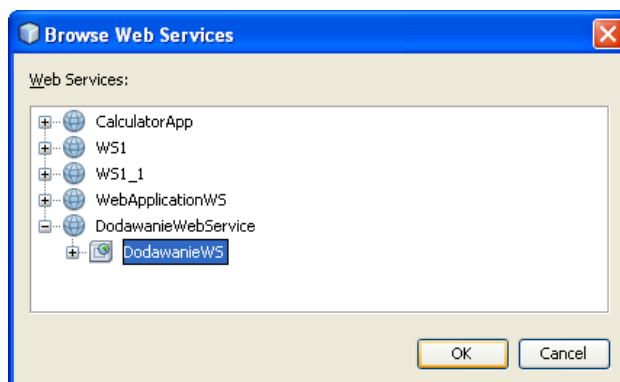
Sposób korzystania z usług typu WebService zostanie omówiony na przykładzie prostej aplikacji konsolowej. Aplikacja ma na celu obsługę usługi utworzonej we wcześniejszym punkcie instrukcji. Poniżej przedstawiono kroki w celu realizacji zadania:

- Z menu, w górnej części okna środowiska Netbeans wybieramy pozycję:  
*File → New Project...*
- W nowym oknie wybieramy kategorię *Java* i rodzaj projektu *Java Application*. Klikamy na przycisk *Next >*
- Nadajemy nazwę tworzonego projektu i klikamy na przycisk *Finish*

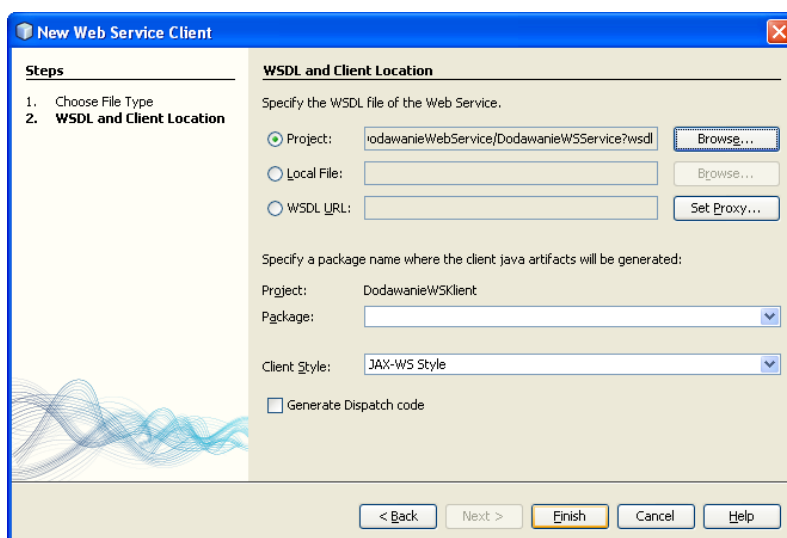


- Następnie klikamy prawym przyciskiem myszy na nazwie projektu i wybieramy *New → Web Service Client...*
- Zostajemy poproszeni o wybór pliku WSDL. Możemy określić plik WSDL na podstawie projektu środowiska NetBeans, lokalizacji (lokalnej) lub adresu URL (rozwiązanie stosowane w większości rozwiązań, gdy korzystamy z usługi znalezionej w internecie). Jako, że nasza usługa znajduje się na komputerze lokalnym możemy skorzystać z dowolnej opcji. Zaznaczamy więc pierwszą opcję *Project* i klikamy na przycisk *Browse*. Szukamy nazwy aplikacji nadanej w poprzednim punkcie instrukcji. Rozwijamy drzewko dla danej aplikacji, zaznaczamy nazwę wcześniej stworzonej usługi a następnie klikamy na przycisk *OK*.





f) Po powrocie do okna tworzenia klienta usługi klikamy na przycisk *Finish*



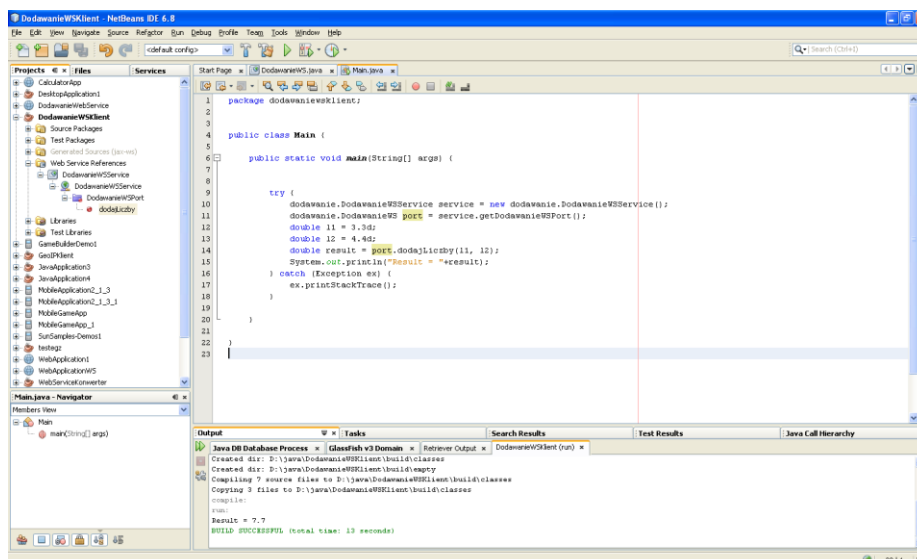
g) Następuje generowanie klas na podstawie pliku WSDL oraz budowanie aplikacji. Po zakończeniu tych czynności pozostaje nam skorzystanie z usługi. W tym celu rozwijamy drzewko projektu. Przechodzimy kolejno: *Web Service References* → *DodawanieWSService* → *DodawanieWSService* → *DodawanieWSPort* → *dodajLiczby*. „Łapiąc” za element *dodajLiczby* przeciągamy go tak, aby kursor znajdował się wewnątrz klasy *aplikacji klienta*. Zostaje wygenerowany kod metody dodającej liczby:

```
private static double dodajLiczby(double l1, double l2) {
    dodawanie.DodawanieWS_Service service = new dodawanie.DodawanieWS_Service();
    dodawanie.DodawanieWS_port = service.getDodawanieWSPort();
    return port.dodajLiczby(l1, l2);
}
```

h) Zmieniamy wartości zmiennych l1 i l2. Obsługujemy też wyjątek.

```
try {
    double l1 = 3.3d;
    double l2 = 4.4d;
    double result = dodajLiczby(l1, l2);
    System.out.println("Result = " + result);
}
catch (Exception ex) {
    System.out.println("Exception: " + ex);
}
```

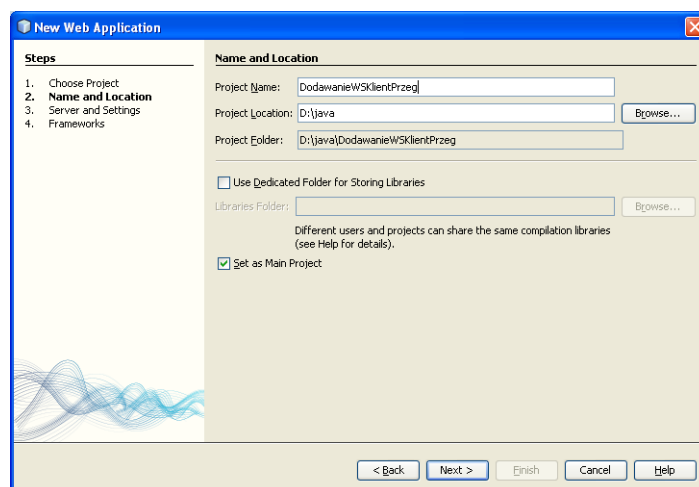
i) Otrzymujemy żądany wynik:



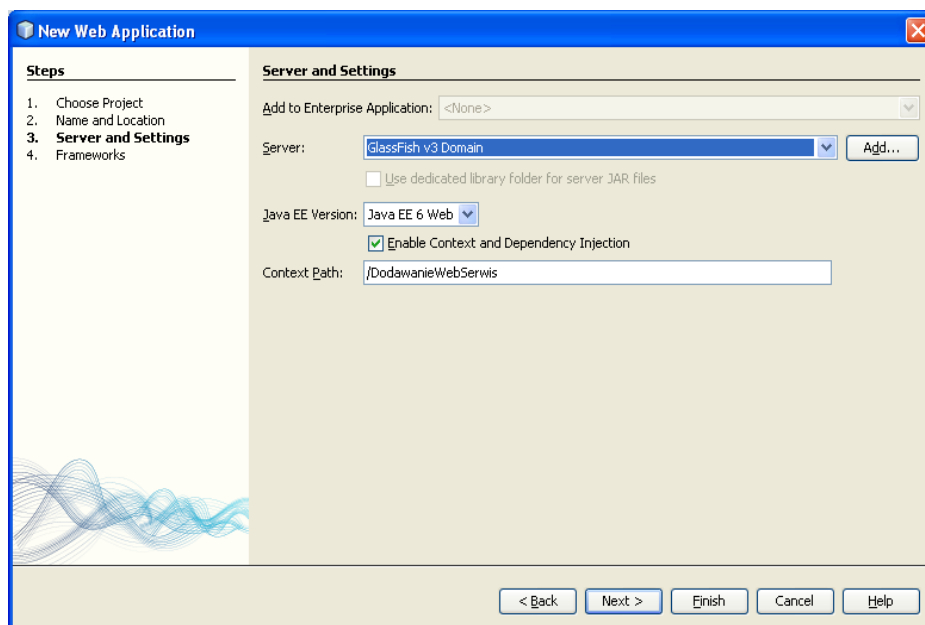
### 3.1.3. Korzystanie z usługi sieciowej z poziomu strony JSP w przeglądarce

Poniżej przedstawiono kolejne kroki w celu realizacji zadania:

- Z menu, w górnej części okna środowiska Netbeans wybieramy pozycję: **File → New Project...**
- W nowym oknie wybieramy kategorię **JavaWeb** i rodzaj projektu **Web Application**. Klikamy na przycisk **Next >**
- Nadajemy nazwę tworzonego projektu i klikamy na przycisk **Next >**

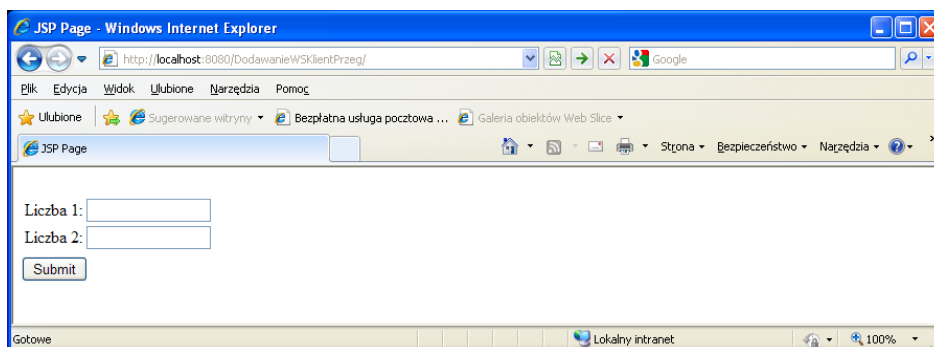


- Wybieramy serwer (skorzystamy z serwera GlassFish) oraz pakiet Java EE, klikamy na przycisk **Next >**



- e) W ostatnim kroku tworzenia projektu istnieje możliwość wyboru dodatkowych frameworków. W naszym przypadku pozostawiamy wszystkie pola odznaczone i klikamy na przycisk *Finish*
- f) Po utworzeniu projektu zostajemy automatycznie przekierowani do wygenerowanego pliku *index.jsp*. Posłużymy nam on jako strona, dzięki której będziemy odwoływali się do usługi sieciowej i wyświetlali wynik. Tworzymy więc formularz z dwoma polami *input* typu *text* oraz przyciskiem służącym do rozpoczęcia wysyłania danych (typ *submit*).

Wynik wykonanych działań (*Run* → *Run Main Project* lub przycisk *F6*):



Kod strony JSP (zamieszczono tylko edytowaną część `<body>`):

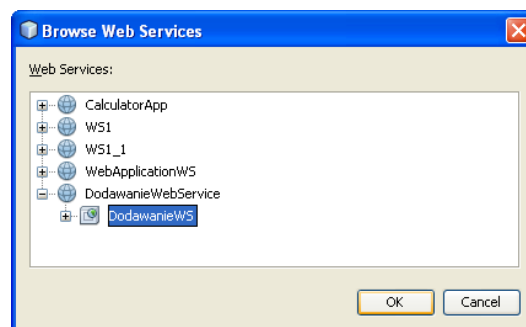
```
<body>
    <%//utworzenie zmiennych do przechowywania parametrów wywołania usługi
        Double l1 = null;
        Double l2 = null;
        //utworzenie zmiennych pobierających wartości z formularza
        String liczba1 = request.getParameter("liczba1");
        String liczba2 = request.getParameter("liczba2");
        //warunek początkowy dla liczby 1 istnienia strony jsp
        //aby w polach formularza nie były wyświetlone wartości "null"
        if(liczba1==null)
        {
            liczba1=" ";
            l1 = 0.0;
        }
        else //jesli wpisano wartosc w polu 1 do formularza, to zamien ja
            //na liczbę
            l1 = Double.parseDouble(liczba1);
        //warunek początkowy dla liczby 2 istnienia strony jsp
        //aby w polach formularza nie były wyświetlone wartości "null"
        if(liczba2==null)
```

```

    {
        liczba2=" ";
        l2 = 0.0;
    }
    else//jesli wpisano wartosc w polu 2 do formularza, to zamien ja
        //na liczbe
        l2 = Double.parseDouble(liczba2);
    %>
    <br/>
    <form action="index.jsp" method="get">
        <table border="0">
            <tr>
                <td>
                    Liczba 1:
                </td>
                <td>
                    <input type="text" name="liczba1" size="14" value="<%=liczba1 %>">
                </td>
            </tr>
            <tr>
                <td>
                    Liczba 2:
                </td>
                <td>
                    <input type="text" name="liczba2" size="14" value="<%=liczba2 %>">
                </td>
            </tr>
        </table>
        <input type="Submit" value="Submit"/>
    </form>
</body>

```

- g) Po utworzeniu odpowiedniego formularza należy odwołać się do żądanej usługi sieciowej w celu zwrócenia wyniku. W tym celu klikamy prawym przyciskiem myszy na nazwie projektu i wybieramy **New** → **Web Service Client...**
- h) Zostajemy poproszeni o wybór pliku WSDL. Podobnie jak wcześniej (w drugim punkcie instrukcji) wybieramy pierwszą opcję **Project** i klikamy na przycisk **Browse**. Szukamy nazwy aplikacji odpowiedzialnej za stworzenie usługi. Rozwijamy drzewko dla danej aplikacji, zaznaczamy nazwę wcześniej stworzonej usługi a następnie klikamy na przycisk **OK**.

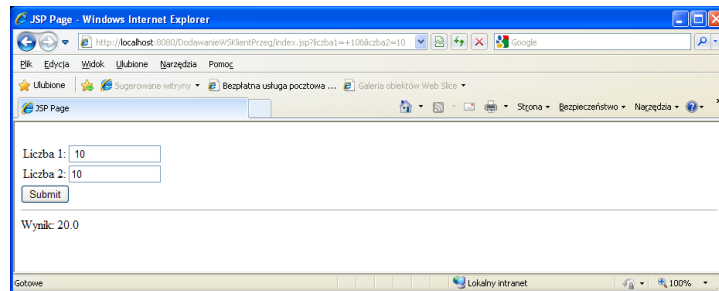


- i) Po powrocie do okna tworzenia klienta usługi klikamy na przycisk **Finish**
- j) Następuje generowanie klas na podstawie pliku WSDL. Po zakończeniu tych czynności postępujemy identycznie jak w punkcie 2(d) instrukcji. Przeciągamy zaznaczoną nazwę **dodajLiczby()** do części okna z kodem programu (najlepiej pomiędzy znacznikiem końca formularza **</form>** a znacznikiem końca ciała dokumentu **</body>**).
- k) Edytujemy wygenerowany kod tak, aby spełniał postawione zadanie, tj. umieszczamy deklarację zmiennej **result** przed blokiem **try{ }catch{ }** oraz wyświetlamy wynik zwracany przez usługę na stronę JSP.

```

</form>
<!-- start web service invocation --><hr/>
<%
    Double result=null;
    try{
        dodawanie.DodawanieWSService service = new
                                dodawanie.DodawanieWSService();
        dodawanie.DodawanieWS port =
                                service.getDodawanieWSPort();
        result = port.dodajLiczby(11, 12);
    } catch (Exception ex) {}
%>
<!-- end web service invocation -->
Wynik: <%=result %>
</body>

```



### 3.1.4. Korzystanie z usługi sieciowej z poziomu serwletu

Przykładowy kod źródłowy serwletu KlientSerwlet.java

```

import dodawanie.DodawanieWS_Service;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.xml.ws.WebServiceRef;

@WebServlet(name = "KlientSerwlet", urlPatterns = {"/KlientSerwlet"})
public class KlientSerwlet extends HttpServlet {

    @WebServiceRef(wsdlLocation =
        "INF/wsdl/localhost_8080/DodawanieWebSerwis/DodawanieWS.wsdl")
    private DodawanieWS_Service service;

    protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {

            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet KlientSerwlet</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Servlet KlientSerwlet at " + request.getContextPath() +
"</h1>");
            double l1 = 3.3d;

```

```

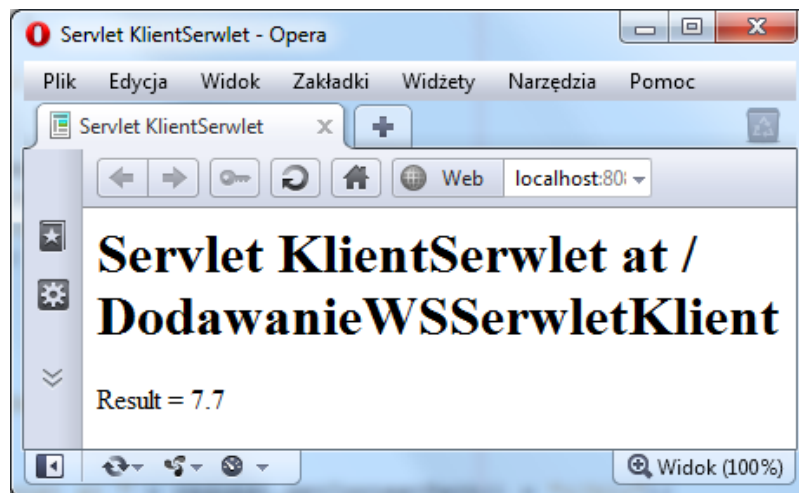
        double l2 = 4.4d;
        double result = dodajLiczby(l1, l2);
        out.println("Result = " + result);
        out.println("</body>");
        out.println("</html>");
    } finally {
        out.close();
    }
}

@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    processRequest(request, response);
}

@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    processRequest(request, response);
}

private double dodajLiczby(double l1, double l2) {
    dodawanie.DodawanieWS port = service.getDodawanieWSPort();
    return port.dodajLiczby(l1, l2);
}
}

```



## 3.2. Komunikacja sieciowa

### 3.2.1. Przykład analizy składniowej obiektu URL

```

import java.net.*;
import java.io.*;

public class ParseURL {
    public static void main(String[] args) throws Exception {
        URL url = new URL("http://host.prz.rzeszow.pl:80/directory/
        index.html#anchor");
        System.out.println("protocol = " + url.getProtocol());
        System.out.println("host = " + url.getHost());
        System.out.println("filename = " + url.getFile());
        System.out.println("port = " + url.getPort());
        System.out.println("ref = " + url.getRef());
    }
}

```

### 3.2.2. Program do sprawdzania na którym porcie działa serwer - program

LocalPortScanner.java

```
import java.net.*;
import java.io.*;
public class LocalPortScanner {
    public static void main(String args[]) {
        for (int port = 1; port <= 65535; port++) {
            try {
                ServerSocket server = new ServerSocket(port);
                server.close();
            }
            catch (IOException e) {
                // Jeśli w danym porcie działa już serwer
                System.out.println("W porcie " + port + " działa
                    serwer");
            }
        }
    }
}
```

### 3.2.3. Przykład komunikacji klient-serwer

#### o Program serwera

```
import java.net.*;
import java.io.*;
public class SimpleServer {
    public static void main(String args[]) throws IOException {
        ServerSocket s = null;
        try {
            // Gniazdo serwera w porcie 4444
            s = new ServerSocket(4444);
        } catch (IOException e) { System.err.println(e); }
        while (true) {
            try {
                // Czekanie na połączenie klienta
                Socket s1 = s.accept();
                // Strumień wyjścia związany z gniazdem
                OutputStream slout = s1.getOutputStream();
                DataOutputStream dos = new DataOutputStream(slout);
                // Wysłanie wiadomości z serwera!
                dos.writeUTF("Tu moj serwer na porcie 4444 !");
                // Zamknięcie połączenia
                dos.close();
                s1.close();
            }
            catch (IOException e) { }
        }
    }
}
```

#### o Program klienta

```
import java.net.*;
import java.io.*;
public class SimpleClient {
    public static void main(String args[]) {
        try {
            // Otwarcie połączenia z "local hosta" na porcie 4444
            Socket s1 = new Socket("127.0.0.1", 4444);
            // Strumień wejściowy z gniazda
            InputStream is = s1.getInputStream();
        }
    }
}
```

```

        DataInputStream dis = new DataInputStream(is);
        System.out.println(dis.readUTF());
        // Zamknięcie strumienia i połączenia
        dis.close();
        s1.close();
    }
    catch (ConnectException connExc) {
        System.err.println("Brak połączenia do serwera !"); }

    catch (IOException e) { } }
}

```

## 4. Elementy gramatyki języka

### 4.1. Typy sparametryzowane

#### 4.1.1. Klasa sparametryzowana

Kod klasy generycznej myGeneric.java	Przykład OneGenericDemo.java
<pre> class myGeneric&lt;T&gt; {     T ob;      myGeneric(T o) {         ob = o;     }      T getobj() {         return ob;     }      void showType() {         System.out.println("Type T: " + ob.getClass().getName());     } } </pre>	<pre> public class OneGenericDemo {     public static void main(String[] args) {         myGeneric&lt;Integer&gt; intObj;         // obiekt myGeneric dla typu Integer         intObj = new myGeneric&lt;Integer&gt;(123);         intObj.showType();         int i = intObj.getobj(); // brak rzutowania !         System.out.println("Value: " + i);         // obiekt myGeneric dla typu String         myGeneric&lt;String&gt; strObj = new             myGeneric&lt;String&gt;("Java");         strObj.showType();         String str = strObj.getobj();         System.out.println("Value: " + str);     } } </pre>

#### Zadania:

W powyższym przykładzie zastosować typ ograniczony `<T extends Number>` i sprawdzić działanie programu dla typów `String`, `Integer` i `Double`

Sparametryzować klasę generyczną **trzema** parametrami `<T, V, W>` i sprawdzić działanie zmodyfikowanego programu dla typów `String`, `Integer` i `Double`

#### 4.1.2. Metoda sparametryzowana - sparametryzować metodę dwoma parametrami

Klasa ze sparametryzowaną metodą
<pre> public class GenericMethods {     static &lt;T&gt; void printType(T anyType) {         System.out.println(anyType.getClass().getName());     }     public static void main(String[] args) {         GenericMethods.printType(123);         GenericMethods.printType(123.123);         GenericMethods.printType("Java");         GenericMethods.printType(String.class);         GenericMethods.printType(new String(""));     } } </pre>

#### 4.1.3. Interfejsy sparametryzowane - napisać program implementujący interfejs o postaci

```

interface MyInterface<T> {
    void myMethod(T arg);
}

```



## 4.2. Kolekcje

### 4.2.1. Przykład zastosowania klasy **ArrayList<E>**

```
import java.util.*;
public class ArrayListDemo {
    public static void main(String[] args) {
        ArrayList<String> arrayList = new ArrayList<String>();
        arrayList.add("Element 1");
        arrayList.add("Element 2");
        arrayList.add("Element 3");
        arrayList.add("Element 4");
        System.out.println("ArrayList contains ");
        for (int i = 0 ; i < arrayList.size() ; i++){
            System.out.println(arrayList.get(i));
        }
        System.out.println("Adding value into ArrayList");
        arrayList.set(2, "Element 33");
        for (String s:arrayList){
            System.out.println(s);
        }
    }
}
```

**Sprawdzić działanie bez stosowania typów sparametryzowanych !!!**

```
ArrayList arrayList =
new ArrayList();
```

```
arrayList.add("100");
String str = (String)
arrayList.get(0);
```

```
Integer i = (Integer)
arrayList.get(0);
```

#### Zadania:

- Wyświetlić zawartość listy za pomocą iteratora - metoda `iterator()`. Pobrać i wyświetlić element z listy o indeksie 2, usunąć dowolny element i wyświetlić zawartość listy.
- Przekonwertować zawartość listy do zwykłej tablicy Javy i wyświetlić jej rozmiar i zawartość.

### 4.2.2. Przykład zastosowania klasy **LinkedList<E>**

```
import java.util.LinkedList;
public class LinkedListDemo {
    public static void main(String[] args){
        LinkedList<String> link = new LinkedList<String>();
        link.add("Red");
        link.add("Black");
        link.add("Green");
        link.add("Blue");
        link.add("Black");
        link.add("Yellow");
        System.out.println("Elements\r\n");
        for (int i = 0 ; i < link.size() ; i++){
            String element = link.get(i);
            System.out.println(element);
        }
        System.out.println("\r\nSet Element\r\n");
        link.addFirst("Pink");
        link.set(5, "White");
        for (int i = 0 ; i < link.size() ; i++){
            String element = link.get(i);
            System.out.println(element);
        }
    }
}
```

#### Zadania:

- Zbadać działanie innych metod klasy `LinkedList` pozwalających na usunięcie pierwszego elementu, usunięcie ostatniego elementu, itp.
- Zliczyć i wyświetlić występowanie na liście elementu o nazwie "Black".
- Zbudować listę sparametryzowaną dwoma parametrami typu `Integer` i `Double`

### 4.2.3. Przykład zastosowania klasy **HashMap<K, V>**

```
import java.util.*;
```

```
public class HashMapDemo {
    public static void main(String[] args) {
        HashMap<Integer, String> hashMap = new HashMap<Integer, String>();
        map.put(new Integer(2) , "Two");
        map.put(new Integer(5) , "Five");
        map.put(new Integer(3) , "Three");
        map.put(new Integer(6) , "Six");
        map.put(new Integer(4) , "Four");
        map.put(new Integer(1) , "One");
        System.out.println("HashMap: " + hashMap);
        System.out.println("KeySet: " + hashMap.keySet());
        System.out.println("Values: " + hashMap.values());
        System.out.println("Key/Value");
        for (Integer key: hashMap.keySet()) {
            System.out.println(key + " = " + hashMap.get(key));
        }
        System.out.println("Demonstrate modification");
        hashMap.put(new Integer(7), "Seven");
        hashMap.put(new Integer(4), "Cztery");
        hashMap.remove(3);
        System.out.println("HashMap: " + hashMap);
        System.out.println("Size hashMap: " + hashMap.size());
        if (hashMap.containsKey(5)){
            System.out.print("Element 5: ");
            System.out.println(hashMap.get(5));
        } else {
            System.out.println("No element 5 !");
        }
    }
}
```

**Zadanie** Przerobić powyższy program z zastosowaniem klasy `TreeMap<K, V>`

#### 4.2.4. Przykład zastosowania algorytmów kolekcji

```
import java.util.*;
public class AlgorithmsDemo {
    public static void main(String args[]) {
        LinkedList<Integer> myList = new LinkedList<Integer>();
        myList.add(12); myList.add(-5); myList.add(3); myList.add(0);
        System.out.println("Original list");
        for (int s: myList){
            System.out.print(s + " ");
        } System.out.println();
        Comparator<Integer> comp = Collections.reverseOrder();
        Collections.sort(myList, comp); System.out.println("Sorted list");
        for (int s: myList){
            System.out.print(s + " ");
        } System.out.println();
        Collections.shuffle(myList); System.out.println("Shuffle list");
        for (int s: myList){
            System.out.print(s + " ");
        }
        System.out.println();
        System.out.println("Minimum: " + Collections.min(myList));
        System.out.println("Maximum: " + Collections.max(myList));
    }
}
```

**Zadanie** Posortować i wyświetlić zawartość kolekcji zarówno dla danych liczbowych oraz dla danych typu `String`

## 5. Zadania

Zadania podaje prowadzący dla każdej grupy laboratoryjnej