

Języki i metody programowania Java

Lab5 – pliki

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

http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/pojava/java_wyklad4.pdf

Zofia Kruczkiewicz

1. Należy dokonać analizy kodu programu Operacje_we_wy_Scaner

http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/pojava/Operacje_we_wy_Scaner.rar

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

```
1 package operacje_we_wy;
2
3 import java.io.BufferedReader;
4 import java.io.IOException;
5 import java.io.InputStreamReader;
6 import java.util.Scanner;
7 import java.util.StringTokenizer;
8
9 public class Operacje_we_wy {
10
11     String menu=
12         "1. We typu char\n"
13         + "2. WE typu boolean\n"
14         + "3. We typu String\n"
15         + "4. We typu byte\n"
16         + "5. We typu short\n"
17         + "6. We typu int\n"
18         + "7. We typu long\n"
19         + "8. We typu float\n"
20         + "9. We typu double\n"
21         + "x - koniec programu\n"
22         + "Podaj numer wejścia: ";
23
24     //wejście - pobranie ciągu znaków standardowego wejścia
25     InputStreamReader wejście = new InputStreamReader(System.in);
26     //bufor - klasa odczytuje wejściowy strumień znakowy ze strumienia
27     //o nazwie wejście i przechowuje w buforze
28     BufferedReader bufor = new BufferedReader(wejście);
29     //klasa do analizy składniowej jednostek leksykalnych tzw. leksemow
30     //(tokens) pobieranych metodą nextToken()
31     StringTokenizer bon;
32     Scanner scanner = new Scanner(System.in);
33 }
```

The left sidebar shows the project structure with the following components:

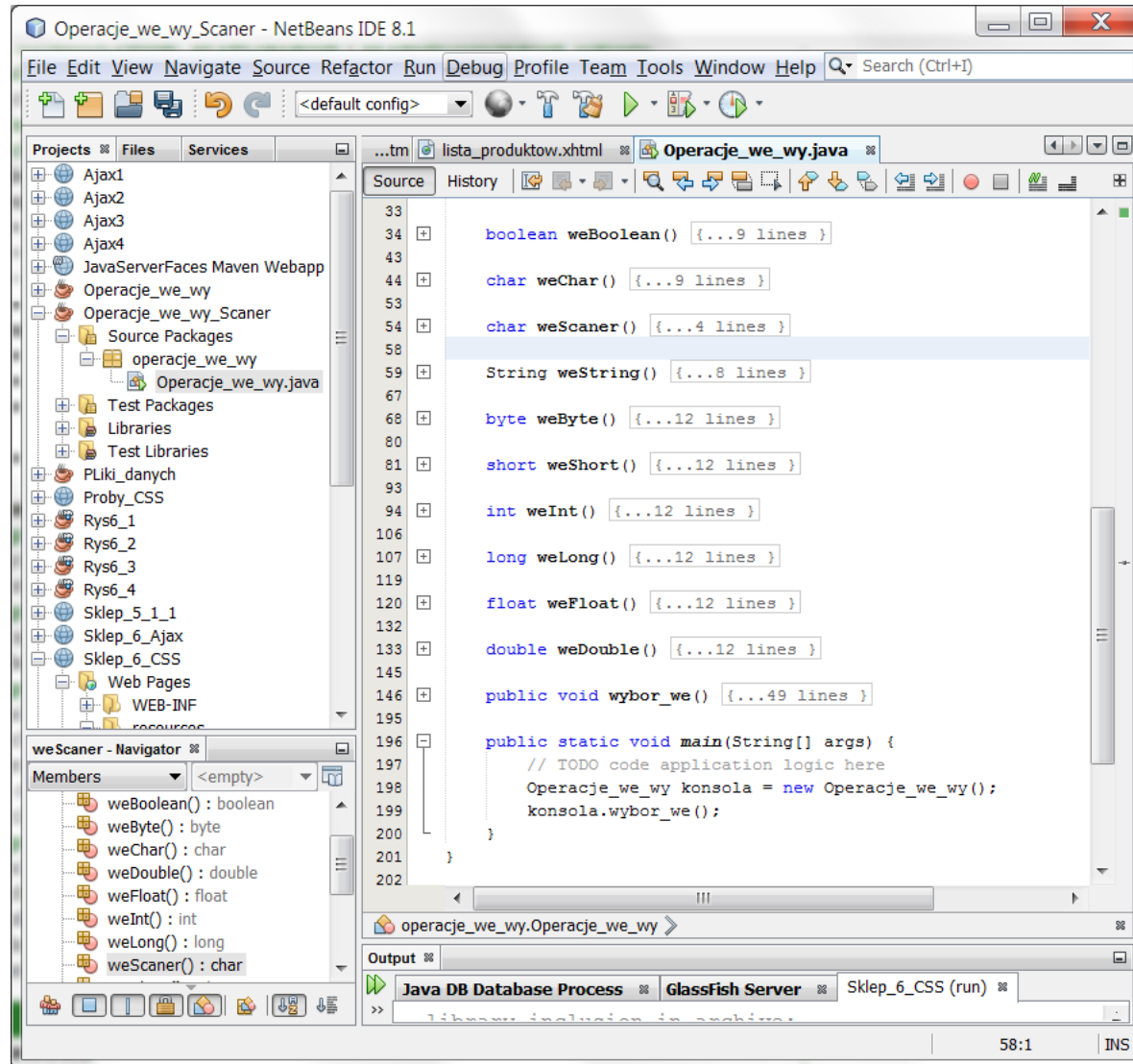
- Proj... (Project)
- Files
 - Ajax1
 - Ajax2
 - Ajax3
 - Ajax4
 - JavaServerFaces Maven V
 - Operacje_we_wy
 - Operacje_we_wy_Scaner
 - Source Packages
 - operacje_we_wy
 - Test Packages
 - Libraries
 - Test Libraries
 - PLiki_danych
 - Proby_CSS
 - Rys6_1
 - Rys6_2
 - Rys6_3
 - Rys6_4
 - Sklep_5_1_1
 - Sklep_6_Ajax
 - Sklep_6_CSS
 - Web Pages
 - WEB-INF
 - resources

The bottom status bar shows the following information:

- Output
- Java DB Database Process
- GlassFish Server
- Sklep_6_CSS (run)
- 1:1
- INS

1. Należy dokonać analizy kodu programu Operacje_we_wy_Scaner (cd)

http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/pojava/Operacje_we_wy_Scaner.rar



1. Należy dokonać analizy kodu programu Operacje_we_wy_Scaner (cd)

http://zofia.kruckiewicz.staff.iar.pwr.wroc.pl/wyklady/pojava/Operacje_we_wy_Scaner.rar

The screenshot displays the NetBeans IDE 8.1 interface. The main editor window shows the source code for the `wybor_we()` method, which is part of a `while(true)` loop. The code uses a `switch` statement to handle different menu options ('1' through 'k'). Each option calls a corresponding `we*` method (e.g., `weChar()`, `weBoolean()`, etc.) and then prints a message. The `main` method is also visible, showing the creation of an `Operacje_we_wy` object and the call to `wybor_we()`.

```
146     public void wybor_we() {
147         char ktory;
148         do {
149             System.out.print(menu);
150             ktory = weScanner();
151             switch (ktory) {
152                 case ('1'):System.out.print("Podaj char: ");
153                     System.out.println(weChar());break;
154                 case ('2'):System.out.print("Podaj boolean: ");
155                     System.out.println(weBoolean());break;
156                 case ('3'):System.out.print("Podaj String: ");
157                     System.out.println(weString());break;
158                 case ('4'):System.out.print("Podaj byte: ");
159                     System.out.println(weByte());break;
160                 case ('5'):System.out.print("Podaj short: ");
161                     System.out.println(weInt());break;
162                 case ('6'):System.out.print("Podaj int: ");
163                     System.out.println(weShort());break;
164                 case ('7'):System.out.print("Podaj long: ");
165                     System.out.println(weLong());break;
166                 case ('8'):System.out.print("Podaj float: ");
167                     System.out.println(weFloat());break;
168                 case ('9'):System.out.print("Podaj double: ");
169                     System.out.println(weDouble());break;
170                 case ('k'):System.exit(0);
171                 default:System.out.println("Zla opcja");
172             }
173         } while (true);
174     }
175     public static void main(String[] args) {
176         // TODO code application logic here
177         Operacje_we_wy konsola = new Operacje_we_wy();
178         konsola.wybor_we();
179     }
180 }
181
```

The left sidebar shows the project structure, including the `Operacje_we_wy` package and its sub-packages. The bottom status bar indicates the current execution context: `operacje_we_wy.Operacje_we_wy > wybor_we > do ... while (true) > switch (ktory) > case ('&apo; ...`.

1. Należy dokonać analizy kodu programu Operacje_we_wy_Scaner (cd)

http://zofia.kruckiewicz.staff.iar.pwr.wroc.pl/wyklady/pojava/Operacje_we_wy_Scaner.rar

```
boolean weBoolean() {
    try {
        bon = new StringTokenizer(bufor.readLine());
        return Boolean.parseBoolean(bon.nextToken());
    } catch (IOException e) {
        System.err.println("Blad IO Boolean " + e);
        return false;
    }
}

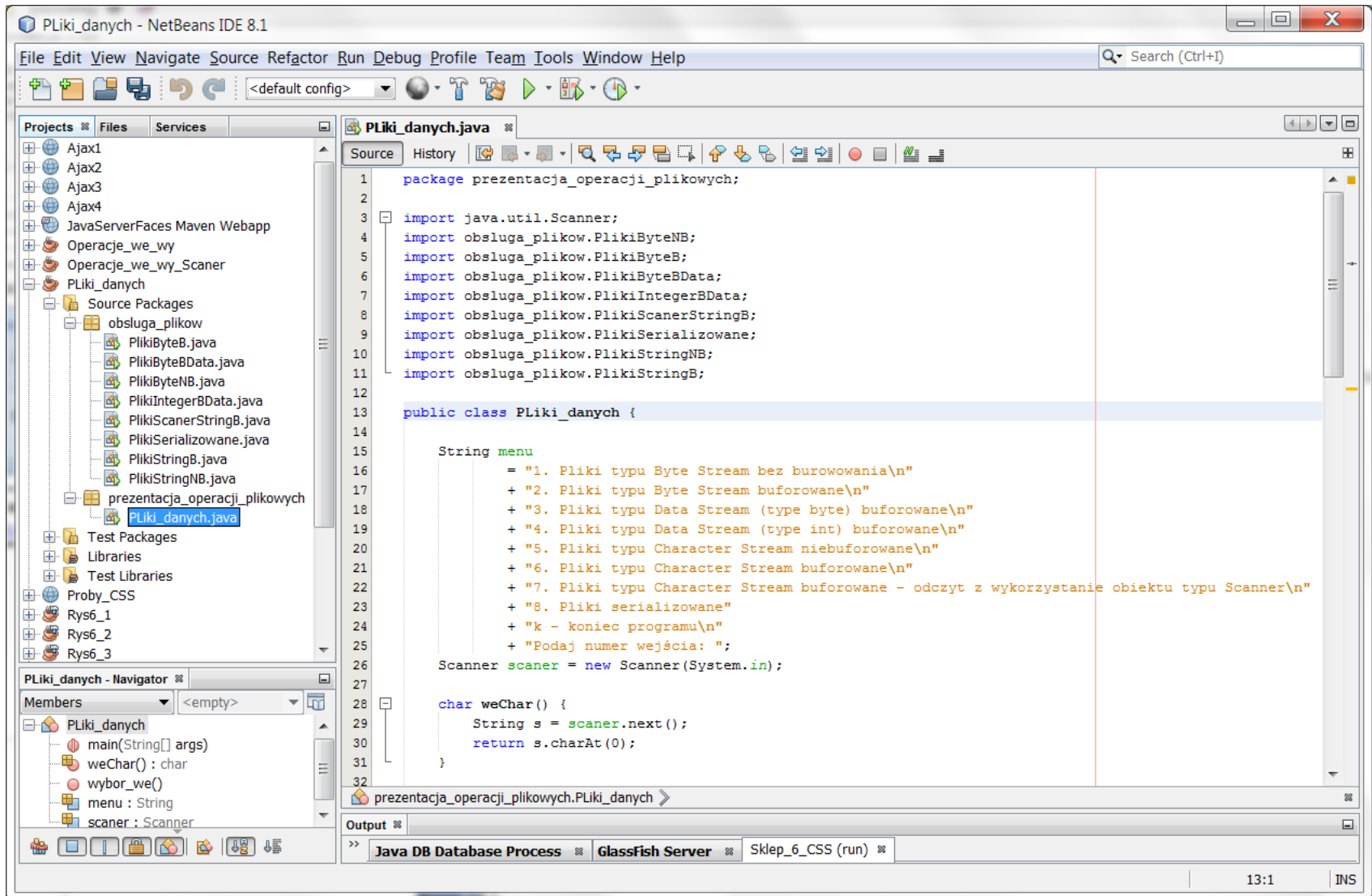
char weChar() {...9 lines }
char weScanner() {...4 lines }
String weString() {...8 lines }
byte weByte() {...12 lines }
short weShort() {...12 lines }
int weInt() {
    try {
        bon = new StringTokenizer(bufor.readLine());
        return Integer.parseInt(bon.nextToken());
    } catch (IOException e) {
        System.err.println("Blad IO int " + e);
        return 0;
    } catch (NumberFormatException e) {
        System.err.println("Blad formatu int " + e);
        return 0;
    }
}

long weLong() {...12 lines }
float weFloat() {
    try {
        bon = new StringTokenizer(bufor.readLine());
        return Float.parseFloat(bon.nextToken());
    } catch (IOException e) {
        System.err.println("Blad IO float " + e);
        return 0.0F;
    } catch (NumberFormatException e) {
        System.err.println("Blad formatu float " + e);
        return 0.0F;
    }
}

double weDouble() {...12 lines }
```

2. Należy dokonać analizy kodu programu Pliki_danych

http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/pojava/Pliki_danych.rar



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

```
1 package prezentacja_operacji_plikowych;
2
3 import java.util.Scanner;
4 import obsługa_plikow.PlikiByteNB;
5 import obsługa_plikow.PlikiByteB;
6 import obsługa_plikow.PlikiByteBData;
7 import obsługa_plikow.PlikiIntegerBData;
8 import obsługa_plikow.PlikiScannerStringB;
9 import obsługa_plikow.PlikiSerializowane;
10 import obsługa_plikow.PlikiStringNB;
11 import obsługa_plikow.PlikiStringB;
12
13 public class Pliki_danych {
14
15     String menu
16         = "1. Pliki typu Byte Stream bez buforowania\n"
17         + "2. Pliki typu Byte Stream buforowane\n"
18         + "3. Pliki typu Data Stream (type byte) buforowane\n"
19         + "4. Pliki typu Data Stream (type int) buforowane\n"
20         + "5. Pliki typu Character Stream niebuforowane\n"
21         + "6. Pliki typu Character Stream buforowane\n"
22         + "7. Pliki typu Character Stream buforowane - odczyt z wykorzystaniem obiektu typu Scanner\n"
23         + "8. Pliki serializowane"
24         + "k - koniec programu\n"
25         + "Podaj numer wejścia: ";
26     Scanner scanner = new Scanner(System.in);
27
28     char weChar() {
29         String s = scanner.next();
30         return s.charAt(0);
31     }
32 }
```

The IDE interface includes a Project Explorer on the left showing the project structure, a Members window below it, and an Output window at the bottom. The status bar at the bottom right shows the time 13:1 and the text "INS".

2. Należy dokonać analizy kodu programu Pliki_danych (cd)

http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/pojava/Pliki_danych.rar

```
33 public void wybor_we() {
34     char ktory = 0;
35     do {
36         System.out.print(menu);
37         ktory = weChar();
38         switch (ktory) {
39             case ('1'):
40                 PlikiByteNB.Zapiszplik();
41                 PlikiByteNB.Odczytajplik();
42                 break;
43             case ('2'):
44                 PlikiByteB.Zapiszplik();
45                 PlikiByteB.Odczytajplik();
46                 break;
47             case ('3'):
48                 PlikiIntegerBData.Zapiszplik();
49                 PlikiIntegerBData.Odczytajplik();
50                 break;
51             case ('4'):
52                 PlikiIntegerBData.Zapiszplik();
53                 PlikiIntegerBData.Odczytajplik();
54                 break;
55             case ('5'):
56                 PlikiStringNB.Zapiszplik();
57                 PlikiStringNB.Odczytajplik();
58                 break;
59             case ('6'):
60                 PlikiStringB.Zapiszplik();
61                 PlikiStringB.Odczytajplik();
62                 break;
63             case ('7'):
64                 PlikiScannerStringB.Zapiszplik();
65                 PlikiScannerStringB.Odczytajplik();
66                 break;
67             case ('8'):
68                 PlikiSerializowane.Zapiszobiektzdopliku();
69                 PlikiSerializowane.Odczytajobiektzpliku();
70                 break;
71             case ('k'):
72                 System.exit(0);
73             default:
74                 System.out.println("Zla opcja");
75         }
76     } while (true);
77 }
```

Members

- PLiki_danych
 - main(String[] args)
 - weChar(): char
 - wybor_we()
 - menu: String
 - scaner: Scanner

Output

Java DB Database Process GlassFish Server Sklep_6_CSS (run)

41:48 INS

2. Należy dokonać analizy kodu programu Pliki_danych (cd) – opcja 6 (klasa PlikiStringB)

http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/pojava/Pliki_danych.rar

The screenshot displays the NetBeans IDE interface for a Java project named "PLiki_danych". The main editor window shows the source code of the class "PlikiStringB.java". The code is as follows:

```
17 static public void Zapiszplik() {
18     String dane = "1";
19     try {
20         FileWriter plik = new FileWriter("plikStringB.txt");
21         try (BufferedWriter bufor1 = new BufferedWriter(plik)) {
22             while (!dane.equals("")) {
23                 System.out.print("Podaj dane: ");
24                 dane = weString();
25                 if (!dane.equals("")) {
26                     bufor1.write(dane, 0, dane.length());
27                 }
28             }
29             bufor1.close();
30         }
31     } catch (IOException e) {
32         System.out.println("Bład zapisu pliku tekstowego" + e);
33     }
34 }
35 static public void Odczytajplik() {
36     String dane;
37     try {
38         FileReader plik = new FileReader("plikStringB.txt");
39         try (BufferedReader bufor1 = new BufferedReader(plik)) {
40             dane = bufor1.readLine();
41             while (dane != null) {
42                 System.out.println(dane);
43                 dane = bufor1.readLine();
44             }
45             bufor1.close();
46         }
47         System.out.println("");
48     } catch (IOException e) {
49         System.out.println("Bład odczytu pliku tekstowego" + e);
50     }
51 }
52
53 public static void main(String[] args) {
54     Zapiszplik();
55     Odczytajplik();
56 }
57
58 }
```

The left sidebar shows the project structure with the following packages and classes:

- Source Packages
 - obsługa_plikow
 - PlikiByteB.java
 - PlikiByteBData.java
 - PlikiByteNB.java
 - PlikiIntegerBData.java
 - PlikiScannerStringB.java
 - PlikiSerializowane.java
 - PlikiStringB.java
 - PlikiStringNB.java
 - prezentacja_operacji_plikowych
 - PLiki_danych.java
- Test Packages
- Libraries
- Test Libraries

The bottom status bar shows the current cursor position at line 58, column 70. The output window at the bottom shows the following text:

```
>> Java DB Database Process GlassFish Server PLiki_danych (run)
```


2. Należy dokonać analizy kodu programu Pliki_danych (cd) – opcja 8(klasa PlikiSerializowane)

http://zofia.kruczkiewicz.staff.iar.pwr.wroc.pl/wyklady/pojava/Pliki_danych.rar

```
1 package obsługa_plikow;
2
3 import java.io.BufferedReader;
4 import java.io.FileInputStream;
5 import java.io.FileOutputStream;
6 import java.io.IOException;
7 import java.io.InputStreamReader;
8 import java.io.ObjectInputStream;
9 import java.io.ObjectOutputStream;
10 import java.io.Serializable;
11 import java.util.Date;
12
13 class Wiadomosc implements Serializable {
14     String dane;
15     Date data;
16     static String weString() {
17         InputStreamReader wejście = new InputStreamReader(System.in);
18         BufferedReader bufor = new BufferedReader(wejście);
19         System.out.print("Podaj wiadomosc: ");
20         try {
21             return bufor.readLine();
22         } catch (IOException e) {
23             System.err.println("Bład IO String");
24             return "";
25         }
26     }
27     public void zapiszWiadomosc(Date d) {
28         data = d;
29         System.out.println(data);
30         dane = weString();
31     }
32     public void odczytajWiadomosc() {
33         System.out.println(data);
34         System.out.println(dane);
35     }
36 }
```

Output window: Java DB Database Process, GlassFish Server, Pliki_danych (run)

2. Należy dokonać analizy kodu programu Pliki_danych (cd) – opcja 8(klasa PlikiSerializowane)

http://zofia.kruczkiewicz.staff.iar.pwr.wroc.pl/wyklady/pojava/Pliki_danych.rar

Sat Dec 17 17:10:26 CET 2016

Podaj wiadomosc: w1

Sat Dec 17 17:10:26 CET 2016

w1

Thu Jan 01 01:00:00 CET 1970

w1

Sat Dec 17 17:10:26 CET 2016

w1

Sat Dec 17 17:10:26 CET 2016

w1

```
38 public class PlikiSerializowane {
39     public static void Zapiszobiektzdopliku() {
40         Date d = new Date();
41         Wiadomosc wiadomosc = new Wiadomosc();
42         wiadomosc.zapiszWiadomosc(d);
43     }
44     try {
45         FileOutputStream plikobiektow = new FileOutputStream("Wiadomosc.obj");
46         ObjectOutputStream strumienobiektow = new ObjectOutputStream(plikobiektow);
47         strumienobiektow.writeObject(wiadomosc);
48         wiadomosc.odczytajWiadomosc();
49         d.setTime(1); //zmiana daty w drugim obiekcie typu Wiadomosc
50         strumienobiektow.writeObject(wiadomosc);
51         System.out.println("Obiekt wiadomosc zostal zapisany do pliku dwa razy");
52         wiadomosc.odczytajWiadomosc(); //obiekt wiadomosc ze zmieniona data
53         strumienobiektow.close();
54     } catch (IOException e) {
55         System.out.println("Blad zapisu pliku obiektowego" + e);
56     }
57 }
58 public static void Odczytajobiektyzpliku() {
59     Wiadomosc wiadomosc;
60     Wiadomosc wiadomosc1;
61     try {
62         FileInputStream plikobiektow = new FileInputStream("Wiadomosc.obj");
63         ObjectInputStream strumienobiektow = new ObjectInputStream(plikobiektow);
64     }
65     wiadomosc = (Wiadomosc) strumienobiektow.readObject();
66     System.out.println("Obiekt wiadomosc zostal odczytany z pliku");
67     if (wiadomosc != null)
68         wiadomosc.odczytajWiadomosc(); //obiekt zawiera dane wstawiona przed zapisem
69     wiadomosc1 = (Wiadomosc) strumienobiektow.readObject();
70     System.out.println("Obiekt wiadomosc zostal odczytany z pliku");
71     if (wiadomosc1 != null)
72         wiadomosc1.odczytajWiadomosc(); //zmieniona data nie zostala zapisana do strumienia
73     strumienobiektow.close();
74 }
75 } catch (IOException | ClassNotFoundException e) {
76     System.out.println("Blad odczytu pliku obiektowego" + e);
77 }
78 }
79 }
80 public static void main(String[] args) {
81     Zapiszobiektzdopliku();
82     Odczytajobiektyzpliku();
83 }
```

3. Wykonanie pliku tekstowego (typu Character Stream)

http://zofia.kruczkiewicz.staff.iar.pwr.wroc.pl/wyklady/pojava/PLiki_danych.rar

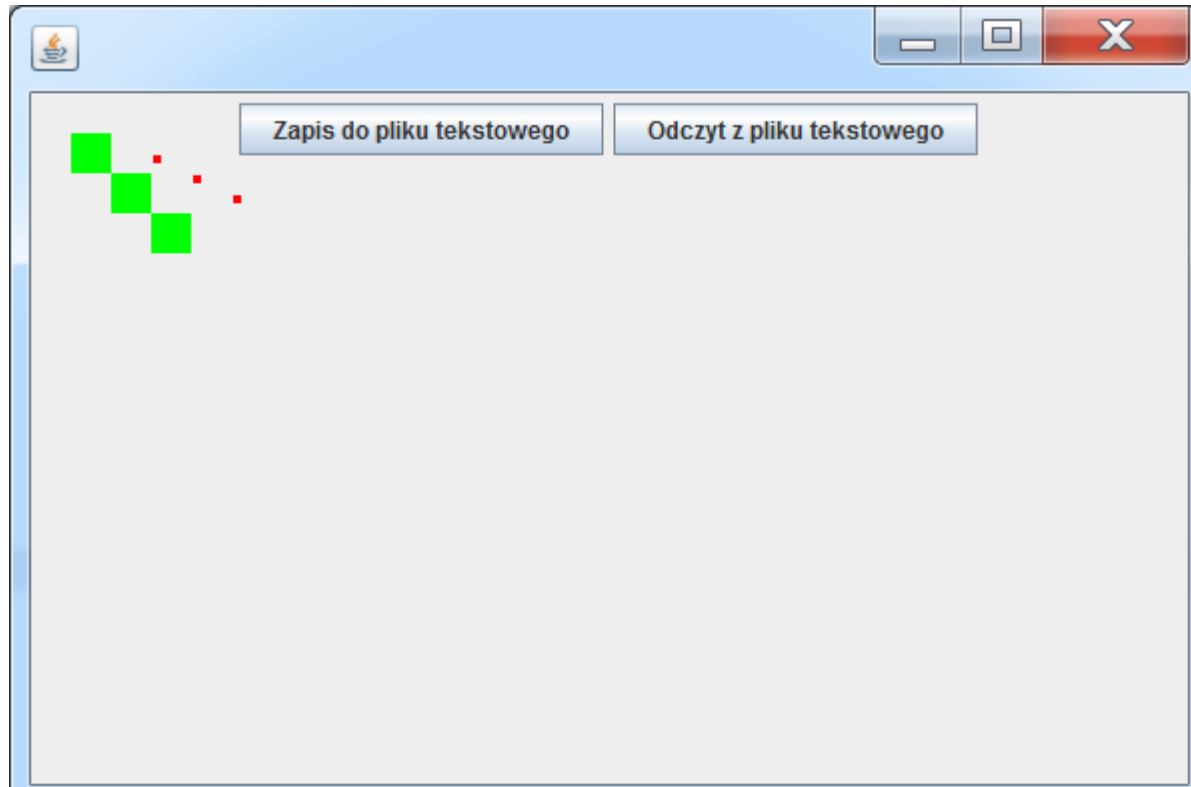
Wykorzystanie kodu klasy PlikuStringB

1. Należy w kopii programu wykonanego w lab4, jako rozwinięcie programu Rys1_1 dodać dwa przyciski JButton: jeden do zapisu danych o figurach, drugi do odczytu z pliku danych o figurach i wydruk na ekranie.

1.1. Należy wykonać kopię wybranego programu z p.1 (lub jego rozwinięcia)

1.2. Należy zmodyfikować kod metody toString() w klasach z rodziny Punkt lub pozostawić kod istniejącej metody

1.3. Należy dodać dwa przyciski typu JButton w klasie Figury_panel (p.1.4)



3. Wykonanie pliku tekstowego (typu Character Stream) cd

http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/pojava/Pliki_danych.rar

1.4. Należy dodać w klasie Figury_panel dwa przyciski i obsługę zdarzeń dla tych przycisków.

```
public class Figury_panel extends JPanel implements ActionListener {  
    FiguryHashSet kontroler;  
    protected JButton zapis = new JButton("Zapis do pliku tekstowego");  
    protected JButton odczyt = new JButton("Odczyt z pliku tekstowego");
```

@Override

```
    protected void paintComponent(Graphics g) {  
        super.paintComponent(g);  
        kontroler.rysuj_figury(g);  
    }  
    public void init() {  
        kontroler = new FiguryHashSet();  
        kontroler.pojemnik();  
        kontroler.wypelnij();  
        zapis.addActionListener(this);  
        odczyt.addActionListener(this);  
        add(zapis);  
        add(odczyt);  
    }
```

3. Wykonanie pliku tekstowego (typu Character Stream) cd

http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/pojava/Pliki_danych.rar

1.5. Należy dodać metodę obsługi zdarzeń **actionPerformed(ActionEvent evt)** w klasie `Figury_panel`.

```
public void actionPerformed(ActionEvent evt) {
    Object zrodlo = evt.getSource();
    if (zrodlo == zapis) {
        kontroler.Zapis_do_pliku();
    }
    else if (zrodlo == odczyt) {
        kontroler.Odczytaj_plik();
    }
    requestFocus();
        //przywrocenie zdolnosc do obslugi zdarzen
    repaint();
}
```

3. Wykonanie pliku tekstowego (typu Character Stream) cd

http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/pojava/Pliki_danych.rar

Wykorzystanie kodu klasy PlikiStringB

1.6. Należy wykonać dwie nowe metody w klasie FiguryHashSet, stosując linię try do otwierania i zamykania strumieni (wg wykładu

http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/pojava/java_wyklad4.pdf):

```
public void Zapis_do_pliku() {
    try {
        FileWriter plik = new FileWriter("Figury_text.txt");
        BufferedWriter bufor1 = new BufferedWriter(plik);
        /* zapis w pętli łańcucha znaków zwracanego z metody toString() wywołanej od
        każdego elementu z tablicy figury*/
    }
}
```

```
public void Odczytaj_plik() {
    String dane;
    try {
        FileReader plik = new FileReader("Figury_text.txt");
        BufferedReader bufor1 = new BufferedReader(plik);
        /* odczyt zawartości pliku tekstowego na ekran */
    }
}
```

3. Wykonanie pliku tekstowego (typu Character Stream) cd

http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/pojava/Pliki_danych.rar

1.7. W klasie
Obraz1_1 należy
zmodyfikować
rozmiar ramki

```
void rysunek_Swing() {  
    JFrame ramka = new JFrame();  
    Figury_panel panel = new Figury_panel();  
    panel.init();  
    ramka.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);  
    ramka.setSize(600, 400);  
    ramka.setContentPane(panel);  
    ramka.setVisible(true);  
}
```

1.8. Przykład wyniku tworzenia i wyświetlania zawartości pliku z informacją o figurach

```
Punkt{x=60, y=30}  
Kwadrat{dlugosc=20} i dziedzicze od  
Punkt{x=20, y=20}  
Kwadrat{dlugosc=20} i dziedzicze od  
Punkt{x=40, y=40}  
Kwadrat{dlugosc=20} i dziedzicze od  
Punkt{x=60, y=60}  
Punkt{x=100, y=50}  
Punkt{x=80, y=40}
```


4. Wykonanie pliku serializowanych obiektów (typu Object Stream) cd

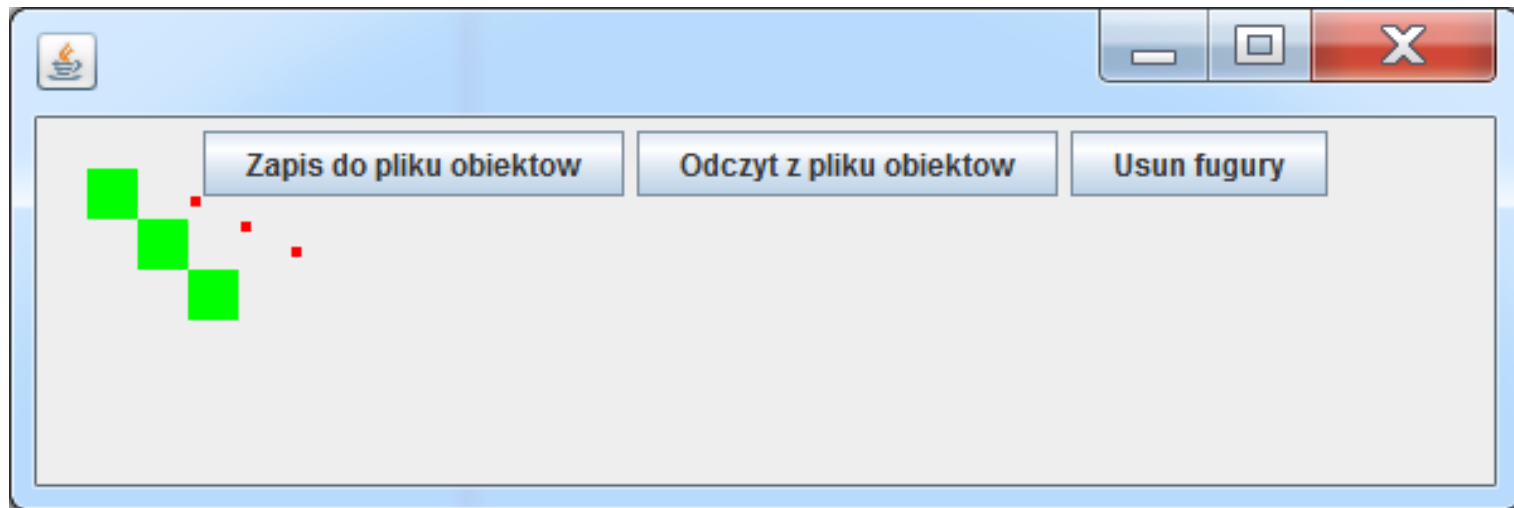
http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/pojava/Pliki_danych.rar

Wykorzystanie kodu klasy **PlikiSerializowane**

1. Należy w kopii programu wykonanego w lab4, jako rozwinięcie programu Rys1_2 dodać trzy przyciski JButton: jeden do zapisu obiektów z kolekcji **figury** z rodziny Punkt, drugi do odczytu z pliku kolekcji **figury** z obiektami z rodziny Punkt, a trzeci do usunięcia wszystkich obiektów z kolekcji figury.

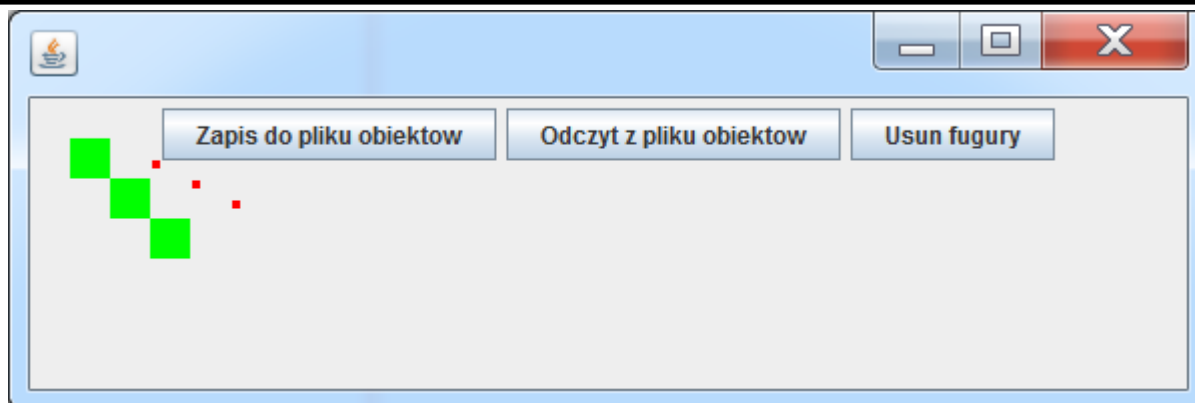
1.1. Należy wykonać kopię wybranego programu z p.1 (lub jego rozwinięcia)

1.2. Należy dodać trzy przyciski typu JButton w klasie Figury_panel (p.1.4)

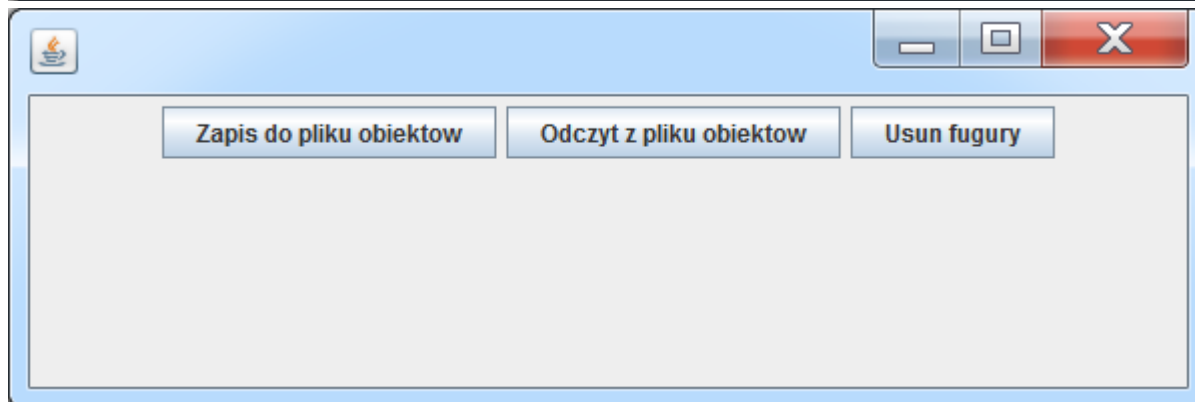


4. Wykonanie pliku serializowanych obiektów (typu Object Stream) cd http://zofia.kruczkiewicz.staff.iar.pwr.wroc.pl/wyklady/pojava/Pliki_danych.rar

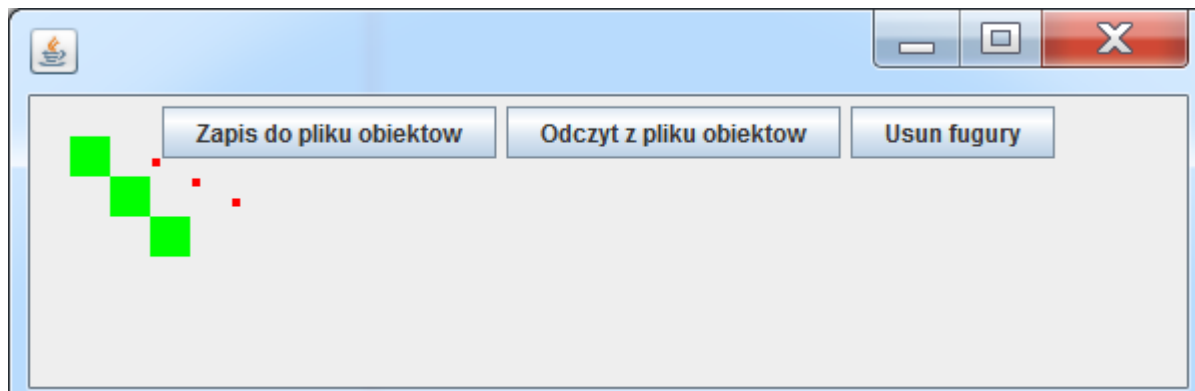
Wykorzystanie kodu klasy **PlikiSerializowane**



Zapis figur do pliku
za pomocą przycisku
Zapisz do pliku obiektow



Usunięcie figur z
programu za pomocą
przycisku **Usun figury**



Odczyt figur z pliku
i zapis w kolekcji figury
Odczyt z pliku obiektow

4. Wykonanie pliku serializowanych obiektów (typu Object Stream) cd

http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/pojava/Pliki_danych.rar

Wykorzystanie kodu klasy **PlikiSerializowane**

1.3. Należy dodać w klasie Figury_panel trzy przyciski i obsługę zdarzeń dla tych przycisków.

```
public class Figury_panel extends JPanel implements ActionListener {  
    FiguryHashSet kontroler;  
    protected JButton zapis = new JButton("Zapis do pliku obiektow");  
    protected JButton odczyt = new JButton("Odczyt z pliku obiektow");  
    protected JButton usun=new JButton("Usun fugury");
```

@Override

```
protected void paintComponent(Graphics g) {  
    super.paintComponent(g);  
    kontroler.rysuj_figury(g); }
```

public void init()

```
{ kontroler=new FiguryArrayList();  
  kontroler.pojemnik();  
  kontroler.wypelnij();  
  zapis.addActionListener(this);  
  odczyt.addActionListener(this);  
  usun.addActionListener(this);  
  add(zapis);  
  add(odczyt);  
  add(usun); }
```

4. Wykonanie pliku serializowanych obiektów (typu Object Stream) cd

http://zofia.kruckiewicz.staff.iiar.pwr.wroc.pl/wyklady/pojava/Pliki_danych.rar

Wykorzystanie kodu klasy **PlikiSerializowane**

1.4. Należy dodać metodę obsługi zdarzeń **actionPerformed(ActionEvent evt)** w klasie **Figury_panel**.

```
public void actionPerformed(ActionEvent evt) {  
    Object zrodlo = evt.getSource();  
    if (zrodlo == zapis) {  
        kontroler.Zapis_do_pliku();  
    } else if (zrodlo == odczyt) {  
        kontroler.Odczytaj_plik();  
    } else if (zrodlo == usun) {  
        kontroler.figury.clear();  
    }  
    requestFocus();  
    repaint();  
}
```

4. Wykonanie pliku serializowanych obiektów (typu Object Stream) cd

http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/pojava/Pliki_danych.rar

Wykorzystanie kodu klasy **PlikiSerializowane**

1.5. Należy wykonać dwie nowe metody w klasie **FiguryArrayList** stosując linię try do otwierania i zamykania strumieni (wg wykładu

http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/pojava/java_wyklad4.pdf)

```
public void Zapis_do_pliku() {
```

```
    try {
```

```
        FileOutputStream plikobiekow = new FileOutputStream("Wiadomosc.obj");
```

```
        ObjectOutputStream strumienobiekow = new ObjectOutputStream(plikobiekow);
```

```
        /* Zapis kolekcji figury*/
```

```
    }
```

```
public void Odczytaj_plik() {
```

```
    try {
```

```
        FileInputStream plikobiekow = new FileInputStream("Wiadomosc.obj");
```

```
        ObjectInputStream strumienobiekow = new ObjectInputStream(plikobiekow);
```

```
        /* Odczyt kolekcji figury. To pozwala odtworzyć stan programu przed zapisem figur do pliku.*/
```

```
    }
```

4. Wykonanie pliku serializowanych obiektów (typu Object Stream) cd

http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/pojava/Pliki_danych.rar

Wykorzystanie kodu klasy **PlikiSerializowane**

1.7. W klasie `Obraz1_1` należy zmodyfikować rozmiar ramki

```
void rysunek_Swing() {  
    JFrame ramka = new JFrame();  
    Figury_panel panel = new Figury_panel();  
    panel.init();  
    ramka.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);  
    ramka.setSize(600, 400);  
    ramka.setContentPane(panel);  
    ramka.setVisible(true);  
}
```