Wykład 7

Refaktoryzacja oprogramowania

autor: Zofia Kruczkiewicz

Definicja

Refaktoryzacja (*ang. Refactoring*) jest procesem dokonywania zmian w strukturze systemu informatycznego, które nie mają wpływu na jego zasadniczą funkcjonalność, natomiast poprawiają:

- **wydajność**
- **funkcjonalność**
- **koszt**
- **jakość oprogramowania:**
  - **Testowalność**
  - **Pielęgnowalność**
  - **Wielożywalność**
  - **Zrozumiałość**
  - **Stopień osiągniętej abstrakcji**

Rodzaje refaktoryzacji

- Kodu źródłowego
- Projektu
- Warstw systemu: klienta, prezentacji, biznesowej, integracji, bazy danych

Sposoby dokonania refaktoryacji

- Ręczna
- Częściowo zautomatyzowana
- Automatyczna

Kiedy refaktoryzować?

- Dalszy rozwój i konieczność wprowadzania zmian w systemie – a struktura systemu utrudnia dodanie nowego przypadku użycia lub jego zmianę,
- Wprowadzenie wielużywalności części systemu
- Utrudniona lokalizacja błędów
- Utrudniona inspekcja systemu = zbyt duża jego złożoność
- Wystąpienie tzw. ***bad smells***, czyli wystąpienia powtarzającego się kodu, długiej listy parametrów przekazywanych do metod, zbytecznych fragmentów kodu , zbyt długich i zawiłych metod lub za dużych klas = prowadzących do złych wartości metryk oprogramowania
- Generowanie dużego ruchu w sieci przez system

Bad smells w kodzie

- Powtarzający się kod, niejednoznaczny wynik metod w przypadku normalnej pracy oraz błędnego wykonania metody
- Długa metoda - trudna do zrozumienia
- Duża klasa – duże wartości LCOM
- Pola tymczasowe - istnienie atrybutów, które są tylko czasem używane
- Długa lista parametrów
- Rozbieżna zmiana – jeśli klasa w różnych zastosowaniach jest wykorzystywana tylko częściowo, wtedy należy wykonać kilka klas
- Zazdrość o kod - występuje wtedy, gdy metody jednej klasy często wywołują metody i używają dane innej klasy.
- Poszatkowanie - to przeciwieństwo rozbieżnej zmiany. Pojawia się w chwili, gdy drobna zmiana w jednej klasie wymaga dokonania drobnych modyfikacji w innych klasach
- Zbitki danych - pojawiają się wtedy, gdy w wielu klasach wykorzystywane są grupy podobnych, a nawet tych samych pól.
- Instrukcje ***switch*** świadczą często o braku polimorfizmu - należy ją zastąpić wywoływaniem metod wirtualnych
- Leniwa klasa - niepotrzebna klasa
- Nie wykorzystane fragmenty kodu lub całe klasy - zostały dodane z myślą o tym, że w przyszłości mogą się przydać,
- Komentarze - powinny zawierać istotne informacje,

Co można refaktoryzować?

- Enkapsulacja atrybutu
- Ekstrakcja zmiennej lokalnej
- Ekstrakcja metody
- Zmiana nazwy
- Ekstrakcja stałej
- Ekstrakcja superklasy/interfejsu

Opis operacji refraktoryzacji dostępnych w środowisku NetBeans

- **Rename** Enables you to change the name of a class, variable, or method to something more meaningful. In addition, it updates all source code in your project to reference the element by its new name.
- **Introduce Variable, Constant, Field, or Method** Enables you to generate a statement based on the selected code and replace that block of code with a call to the statement.
- **Change Method Parameters** Enables you to add parameters to a method and change the access modifier.
- **Encapsulate Fields** Generates a getter method and and a setter method for a field and optionally updates all referencing code to access the field using the getter and setter methods.
- **Pull Up** Moves methods and fields to a class that their current class inherits from.
- **Push Down** Moves inner classes, methods, and fields to all subclasses of their current class.
- **Move Class** Moves a class to another package or into another class. In addition, all source code in your project is updated to reference the class in its new location.
- **Copy Class** Copies a class to the same or a different package.
- **Move Inner to Outer Level** Moves an inner class one level up in hierarchy.
- **Convert Anonymous Class to Inner** Converts an anonymous class to an inner class that contains a name and constructor. The anonymous inner class is replaced with a call to the new inner class.
- **Extract Interface** Creates a new interface from the selected public non-static methods in a class or interface.
- **Extract Superclass** Creates a new abstract class, changes the current class to extend the new class, and moves the selected methods and fields to the new class.
- **Use Supertype Where Possible** Changes code that references the selected class (or other type) to instead use a supertype of that type.
- **Safely Delete** Checks for references to a code element and then automatically deletes that element if no other code references it.

Przykłady refaktoryzacji

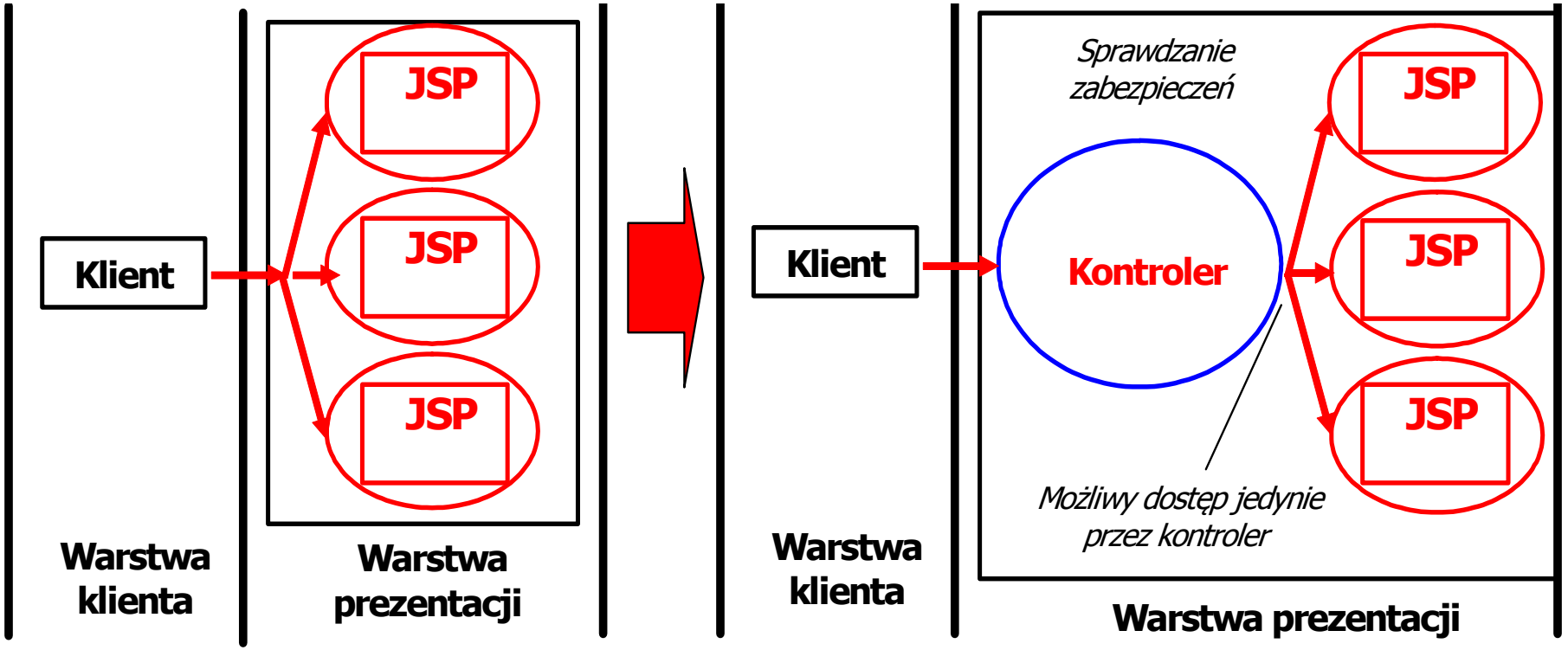
1. Przykłady refaktoryzacji architektury aplikacji
2. Przykład aplikacji typu Java Application z modelem obiektowym opartym na klasach zdefiniowanych przez użytkownika oraz klasach typu *Controller* technologii JPA
3. Przykład aplikacji typu Visual Web Java Server Faces
4. Przykłady metod zawierających równoważne algorytmy o różnych wartościach metryk MC Cabe

1. Refaktoryzacja architektury wielowarstwowej systemu informatycznego

(wg. D.Alur, J.Crupi, D. Malks, Core J2EE. Wzorce projektowe.)

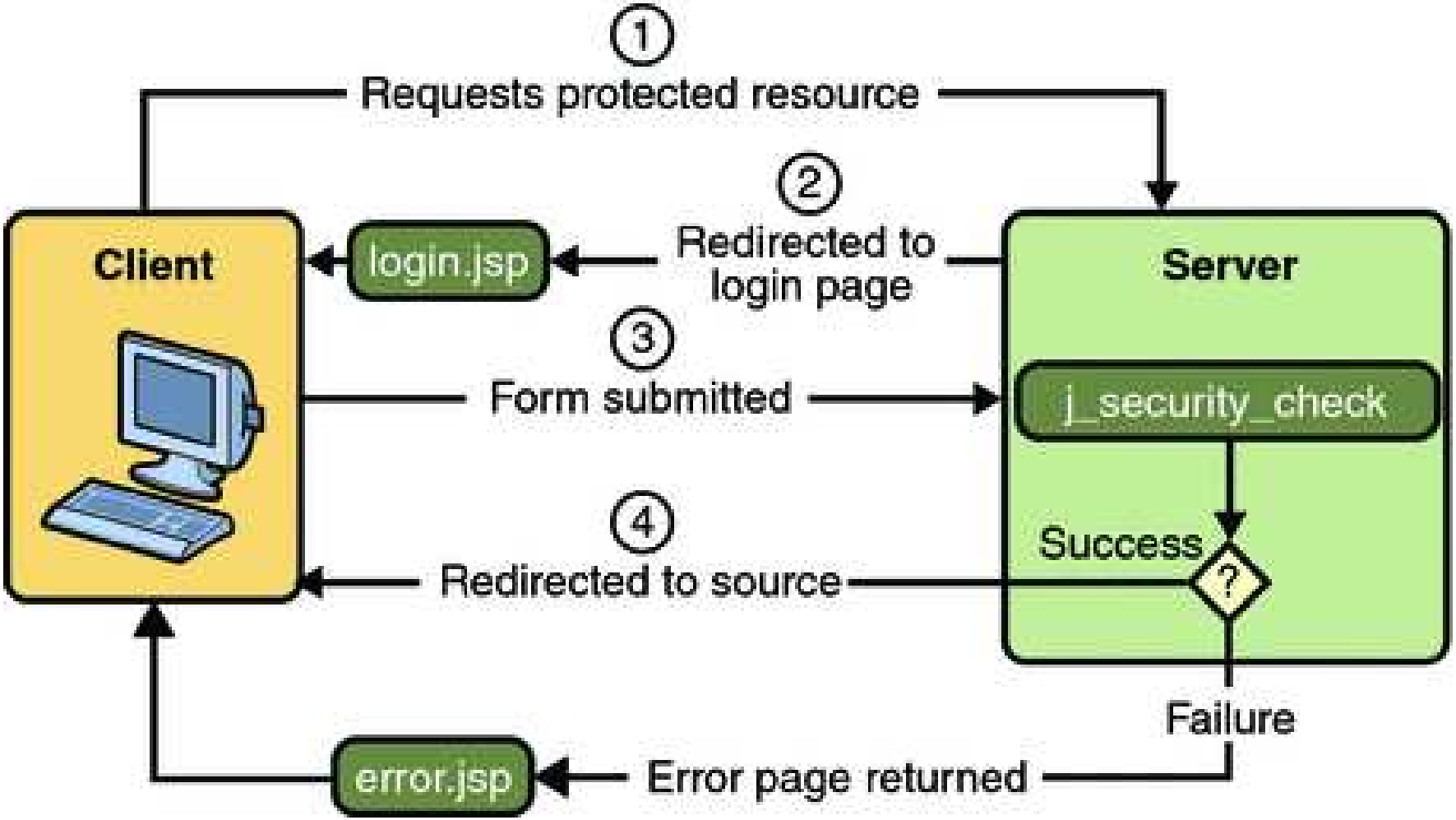
1.1. Refaktoryzacja warstwy prezentacji

Ukrywanie zasobów przed klientem za pomocą konfiguracji kontenera – uwierzytelnianie i autoryzacja



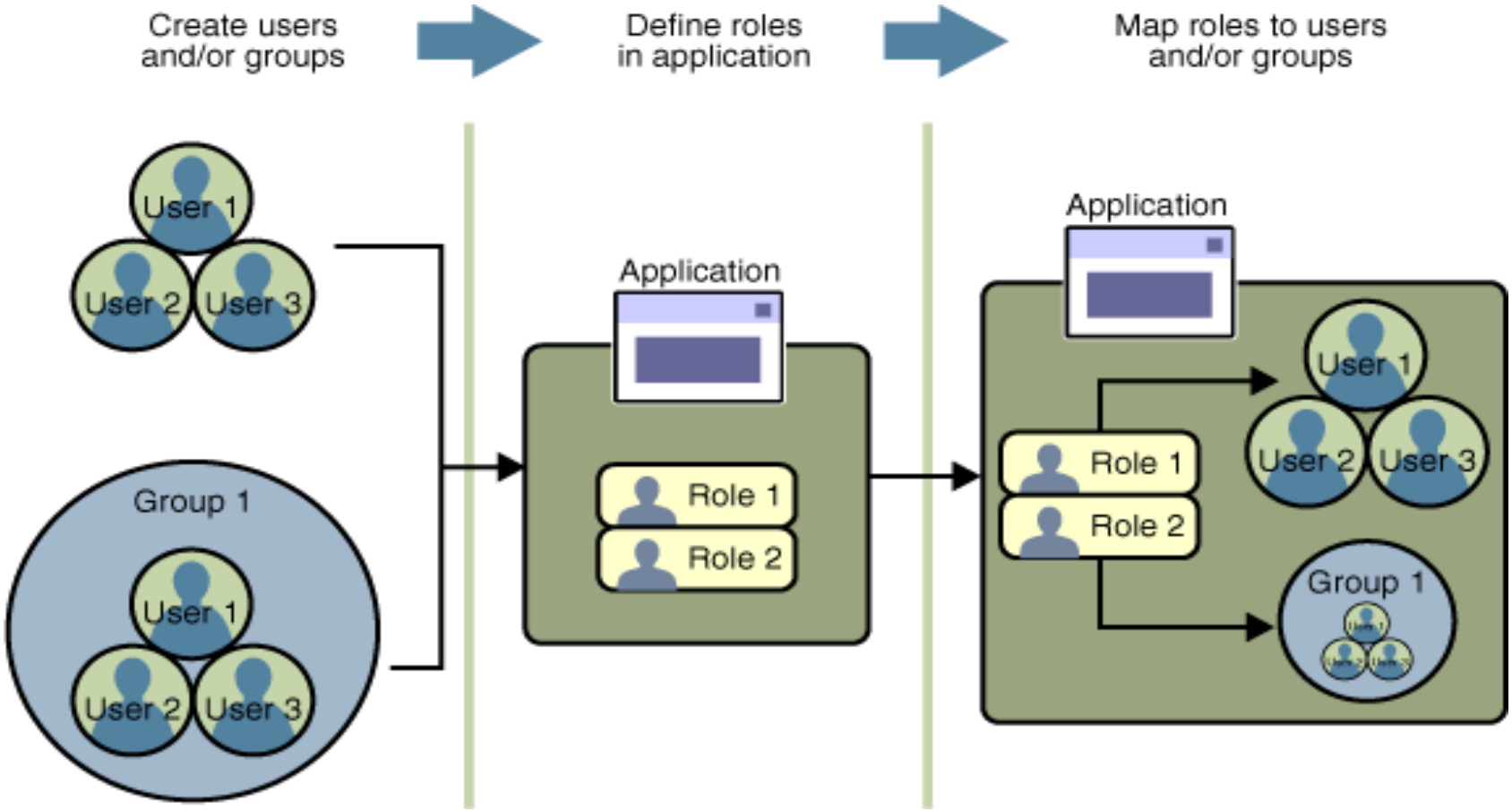
Przebieg uwierzytelniania (logowania)

http://download.oracle.com/javaee/5/tutorial/doc/bncbe.html



Bazy użytkowników i grup, Użytkownik, Grupa, Rola

http://download.oracle.com/javaee/5/tutorial/doc/bnbxj.html



Rodzaje mechanizmów bezpieczeństwa w kontenerach

- **Deklaratywne mechanizmy bezpieczeństwa** – deklarowane za pomocą tzw. *„deployment descriptors" (deskryptory aplikacji np.* ***web.xml*** dla aplikacji typu ***web*** *)*. Deskryptory jako zewnętrzny element aplikacji zawierają informację specyfikującą role bezpieczeństwa i wymagania dostępu są mapowane w role specyficzne dla środowiska oraz użytkowników i polisy bezpieczeństwa.

- ***Programowe mechanizmy bezpieczeństwa*** - są osadzone w aplikacji i służą do podejmowanie decyzji o bezpieczeństwie. Uzupełniają deklaratywne mechanizmy bezpieczeństwa – lepiej wyrażają model bezpieczeństwa aplikacji. API mechanizmów programowych:
  - metody interfejsu EJBContext
  - metody interfejsu HttpServletRequest. Metody te pozwalają na podejmowanie decyzji biznesowych opartych na rolach bezpieczeństwa nadawcy lub zdalnego odbiorcy

- **Adnotacje lub metadane** są używane do specyfikowania informacji wewnątrz pliku z kodem klasy. Kiedy aplikacja jest uruchamiana, informacja ta jest używana lub pokrywana przez deskryptor aplikacji.

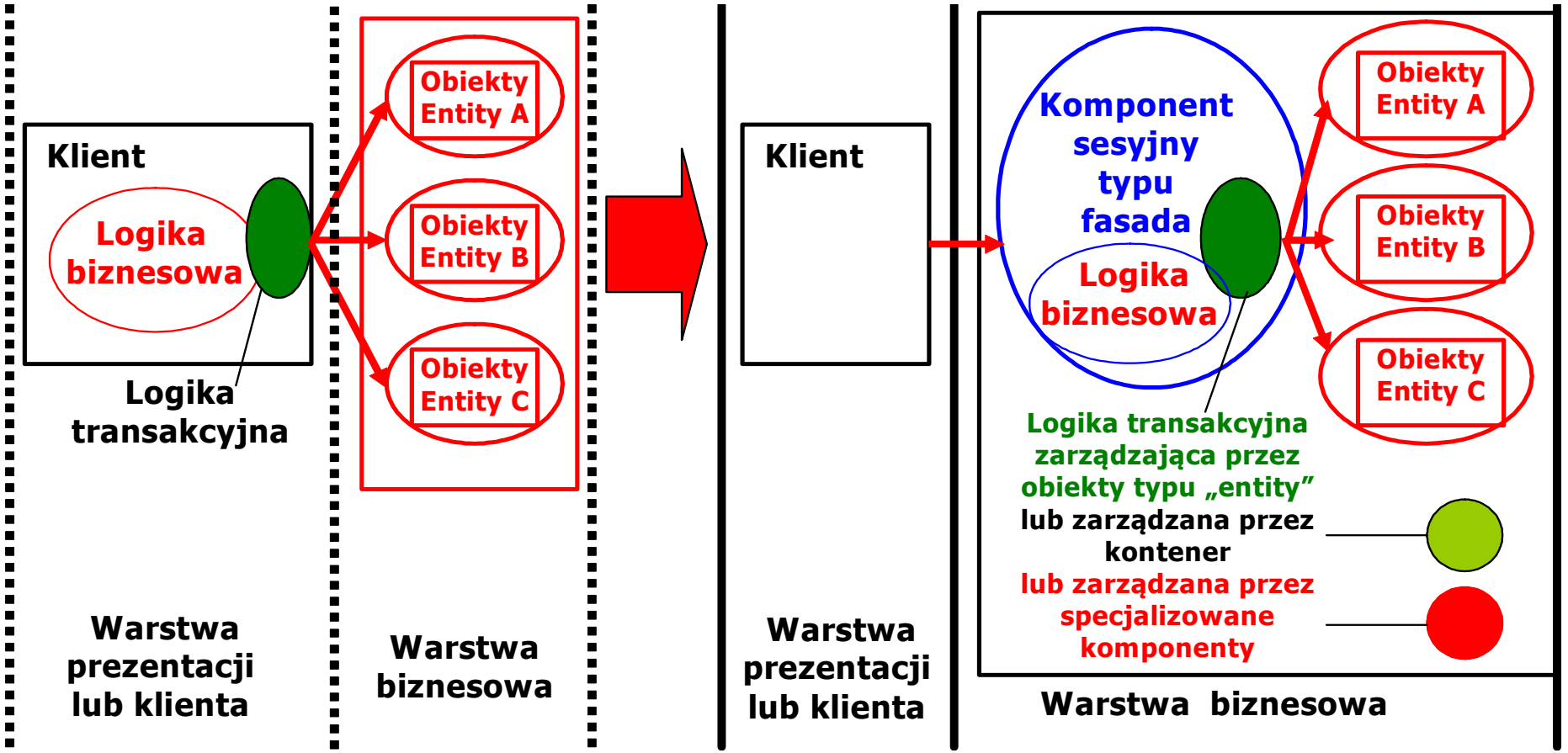
Np.

```
@DeclareRoles(„klient") public class Page1 extends AbstractPageBean
  {   //... }
```

1.2. Refaktoryzacja warstwy biznesowej

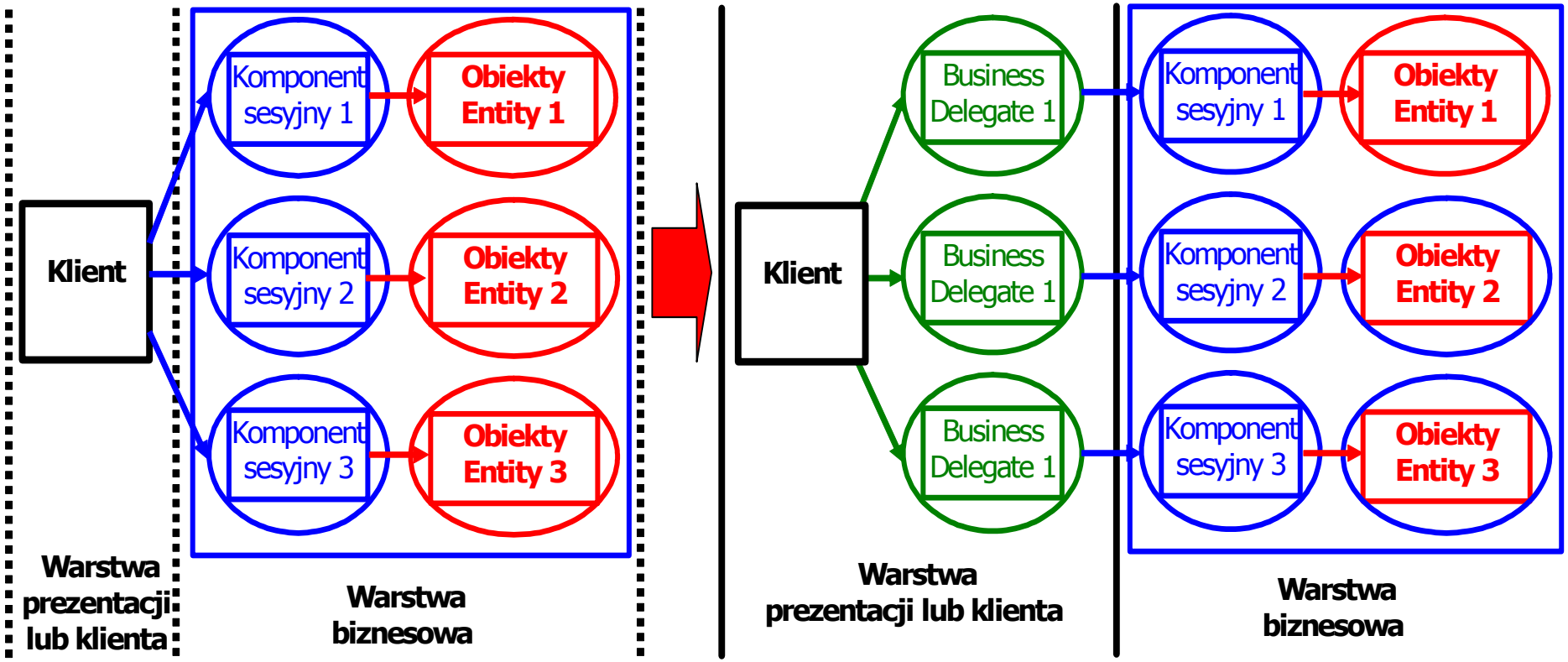
Refaktoryzacja warstwy biznesowej 1

Obiekty danych typu „Entity” (obiekty biznesowe) z warstwy biznesowej są udostępniane klientom w innych warstwach za pomocą **fasadowych komponentów sesyjnych typu „Control” (komponent typu fasada - hermetyzujący dostęp do usług biznesowych)**



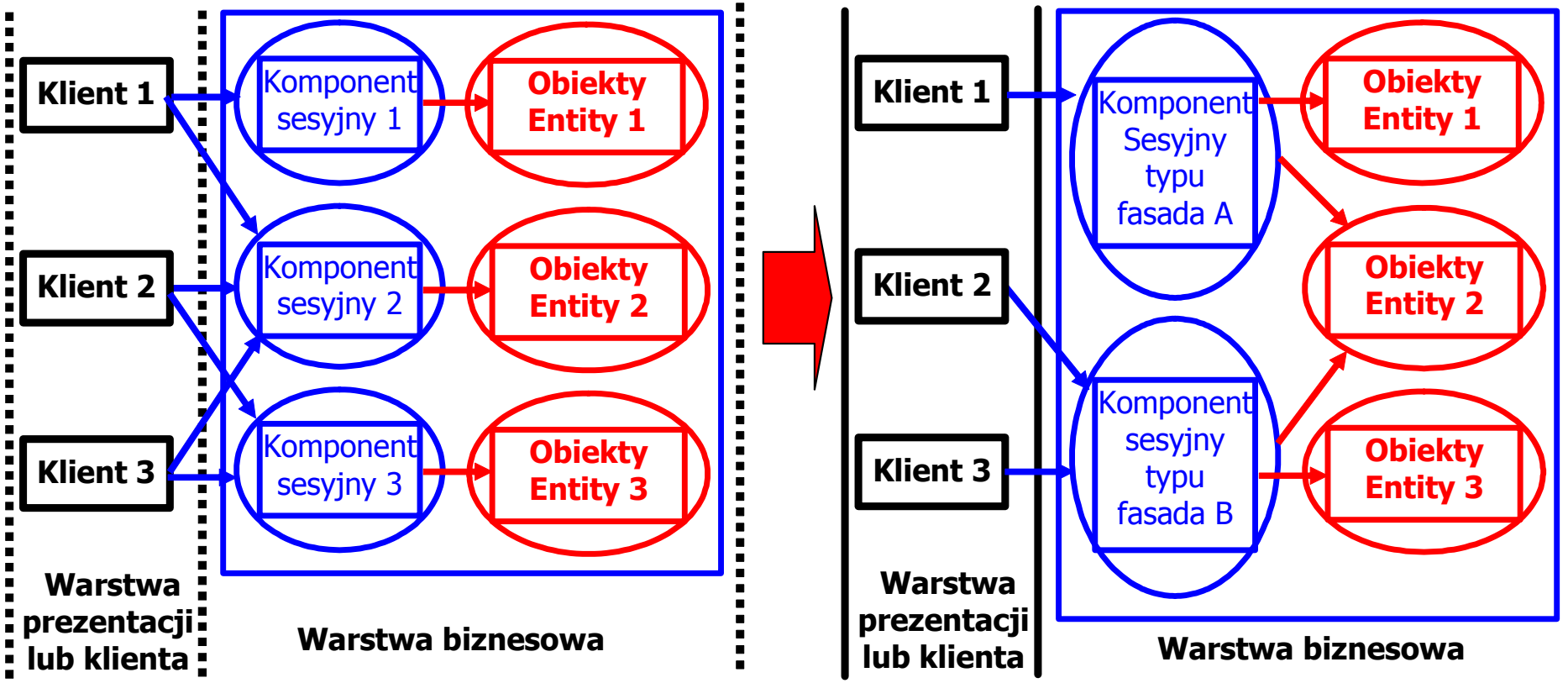
Refaktoryzacja warstwy biznesowej 2

Komponenty sesyjne typu „Control” (pośredniczące w dostępie do **obiektów danych typu „Entity”**) z warstwy biznesowej są udostępniane klientom w innych warstwach za pomocą **obiektów fasadowych typu „Control” (hermetyzujących dostęp do warstwy biznesowej- komponentów Business Delegate)**



Refaktoryzacja warstwy biznesowej 3

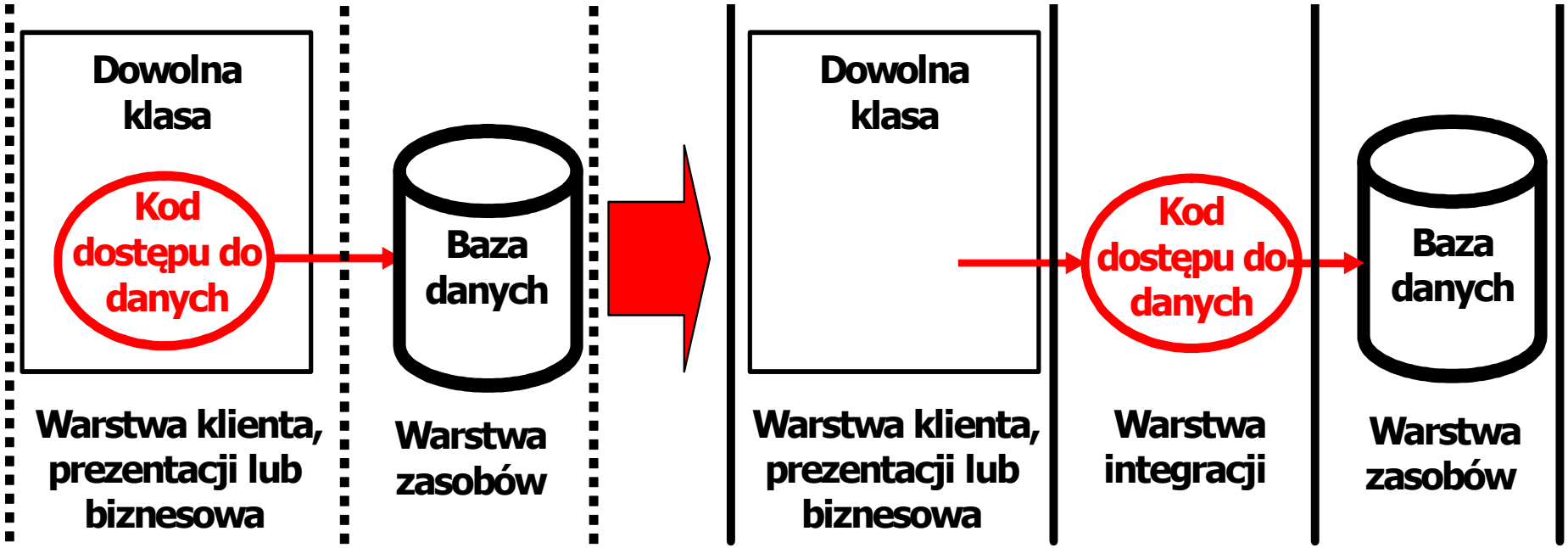
Sesyjne komponenty fasadowe typu „Control” (każdy komponent jako odrębna usługa biznesowa), hermetyzujące **obiekty danych typu „Entity”** z warstwy biznesowej są udostępniane klientom w innych warstwach. Zwykłe obiekty sesyjne są jedynie pośrednikami obiektów **„Entity”,** natomiast nie hermetyzują całych usług, które wymagają odwołania do wielu zwykłych komponentów sesyjnych.



1.3. Tworzenie warstwy integracji

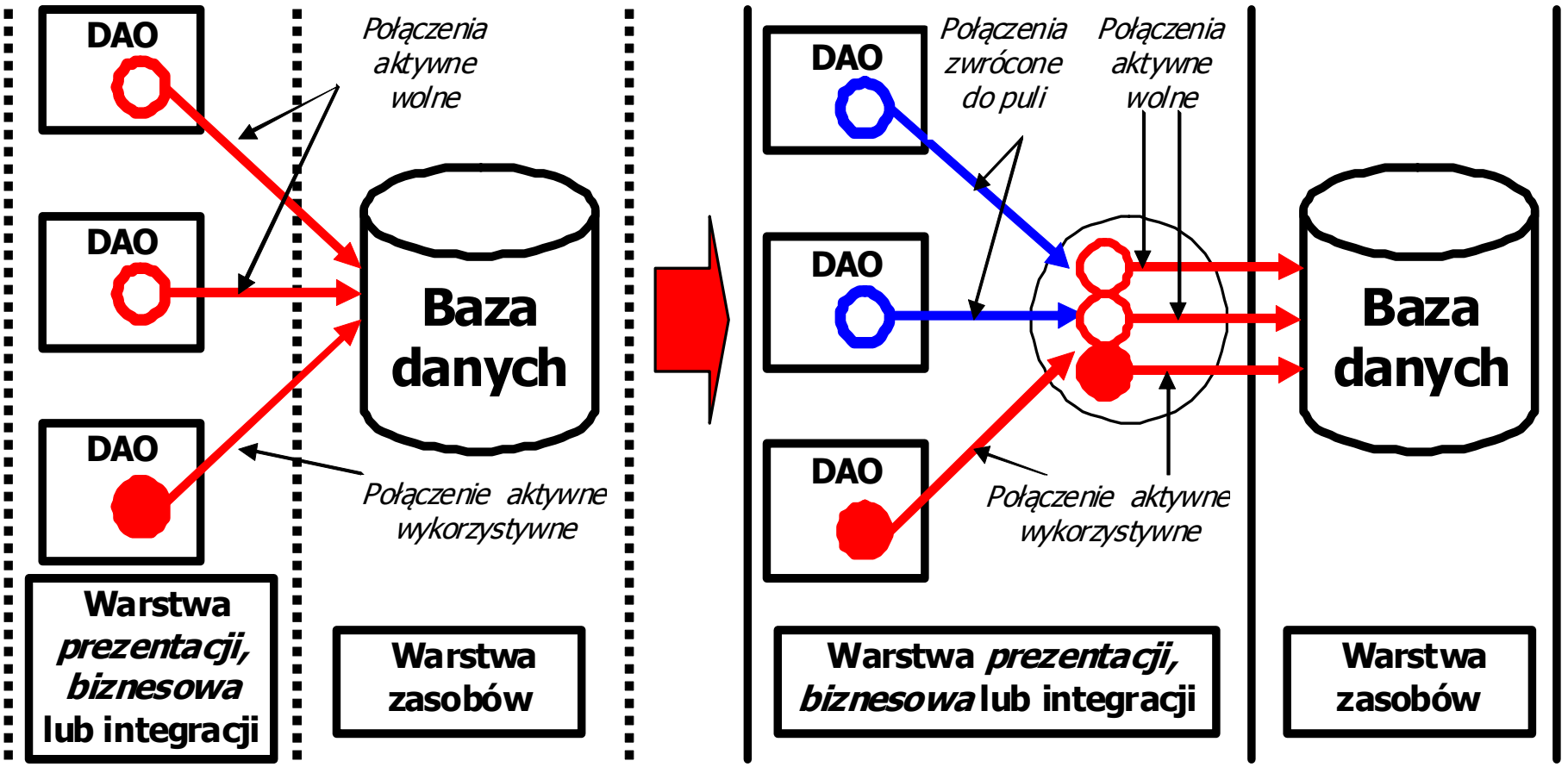
Wydzielanie kodu dostępu do danych

- Kod dostępu do danych jest wydzielany z klas, które są używane do spełniania również innych celów
- Kod dostępu do danych powinno umieszczać się logicznie i fizycznie bliżej źródła danych



Refaktoryzacja dostępu do danych – pula obiektów

- **Liczba połączeń kodu dostępu do danych (DAO) z bazą danych jest ograniczona**
- **Połączenia kodu dostępu do danych (DAO) nie zawsze są wykorzystywane, lecz są utrzymywane, ponieważ otwarcie połączenia z bazą danych zabiera i zasoby**
- **Pula połączeń kodu dostępu do danych (DAO) pozwala racjonalnie zarządzać połączeniami aplikacji z bazą danych**



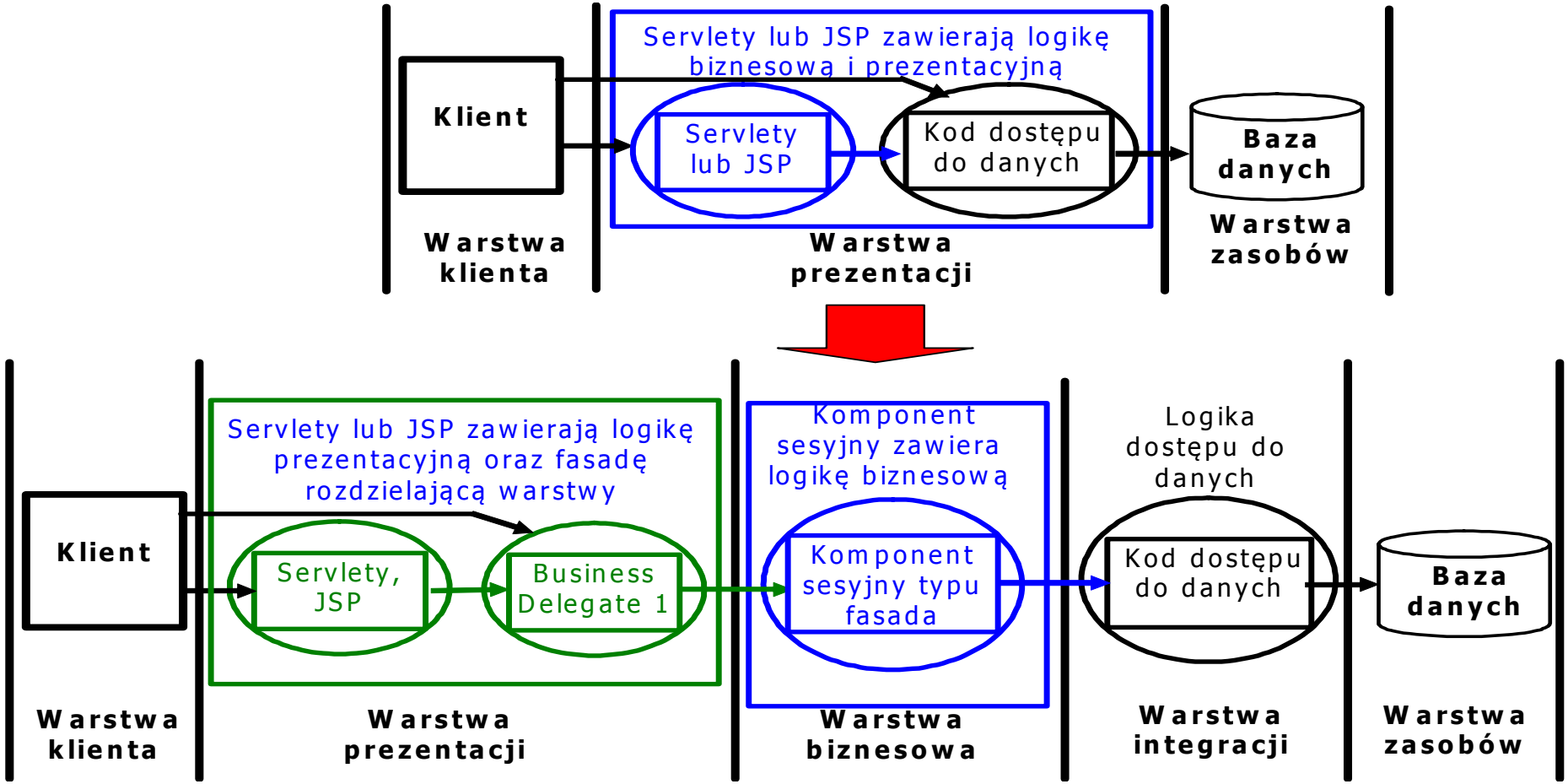
1.4. Podsumowanie refaktoryzacji warstw pięciowarstwowych systemów informatycznych

wg. D.Alur, J.Crupi, D. Malks, Core J2EE. Wzorce projektowe.)

Refaktoryzacja architektury wielowarstwowej 1

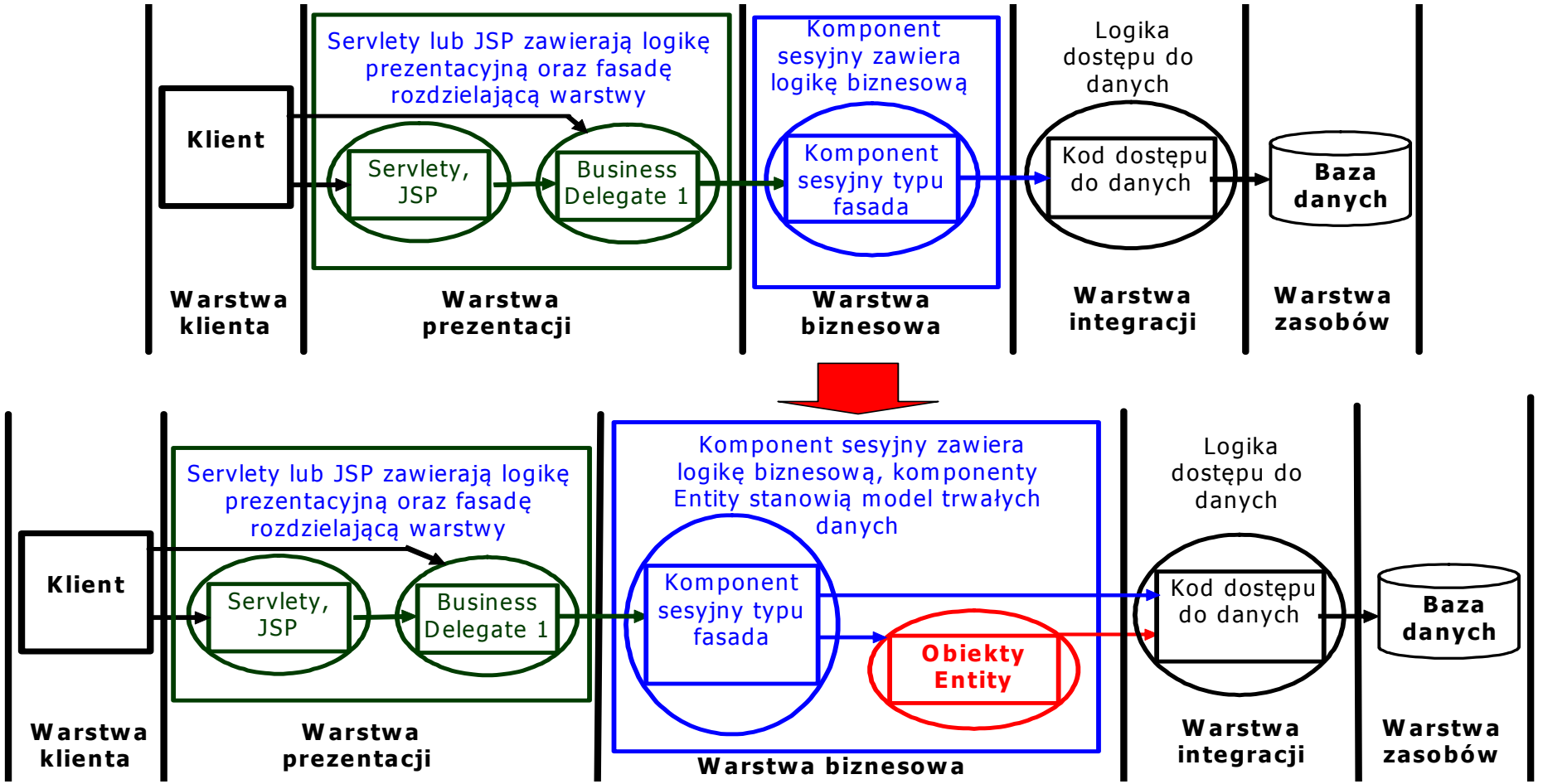
Należy przenieść kod dostępu do danych logicznie lub fizycznie bliżej rzeczywistego źródła danych, a logikę przetwarzania z klienta i warstwy prezentacji do warstwy biznesowej zawierającej **fasadowe komponenty sesyjne typu „Control”.**

Komponenty Business Delegate typu „Control” hermetyzują dostęp do warstwy biznesowej z warstwy prezentacji – stanowią przedłużenie warstwy biznesowej.

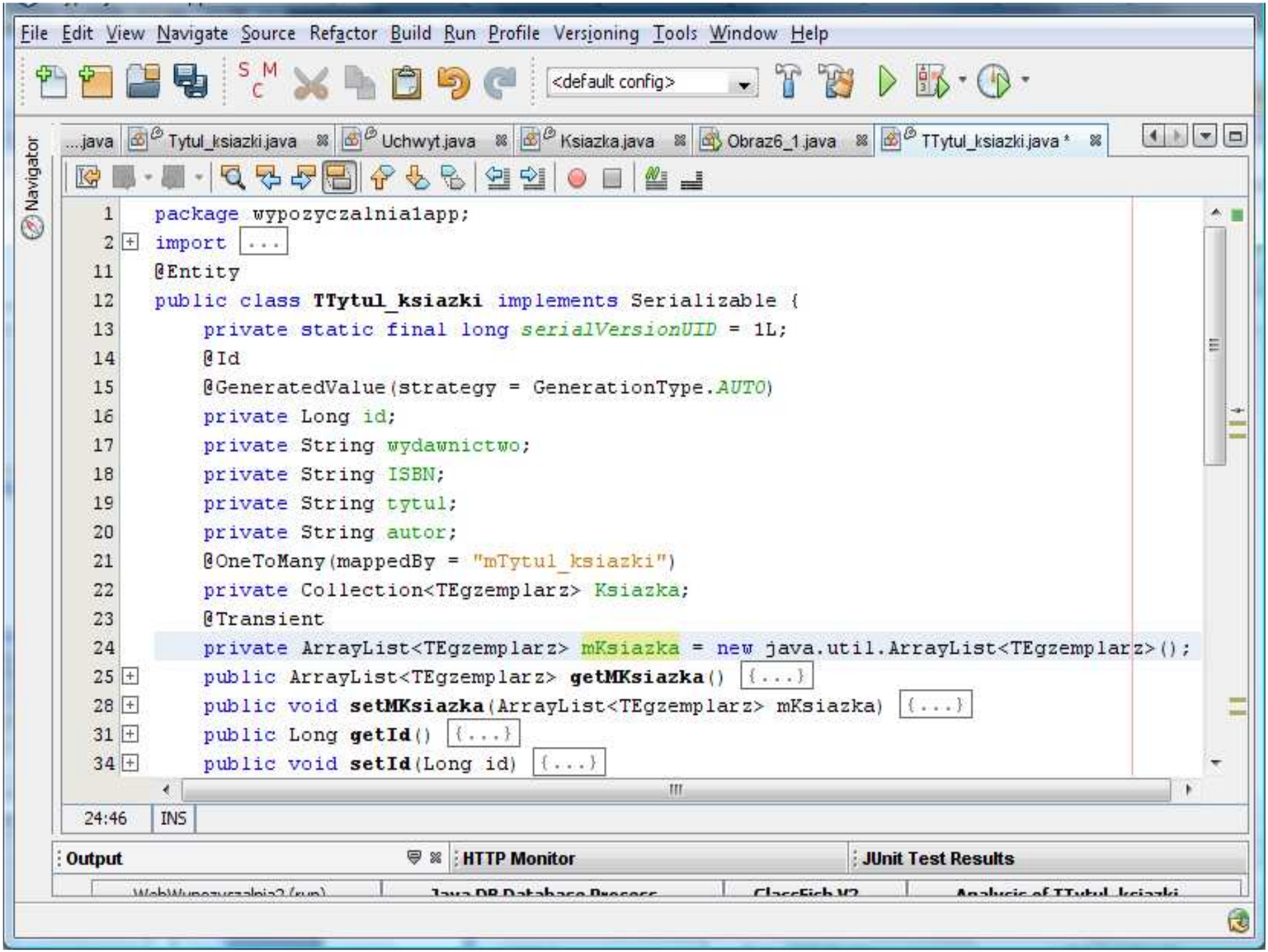


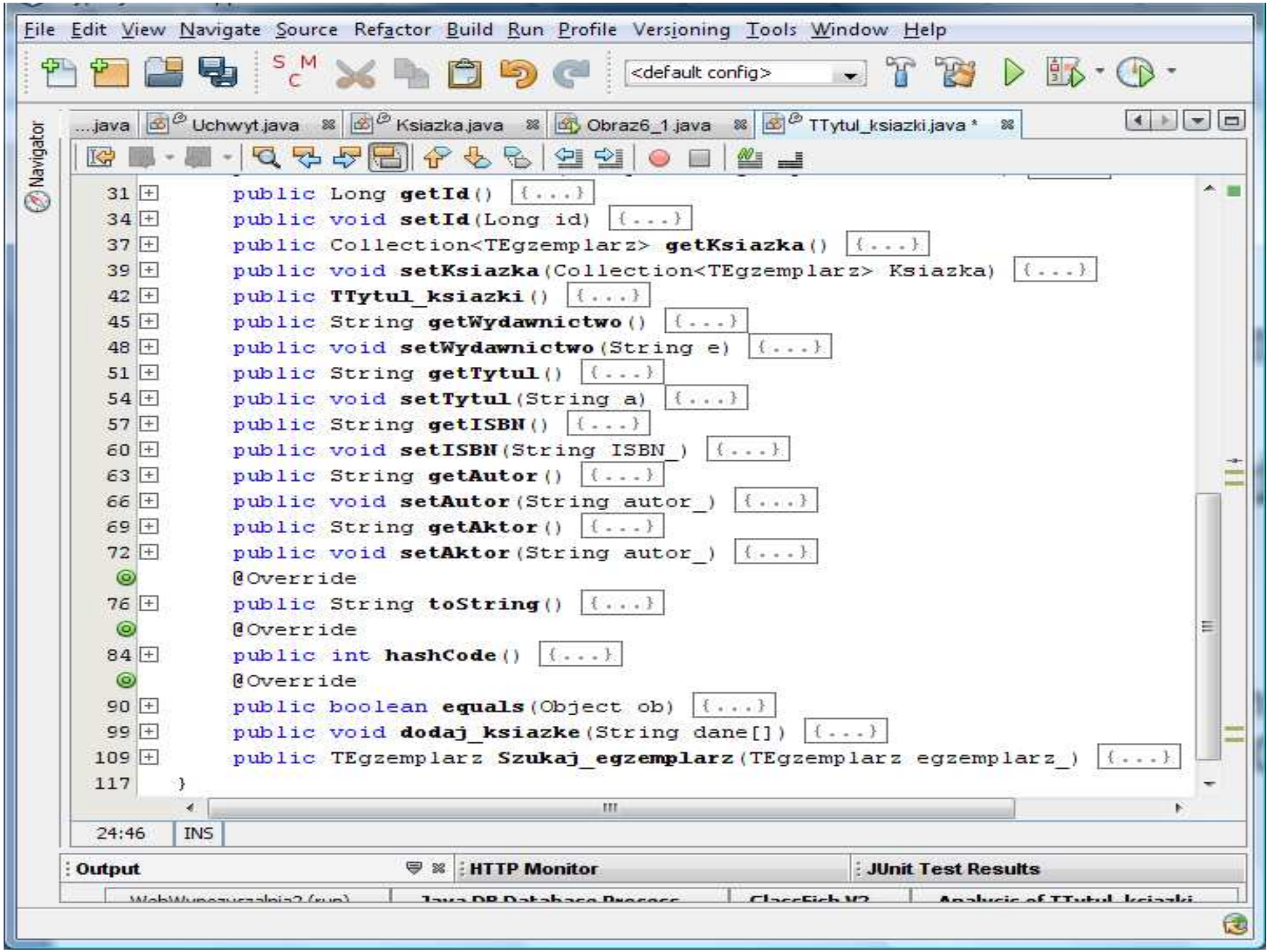
Refaktoryzacja architektury wielowarstwowej 2

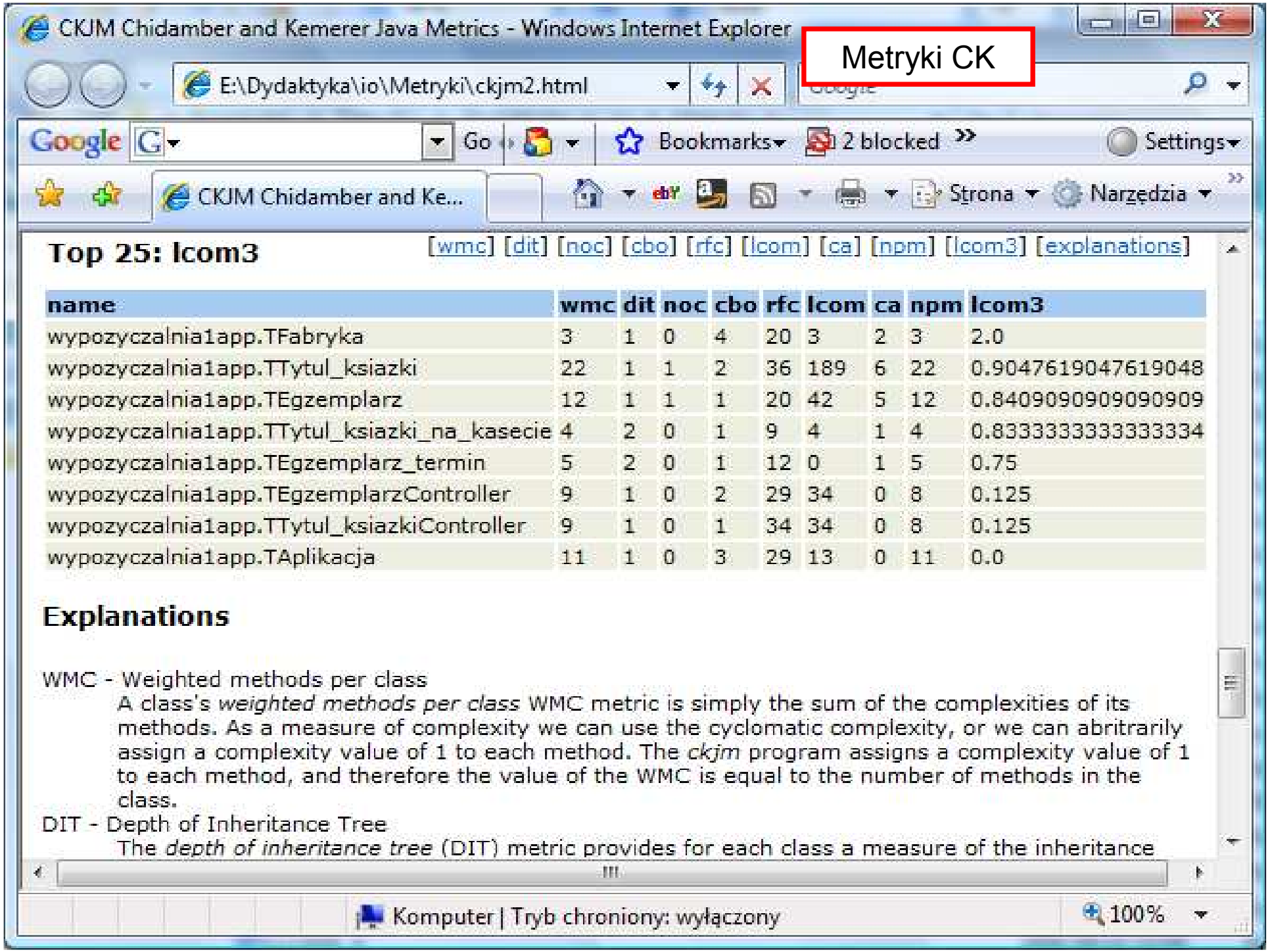
Należy przenieść kod dostępu do danych logicznie lub fizycznie bliżej rzeczywistego źródła danych, a złożoną logikę przetwarzania z klienta i warstwy prezentacji typu do warstwy biznesowej zawierającą **obiekty danych typu „Entity” i hermetyzujace dostep do tych komponentów fasadowe komponenty sesyjne typu „Control”.** Komponenty Business Delegate typu „Control” hermetyzują dostęp do warstwy biznesowej z warstwy prezentacji.



2. Przykład aplikacji typu Java Application z modelem obiektowym opartym na klasach zdefiniowanych przez użytkownika oraz klasach typu *Controller* technologii JPA


File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
<default config>
Navigator
...java
Tytul_ksiazki.java
Uchwyt.java
Ksiazka.java
Obraz6_1.java
TTytul_ksiazki.java *
1 package wypozyczalnia1app;
2 import ...
11 @Entity
12 public class TTytul_ksiazki implements Serializable {
13 private static final long serialVersionUID = 1L;
14 @Id
15 @GeneratedValue(strategy = GenerationType.AUTO)
16 private Long id;
17 private String wydawnictwo;
18 private String ISBN;
19 private String tytul;
20 private String autor;
21 @OneToMany(mappedBy = "mTytul_ksiazki")
22 private Collection<TEgzemplarz> Ksiazka;
23 @Transient
24 private ArrayList<TEgzemplarz> mKsiazka = new java.util.ArrayList<TEgzemplarz>();
25 public ArrayList<TEgzemplarz> getMKsiazka() {...}
28 public void setMKsiazka(ArrayList<TEgzemplarz> mKsiazka) {...}
31 public Long getId() {...}
34 public void setId(Long id) {...}
24:46
INS
Output
HTTP Monitor
JUnit Test Results


File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
<default config>
Navigator
...java
Uchwyt.java
Ksiazka.java
Obraz6_1.java
TTytul_ksiazki.java *
31 public Long getId() {...}
34 public void setId(Long id) {...}
37 public Collection<TEgzemplarz> getKsiazka() {...}
39 public void setKsiazka(Collection<TEgzemplarz> Ksiazka) {...}
42 public TTytul_ksiazki() {...}
45 public String getWydawnictwo() {...}
48 public void setWydawnictwo(String e) {...}
51 public String getTytul() {...}
54 public void setTytul(String a) {...}
57 public String getISBN() {...}
60 public void setISBN(String ISBN_) {...}
63 public String getAutor() {...}
66 public void setAutor(String autor_) {...}
69 public String getAktor() {...}
72 public void setAktor(String autor_) {...}
@Override
76 public String toString() {...}
@Override
84 public int hashCode() {...}
@Override
90 public boolean equals(Object ob) {...}
99 public void dodaj_ksiazke(String dane[]) {...}
109 public TEgzemplarz Szukaj_egzemplarz(TEgzemplarz egzemplarz_) {...}
117 }
24:46
INS
Output
HTTP Monitor
JUnit Test Results


CKJM Chidamber and Kemerer Java Metrics - Windows Internet Explorer
Metryki CK
E:\Dydaktyka\io\Metryki\ckjm2.html
Google
Go
Bookmarks
2 blocked
Settings
CKJM Chidamber and Ke...
Strona
Narzędzia
Top 25: lcom3
[wmc] [dit] [noc] [cbo] [rfc] [lcom] [ca] [npm] [lcom3] [explanations]
name wmc dit noc cbo rfc lcom ca npm lcom3
wypozyczalnia1app.TFabryka 3 1 0 4 20 3 2 3 2.0
wypozyczalnia1app.TTytul_ksiazki 22 1 1 2 36 189 6 22 0.9047619047619048
wypozyczalnia1app.TEgzemplarz 12 1 1 1 20 42 5 12 0.8409090909090909
wypozyczalnia1app.TTytul_ksiazki_na_kasecie 4 2 0 1 9 4 1 4 0.8333333333333334
wypozyczalnia1app.TEgzemplarz_termin 5 2 0 1 12 0 1 5 0.75
wypozyczalnia1app.TEgzemplarzController 9 1 0 2 29 34 0 8 0.125
wypozyczalnia1app.TTytul_ksiazkiController 9 1 0 1 34 34 0 8 0.125
wypozyczalnia1app.TAplikacja 11 1 0 3 29 13 0 11 0.0
Explanations
WMC - Weighted methods per class
A class's weighted methods per class WMC metric is simply the sum of the complexities of its methods. As a measure of complexity we can use the cyclomatic complexity, or we can abritrarily assign a complexity value of 1 to each method. The ckjm program assigns a complexity value of 1 to each method, and therefore the value of the WMC is equal to the number of methods in the class.
DIT - Depth of Inheritance Tree
The depth of inheritance tree (DIT) metric provides for each class a measure of the inheritance
Komputer | Tryb chroniony: wyłączony
100%

Poprawa spójności w klasie typu Entity po dodaniu metod związanych z przypadkami użycia: *usun ksiazke, zmien_ksiazke, zmien (zmienia dane tytulu),* oraz usunięciu zbędnego atrybutu typu *Transient* wraz z metodami typu set i get. Teraz dostęp do kolekcji książek w klasie TTytul_ksiazki jest możliwy za pomocą getKsiazka (wcześnie getMKsiazka)

```
@Entity
public class TTytul_ksiazki implements Serializable {

    private static final long serialVersionUID = 1L;
    private String wydawnictwo;
    private String ISBN;
    private String tytul;
    private String autor;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    public Long getId() {...}
    public void setId(Long id) {...}

    @OneToMany(mappedBy = "mTytul_ksiazki")
    private Collection<TEgzemplarz> Ksiazka;
    public Collection<TEgzemplarz> getKsiazka() {...}
    public void setKsiazka(Collection<TEgzemplarz> Ksiazka) {...}
```

```
public TTytul_ksiazki() {    //id = new Long(-1);
    id = null;
}

public TTytul_ksiazki(int a) {    //id = new Long(-1);
    Ksiazka = new ArrayList<TEgzemplarz>();
    id = null;
}
public String getWydawnictwo() {...}
public void setWydawnictwo(String e) {...}
public String getTytul() {...}
public void setTytul(String a) {...}
public String getISBN() {...}
public void setISBN(String ISBN_) {...}
public String getAutor() {...}
public void setAutor(String autor_) {...}
public String getAktor() {...}
public void setAktor(String autor_) {...}
@Override
public String toString() {...}
@Override
public int hashCode() {...}
@Override
public boolean equals(Object ob) {...}
public void dodaj_ksiazke(String dane[]) {...}
```

Kostruktor z parametrami pozwala implementować relację 1 do wiele między klasą TTytul_ksiazki i klasą TEgzemplarz, która jest używana w wartswie biznesowej aplikacji. Konstruktor bezparamatrowy jest potrzebny w warstwie integracji opartej na technologii ORM (np TopLink)

```
public TEgzemplarz Szukaj_egzemplarz(TEgzemplarz egzemplarz_) {
    int idx;
    ArrayList<TEgzemplarz> pom = (ArrayList<TEgzemplarz>) Ksiazka;
    if ((idx = pom.indexOf(egzemplarz_)) != -1) {
        egzemplarz_ = pom.get(idx);
        return egzemplarz_;
    }
    return null;
}

public void zmien(String dane[]) {
    wydawnictwo = dane[3];
    ISBN = dane[2];
    tytul = dane[0];
    autor = dane[1];
    setAktor(dane[4]);
}
```

Należy rzutować typ Collection<TEgzemplarz> kolekcji Ksiazka na aktualny typ ArrayList<TEgzemplarz> w celu użycia metody indexOf, należącej do API intefejsu List, dziedziczącego po interfejsie Collection implementowanego przez klasę ArrayList.

```
    public void usun_ksiazke(String dane[]) {
        TFabryka fabryka = new TFabryka();
        TEgzemplarz nowa;
        nowa = fabryka.Podaj_egzemplarz(dane);
        Ksiazka.remove(nowa);
    }

    public void zmien_ksiazke(String dane1[], String dane2[]) {
        TFabryka fabryka = new TFabryka();
        TEgzemplarz nowa;
        nowa = fabryka.Podaj_egzemplarz(dane1);
        if ((nowa = Szukaj_egzemplarz(nowa)) != null) {
            nowa.setNumer(Integer.parseInt(dane2[0]));
            nowa.setTermin(new Date(Long.parseLong(dane2[1])));
        }
    }
}
```

```
@Entity
public class TTytul_ksiazki_na_kasecie extends TTytul_ksiazki
                                        implements Serializable {

    private static final long serialVersionUID = 1L;
    private String aktor;

    public TTytul_ksiazki_na_kasecie(int a)
    {
        super(a);
    }
    public TTytul_ksiazki_na_kasecie()
    {
    }
    @Override
    public String getAktor() {...}
    @Override
    public void setAktor(String aktor) {...}
    @Override
    public String toString() {...}
}
```

Kostruktor z parametrami w klasie TTytul_ksiazki wymaga zdefiniowania konstruktorów: bez parametrów i z parametrami w klasie dziedziczącej TTytul_ksiazki_na_kasecie, aby spełnić ten sam cel klasy bazowej.

Przykłady zmian w programie, korzystającym z klasy Tytul_ksiazki po jej refaktoryzacji

- W warstwie prezentacji oraz w klasach warstwy integracji należy zmienić metodę getMKsiazka() na getKsiazka()
- Wynik metody getKsiazka jest typu Collection np

```
public void przygotujksiazki(TTytul_ksiazki tytul) {
  if (tytul == null) {
     return; }
  Collection<TEgzemplarz> ksiazki = tytul.getKsiazka();
  int ile = ksiazki.size();
  if (ile > 0) {
    Option pom[] = new Option[ile];
    Iterator iterator = ksiazki.iterator();
    int i = 0;
    while (iterator.hasNext()) {
     pom[i++] =
      new Option(Integer.toString(i),iterator.next().toString());}
    ksiazki_ = pom;
  }
}
```

```
public class TFabryka {

    public TTytul_ksiazki Podaj_tytul(String dane[])
     { TTytul_ksiazki tytul=null;
       switch(Integer.parseInt(dane[0]))
       {
           case 0: tytul= new  TTytul_ksiazki(1);
                   tytul.setISBN(dane[1]);
                   break;
           case 1: tytul= new  TTytul_ksiazki(1);
                   tytul.setAutor(dane[1]);
                   tytul.setTytul(dane[2]);
                   tytul.setISBN(dane[3]);
                   tytul.setWydawnictwo(dane[4]);
                   break;
           case 2: tytul= new TTytul_ksiazki_na_kasecie(1);
                   tytul.setISBN(dane[1]);
                   ((TTytul_ksiazki_na_kasecie)tytul).setAktor(dane[2]);
                   break;
           case 3: tytul= new TTytul_ksiazki_na_kasecie(1);
                   tytul.setAutor(dane[1]);
                   tytul.setTytul(dane[2]);
                   tytul.setISBN(dane[3]);
                   tytul.setWydawnictwo(dane[4]);
                   ((TTytul_ksiazki_na_kasecie)tytul).setAktor(dane[5]);
                   break;
        }
       return tytul;
     }
```

Zmiana tworzenia obiektów typu TTytul_ksiazki i TTytul_ksiazki_na _kasecie

Metryki CK - poprawa metryk LCOM w klasie TTytul_ksiazki

Top 25: lcom3

[wmc] [dit] [noc] [cbo] [rfc] [lcom] [ca] [npm] [lcom3] [explanations]

name	wmc	dit	noc	cbo	rfc	lcom	ca	npm	lcom3
wypozyczalnia1app.TFabryka	3	1	0	4	19	3	2	3	2.0
wypozyczalnia1app.TTytul_ksiazki_na_kasecie	5	2	0	1	11	8	1	5	0.875
wypozyczalnia1app.TTytul_ksiazki	24	1	1	2	43	202	6	24	0.8633540372670808
wypozyczalnia1app.TEgzemplarz	12	1	1	1	19	42	5	12	0.8409090909090909
wypozyczalnia1app.TEgzemplarz_termin	5	2	0	1	12	0	1	5	0.75
wypozyczalnia1app.TEgzemplarzController	9	1	0	2	31	34	0	8	0.125
wypozyczalnia1app.TTytul_ksiazkiController	9	1	0	1	34	34	0	8	0.125
wypozyczalnia1app.TAplikacja	11	1	0	3	31	13	0	11	0.0

3. Przykład aplikacji typu Visual Web Java Server Faces – klasy przed refaktoryzacją

```
package webwypozyczalnia2;
import ...
/**...*/
public class ApplicationBean1 extends AbstractApplicationBean {
    Managed Component Definition
    /**...*/
    public ApplicationBean1() {...}
    /**...*/
  @Override
  public void init() {...}
  private TAplikacja aplikacja = new TAplikacja();
  public TAplikacja getAplikacja() {...}
  public void setAplikacja(TAplikacja aplikacja) {...}
  public void updateAplikacja() {...}
  public void  dodaj_tytul(String dane[]) {...}//przypadek użycia '
  public void  dodaj_ksiazke(String dane1[], String dane2[]) {...}

  private Option tytuly_[] = new Option[0];
  public Option[] getTytuly_() {...}
  public void setTytuly_(Option[] tytuly_) {...}
  public void przygotujtytuly() {...}

  private Option ksiazki_[] = new Option[0];
  public Option[] getKsiazki_() {...}
  public void setKsiazki_(Option[] ksiazki_) {...}
  public void przygotujksiazki(TTytul_ksiazki tytul) {...}
```

```
    private TTytul_ksiazki tytuly[];
    public TTytul_ksiazki[] getTytuly() {...}
    public void setTytuly(TTytul_ksiazki[] tytuly) {...}
    public void updateTytuls() {...}

    private TEgzemplarz ksiazki[];
    public TEgzemplarz[] getKsiazki() {...}
    public void setKsiazki(TEgzemplarz[] ksiazki) {...}
    public void updateKsiazkis() {...}

    public void zapisz_tytul_do_bazy(String dane[]) {...}
    public void zapisz_tytuly_do_bazy() {...}
    public void zapisz_ksiazki_do_bazy() {...}
    /**...*/
    @Override
    public void destroy() {...}
    @Override
    public String getLocaleCharacterEncoding() {...}
}
```

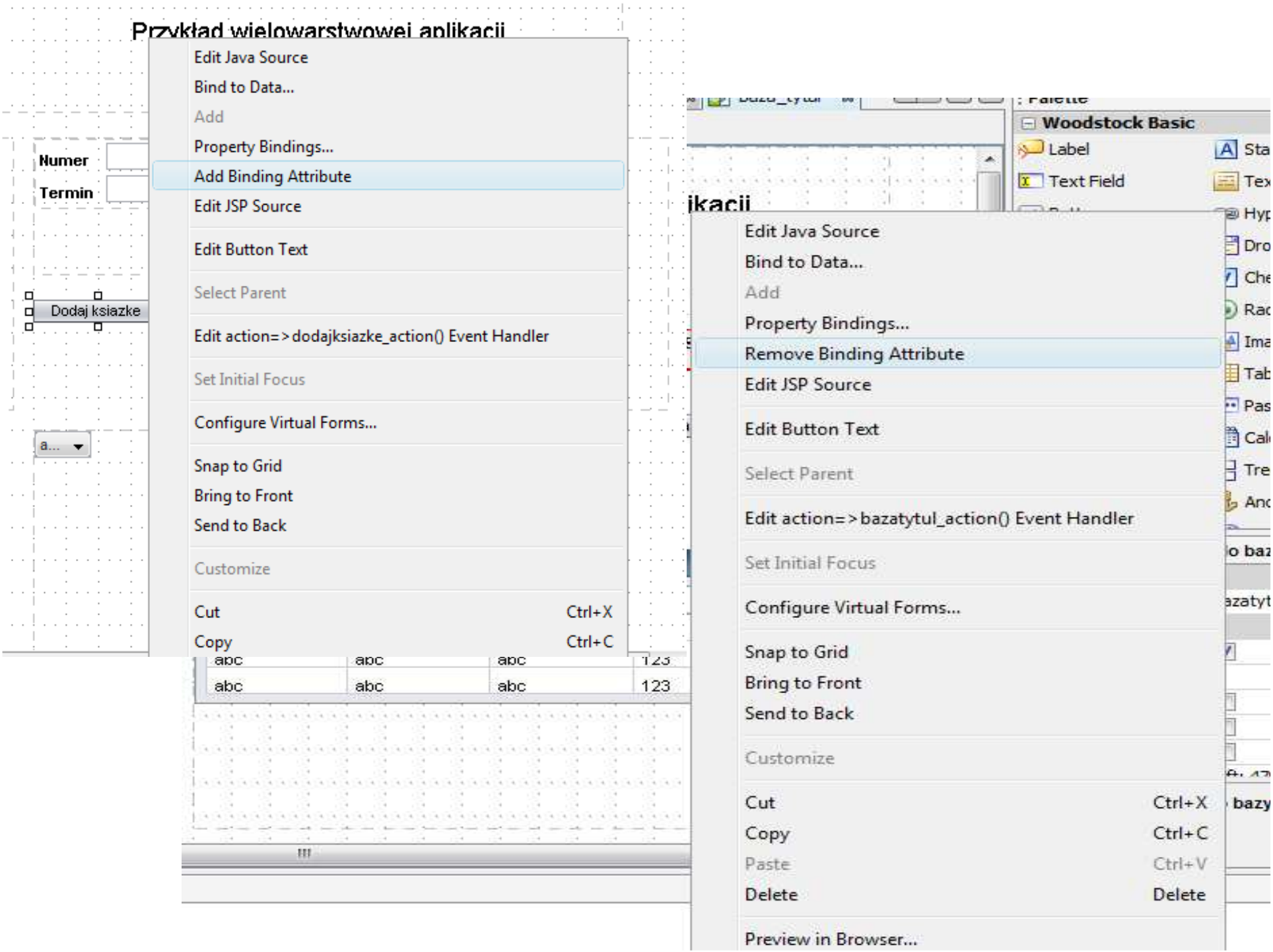
Metryki CK klasy ApplicationBean1 przed refaktoryzacją

Top 25: lcom3

[wmc] [dit] [noc] [cbo] [rfc] [lcom] [ca] [npm] [lcom3] [explanations]

name	wmc	dit	noc	cbo	rfc	lcom	ca	npm	lcom3
webwypozyczalnia2.RequestBean1	6	3	0	3	11	15	14	3	2.0
webwypozyczalnia2.Tytuly	26	3	0	15	46	277	0	22	0.92
webwypozyczalnia2.Menu	26	3	0	5	32	283	6	22	0.92
webwypozyczalnia2.Baza_tytul	26	3	0	15	45	277	0	22	0.92
webwypozyczalnia2.Ksiazkibaza	25	3	0	9	35	246	0	21	0.9166666666666666
webwypozyczalnia2.Baza_ksiazki	24	3	0	13	40	234	0	20	0.9130434782608695
webwypozyczalnia2.Baza_tytuly	24	3	0	13	40	234	0	20	0.9130434782608695
webwypozyczalnia2.Ksiazki	24	3	0	18	49	234	0	20	0.9130434782608695
webwypozyczalnia2.Page1	21	3	0	12	34	174	0	17	0.9
webwypozyczalnia2.FormTytul	31	3	0	6	43	349	2	27	0.9
webwypozyczalnia2.FormKsiazka	19	3	0	6	31	115	1	15	0.8333333333333334
webwypozyczalnia2.Tytulybaza	13	3	0	7	21	60	0	9	0.8333333333333334
webwypozyczalnia2.Logo	11	3	0	6	18	43	0	7	0.8
webwypozyczalnia2.ApplicationBean1	25	3	0	7	54	234	16	24	0.775
webwypozyczalnia2.SessionBean1	9	3	0	2	14	30	15	7	0.75
webwypozyczalnia2.Ksiazkiaplikacja	9	3	0	5	15	30	0	5	0.75
webwypozyczalnia2.Tytulyaplikacja	11	3	0	8	25	43	1	7	0.6

4. Przykład aplikacji typu Visual Web Java Server Faces – klasy typu PageBean po usunięciu niepotrzebnych atrybutów oraz metod typu set i get reprezentujących elementy widoków w klasach typu PageBean każdego z formularzy jsp wprowadzonych poleceniem **Add Binding Attribute**. Należy usunąć te składowe poleceniem **Remove Binding Attribute**.


Przykład wielowarstwowej aplikacji
Numer
Termin
Dodaj ksiazke
a...
Edit Java Source
Bind to Data...
Add
Property Bindings...
Add Binding Attribute
Edit JSP Source
Edit Button Text
Select Parent
Edit action=>dodajksiazke_action() Event Handler
Set Initial Focus
Configure Virtual Forms...
Snap to Grid
Bring to Front
Send to Back
Customize
Cut Ctrl+X
Copy Ctrl+C
abc abc abc 123
abc abc abc 123
Woodstock Basic
Label
Text Field
Edit Java Source
Bind to Data...
Add
Property Bindings...
Remove Binding Attribute
Edit JSP Source
Edit Button Text
Select Parent
Edit action=>bazatytul_action() Event Handler
Set Initial Focus
Configure Virtual Forms...
Snap to Grid
Bring to Front
Send to Back
Customize
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Delete Delete
Preview in Browser...

```
public class Ksiazki extends AbstractPageBean {
    // <editor-fold defaultstate="collapsed" desc="Managed Compc

    /**...*/
    private void _init() throws Exception {...}

    private Page page1 = new Page();
    public Page getPage1() {...}
    public void setPage1(Page p) {...}
    private Html html1 = new Html();
    public Html getHtml1() {...}
    public void setHtml1(Html h) {...}
    private Head head1 = new Head();
    public Head getHead1() {...}
    public void setHead1(Head h) {...}
    private Link link1 = new Link();
    public Link getLink1() {...}
    public void setLink1(Link l) {...}
    private Body body1 = new Body();
    public Body getBody1() {...}
    public void setBody1(Body b) {...}
    private Form form1 = new Form();
    public Form getForm1() {...}
    public void setForm1(Form f) {...}
    private Button dodajksiazke1 = new Button();
    public Button getDodajksiazke1() {...}
    public void setDodajksiazke1(Button b) {...}
```

Przykład niepotrzebnych składowych w klasie Ksiazki typu PageBean reprezentujących znaczniki pliku jsp (JSP Tags)

```
public class Ksiazki extends AbstractPageBean {
    // <editor-fold defaultstate="collapsed" desc="Managed Component Definit

    /**
     * <p>Automatically managed component initialization.  <strong>WARNING:<
     * This method is automatically generated, so any user-specified code in
     * here is subject to being replaced.</p>
     */
    private void _init() throws Exception {
    }

    // </editor-fold>

    /**
     * <p>Construct a new Page bean instance.</p>
     */
    public Ksiazki() {
    }
```

Klasa Ksiazka typu PageBean pozbawiona niepotrzebnych składowych reprezentujących znaczniki pliku jsp (JSP Tags)

Metryki CK klas PageBean po refaktoryzacji (usunięcie niepotrzebnych składowych)

Top 25: lcom3

[wmc] [dit] [noc] [cbo] [rfc] [lcom] [ca] [npm] [lcom3] [explanations]

name	wmc	dit	noc	cbo	rfc	lcom	ca	npm	lcom3
webwypozyczalnia2.RequestBean1	6	3	0	3	11	15	14	3	2.0
webwypozyczalnia2.Tytuly	10	3	0	7	22	45	0	6	2.0
webwypozyczalnia2.Baza_ksiazki	10	3	0	6	19	45	0	6	2.0
webwypozyczalnia2.Baza_tytuly	10	3	0	6	19	45	0	6	2.0
webwypozyczalnia2.Ksiazkibaza	7	3	0	4	12	21	0	3	2.0
webwypozyczalnia2.Page1	9	3	0	6	16	36	0	5	2.0
webwypozyczalnia2.Ksiazki	10	3	0	11	28	45	0	6	2.0
webwypozyczalnia2.Ksiazkiaplikacja	7	3	0	4	12	21	0	3	2.0
webwypozyczalnia2.Logo	7	3	0	4	12	21	0	3	2.0
webwypozyczalnia2.Baza_tytul	10	3	0	7	21	45	0	6	2.0
webwypozyczalnia2.Tytulybaza	7	3	0	4	12	21	0	3	2.0
webwypozyczalnia2.Menu	26	3	0	5	32	283	6	22	0.92
webwypozyczalnia2.FormTytul	22	3	0	5	33	145	2	18	0.8174603174603176
webwypozyczalnia2.ApplicationBean1	25	3	0	7	54	234	16	24	0.775
webwypozyczalnia2.SessionBean1	9	3	0	2	14	30	15	7	0.75
webwypozyczalnia2.FormKsiazka	15	3	0	5	26	61	1	11	0.738095238095238
webwypozyczalnia2.Tytulyaplikacja	11	3	0	8	25	43	1	7	0.6

5. Przykład aplikacji typu Visual Web Java Server Faces – klasy typu PageBean po usunięciu nadmiarowych wywołań typu getBean

```
public class Baza_tytuly extends AbstractPageBean {
    // <editor-fold defaultstate="collapsed" desc="Managed Component
    private ApplicationBean1 a;
    private Menu menudiv;
    /**...*/
    private void _init() throws Exception {...}
    // </editor-fold>
    /**...*/
    public Baza_tytuly() {...}
    /**...*/
    @Override
    public void init() { // Perform initializations inherited from o
        super.init(); // Perform application initialization that mus
        // *before* managed components are initialized
        // TODO - add your own initialiation code here
        Managed Component Initialization
        // Perform application initialization that must complete
        // *after* managed components are initialized
        // TODO - add your own initialization code here
        a=getApplicationBean1();
        menudiv=(Menu)getBean("Menu");
    }
    /**...*/
    @Override
    public void preprocess() {...}
    /**...*/
    @Override
    public void prerender() {
        a.updateTytuls();
        Hyperlink link4 = menudiv.getHyperlink4();
        link4.setDisabled(true);
    }
```

Metryki CK klas PageBean po wprowadzeniu buforowania wyników wywołań typu getBean

Top 25: lcom3

[wmc] [dit] [noc] [cbo] [rfc] [lcom] [ca] [npm] [lcom3] [explanations]

name	wmc	dit	noc	cbo	rfc	lcom	ca	npm	lcom3
webwypozyczalnia2.RequestBean1	6	3	0	3	11	15	14	3	2.0
webwypozyczalnia2.Ksiazkibaza	7	3	0	4	12	21	0	3	2.0
webwypozyczalnia2.Ksiazkiaplikacja	7	3	0	4	12	21	0	3	2.0
webwypozyczalnia2.Logo	7	3	0	4	12	21	0	3	2.0
webwypozyczalnia2.Tytulybaza	7	3	0	4	12	21	0	3	2.0
webwypozyczalnia2.Menu	26	3	0	5	32	283	6	22	0.92
webwypozyczalnia2.Page1	9	3	0	6	16	34	0	5	0.875
webwypozyczalnia2.Ksiazki	10	3	0	11	28	39	0	6	0.8611111111111112
webwypozyczalnia2.Baza_ksiazki	10	3	0	6	19	39	0	6	0.8333333333333334
webwypozyczalnia2.Baza_tytuly	10	3	0	6	19	39	0	6	0.8333333333333334
webwypozyczalnia2.Tytuly	10	3	0	7	22	39	0	6	0.8148148148148149
webwypozyczalnia2.Baza_tytul	10	3	0	7	21	39	0	6	0.8148148148148149
webwypozyczalnia2.FormTytul	21	3	0	5	32	124	2	17	0.8083333333333333
webwypozyczalnia2.ApplicationBean1	25	3	0	7	54	234	16	24	0.775
webwypozyczalnia2.SessionBean1	9	3	0	2	14	30	15	7	0.75
webwypozyczalnia2.FormKsiazka	15	3	0	5	26	61	1	11	0.738095238095238
webwypozyczalnia2.Tytulyaplikacja	11	3	0	8	25	37	1	7	0.7

6. Przykład aplikacji typu Visual Web Java Server Faces – klasa ApplicationBean1 po refaktoryzacji. Dokonano podziału na dwie klasy o czasie życia **application**.
Klasa ApplicationBean1 udostępnia usługi warstwy biznesowej, a klasa ApplicationBean2 stanowi fasadę warstwy integracji z bazą danych

Należy wstawić definicję nowej klasy **ApplicationBean2** do pliku **faces-config.xml**

```
<?xml version='1.0' encoding='UTF-8'?>

<!-- ========== FULL CONFIGURATION FILE ========================================== -->

<faces-config version="1.2"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/ja
    <managed-bean>
        <managed-bean-name>SessionBean1</managed-bean-name>
        <managed-bean-class>webwypozyczalnia2.SessionBean1</managed-bean-class>
        <managed-bean-scope>session</managed-bean-scope>
    </managed-bean>
    <managed-bean>
        <managed-bean-name>Page1</managed-bean-name>
        <managed-bean-class>webwypozyczalnia2.Page1</managed-bean-class>
        <managed-bean-scope>request</managed-bean-scope>
    </managed-bean>
    <managed-bean>
        <managed-bean-name>ApplicationBean1</managed-bean-name>
        <managed-bean-class>webwypozyczalnia2.ApplicationBean1</managed-bean-class>
        <managed-bean-scope>application</managed-bean-scope>
    </managed-bean>
    <managed-bean>
        <managed-bean-name>ApplicationBean2</managed-bean-name>
        <managed-bean-class>webwypozyczalnia2.ApplicationBean2</managed-bean-class>
        <managed-bean-scope>application</managed-bean-scope>
    </managed-bean>
```

```
public class ApplicationBean1 extends AbstractApplicationBean {
    Managed Component Definition
    /**...*/
    public ApplicationBean1() {...}
    /**...*/
    @Override
    public void init() {...}
    private TAplikacja aplikacja = new TAplikacja();
    public TAplikacja getAplikacja() {...}
    public void setAplikacja(TAplikacja aplikacja_) {...}
    public void dodaj_tytul(String dane[]) {...}//przypadek użycia '
    public void dodaj_ksiazke(String dane1[], String dane2[]) {...}

    private Option tytuly_[] = new Option[0];
    public Option[] getTytuly_() {...}
    public void setTytuly_(Option[] tytuly) {...}
    public <T> Option[] przygotuj(ArrayList<T> kol) {...}
    public void przygotujtytuly() {...}

    private Option ksiazki_[] = new Option[0];
    public Option[] getKsiazki_() {...}
    public void setKsiazki_(Option[] ksiazki) {...}
    public void przygotujksiazki(TTytul_ksiazki tytul) {...}
    /**...*/
    @Override
    public void destroy() {...}
    @Override
    public String getLocaleCharacterEncoding() {...}
}
```

Klasa ApplicationBean1 po ekstrakcji klasy ApplicationBean2

Przykład refaktoryzacji metod przygotujtytuly i przygotujksiazki – powtarzający się kod zastąpiono metodą przygotuj

```
public <T> Option[] przygotuj(Collection<T> kol) {
    int ile = kol.size();
    if (ile > 0) {
        Option pom[] = new Option[ile];
        Iterator iterator = kol.iterator();
        int i = 0;
        while (iterator.hasNext()) {
            pom[i++] =
             new Option(Integer.toString(i), iterator.next().toString());
        }
        return pom; }
    return null;
}
public void przygotujtytuly() {
    ArrayList<TTytul_ksiazki> tytuly = aplikacja.getTytul_ksiazki();
    Option pom[] = przygotuj(tytuly);
    if (pom != null) {
        tytuly_ = pom; }
}
public void przygotujksiazki(TTytul_ksiazki tytul) {
    if (tytul == null) {
        return;  }
    Collection<TEgzemplarz> ksiazki = tytul.getKsiazka();
    Option pom[] = przygotuj(ksiazki);
    if (pom != null) {
        ksiazki_ = pom; }
}
```

```
public class ApplicationBean2 extends AbstractApplicationBean {
    // <editor-fold defaultstate="collapsed" desc="Managed Component Definition">
    private ApplicationBean1 a;
    private TAplikacja aplikacja;
    TTytul_ksiazkiController tytulController = new TTytul_ksiazkiController();
    TEgzemplarzController egzController = new TEgzemplarzController();
    /**...*/
    private void _init() throws Exception {
    }
    // </editor-fold>
    /**...*/
    public ApplicationBean2() {...}
    /**...*/
    @Override
    public void init() {...}

    private TTytul_ksiazki tytuly[];
    public TTytul_ksiazki[] getTytuly() {...}
    public void setTytuly(TTytul_ksiazki[] tytuly_) {...}
    public void updateTytuls() {...}

    private TEgzemplarz ksiazki[];
    public TEgzemplarz[] getKsiazki() {...}
    public void setKsiazki(TEgzemplarz[] ksiazki_) {...}
    public void updateAplikacja() {...}
    public void updateKsiazkis() {...}

    public void zapisz_tytul_do_bazy(String dane[]) {...}
    public void zapisz_tytuly_do_bazy() {...}
    public void zapisz_ksiazki_do_bazy() {...}
```

Klasa ApplicationBean2 po ekstrakcji z klasy ApplicationBean2

```
public class ApplicationBean2 extends AbstractApplicationBean {
    // <editor-fold defaultstate="collapsed" desc="Managed Component Definition">
    private ApplicationBean1 a;
    private TAplikacja aplikacja;
    TTytul_ksiazkiController tytulController = new TTytul_ksiazkiController();
    TEgzemplarzController egzController = new TEgzemplarzController();
    /**...*/
    private void _init() throws Exception {
    }
    // </editor-fold>
    /**...*/
    public ApplicationBean2() {...}
    /**...*/
    @Override
    public void init() { // Perform initializations inherited from our superclass
        super.init();
        // Perform application initialization that must complete
        // *before* managed components are initialized
        // TODO - add your own initialiation code here
        Managed Component Initialization
        // *after* managed components are initialized
        // TODO - add your own initialization code here
        a = getApplicationBean1();
        aplikacja = a.getAplikacja();
        updateTytuls();
        updateKsiazkis();
        updateAplikacja();
        a.przygotujtytuly();
    }
```

Ekstrakcja klasy ApplicationBean2 po refaktoryzacji klasy ApplicationBean1 cd.

Metryki CK klas ApplicationBean1 i ApplicationBean2 po refaktoryzacji

Top 25: rfc

[wmc] [dit] [noc] [cbo] [rfc] [lcom] [ca] [npm] [lcom3] [explanations]

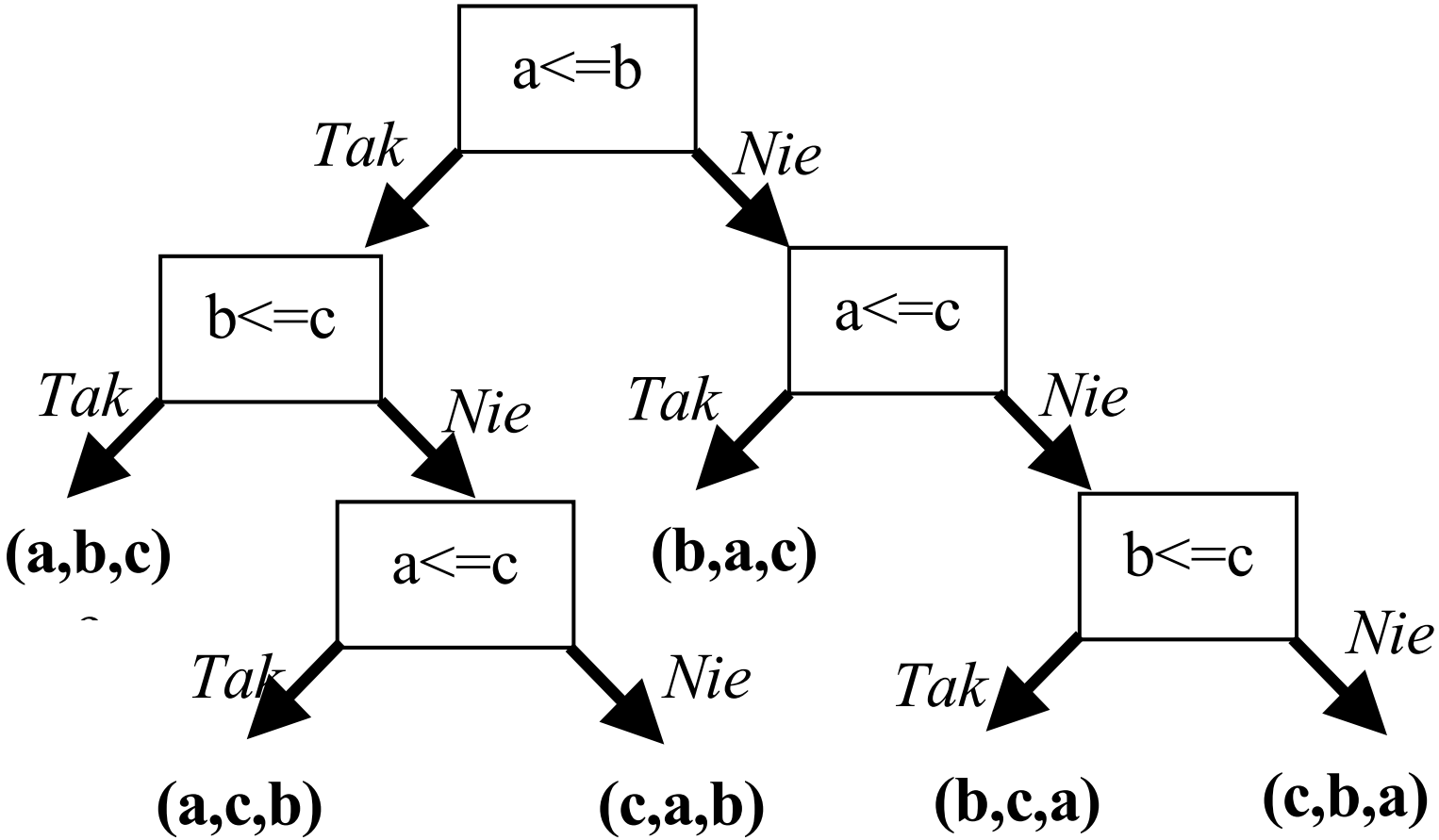
name	wmc	dit	noc	cbo	rfc	lcom	ca	npm	lcom3
webwypozyczalnia2.ApplicationBean2	16	3	0	7	40	60	4	14	0.7777777777777779
webwypozyczalnia2.ApplicationBean1	16	3	0	4	33	68	14	15	0.7333333333333333
webwypozyczalnia2.Menu	26	3	0	5	32	283	6	22	0.92
webwypozyczalnia2.FormTytul	21	3	0	5	32	124	2	17	0.8083333333333333
webwypozyczalnia2.Ksiazki	10	3	0	11	28	39	0	6	0.8611111111111112
webwypozyczalnia2.FormKsiazka	15	3	0	5	26	61	1	11	0.738095238095238
webwypozyczalnia2.Tytulyaplikacja	11	3	0	8	25	37	1	7	0.7
webwypozyczalnia2.Tytuly	11	3	0	8	23	49	0	6	0.875
webwypozyczalnia2.Baza_tytul	10	3	0	7	21	39	0	6	0.8148148148148149
webwypozyczalnia2.Baza_ksiazki	10	3	0	6	19	39	0	6	0.8333333333333334
webwypozyczalnia2.Baza_tytuly	10	3	0	6	19	39	0	6	0.8333333333333334
webwypozyczalnia2.Page1	9	3	0	6	16	34	0	5	0.875
webwypozyczalnia2.Ksiazkibaza	9	3	0	5	15	30	0	5	0.75
webwypozyczalnia2.SessionBean1	9	3	0	2	14	30	15	7	0.75
webwypozyczalnia2.Ksiazkiaplikacja	7	3	0	4	12	21	0	3	2.0
webwypozyczalnia2.Logo	7	3	0	4	12	21	0	3	2.0
webwypozyczalnia2.Tytulybaza	7	3	0	4	12	21	0	3	2.0
webwypozyczalnia2.RequestBean1	6	3	0	3	11	15	14	3	2.0

7. Przykłady metod zawierających równoważne algorytmy o różnych wartościach metryk MC Cabe

7.1. Rozwiązanie równania liniowego

```
public class MCCABE1 {  //Average cyclomatic complexity: 9.166666666666666
    static void rownanie()          // Liczba MCCABE1::rownanie:    7
    {   float a,b,c,x,d;            // Trzy porównania
        String s;
        JOptionPane.showMessageDialog(null,„Rownanie liniowe ax+by=c\n");
        s=JOptionPane.showInputDialog(null, "Podaj a");      a = Float.parseFloat(s);
        s=JOptionPane.showInputDialog(null, "Podaj b");      b = Float.parseFloat(s);
        s=JOptionPane.showInputDialog(null, "Podaj c");      c = Float.parseFloat(s);
        s=JOptionPane.showInputDialog(null, "Podaj x");      x = Float.parseFloat(s);
        if(a==0)
            if(b==0)
                if(c==0)          s="Dla dowolnego x i y";
                else              s="Brak rozwiazania";
            else  if(c==0)        s="dowolne x,y="+0;
                  else            s="dowolne x, y="+(c/b);
        else
            if(b==0)              s="dowolne y"+((a*x)==c);
            else  if(c==0)        s="y="+((a*x)/b);
                  else            s="y="+((c-(a*x))/b);
        JOptionPane.showMessageDialog(null, s);
    }
```

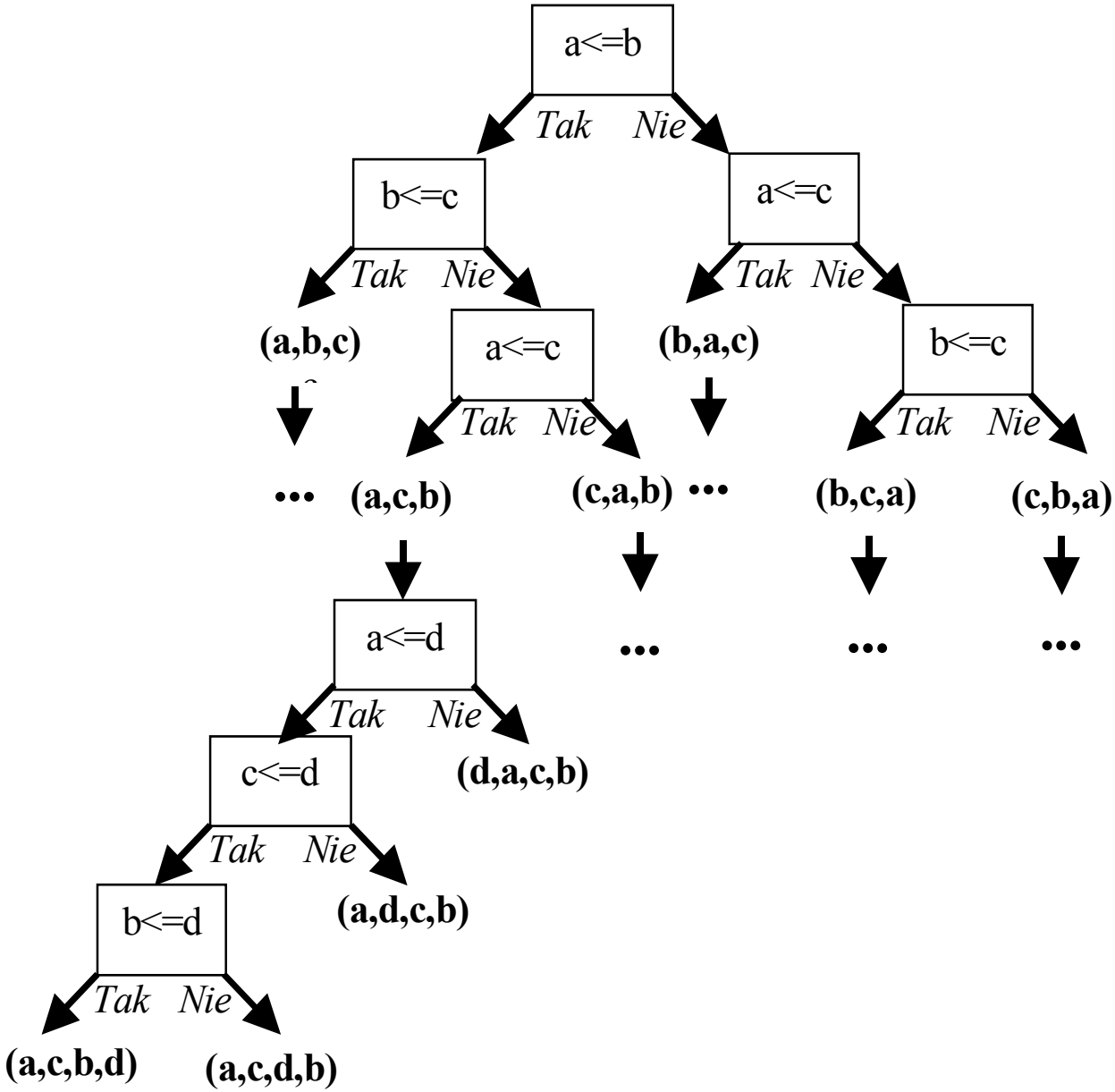
7.2. Sortowanie trzech liczb – drzewo algorytmu (dwa lub trzy porównania)



Sortowanie trzech liczb

```
static void liczby3()          //Liczba MCCABE1::liczby3:      6
  {  float a,b,c,d;
     String s;
     JOptionPane.showMessageDialog(null,„Sortowanie trzech liczb");
     s=JOptionPane.showInputDialog(null, "Podaj a");      a = Float.parseFloat(s);
     s=JOptionPane.showInputDialog(null, "Podaj b");      b = Float.parseFloat(s);
     s=JOptionPane.showInputDialog(null, "Podaj c");      c = Float.parseFloat(s);
    if(a<=b)
      if(b<=c)
                        JOptionPane.showMessageDialog(null, "a,b,c");
      else if(a<=c)
                        JOptionPane.showMessageDialog(null, "a,c,b");
           else       JOptionPane.showMessageDialog(null, "c,a,b");
     else if(a<=c)
                       JOptionPane.showMessageDialog(null, "b,a,c");
        else if (b<=c)
                       JOptionPane.showMessageDialog(null, "b,c,a");
             else    JOptionPane.showMessageDialog(null, "c,b,a");
}
```

7.3.1. Sortowanie czterech liczb (pierwszy sposób) – drzewo algorytmu pięć lub sześć porównań



Sortowanie czterech liczb (pierwszy sposób) - program

```
static void liczby4_1()                    // Liczba MCCABE1::liczby4_1:  24
    {   float a,b,c,d;
        String s;
        JOptionPane.showMessageDialog(null, "Sortowanie 4 liczb");
        s= JOptionPane.showInputDialog(null, "Podaj a");
        a = Float.parseFloat(s);
        s= JOptionPane.showInputDialog(null, "Podaj b");
        b = Float.parseFloat(s);
        s= JOptionPane.showInputDialog(null, "Podaj c");
        c = Float.parseFloat(s);
        s= JOptionPane.showInputDialog(null, "Podaj d");
        d = Float.parseFloat(s);
```

```
if(a<=b)    //1
    if(b<=c)  //2   //a,b,c
            if(a<=d) //3
              if(b<=d) //4
                if(c<=d)          JOptionPane.showMessageDialog(null, "a,b,c,d"); //5
                else              JOptionPane.showMessageDialog(null, "a,b,d,c"); //5
              else                JOptionPane.showMessageDialog(null, "a,d,b,c"); //4
            else                  JOptionPane.showMessageDialog(null, "d,a,b,c"); //3
    else  //2
        if(a<=c) //6  //a,c,b
          if(a<=d) //3
            if(c<=d) //4
              if(b<=d)          JOptionPane.showMessageDialog(null, "a,c,b,d"); //5
              else              JOptionPane.showMessageDialog(null, "a,c,d,b"); //5
            else                JOptionPane.showMessageDialog(null, "a,d,c,b"); //4
          else                  JOptionPane.showMessageDialog(null, "d,a,c,b"); //3
        else //6     //c,a,b
          if(c<=d) //3
            if(a<=d) //4
              if(b<=d)          JOptionPane.showMessageDialog(null, "c,a,b,d"); //5
              else              JOptionPane.showMessageDialog(null, "c,a,d,b"); //5
            else                JOptionPane.showMessageDialog(null, "c,d,a,b"); //4
          else                  JOptionPane.showMessageDialog(null, "d,c,a,b"); //3
```

```
    else  //1
        if(a<=c)  //7  //b,a,c
                if(b<=d) //3
                  if(a<=d) //4
                    if(c<=d)      JOptionPane.showMessageDialog(null, "b,a,c,d"); //5
                    else          JOptionPane.showMessageDialog(null, "b,a,d,c"); //5
                  else            JOptionPane.showMessageDialog(null, "b,d,a,c"); //4
                else              JOptionPane.showMessageDialog(null, "d,b,a,c"); //3
        else  //7
              if (b<=c)  //8  //b,c,a
                if(b<=d) //3
                  if(c<=d) //4
                    if(a<=d)     JOptionPane.showMessageDialog(null, "b,c,a,d"); //5
                    else         JOptionPane.showMessageDialog(null, "b,c,d,a"); //5
                  else           JOptionPane.showMessageDialog(null, "b,d,c,a"); //4
                else             JOptionPane.showMessageDialog(null, "d,b,c,a"); //3
              else //8      //c,b,a
                 if(c<=d) //3
                   if(b<=d) //4
                     if(a<=d)     JOptionPane.showMessageDialog(null, "c,b,a,d"); //5
                     else         JOptionPane.showMessageDialog(null, "c,b,d,a"); //5
                   else           JOptionPane.showMessageDialog(null, "c,d,b,a"); //4
                 else             JOptionPane.showMessageDialog(null, "d,c,b,a"); //3
    }
```

7.3.2. Sortowanie czterech liczb – drugi sposób (wg Maciej Sysło, Algorytmy)

1) Koncepcja sortowania

Należy posortować niemalejąco 4 dane używając **5 porównań**, gdy dopuszczalna liczba porównań jest równa **6 (4*3/2=6).**

Projekt programu

Dane: tablica zawierająca 5 liczb

- porównaj **(x[1] > x[2])** – jeśli prawda, zamień zawartość obu elementów, aby x[1]<=x[2]
- **wstaw zawartość elementu c=x[3]** we właściwe miejsce między x[1], x[2]
  - jeśli prawda, że **(x[2]>c),** to wstaw x[3]=x[2]
    - jeśli prawda, że **(x[1]>c)**, to wstaw x[2]=x[1] oraz x[1]=c,
    - jeśli nieprawda, to wstaw x[2]=c
- **wstaw zawartość elementu d=x[4]** we właściwe miejsce między elementy x[1], x[2], x[3]
  - jeśli prawda, że **(x[2]>d)**, to wstaw x[4]=x[3], x[3]=x[2]
    - jeśli prawda, że **(x[1]>d)**, to wstaw x[2]=x[1], x[1]=d,
    - jeśli nieprawda, to wstaw x[2]=d
  - jeśli nieprawda,
    - jeśli prawda, że (x[3]>d), to wstaw x[4]=x[3], x[3]=d

Sortowanie czterech liczb – drugi sposób (wg Maciej Sysło, Algorytmy)

```
static void liczby4_2()                    // lLiczba MCCABE1::liczby4_2:    7
  { int x[]= {4,1,9,0};
    int pom, c, d;
    if (x[0]>x[1])     { pom=x[0]; x[0]=x[1];x[1]=pom; }
    c=x[2];
    if (x[1]>c)
      {  x[2]=x[1];
         if (x[0]>c)     {  x[1]=x[0];  x[0]=c; }
         else              x[1]=c;
      }
   d=x[3];
   if (x[1]>d)
      {  x[3]=x[2]; x[2]=x[1];
        if (x[0]<=d)        x[1]=d;
        else               { x[1]=x[0]; x[0]=d; }
      }
   else  if (x[2]>d)          { x[3]=x[2]; x[2]=d; }
JOptionPane.showMessageDialog(null,
              "x[0] "+x[0]+",x[1] "+x[1]+",x[2] "+x[2]+",x[3]"+ x[3]);
  }
```

7.4.1. Sortowanie pięciu liczb – drugi sposób (wg Maciej Sysło, Algorytmy)

1) Koncepcja sortowania

Należy posortować niemalejąco 5 danych używając **7 porównań**, gdy dopuszczalna liczba porównań jest równa **10 (5*4/2=10).**

2) Projekt programu

Dane: tablica zawierająca 5 liczb

- porównaj **(x[1] > x[2])** – jeśli prawda, zamień zawartość obu elementów, aby x[1]<=x[2]
- porównaj **(x[4] > x[5])** – jeśli prawda, zamień zawartość obu elementów, aby x[4]<=x[5]
- porównaj **(x[2] > x[5])** – jeśli prawda, zamień zawartość obu par elementów, aby x[2]<=x[5], jednak niekoniecznie będzie prawdą po zamianie, że x[1]<=x[4]
- **wstaw zawartość elementu c=x[3]** we właściwe miejsce między x[1], x[2], x[5]
  - jeśli prawda, że **(x[2]>c),** to wstaw x[3]=x[2]
    - jeśli prawda, że **(x[1]>c)**, to wstaw x[2]=x[1] oraz x[1]=c,
    - jeśli nieprawda, to wstaw x[2]=c
  - jeśli nieprawda,
    - jeśli prawda, że(c>x[5]), to należy zamienić zawartość x[3] oraz x[5], tak aby x[3]<=x[5]
- **wstaw zawartość elementu d=x[4]** we właściwe miejsce między elementy mniejsze niż x[5], czyli x[1], x[2], x[3]
  - jeśli prawda, że **(x[2]>d)**, to wstaw x[4]=x[3], x[3]=x[2]
    - jeśli prawda, że **(x[1]>d)**, to wstaw x[2]=x[1], x[1]=d,
    - jeśli nieprawda, to wstaw x[2]=d
  - jeśli nieprawda,
    - jeśli prawda, że (x[3]>d), to wstaw x[4]=x[3], x[3]=d

7.4.2. Sortowanie 5 liczb – drugi sposób (wg Maciej Sysło, Algorytmy)

```
static void liczby5_2() //Liczba MCCABE1::liczby5_1: 120 (pierwszy sposób)
 { int x[]= {4,1,9,0,20};   //MCCABE1::liczby5_2: 10  (drugi sposób)
   int pom, c, d;
    if (x[0]>x[1])    { pom=x[0]; x[0]=x[1];x[1]=pom; }
    if (x[3]>x[4])    { pom=x[3]; x[3]=x[4];x[4]=pom; }
    if (x[1]>x[4])    { pom=x[1]; x[1]=x[4];x[4]=pom; pom=x[0]; x[0]=x[3];x[3]=pom; }
    c=x[2];
    if (x[1]>c)
       { x[2]=x[1];
         if (x[0]>c)              { x[1]=x[0];  x[0]=c; }
         else                       x[1]=c;
        }
    else   if (c>x[4])             { pom=x[2]; x[2]=x[4]; x[4]=pom; }
    d=x[3];
    if (x[1]>d)
       { x[3]=x[2]; x[2]=x[1];
         if (x[0]<=d)               x[1]=d;
         else                     { x[1]=x[0]; x[0]=d; }
       }
    else  if (x[2]>d)             { x[3]=x[2]; x[2]=d; }
    JOptionPane.showMessageDialog(null,
          "x[0] "+x[0]+",x[1] "+x[1]+",x[2] "+x[2]+",x[3] "+x[3]+",x[4] "+x[4]);    }
```

Poprawa szybkości algorytmu mierzona liczbą porównań **może prowadzić w niektórych wypadkach** do poprawy złożoności kodu. Jednak nie jest to zasadą – należy więc przy poprawie złożoności struktury programu sprawdzać zmianę złożoności obliczeniowej. Dążąc do poprawy złożoności obliczeniowej nie można przekroczyc granic wyznaczonych wartościami metryk złożoności struktury oprogramiowania, ponieważ takie programy mogą stać się nietestowalne.

	Sortowanie 3 liczb	Sortowanie 4 liczb		Sortowanie 5 liczb	
	1 sposób	1 sposób	2 sposób	1 sposób	2 sposób
Liczba porównań	2-3	6	5	10	**7**
Liczba MCCabe1	6	24	7	120	**10**