Metryki

Przykłady pomiaru złożoności modułowej i międzymodułowej **oprogramowania**

autor: Zofia Kruczkiewicz

Dodatek

Powtórka z metryk ilustrowana wynikami pomiaru realizowanymi przez narzędzia ckjm 1.8, SCM (dodatek do NetBeans 6.7.1) oraz RefactorIT (dodatek do NetBeans 5.5.1)

2. Zastosowanie narzędzi do pomiaru złożoności oprogramowania

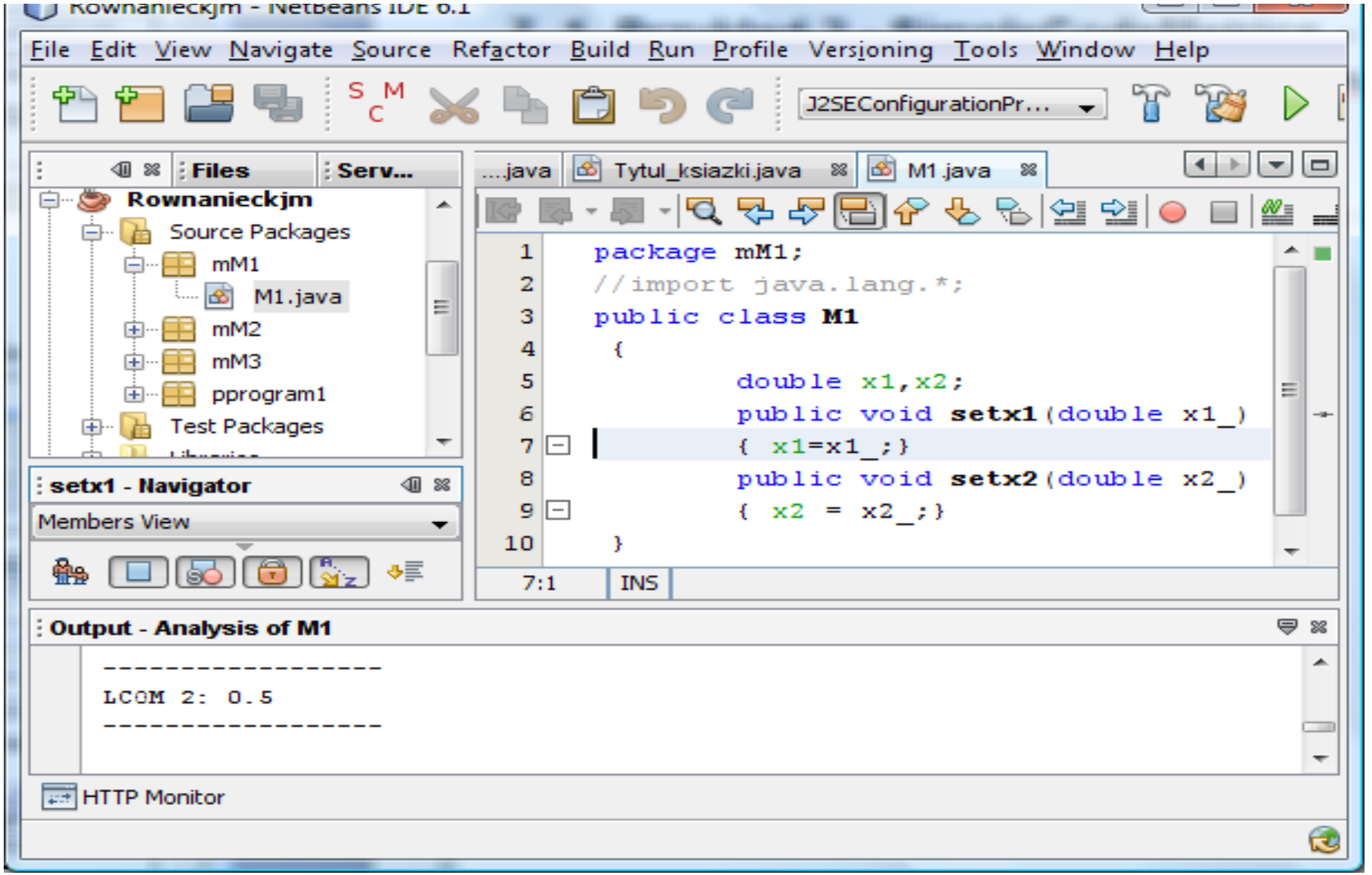
2.1. Przykład 1 – wyznaczanie równania kwadratowego

- SimpleCodeMetrics - Copyright 2008 Krzysztof Dębski (instalacja modułu org-netbeans-modules-scm.nbm)
- CKJM

2.2. Przykład 2 – katalog książek

- SimpleCodeMetrics - Copyright 2008 Krzysztof Dębski (instalacja modułu org-netbeans-modules-scm.nbm)
- CKJM

2. 1. Przykład 1 - wyznaczanie równania kwadratowego SimpleCodeMetrics (SCM) - Copyright 2008 Krzysztof Dębski (instalacja modułu org-netbeans-modules-scm.nbm)



Schemat powiązań międzymodułowych do pomiaru metryk międzymodułowych

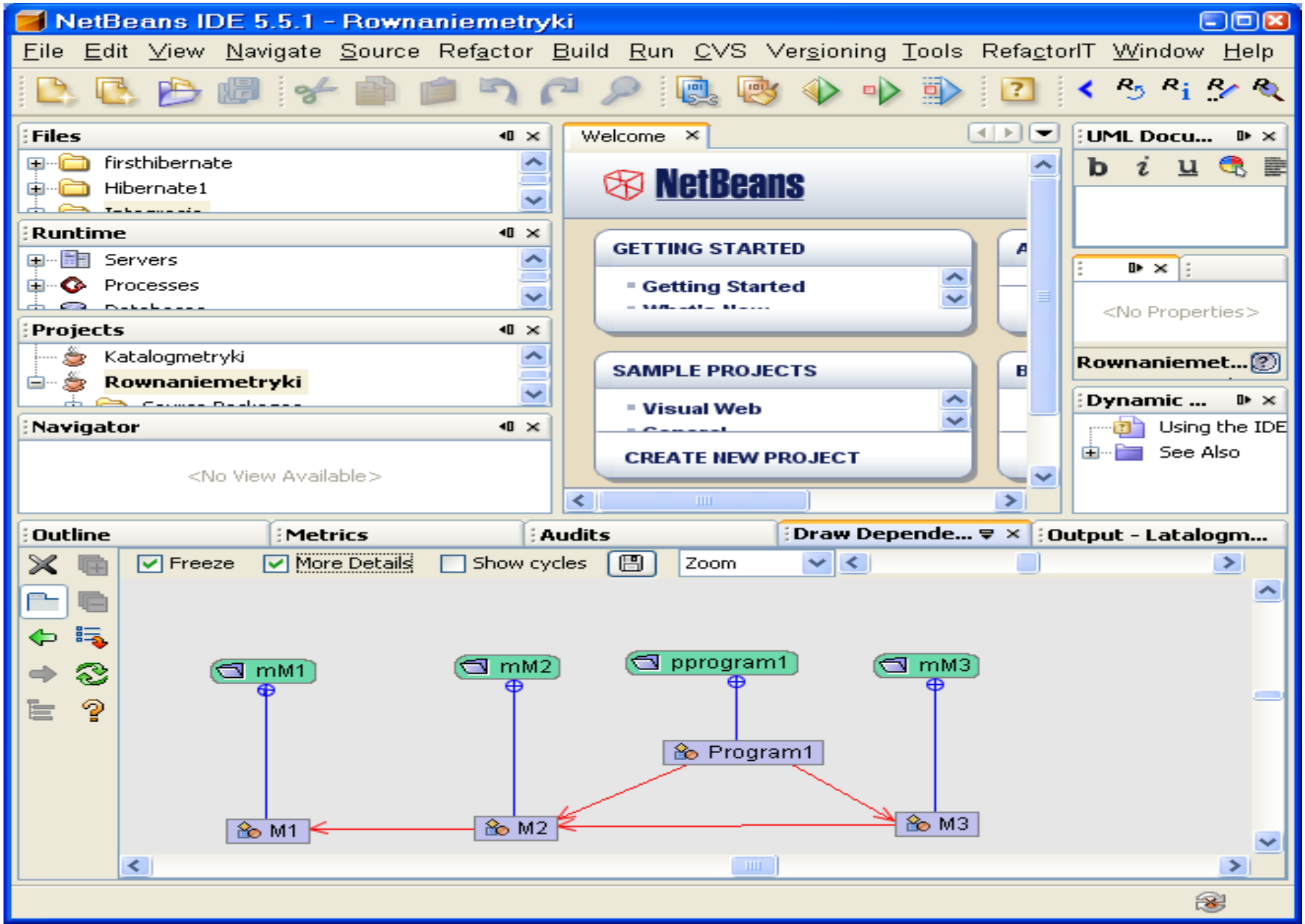
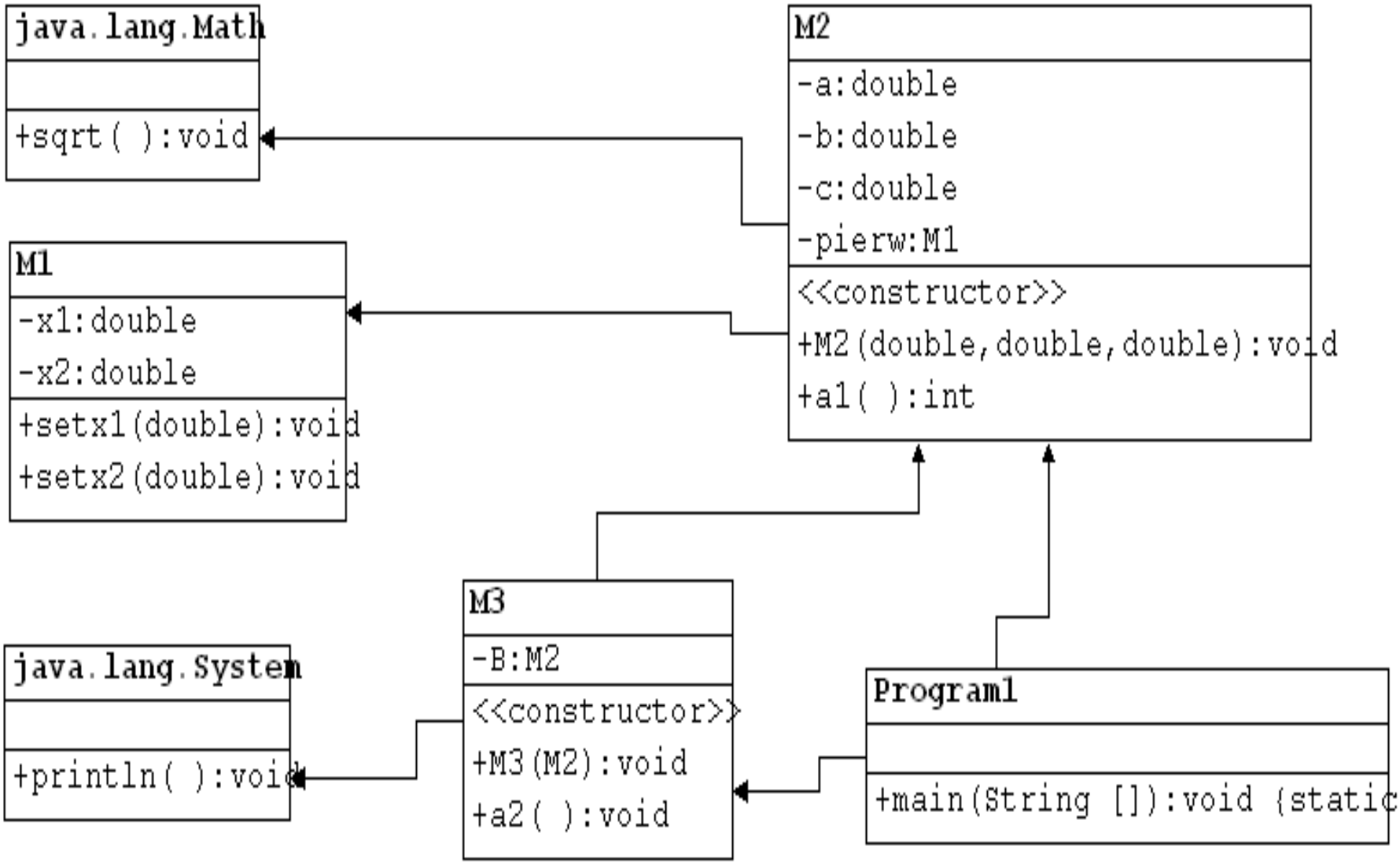
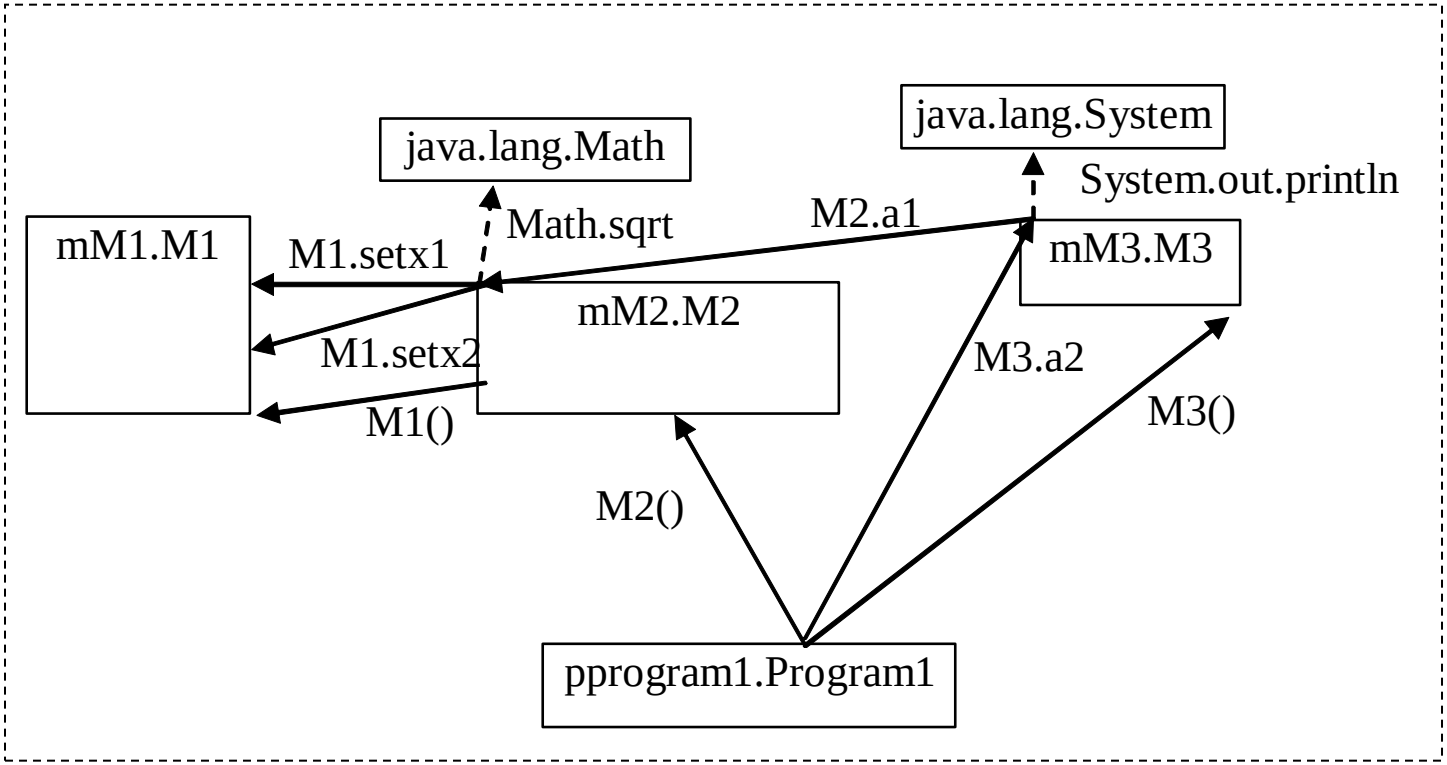


Diagram klas badanego programu

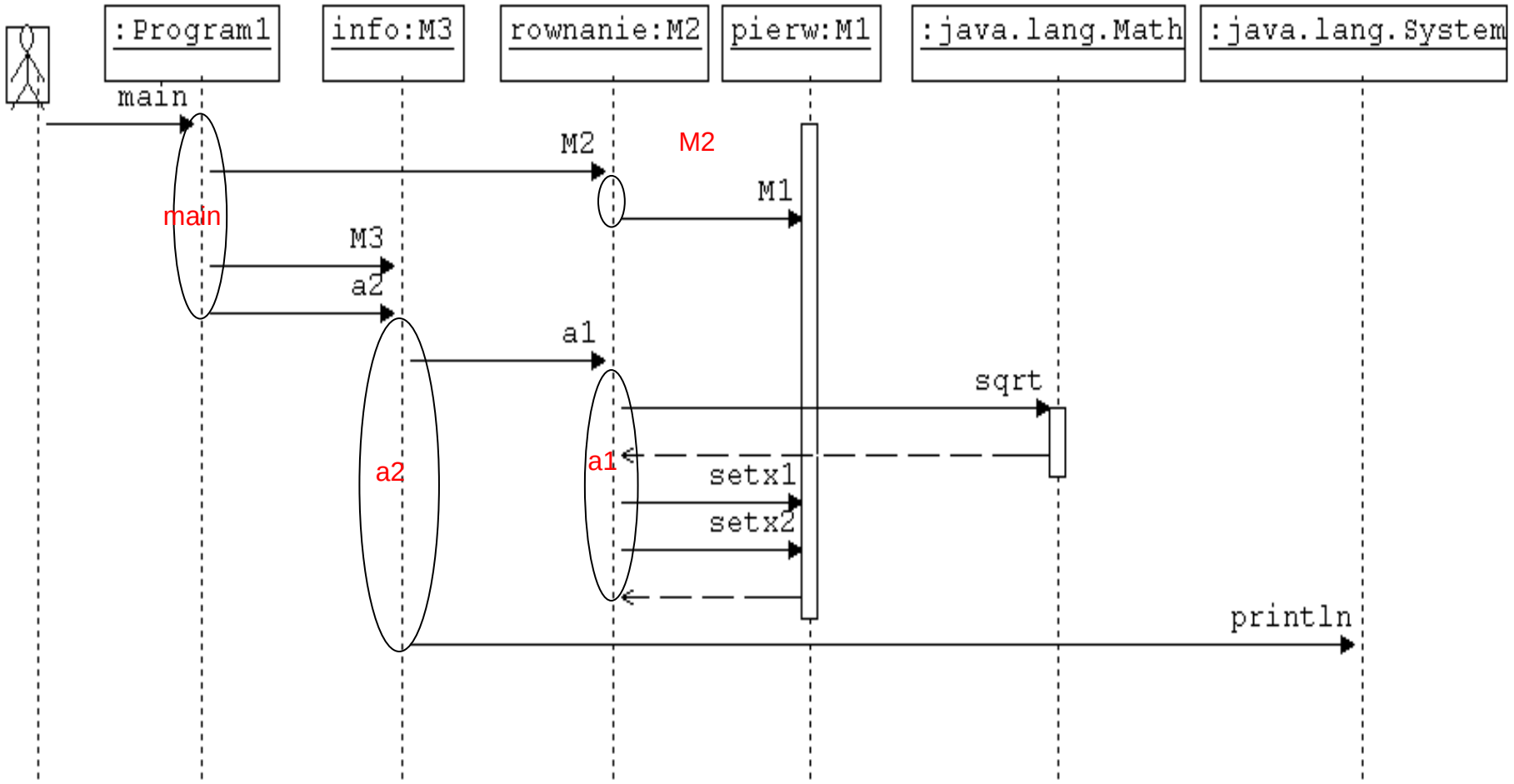


Schemat do pomiaru metryk połączeń międzymodułowych

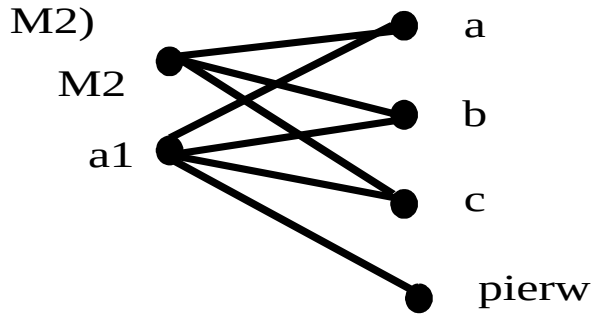


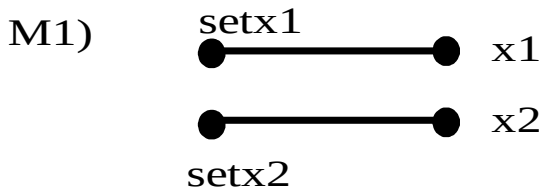
	mM1	mM2	mM3	pprogram1	java.lang.System	java.lang.Math
Fan-out	-	1 + (1)	1+ (1)	2	-	-
Fan-in	1	2	1	-	(1)	(1)
RFC	-	5+(1)	2+(1)	4	-	-
|R|	-	3+(1)	1+(1)	3	-	-

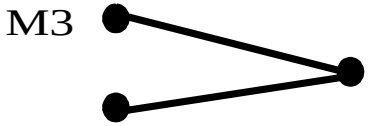
Korzystanie z diagramu sekwencji do pomiaru metryk powiązań międzymodułowych



Obliczanie metryki spójności







Spójność klasy M1

- a=2, m=2, r=2

$$LCOM = \frac{\frac{r}{a} - m}{1 - m} = 1$$

Spójność klasy M2

- a=4, m=2, r=7

$$LCOM = \frac{\frac{r}{a} - m}{1 - m} = \frac{\frac{7}{4} - 2}{1 - 2} = 0.25$$

Spójność klasy M2'

a=4, m=2, r=8

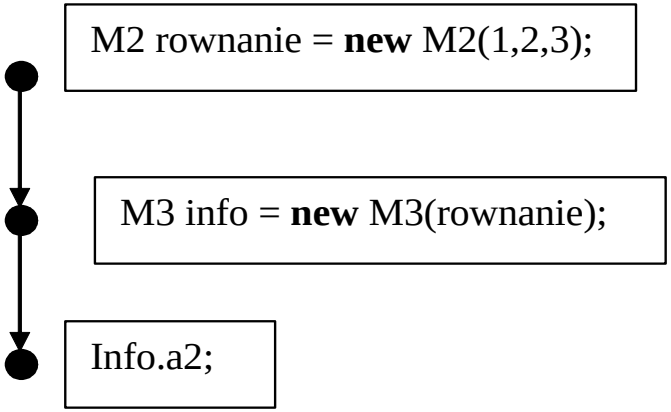
$$LCOM = \frac{\frac{r}{a} - m}{1 - m} = \frac{\frac{8}{4} - 2}{1 - 2} = 0$$

Spójność klasy M3

- a=1, m=2, r=2

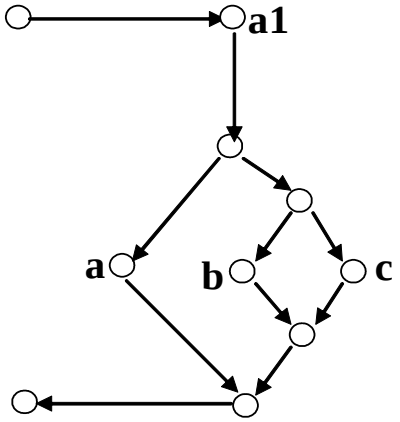
$$LCOM = \frac{\frac{r}{a} - m}{1 - m} = \frac{\frac{2}{1} - 2}{1 - 2} = 0$$

Wyznaczanie metryk MC Cabe



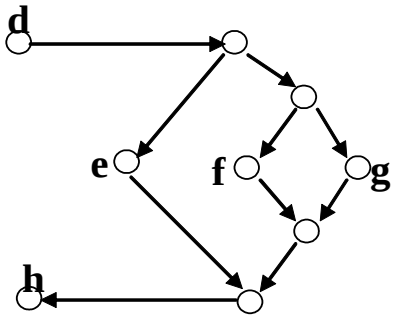
Program1::main: Vli(G)=V(G)=1

M3::a2: Vli(G)=V(G)=3



- a1:**int** b=B.a1();
- a: System.*out*.println("Brak równania kwadratowego\n");
- b: System.*out*.println("Brak pierwiastków rzeczywistych\n");
- c: System.*out*.println("Rownanie ma pierwiastki rzeczywiste\n");
- d: **int** B; **double** pom=2*a, d=b*b-4*a*c;
- e: B=0;
- f: B=1;
- g: B=2;
  d=Math.*sqrt*(d);
  pierw.setx1((-b-d)/pom);
  pierw.setx2((-b+d)/pom);
- h: return B;

M2::a1: Vli(G)=V(G)=3



Kod źródłowy klasy M1

```
package mM1;
public class M1
{   double x1,x2;
    public void setx1(double x1_)      { x1=x1_; }
    public void setx2(double x2_)      { x2 = x2_;}
}
```

SMC - Metryki kodu źródłowego klasy M1

- **LOC** | Total LOC: 11, Classes LOC: M1: 11, Packages LOC:
- **Lines with imports** |Total imports: 0, Classes imports:nM1: 0, Packages imports:
- **Blank lines** |Total blank lines:0, Classes blank lines:M1:0, Packages blank lines:
- **Classes count** |Total classes: 1, Packages with the biggest number of classes:
- **Methods count** |Total methods: 2,Classes with the biggest number of methods: M1: 2
- **Cyclomatic complexity** | Average cyclomatic complexity: 1.0
  Methods with the highest cyclomatic complexity: M1::setx2: 1, M1::setx1: 1
- **LCOM**
- Average LCOM 1: 1,
  Classes with the highest LCOM 1:M1: 1,Packages with the highest average LCOM 1:
- Average LCOM 2: 0.5
  Classes with the highest LCOM 2: M1: 0.5,Packages with the highest average LCOM 2:
- Average LCOM 3: 1.0
  Classes with the highest LCOM 3: M1: 1.0,Packages with the highest average LCOM 3:
- Average LCOM 4: 2
  Classes with the highest LCOM 4: M1: 2,Packages with the highest average LCOM 4:

Kod źródłowy klasy M2

```
package mM2;
import java.lang.Math;
import mM1.M1;
public class M2
 {
     private double a, b, c;
     private M1 pierw = new M1();
     public M2(double a_, double b_, double c_)
      { a=a_; b=b_; c=c_; }
     public int a1 ()
      { int B;
        double pom=2*a, d=b*b-4*a*c;
        if (a==0)  B=0;
        else
          if (d<0)   B=1;
          else
            { B=2;
              d=Math.sqrt(d);
              pierw.setx1((-b-d)/pom);
              pierw.setx2((-b+d)/pom);
            }
       return B;
      }
}
```

SMC - Metryki kodu źródłowego klasy M2

- **LOC** |Total LOC: 27, Classes LOC: M2: 27, Packages LOC:
- **Lines with imports** |Total imports:2, Classes imports: M2: 2, Packages imports:
- **Blank lines** |Total blank lines:1, Classes blank lines:M2:1, Packages blank lines:
- **Classes count** |Total classes: 1, Packages with the biggest number of classes:
- **Methods count** |Total methods: 1, Classes with the biggest number of methods: M2: 1
- **Cyclomatic complexity** | Average cyclomatic complexity: 3.0
  Methods with the highest cyclomatic complexity: M2::a1: 3
- **LCOM**
- Average LCOM 1: 0
  Classes with the highest LCOM 1: M2: 0,
  Packages with the highest average LCOM 1:
- Average LCOM 2: 0.0
  Classes with the highest LCOM 2:M2:0.0,
  Packages with the highest average LCOM 2:
- Average LCOM 3: 0.0
  Classes with the highest LCOM 3:M2:0.0,
  Packages with the highest average LCOM 3:
- Average LCOM 4: 1
  Classes with the highest LCOM 4:M2: 1,
  Packages with the highest average LCOM 4:

Kod źródłowy klasy M3

```
package mM3;
import mM2.M2;

public class M3
{
  M2 B;
  public M3 (M2 B_)
   {      B=B_;    }

 public void a2( )
   {
     int b=B.a1();
     if (b<1)
       System.out.println("Brak równania kwadratowego\n");
     else
      if (b==1)
        System.out.println("Brak pierwiastków rzeczywistych\n");
      else
        System.out.println("Rownanie ma pierwiastki rzeczywiste\n");
   }
}
```

SMC - Metryki kodu źródłowego klasy M3

- **LOC** |Total LOC: 21, Classes LOC: M3: 21, Packages LOC:
- **Lines with imports** |Total imports: 1, Classes imports: M3: 1, Packages imports:
- **Blank lines** |Total blank lines: 2,m Classes blank lines:M3: 2, Packages blank lines:
- **Classes count** |Total classes: 1, Packages with the biggest number of classes:
- **Methods count** |Total methods: 1, Classes with the biggest number of methods:M3: 1
- **Cyclomatic complexity** | Average cyclomatic complexity: 3.0
  Methods with the highest cyclomatic complexity: M3::a2: 3
- **LCOM**
- Average LCOM 1: 0,
  Classes with the highest LCOM 1:M3: 0,
  Packages with the highest average LCOM 1:
- Average LCOM 2: 0.0
  Classes with the highest LCOM 2: M3: 0.0,
  Packages with the highest average LCOM 2:
- Average LCOM 3: 0.0
  Classes with the highest LCOM 3: M3: 0.0,
  Packages with the highest average LCOM 3:
- Average LCOM 4: 1
  Classes with the highest LCOM 4: M3: 1,
  Packages with the highest average LCOM 4:

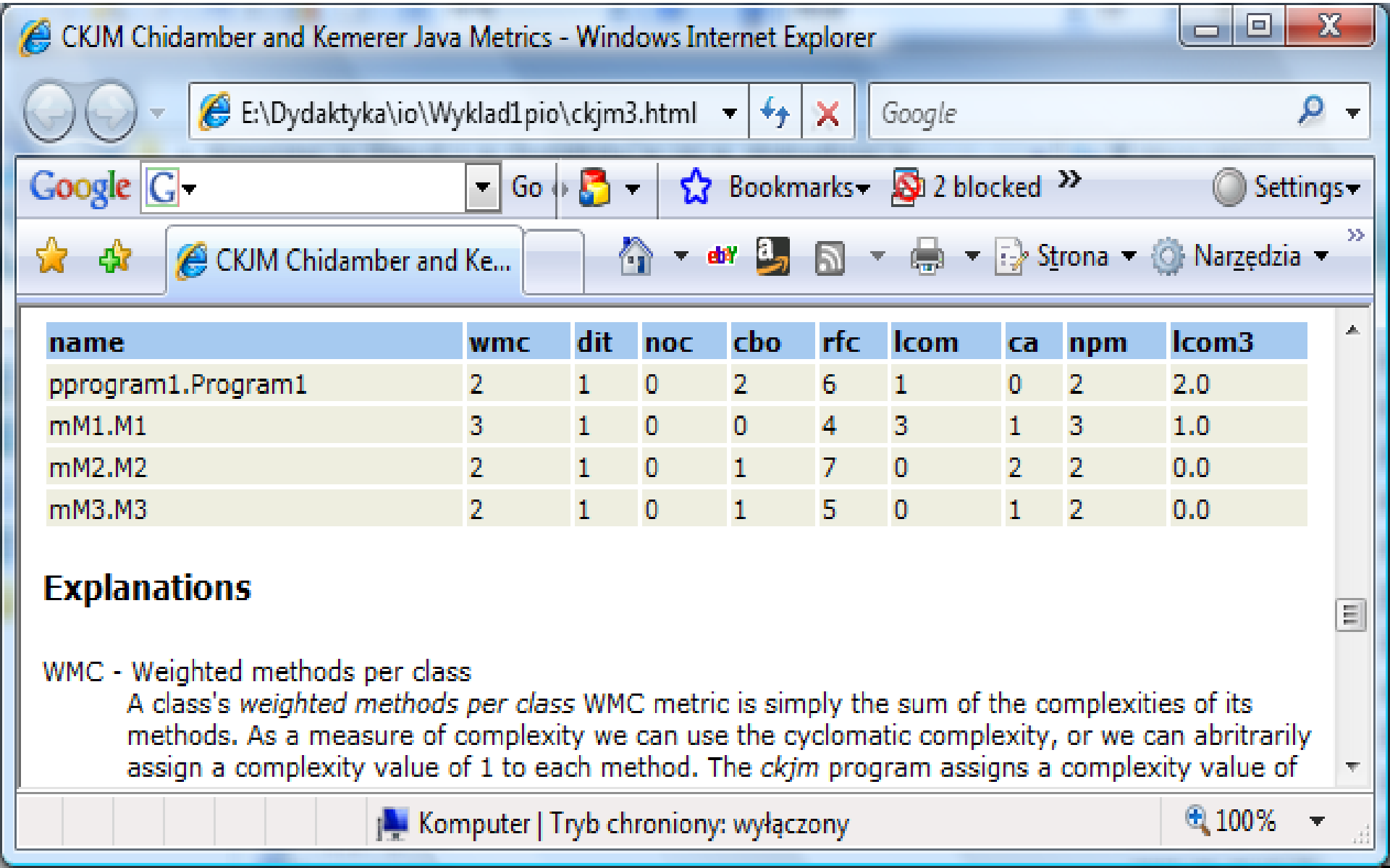
Kod źródłowy klasy Program1

```
package pprogram1;

import mM3.M3;
import mM2.M2;
public class Program1
{
    public static void main(String arg[])
    {
       M2 rownanie = new M2(1,2,3);
       M3 info= new M3(rownanie);
       info.a2();
     }
}
```

SMC - Metryki kodu źródłowego klasy Program1

- **LOC** |Total LOC: 15, Classes LOC: Program1: 15, Packages LOC:
- **Lines with imports** |Total imports: 2, Classes imports:Program1: 2, Packages imports:
- **Blank lines** |Total blank lines:3 |Classes blank lines:Program1: 3, Packages blank lines:
- **Classes count** |Total classes: 1,Packages with the biggest number of classes:
- **Methods count**|Total methods: 1,Classes with the biggest number of methods:Program1: 1
- **Cyclomatic complexity** | Average cyclomatic complexity: 1.0,
- Methods with the highest cyclomatic complexity: Program1::main: 1
- **LCOM**
- Average LCOM 1: 0,
  Classes with the highest LCOM 1:Program1: 0,
  Packages with the highest average LCOM 1:
- Average LCOM 2: 0.0
  Classes with the highest LCOM 2:Program1:0.0,
  Packages with the highest average LCOM 2:
- Average LCOM 3: 0.0
  Classes with the highest LCOM 3:Program1:0.0,
  Packages with the highest average LCOM 3:
- Average LCOM 4: 1
  Classes with the highest LCOM 4:Program1: 1,
  Packages with the highest average LCOM 4:

Wyniki działania programu ckjm do pomiaru metryk CK - wykonanie skryptu build za pomocą programu ant - apache-ant-1.7.1 i utworzenie raportu typu html



name	wmc	dit	noc	cbo	rfc	lcom	ca	npm	lcom3
pprogram1.Program1	2	1	0	2	6	1	0	2	2.0
mM1.M1	3	1	0	0	4	3	1	3	1.0
mM2.M2	2	1	0	1	7	0	2	2	0.0
mM3.M3	2	1	0	1	5	0	1	2	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

Przykład skryptu build.xml dla programu CKJM do pomiaru metryk CK.

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="changeme" default="ckjm" basedir=".">
                <!-- wskazanie na plik ckjm-1.8.extended.jar: -->
  <property name="ckjm.location" value="c:/downloads/ckjm-1.8/ckjm-1.8.extended.jar" />
                <!-- to laduje zadanie <ckjm> -->
  <taskdef name="ckjm" classname="gr.spinellis.ckjm.ant.CkjmTask">
      <classpath>
        <pathelement location="${ckjm.location}"/>
      </classpath>
  </taskdef>
              <!-- Metryki zostana zapisane w pliku ckjm4.html, mozna rowniez wybrac format xml.-->
  <target name="ckjm">
              <!-- ckjm lokalizuje pliki uzywane przez rozwijany projekt, przy zalozeniu, ze 'build/classes' jest
                      katalogiem, w ktorym sa umieszczane 'bajtkody' (*.class)-->
    <ckjm outputfile="ckjm.xml" format="xml„
          classdir="e:/dydaktyka/io/Wyklad1pio/Rownanieckjm/build/classes/">
      <include name="**/*.class" />
      <exclude name="**/*Test.class" />
              <!-- ckjm lokalizuje w tej linii klasy wchodzace w sklad rozwijanego projektu, przy zalozeniu, ze
                      build/classes' jest katalogiem w ktorym sa umieszczane skompilowane klasy (*.class)-->
      <extdirs path="e:/dydaktyka/io/Wyklad1pio/Rownanieckjm/build/classes/"/>
              <!-- ckjm lokalizuje biblioteki uzywane przez rozwijany projekt, przy zalozeniu, ze lib jest
                      katalogiem, w ktorym sa umieszczane biblioteki (*.jar)-->
      <extdirs path="e:/dydaktyka/io/Wyklad1pio/Rownanieckjm/dist/lib"/>
    </ckjm>
    <xslt in="ckjm.xml" style="ckjm_extra.xsl" out="ckjm3.html" />
  </target>
</project>
```

2. 2. Przykład 2 – katalog książek
SimpleCodeMetrics - Copyright 2008 Krzysztof Dębski (instalacja modułu org-netbeans-modules-scm.nbm)

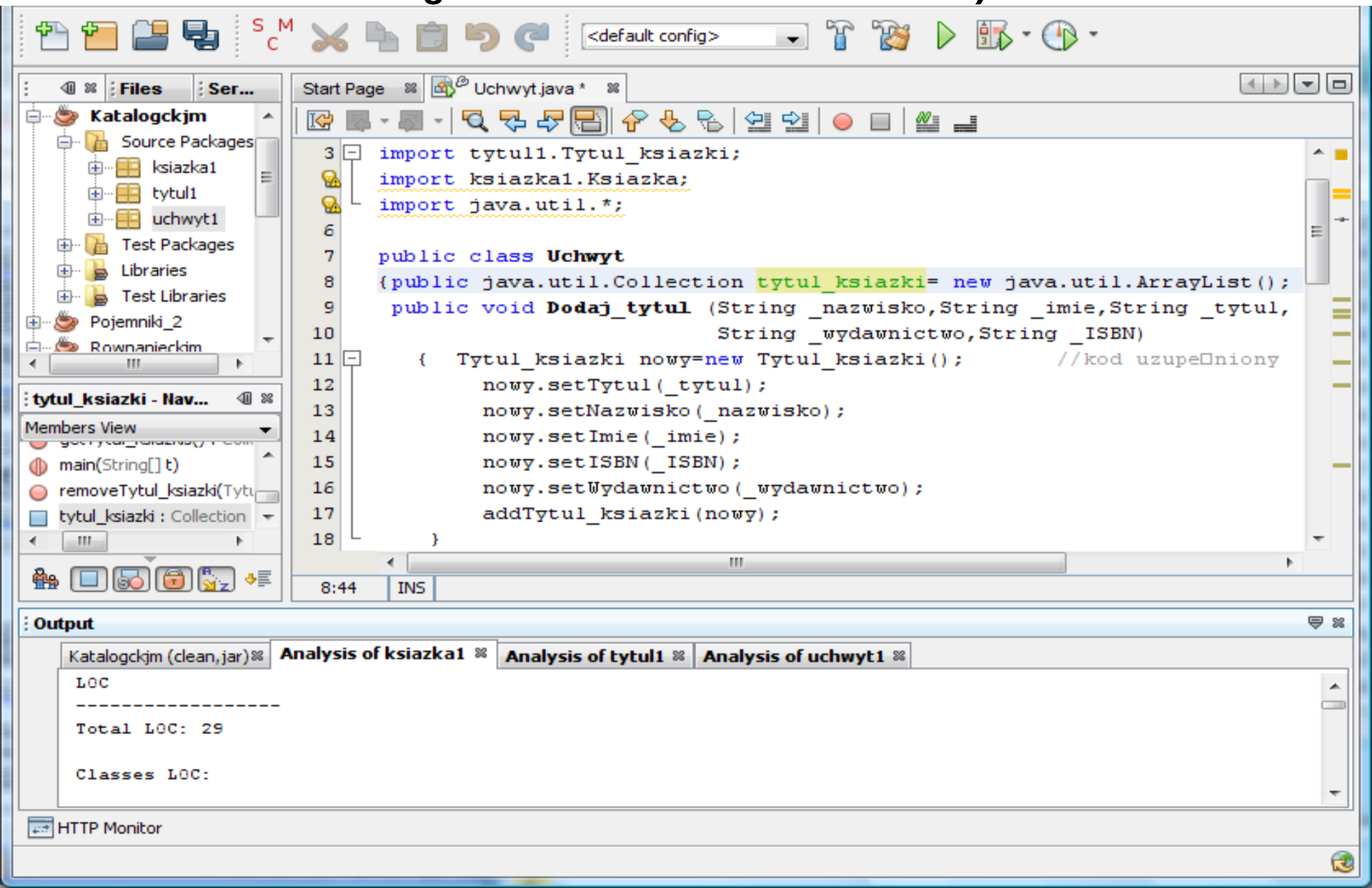
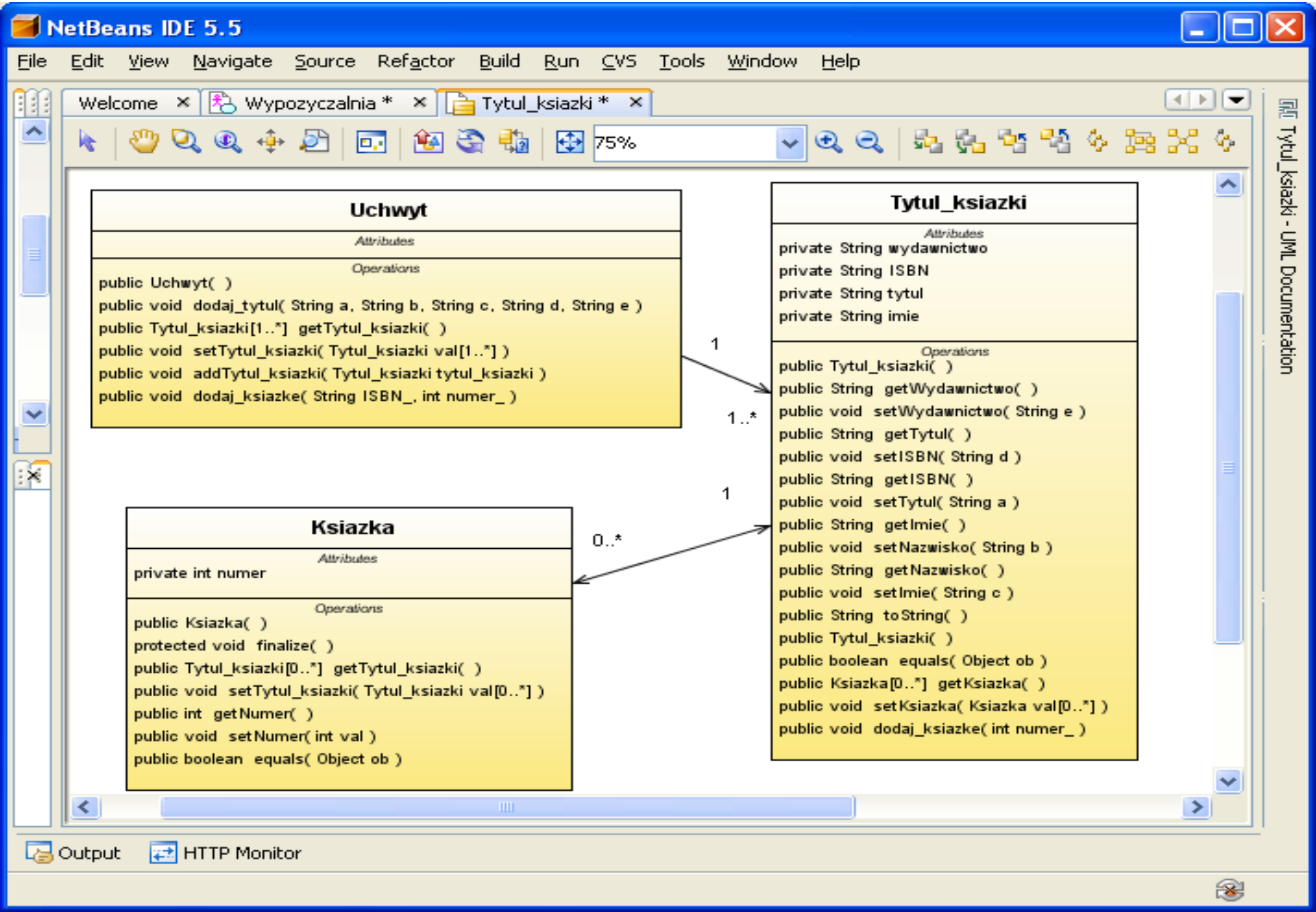
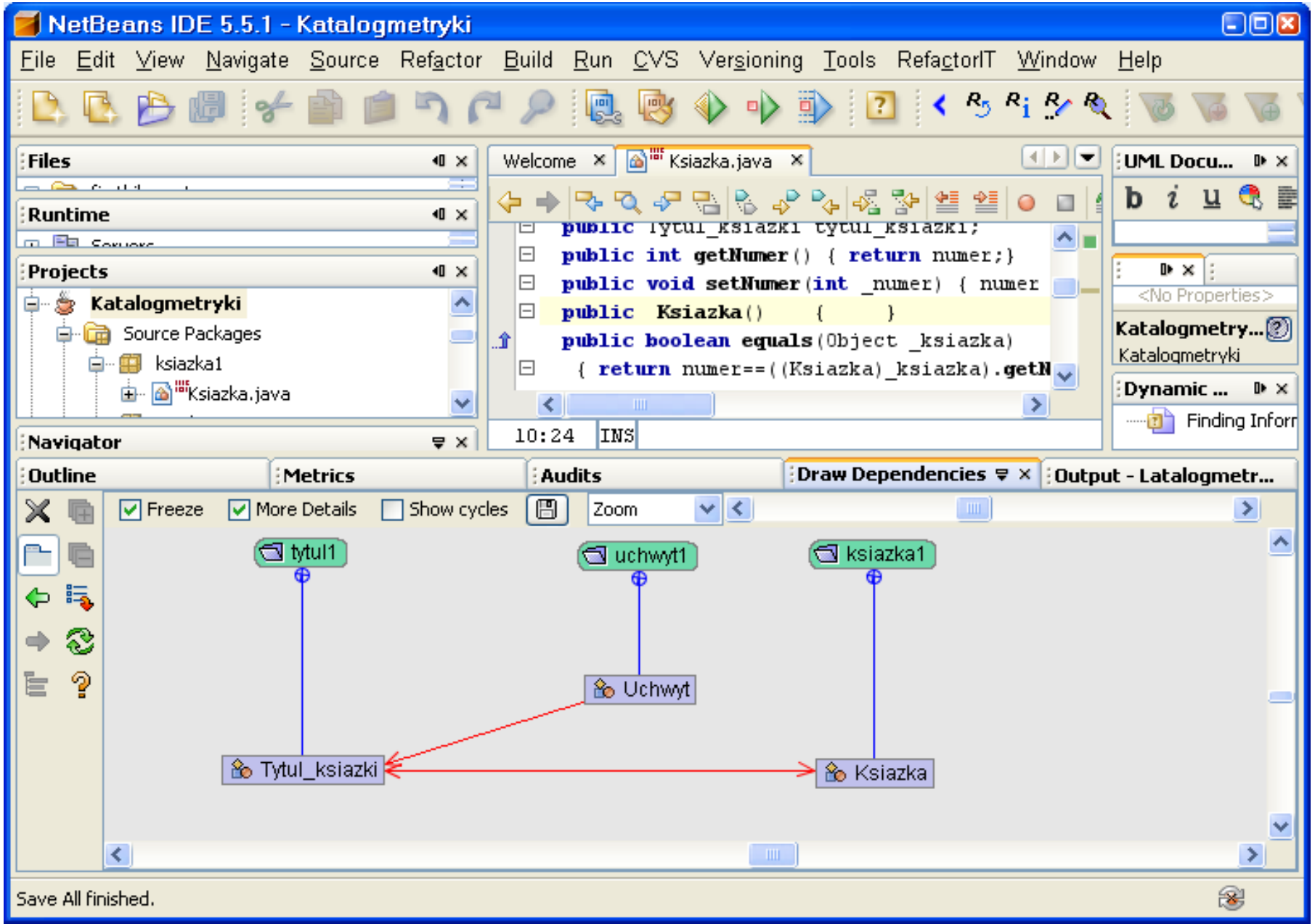


Diagram klas badanego programu



Schemat powiązań międzymodułowych do pomiaru metryk międzymodułowych



Kod źródłowy klasy Ksiazka (klasa typu Entity)

```
package ksiazka1;
import tytul1.Tytul_ksiazki;
public class Ksiazka
{  private int numer;
   public Tytul_ksiazki tytul_ksiazki;
   public  Ksiazka()                          {     }
   public int getNumer()                      { return numer;}
   public void setNumer(int _numer)           { numer = _numer;  }
   public boolean equals(Object _ksiazka)
                                              { return numer==((Ksiazka)_ksiazka).getNumer();}
   public Tytul_ksiazki getTytul_ksiazki()   {   return tytul_ksiazki;}
   public void setTytul_ksiazki(Tytul_ksiazki tytul_ksiazki)
    {  if (this.tytul_ksiazki != tytul_ksiazki)
         { if (this.tytul_ksiazki != null)
             this.tytul_ksiazki.removeKsiazka(this);
           this.tytul_ksiazki = tytul_ksiazki;
           if (tytul_ksiazki != null) tytul_ksiazki.addKsiazka(this);    }
    }
  public String toString()
   {   String pom=tytul_ksiazki.toString();
       pom+=" Numer: "+getNumer();
       return pom;  }
   }
```

SMC - Metryki kodu źródłowego Ksiazka

- **LOC** | Total LOC: 29 , Classes LOC: Ksiazka: 29 Packages LOC:
- **Lines with imports** | Total imports: 1, Classes imports: Ksiazka: 1,Packages imports:
- **Blank lines**| Total blank lines:2, Classes blank lines: Ksiazka:2, Packages blank lines:
- **Classes count** | Total classes: 1, Packages with the biggest number of classes:
- **Methods count** | Total methods: 6, Classes with the biggest number of methods: Ksiazka:6
- **Cyclomatic complexity** | Average cyclomatic complexity: 1.5
  Methods with the highest cyclomatic complexity:
  Ksiazka::setTytul_ksiazki: 4, Ksiazka::getNumer: 1, Ksiazka::toString: 1
  Ksiazka::setNumer: 1, Ksiazka::getTytul_ksiazki: 1
- **LCOM**
- **Average LCOM 1**: 0, Classes with the highest LCOM 1: Ksiazka: 0
  Packages with the highest average LCOM 1:
- **Average LCOM 2**: 0.4166666666666663,
  Classes with the highest LCOM 2:Ksiazka: 0.4166666666666663
  Packages with the highest average LCOM 2:
- **Average LCOM 3**: 0.5, Classes with the highest LCOM 3: Ksiazka: 0.5
  Packages with the highest average LCOM 3:
- **Average LCOM 4**: 2, Classes with the highest LCOM 4:Ksiazka: 2
  Packages with the highest average LCOM 4:

Kod źródłowy klasy Tytul_ksiazki (klasa typu Entity)

```
package tytul1;
import ksiazka1.Ksiazka;
public class Tytul_ksiazki
 {  private String wydawnictwo;
    private String ISBN;
    private String tytul;
    private String nazwisko;
    private String imie;
    public java.util.Collection ksiazka =  new java.util.ArrayList();
     public Tytul_ksiazki()                                          { }
     public String getWydawnictwo()                                  { return wydawnictwo;  }

     public void setWydawnictwo(String _wydawnictwo)      { wydawnictwo = _wydawnictwo;    }
     public String getISBN()                                         { return ISBN;      }
     public void setISBN(String _ISBN)                               { ISBN = _ISBN;    }
     public String getTytul()                                        { return tytul;       }
     public void setTytul(String _tytul)                             { tytul = _tytul;   }
     public String getNazwisko()                                     { return nazwisko; }
     public void setNazwisko(String _nazwisko)                       { nazwisko = _nazwisko;}
     public String getImie()                                         { return imie;    }
     public void setImie(String _imie)                               { imie = _imie;   }
     public boolean equals(Object tytul_ksiazki)
         {    boolean a;
              a = ISBN.equals(((Tytul_ksiazki)tytul_ksiazki).getISBN());
              //System.out.println(a);
               return a; }
```

Kod źródłowy klasy Tytul_ksiazki cd

```
public String toString()
    {   String pom="Tytul: "+getTytul();
         pom+=" Autor:"+getNazwisko() +" "+getImie();
         pom+=" ISBN: "+getISBN();
         pom+=" Wydawnictwo:"+getWydawnictwo();
        return pom;
    }
public void Dodaj_ksiazke(int _numer)
      {       Ksiazka nowa= new Ksiazka();
              if (nowa != null)
               { nowa.setNumer(_numer);
                 addKsiazka(nowa);  }
      }
public java.util.Collection getKsiazkas()              {  return ksiazka;   }

public void addKsiazka(Ksiazka ksiazka)
      { if (!this.ksiazka.contains(ksiazka))
              {  this.ksiazka.add(ksiazka);
                 ksiazka.setTytul_ksiazki(this); }
     }
public void removeKsiazka(Ksiazka ksiazka)
        {    boolean removed = this.ksiazka.remove(ksiazka);
             if (removed)
                   ksiazka.setTytul_ksiazki((Tytul_ksiazki)null);     }
}
```

SMC - Metryki kodu źródłowego Tytul_ksiazki (klasa typu Entity)

- **LOC**| Total LOC: 72, Classes LOC: Tytul_ksiazki: 72, Packages LOC:
- **Lines with imports**|Total imports: 1, Classes imports:Tytul_ksiazki: 1, Packages imports:
- **Blank lines**|Total blank lines:9, Classes blank lines:Tytul_ksiazki:9, Packages blanklines:
- **Classes count**|Total classes: 1, Packages with the biggest number of classes:
- **Methods count**|Total methods:16,
  Classes with the biggest number of methods:Tytul_ksiazki: 16
- **Cyclomatic complexity**| Average cyclomatic complexity: 1.1875
  Methods with the highest cyclomatic complexity:
  Tytul_ksiazki::removeKsiazka: 2, Tytul_ksiazki::Dodaj_ksiazke: 2
  Tytul_ksiazki::addKsiazka: 2, Tytul_ksiazki::getKsiazkas: 1, Tytul_ksiazki::toString: 1
- **LCOM**
- **Average LCOM 1**: 144
  Classes with the highest LCOM 1:Tytul_ksiazki:144,
  Packages with the highest average LCOM 1:
- **Average LCOM 2**: 0.7916666666666666,
  Classes with the highest LCOM 2: Tytul_ksiazki: 0.7916666666666666
  Packages with the highest average LCOM 2:
- **Average LCOM 3:** 0.8444444444444444
  Classes with the highest LCOM 3: Tytul_ksiazki: 0.8444444444444444
  Packages with the highest average LCOM 3:
- **Average LCOM 4**: 6, Classes with the highest LCOM 4: Tytul_ksiazki: 6
  Packages with the highest average LCOM 4:

Kod źródłowy klasy Uchwyt pełniącej rolę fasady

```
import tytul1.Tytul_ksiazki;
import ksiazka1.Ksiazka;
import java.util.*;
public class Uchwyt
{ public java.util.Collection tytul_ksiazki= new java.util.ArrayList();
  public void Dodaj_tytul (String _nazwisko,String _imie,String _tytul,
                                  String _wydawnictwo,String _ISBN)
   {  Tytul_ksiazki nowy=new Tytul_ksiazki();
       nowy.setTytul(_tytul);
       nowy.setNazwisko(_nazwisko);
       nowy.setImie(_imie);
       nowy.setISBN(_ISBN);
       nowy.setWydawnictwo(_wydawnictwo);
       addTytul_ksiazki(nowy);
    }
public java.util.Collection getTytul_ksiazkis()
   {    return tytul_ksiazki;  }

public void addTytul_ksiazki(Tytul_ksiazki tytul_ksiazki)
   {    if (! this.tytul_ksiazki.contains(tytul_ksiazki))
                  this.tytul_ksiazki.add(tytul_ksiazki);  }
```

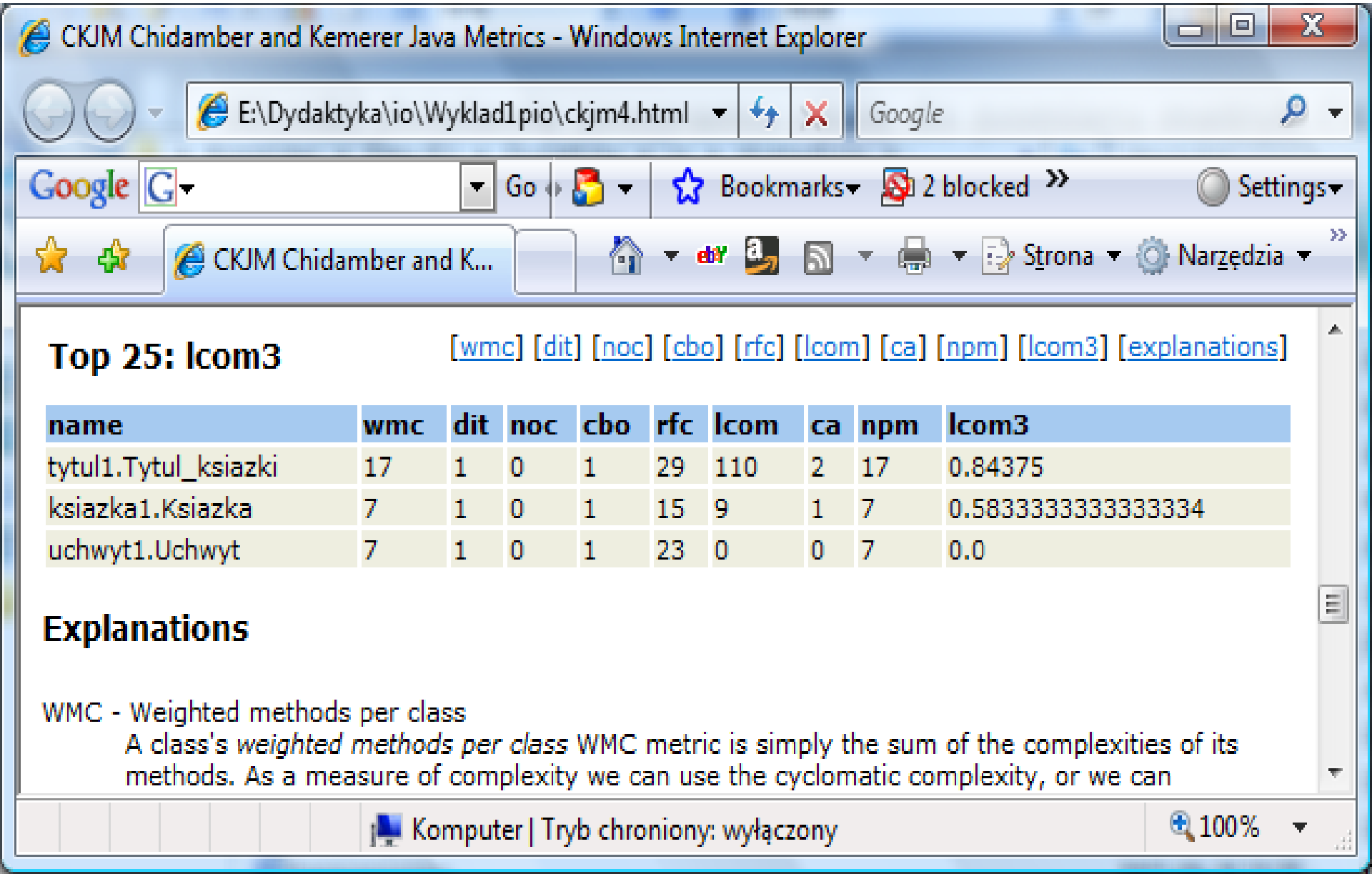
Kod źródłowy klasy Uchwyt pełniącej rolę fasady cd

```
public void removeTytul_ksiazki(Tytul_ksiazki tytul_ksiazki)
  {   this.tytul_ksiazki.remove(tytul_ksiazki); }

public void Dodaj_ksiazke(String _ISBN, int _numer)
 { Tytul_ksiazki pom=new Tytul_ksiazki();
    pom.setISBN(_ISBN);
    int idx=((java.util.ArrayList)tytul_ksiazki).indexOf(pom);
    if (idx!=-1)
      { Tytul_ksiazki pom1= (Tytul_ksiazki)((java.util.ArrayList)tytul_ksiazki).get(idx);
        pom1.Dodaj_ksiazke(_numer); }
     }

public static void main(String t[])          //kod dodany
    { Uchwyt ap=new Uchwyt();
      ap.Dodaj_tytul("1","1","1","1","1");
      ap.Dodaj_tytul("2","2","2","2","2");
      ap.Dodaj_tytul("2","2","2","2","2");
      String lan=ap.tytul_ksiazki.toString();
      System.out.println(lan);
      ap.Dodaj_ksiazke("1",1);
      ap.Dodaj_ksiazke("1",2);
      ap.Dodaj_ksiazke("1",2);
      ap.Dodaj_ksiazke("2",1);
    }
}
```

SMC - Metryki kodu źródłowego klasy Uchwyt

- **LOC** |Total LOC: 57, Classes LOC: Uchwyt: 57, Packages LOC:
- **Lines with imports**| Total imports: 3, Classes imports: Uchwyt: 3, Packages imports:
- **Blank lines** |Total blank lines: 7, Classes blank lines: Uchwyt: 7,Packages blank lines:
- **Classes count** |Total classes: 1, Packages with the biggest number of classes:
- **Methods count** | Total methods: 6, Classes with the biggest number of methods: Uchwyt: 6
- **Cyclomatic complexity**| Average cyclomatic complexity: 1.3333333333333333
  Methods with the highest cyclomatic complexity:
  Uchwyt::Dodaj_ksiazke: 2, Uchwyt::addTytul_ksiazki: 2, Uchwyt::main: 1
  Uchwyt::Dodaj_tytul: 1, Uchwyt::getTytul_ksiazkis: 1
- **LCOM**
- **Average LCOM 1**: 0, Classes with the highest LCOM 1:Uchwyt: 0
  Packages with the highest average LCOM 1:
- **Average LCOM 2**: 0.16666666666666663
  Classes with the highest LCOM 2: Uchwyt: 0.16666666666666663
  Packages with the highest average LCOM 2:
- **Average LCOM 3**: 0.2
  Classes with the highest LCOM 3: Uchwyt: 0.2,
  Packages with the highest average LCOM 3:
- **Average LCOM 4**: 2
  Classes with the highest LCOM 4: Uchwyt: 2,
  Packages with the highest average LCOM 4:

Wyniki działania programu ckjm do pomiaru metryk CK - wykonanie skryptu build za pomocą programu ant - apache-ant-1.7.1 i utworzenie raportu typu html



Top 25: lcom3

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

name	wmc	dit	noc	cbo	rfc	lcom	ca	npm	lcom3
tytul1.Tytul_ksiazki	17	1	0	1	29	110	2	17	0.84375
ksiazka1.Ksiazka	7	1	0	1	15	9	1	7	0.5833333333333334
uchwyt1.Uchwyt	7	1	0	1	23	0	0	7	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

Przykład skryptu build.xml wywołanego domyślnie przez program ant

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="changeme" default="ckjm" basedir=".">
                <!-- wskazanie na plik ckjm-1.8.extended.jar: -->
  <property name="ckjm.location" value="c:/downloads/ckjm-1.8/ckjm-1.8.extended.jar" />
                <!-- to laduje zadanie <ckjm> -->
  <taskdef name="ckjm" classname="gr.spinellis.ckjm.ant.CkjmTask">
      <classpath>
        <pathelement location="${ckjm.location}"/>
      </classpath>
  </taskdef>
              <!-- Metryki zostana zapisane w pliku ckjm4.html, mozna rowniez wybrac format xml.-->
  <target name="ckjm">
              <!-- ckjm lokalizuje pliki uzywane przez rozwijany projekt, przy zalozeniu, ze 'build/classes' jest
                        katalogiem, w ktorym sa umieszczane 'bajtkody' (*.class)-->
    <ckjm outputfile="ckjm.xml" format="xml„
          classdir="e:/dydaktyka/io/Wyklad1pio/Katalogckjm/build/classes/">
      <include name="**/*.class" />
      <exclude name="**/*Test.class" />
              <!-- ckjm lokalizuje w tej linii klasy wchodzace w sklad rozwijanego projektu, przy zalozeniu, ze
                        build/classes' jest katalogiem w ktorym sa umieszczane skompilowane klasy (*.class)-->
      <extdirs path="e:/dydaktyka/io/Wyklad1pio/Katalogckjm/build/classes/"/>
              <!-- ckjm lokalizuje biblioteki uzywane przez rozwijany projekt, przy zalozeniu, ze lib jest
                        katalogiem, w ktorym sa umieszczane biblioteki (*.jar)-->
      <extdirs path="e:/dydaktyka/io/Wyklad1pio/Katalogckjm/dist/lib"/>
    </ckjm>
    <xslt in="ckjm.xml" style="ckjm_extra.xsl" out="ckjm4.html" />
  </target>
</project>
```