

Metryki

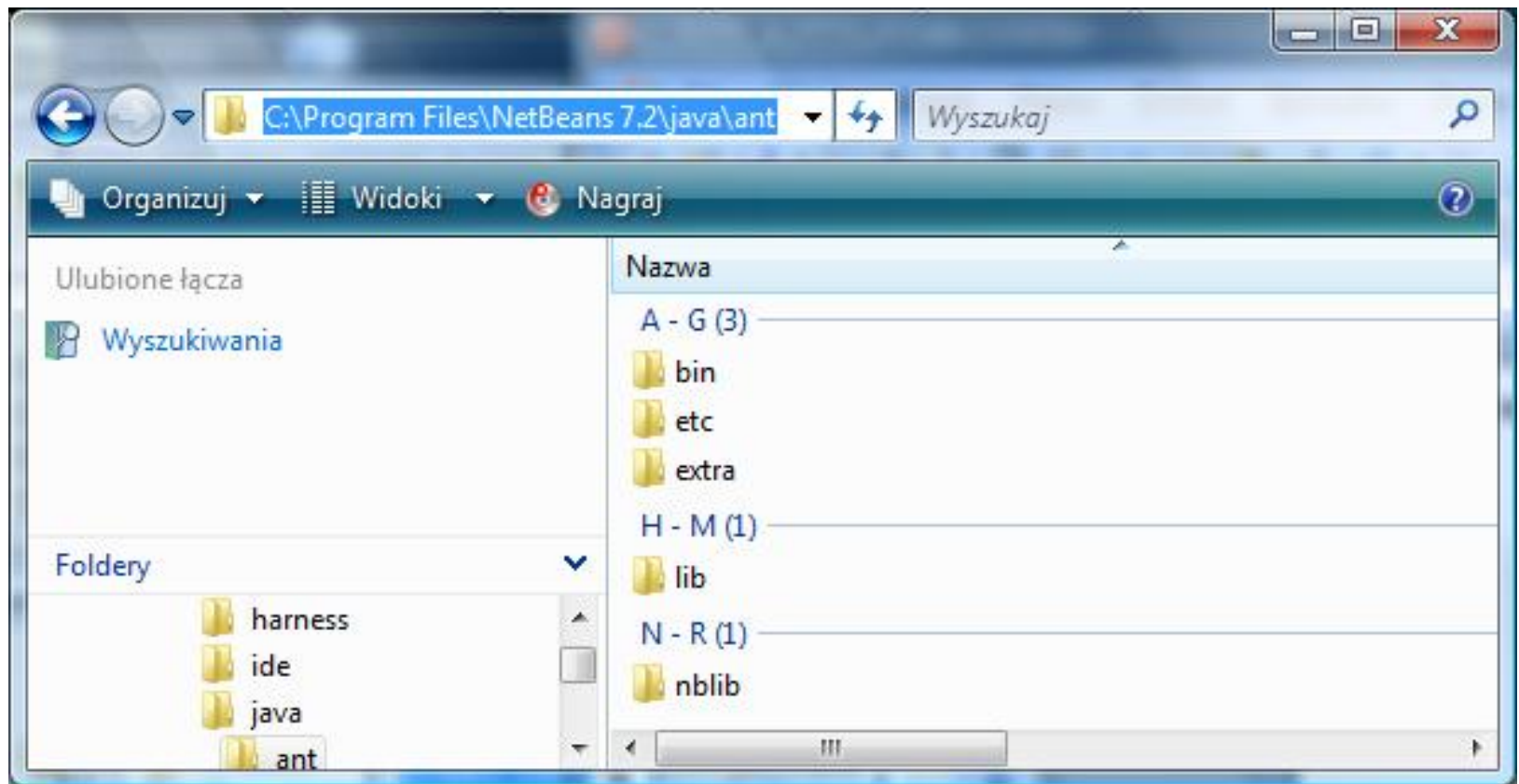
**Narzędzia do pomiaru złożoności
modułowej i międzymodułowej
oprogramowania**

autor: Zofia Kruczkiewicz

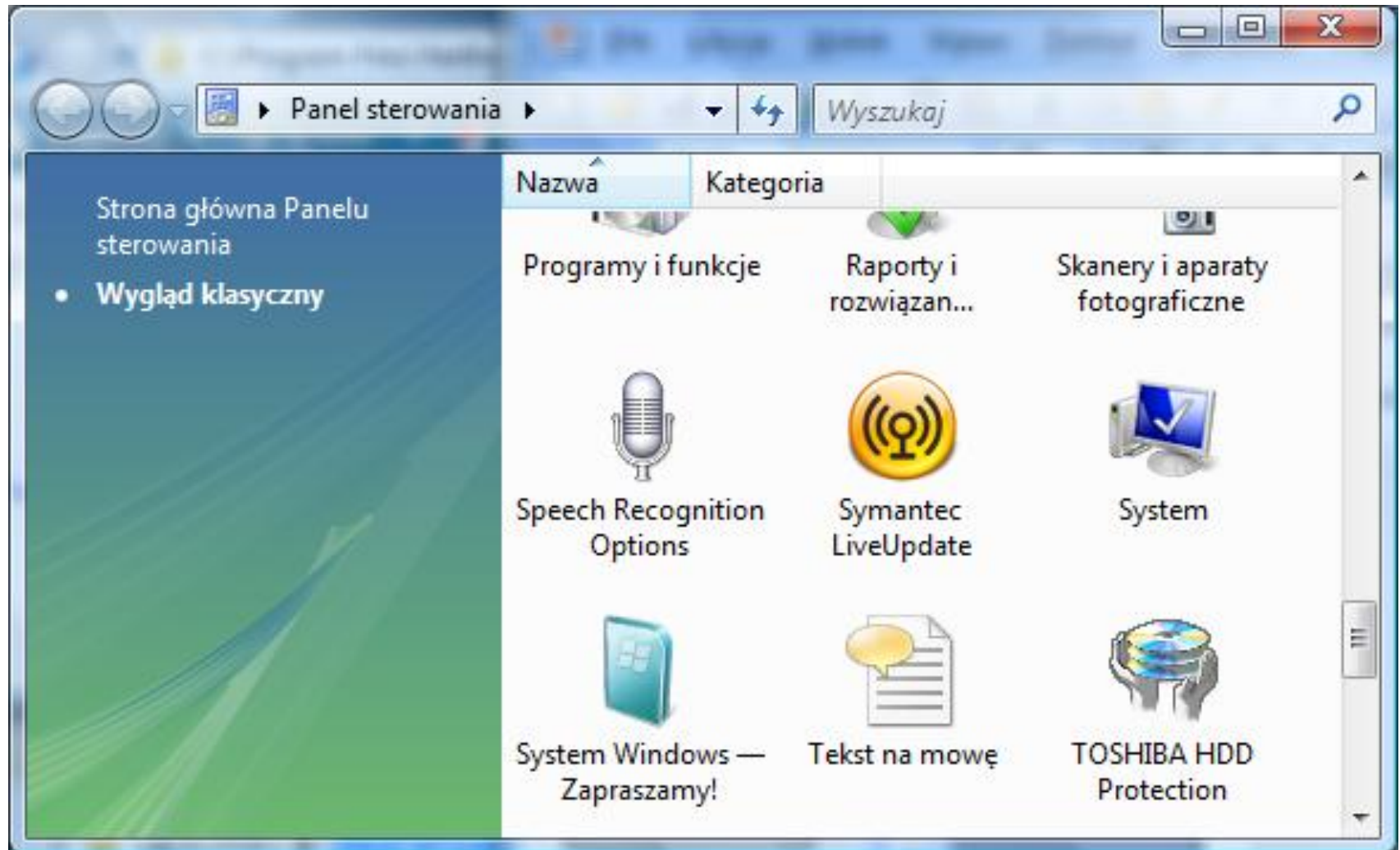
Zastosowanie narzędzi **ant** i **ckjm** do pomiaru złożoności oprogramowania

1. Wskazanie ścieżki dostępu do programu **ant**, umieszczonego w podkatalogu

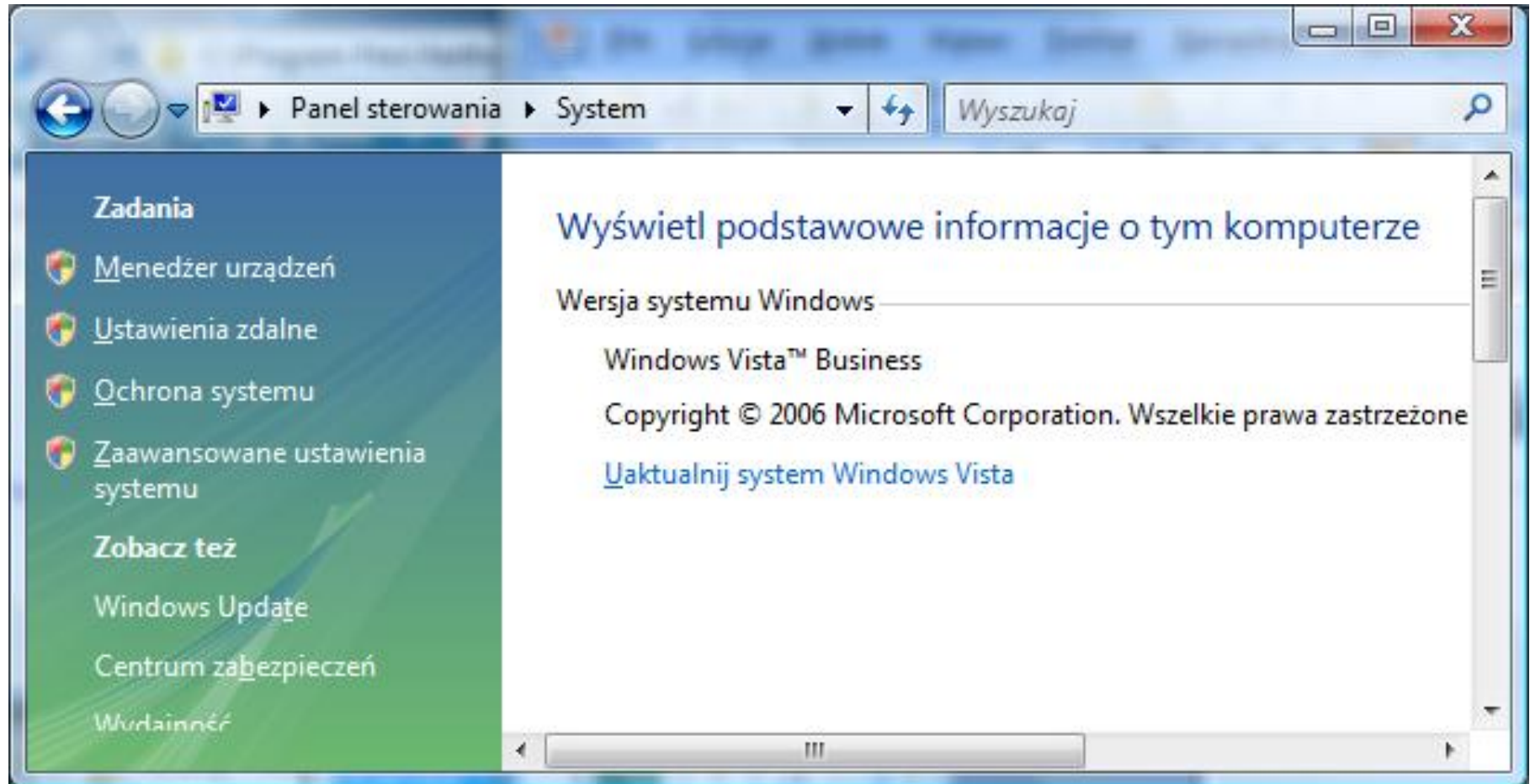
C:\Program Files\NetBeans 7.2\java\ant (instalowanego podczas instalacji NetBeans)



Dostęp do definiowania zmiennych środowiskowych SO Windows w Panelu Sterowania



Przejsięcie w Panelu Sterowania do zadań typu System



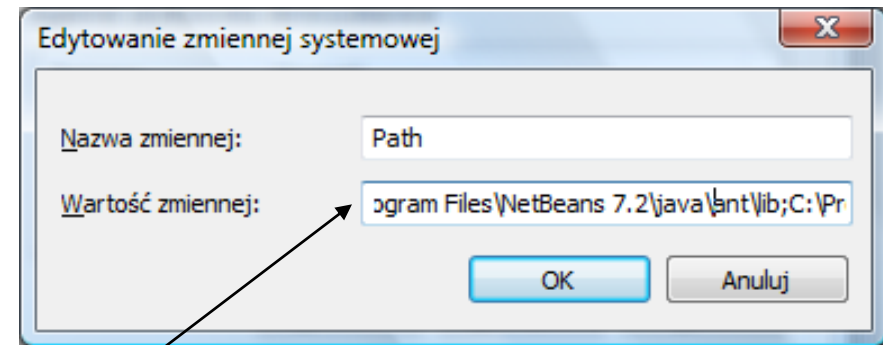
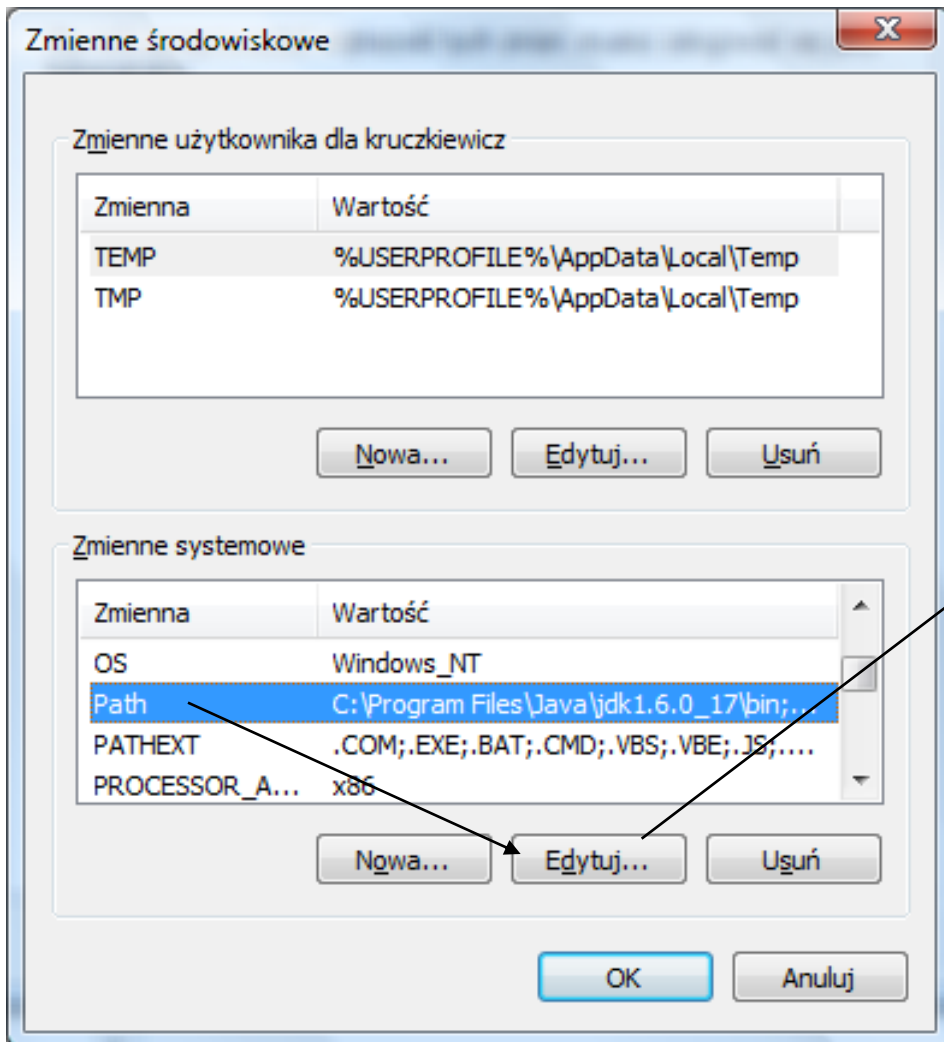
Wybór zadania **Zaawansowane ustawienia systemu**

The screenshot shows the Windows Vista Control Panel. The breadcrumb path is "Panel sterowania > System". The left sidebar lists several tasks, with "Zaawansowane ustawienia systemu" (Advanced System Settings) highlighted. An arrow points from this task to the "Zaawansowane" (Advanced) tab in the "Właściwości systemu" (System Properties) dialog box.

The "Właściwości systemu" dialog box has the "Zaawansowane" tab selected. It contains several sections with "Ustawienia..." buttons: "Wydajność" (Performance), "Profile użytkownika" (User Profiles), and "Uruchamianie i odzyskiwanie" (Startup and Recovery). At the bottom right, there is a "Zmienne środowiskowe..." (Environment Variables...) button. An arrow points from this button to the text below.

Wybór z **Zaawansowane** podzadanie **Zmienne środowiskowe**

Wybór z listy **Zmienne systemowe** zmiennej **Path** i kliknięcie na przycisk **Edytuj**



Wpisanie ścieżek dostępu do katalogów **bin** i **lib** w katalogu **ant**:

C:\Program Files\NetBeans 7.2\java\ant \bin;
C:\Program Files\NetBeans 7.2\java\ant\lib ;

Ustawienie w systemowej zmiennej środowiskowej **Path** ścieżek do podkatalogów **bin** i **lib** podkatalogu **ant** (poprzedni slajd).

```
C:\PROGRA~1\Borland\CBUILD~1\Bin;C:\PROGRA~1\Borland\CBUILD~1\Projects\Bpl;%SystemRoot%\system32;%SystemRoot%;%SystemRoot%\System32\Wbem;c:\Program Files\Microsoft SQL Server\90\Tools\binn\; C:\Program Files\NetBeans 7.2\java\ant\bin; C:\Program Files\NetBeans 7.2\java\ant\lib ;
```


2. Instalacja narzędzia **ckjm**

Linki do strony z programem CKJM

Strona główna:

<http://www.spinellis.gr/sw/ckjm>

Link do pobrania kodu narzędzia:

<http://www.spinellis.gr/sw/ckjm/ckjm-1.9.zip>

Opis narzędzia:

<http://www.spinellis.gr/sw/ckjm/doc/indexw.html>

Budowa skryptu **build.xml** dla programu **ant**, zawierającego informacje, gdzie znajduje się narzędzie **ckjm**, jaki program należy zmierzyć oraz sposób prezentacji wyników pomiaru

Utworzenie i wywołanie skryptu **build.xml**

- Należy napisać skrypt **build.xml** wg wzoru podanego na następnym slajdzie
- Należy umieścić skrypt **build.xml** w wybranym katalogu,
- Należy wywołać program **ant** w wybranym katalogu, gdzie znajduje się plik **build.xml**
- Wynik pomiaru w postaci pliku typu **html** warto umieszczać w wybranym katalogu, gdzie znajduje się skrypt **build.xml**. W skrypcie podanym na następnej stronie w znaczniku **<ckjm outputfile="ckjm.xml" format="xml"** podano, że plik **ckjm.xml** zawiera wyniki pomiaru. W znaczniku **<xslt>** ten plik jest wskazany w atrybucie **in**, jako dane wejściowe przekształcone na podstawie wskazanego arkusza stylu **ckjm_extra.xsl** w atrybucie **style** na plik **ckjm.html** podany przez atrybut **out**. Brak ścieżki oznacza generowanie wyniku w katalogu, gdzie znajduje się skrypt **build.xml**

Przykład skryptu build.xml wywołanego domyślnie przez program ant dla programu typu Java Application

```
<?xml version="1.0" encoding="UTF-8"?>  
<project name="myproject" default="ckjm">
```

```
<target name="compile">  
  <!-- your compile instructions -->  
</target>
```

<!-- classname zawiera nazwę klasy głównej narzędzia ckjm, która jest wskazana za pomocą podanej ścieżki w znaczniku pathelement-->

```
<target name="ckjm" depends="compile">  
  <taskdef name="ckjm" classname="gr.spinellis.ckjm.ant.CkjmTask">  
    <classpath>  
      <pathelement location="C:/downloads/ckjm-1.9/ckjm-1.9/build/ckjm-1.9.jar"/>  
    </classpath>  
  </taskdef>
```

<!-- classdir lokalizuje pakiety z bajkodami używanymi przez projekt za pomocą podanej ścieżki – w przykładzie classes jest katalogiem, w którym są umieszczane pakiety z plikami (*.class)-->

```
<ckjm outputfile="ckjm.xml" format="xml"  
  classdir="E:\EnglishLecture\Laboratory\BT_Library1-Model2\BT_Library1\build\classes">
```

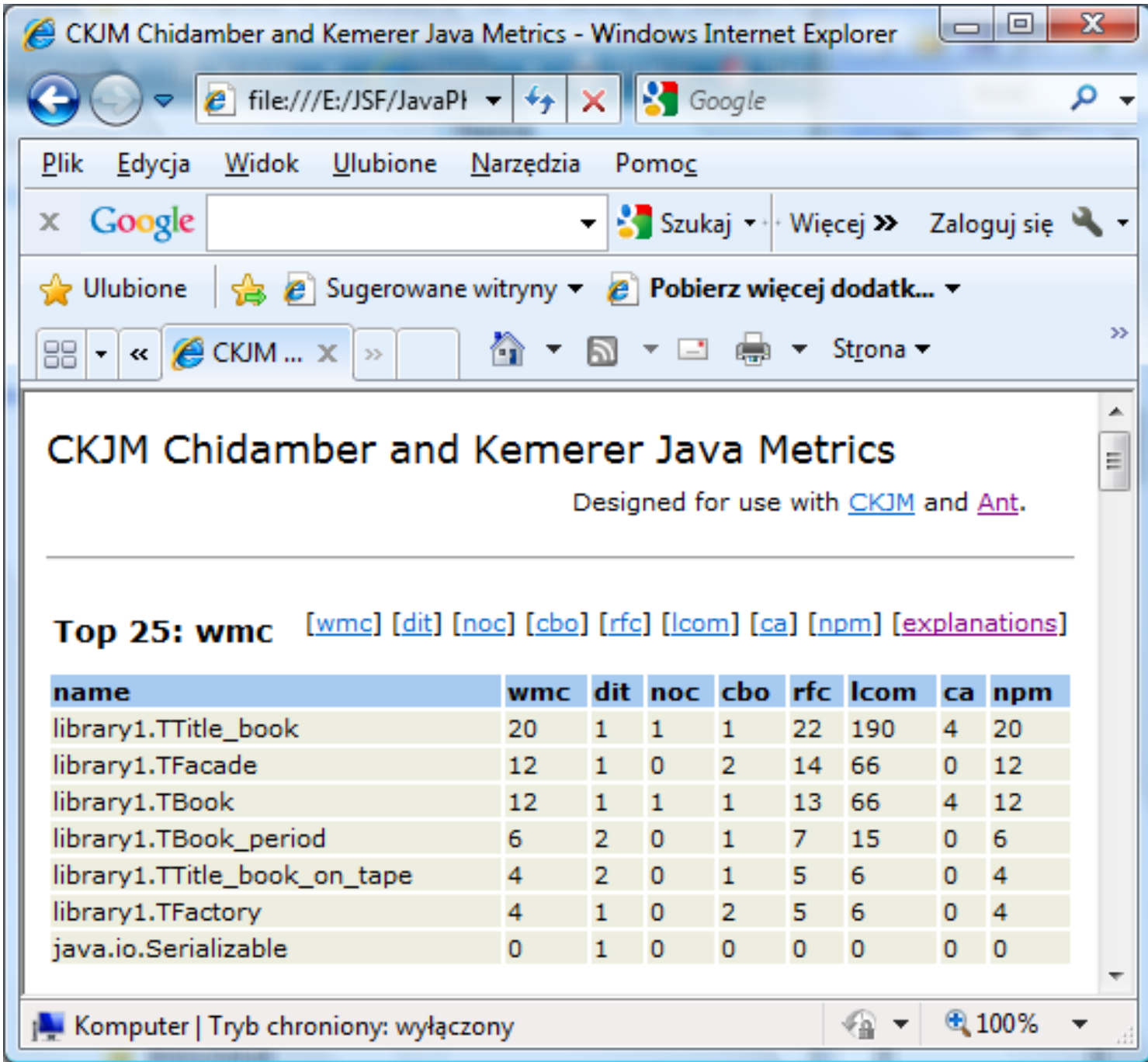
```
<include name="**/*.class" />  
<exclude name="**/*Test.class" />
```

<!-- extdirs lokalizuje biblioteki używane przez klasy projektu za pomocą podanej ścieżki - w przykładzie dist jest katalogiem, w którym są umieszczone spakowane pakiety bibliotek (*.jar)- Można dodać kolejne znaczniki extdirs zawierające różne ścieżki do pozostałych bibliotek używanych przez klasy aplikacji->

Mierzone są pliki .class, wyłączone z pomiarów są pliki *Test.class

```
<extdirs path="E:\EnglishLecture\Laboratory\BT_Library1-Model2\BT_Library1\dist"/>  
</ckjm>
```

```
<xslt in="ckjm.xml" style="C:/downloads/ckjm-1.9/ckjm-1.9/xsl/ckjm_extra.xsl" out="ckjm.html" />  
</target>  
</project>
```



Przykład pomiaru metryk wg skryptu z poprzedniej strony

Przykład skryptu build.xml wywołanego domyślnie przez program ant dla programu typu Java Application

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="myproject" default="ckjm">
```

```
<target name="compile">
  <!-- your compile instructions -->
</target>
```

<!-- classname zawiera nazwę klasy głównej narzędzia ckjm, która jest wskazana za pomocą podanej ścieżki w znaczniku pathelement-->

```
<target name="ckjm" depends="compile">
  <taskdef name="ckjm" classname="gr.spinellis.ckjm.ant.CkjmTask">
    <classpath>
      <pathelement location="C:/downloads/ckjm-1.9/ckjm-1.9/build/ckjm-1.9.jar"/>
    </classpath>
  </taskdef>
```

<!-- classdir lokalizuje pakiety z bajkodami używanymi przez projekt za pomocą podanej ścieżki – w przykładzie classes jest katalogiem, w którym są umieszczane pakiety z plikami (*.class)-->

```
<ckjm outputfile="ckjm.xml" format="xml"
  classdir="E:\EnglishLecture\Laboratory\BT_Library1-Model2\BT_Library1\build\classes">
  <include name="**/*.class" />
  <exclude name="**/*Test.class" />
```

<!-- extdirs lokalizuje biblioteki używane przez klasy projektu za pomocą podanej ścieżki - w przykładzie dist jest katalogiem, w którym są umieszczone spakowane pakiety bibliotek (*.jar)- Można dodać kolejne znaczniki extdirs zawierające różne ścieżki do pozostałych bibliotek używanych przez klasy aplikacji->

```
<!--<extdirs path="E:\EnglishLecture\Laboratory\BT_Library1-Model2\BT_Library1\dist"/>→
</ckjm>
```

```
<xslt in="ckjm.xml" style="C:/downloads/ckjm-1.9/ckjm-1.9/xsl/ckjm_extra.xsl" out="ckjm.html" />
</target>
</project>
```

CKJM Chidamber and Kemerer Java Metrics - Windows Internet Explorer

E:\JSF\JavaPK\Metr Google

Plik Edycja Widok Ulubione Narzędzia Pomoc

Google Szukaj Więcej >> Zaloguj się

Ulubione Sugerowane witryny Pobierz więcej dodatk...

CKJM ...

CKJM Chidamber and Kemerer Java Metrics

Designed for use with [CKJM](#) and [Ant](#).

Top 25: wmc [\[wmc\]](#) [\[dit\]](#) [\[noc\]](#) [\[cbo\]](#) [\[rfc\]](#) [\[lcom\]](#) [\[ca\]](#) [\[npm\]](#) [\[explanations\]](#)

| name | wmc | dit | noc | cbo | rfc | lcom | ca | npm |
|------------------------------|-----|-----|-----|-----|-----|------|----|-----|
| library1.TTitle_book | 20 | 1 | 1 | 1 | 22 | 190 | 4 | 20 |
| library1.TFacade | 12 | 1 | 0 | 2 | 14 | 66 | 0 | 12 |
| library1.TBook | 12 | 1 | 1 | 1 | 13 | 66 | 4 | 12 |
| library1.TBook_period | 6 | 0 | 0 | 1 | 7 | 15 | 0 | 6 |
| library1.TTitle_book_on_tape | 4 | 0 | 0 | 1 | 5 | 6 | 0 | 4 |
| library1.TFactory | 4 | 1 | 0 | 2 | 5 | 6 | 0 | 4 |
| java.io.Serializable | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

Komputer | Tryb chroniony: wyłączony 100%

Przykład pomiaru metryk wg skryptu z poprzedniego slajdu, przy usuniętym znaczniku **extdirs**, która wskazywała klasy bazowe – wynik pomiaru głębokości dziedziczenia **dit** zawiera błędy (wartość 0), gdyż najmniejsza głębokość dziedziczenia w Javie to 1 (dziedziczenie po klasie Object)

Przykład skryptu build.xml wywołanego domyślnie przez program ant dla programu typu Java Application- **zalecane dla pozostałych typów aplikacji**

```
<?xml version="1.0" encoding="UTF-8"?>  
<project name="myproject" default="ckjm">
```

```
<target name="compile">  
  <!-- your compile instructions -->  
</target>
```

<!-- classname zawiera nazwę klasy głównej narzędzia ckjm, która jest wskazana za pomocą podanej ścieżki w znaczniku pathelement-->

```
<target name="ckjm" depends="compile">  
  <taskdef name="ckjm" classname="gr.spinellis.ckjm.ant.CkjmTask">  
    <classpath>  
      <pathelement location="C:/downloads/ckjm-1.9/ckjm-1.9/build/ckjm-1.9.jar"/>
```

```
    </classpath>  
  </taskdef>
```

<!-- classdir lokalizuje wszystkie pakiety z bajkodami używanymi przez projekt za pomocą podanej ścieżki – w przykładzie dist jest katalogiem, w którym są umieszczane spakowane pakiety z plikami (*.lar)-->

```
<ckjm outputfile="ckjm.xml" format="xml"  
  classdir="E:\EnglishLecture\Laboratory\BT_Library1-Model2\BT_Library1\dist">  
  <include name="**/*.class" />  
  <exclude name="**/*Test.class" />  
</ckjm>
```

```
<xslt in="ckjm.xml" style="C:/downloads/ckjm-1.9/ckjm-1.9/xsl/ckjm_extra.xsl"  
  out="ckjm.html" />
```

```
</target>  
</project>
```


CKJM Chidamber and Kemerer Java Metrics - Windows Internet Explorer

E:\JSF\JavaPK\Metr Google

Plik Edycja Widok Ulubione Narzędzia Pomoc

Google Szukaj Więcej Zaloguj się

Ulubione Sugerowane witryny Pobierz więcej dodatk...

CKJM ... Strona

CKJM Chidamber and Kemerer Java Metrics

Designed for use with [CKJM](#) and [Ant](#).

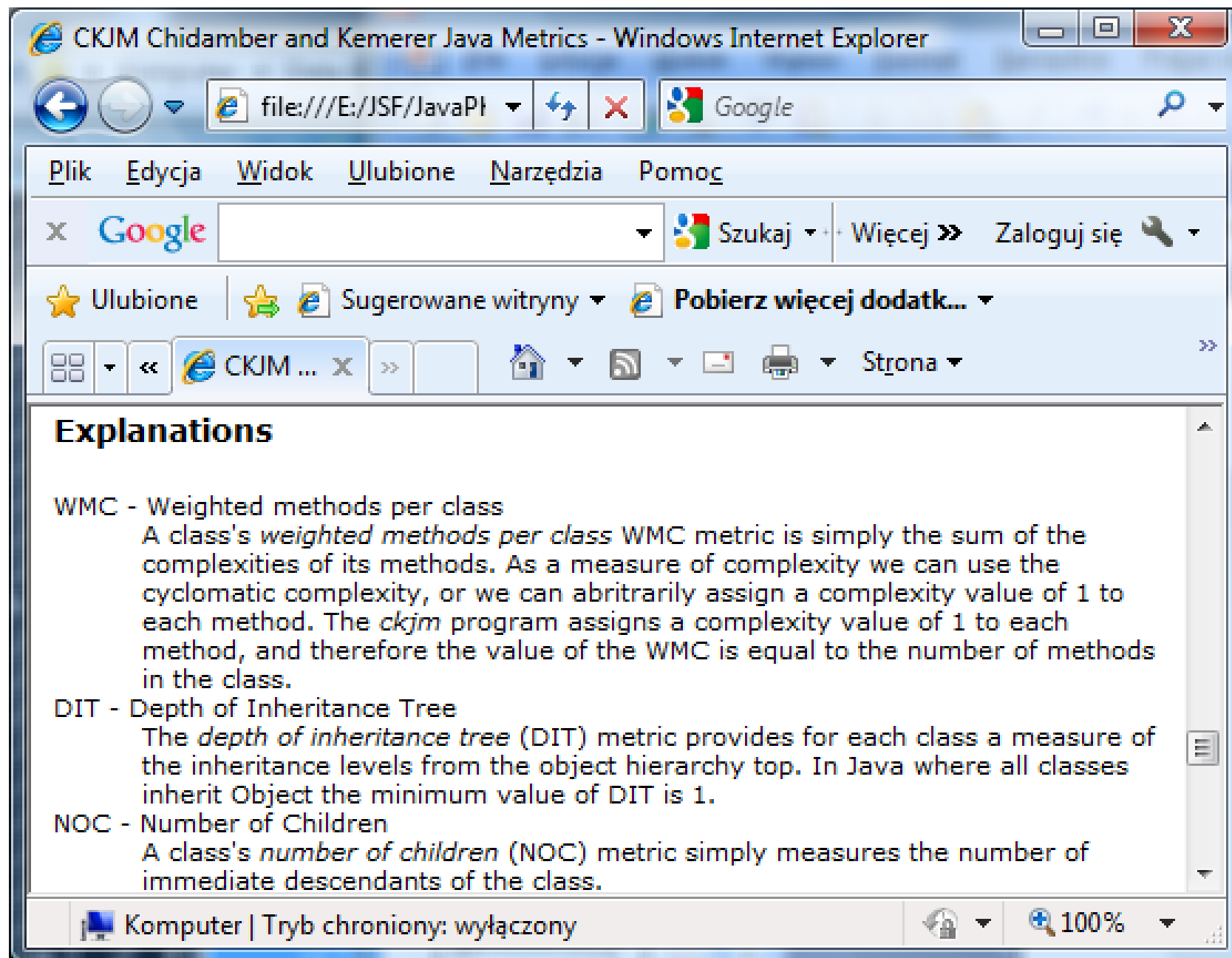
Top 25: wmc [\[wmc\]](#) [\[dit\]](#) [\[noc\]](#) [\[cbo\]](#) [\[rfc\]](#) [\[lcom\]](#) [\[ca\]](#) [\[npm\]](#) [\[explanations\]](#)

name	wmc	dit	noc	cbo	rfc	lcom	ca	npm
library1.TTitle_book	20	1	1	1	22	190	4	20
library1.TFacade	12	1	0	2	14	66	0	12
library1.TBook	12	1	1	1	13	66	4	12
library1.TBook_period	6	2	0	1	7	15	0	6
library1.TTitle_book_on_tape	4	2	0	1	5	6	0	4
library1.TFactory	4	1	0	2	5	6	0	4
java.io.Serializable	0	1	0	0	0	0	0	0

Komputer | Tryb chroniony: wyłączony 100%

Poprawne wyniki pomiarów, gdy w ścieżce zdefiniowanej w atrybucie **classdir** wskazano wszystkie możliwe biblioteki wykorzystywane w aplikacji (skrypt **build.xml** z poprzedniego slajdu)

Wybór zakładki Explanations zawierającej definicje metryk



CKJM Chidamber and Kemerer Java Metrics - Windows Internet Explorer

file:///E:/JSF/JavaPl | Google

Plik Edycja Widok Ulubione Narzędzia Pomoc

Google Szukaj Więcej >> Zaloguj się

Ulubione Sugerowane witryny Pobierz więcej dodatk...

CKJM ...

Explanations

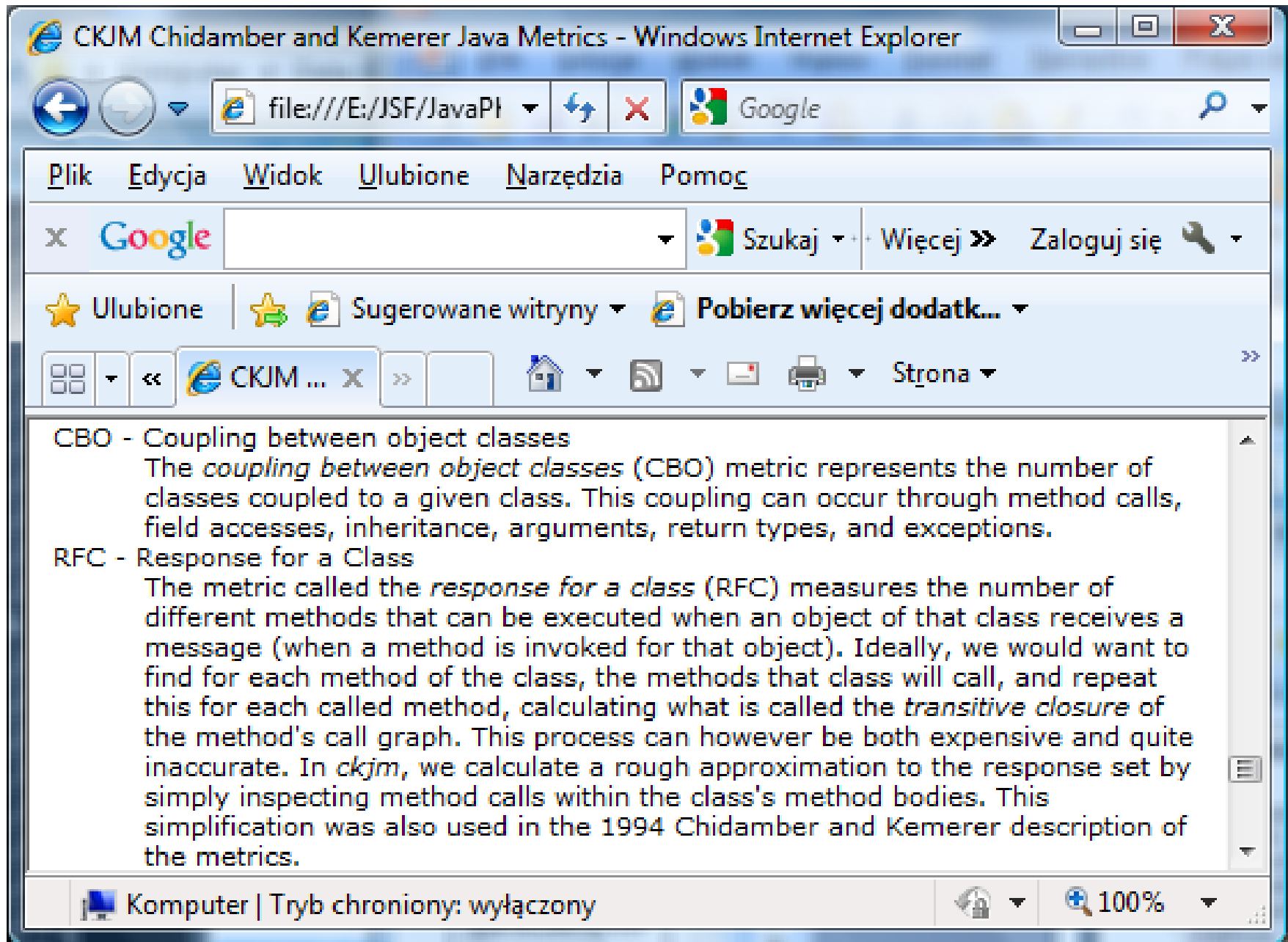
WMC - Weighted methods per class
A class's *weighted methods per class* WMC metric is simply the sum of the complexities of its methods. As a measure of complexity we can use the cyclomatic complexity, or we can arbitrarily assign a complexity value of 1 to each method. The *ckjm* program assigns a complexity 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 levels from the object hierarchy top. In Java where all classes inherit Object the minimum value of DIT is 1.

NOC - Number of Children
A class's *number of children* (NOC) metric simply measures the number of immediate descendants of the class.

Komputer | Tryb chroniony: wyłączony | 100%

Wybór zakładki Explanations zawierającej definicje metryk (cd)



The screenshot shows a Windows Internet Explorer browser window. The title bar reads "CKJM Chidamber and Kemerer Java Metrics - Windows Internet Explorer". The address bar contains the file path "file:///E:/JSF/JavaPl". The menu bar includes "Plik", "Edycja", "Widok", "Ulubione", "Narzędzia", and "Pomoc". The search bar shows "Google" and "Szukaj". The toolbar includes "Ulubione", "Sugerowane witryny", "Pobierz więcej dodatk...", and "Strona". The main content area displays two definitions:

CBO - Coupling between object classes
The *coupling between object classes* (CBO) metric represents the number of classes coupled to a given class. This coupling can occur through method calls, field accesses, inheritance, arguments, return types, and exceptions.

RFC - Response for a Class
The metric called the *response for a class* (RFC) measures the number of different methods that can be executed when an object of that class receives a message (when a method is invoked for that object). Ideally, we would want to find for each method of the class, the methods that class will call, and repeat this for each called method, calculating what is called the *transitive closure* of the method's call graph. This process can however be both expensive and quite inaccurate. In *ckjm*, we calculate a rough approximation to the response set by simply inspecting method calls within the class's method bodies. This simplification was also used in the 1994 Chidamber and Kemerer description of the metrics.

The status bar at the bottom shows "Komputer | Tryb chroniony: wyłączony" and a zoom level of "100%".

Wybór zakładki Explanations zawierającej definicje metryk (cd)

The screenshot shows a Windows Internet Explorer browser window. The title bar reads "CKJM Chidamber and Kemerer Java Metrics - Windows Internet Explorer". The address bar contains the file path "file:///E:/JSF/JavaPt". The browser's menu bar includes "Plik", "Edycja", "Widok", "Ulubione", "Narzędzia", and "Pomoc". The search bar shows "Google" and "Szukaj". The toolbar includes "Ulubione", "Sugerowane witryny", and "Pobierz więcej dodatk...". The main content area displays the following text:

LCOM - Lack of cohesion in methods
A class's *lack of cohesion in methods* (LCOM) metric counts the sets of methods in a class that are not related through the sharing of some of the class's fields. The original definition of this metric (which is the one used in *ckjm*) considers all pairs of a class's methods. In some of these pairs both methods access at least one common field of the class, while in other pairs the two methods do not share any common field accesses. The lack of cohesion in methods is then calculated by subtracting from the number of method pairs that don't share a field access the number of method pairs that do. Note that subsequent definitions of this metric used as a measurement basis the number of disjoint graph components of the class's methods. Others modified the definition of connectedness to include calls between the methods of the class. The program *ckjm* follows the original (1994) definition by Chidamber and Kemerer.

Ca - Afferent couplings
A class's *afferent couplings* is a measure of how many other classes use the specific class. *Ca* is calculated using the same definition as that used for calculating CBO.

NPM - Number of Public Methods
The NPM metric simply counts all the methods in a class that are declared as public. It can be used to measure the size of an API provided by a package.

The status bar at the bottom shows "Komputer | Tryb chroniony: wyłączony" and a zoom level of "100%".

Przykład metryk CK trzech systemów

System analyzed	Java	Java	C++
Classes	46	1000	1617
Lines	50,000	300,000	500,000
Quality	"Low"	"High"	"Medium"
CBO	2.48	1.25	2.09
LCOM1	447.65	78.34	113.94
RFC	80.39	43.84	28.60
NOC	0.07	0.35	0.39
DIT	0.37	0.97	1.02
WMC	45.7	11.10	23,97 ₂₀