

# Narzędzia programisty Java

Jacek Bzdak

2 kwietnia 2012

Używanie IDE

Debugging kodu

Findbugs — przykład narzędzia do statycznej analizy kodu

# Na wstępie

- ▶ Gdybym mówił Państwu rzeczy oczywiste to proszę mi wybaczyć, mam nadzieję że dla części osób na sali te rzeczy oczywiste nie są.

## Import do workspace eclipse

Jedną ze standardowych czynności jest importowanie kodu z poza workspace do workspace.

- ▶ Na przykład pobierają Państwo przykładowy kod ze strony dr. Dudy i chcą go użyć w swoim projekcie.
- ▶ Albo przynoszą mi Państwo stworzony w domu projekt i chcą go uruchomić w Eclipse.

# Import projektu eclipseowego

- ▶ Otwierają Państwo workspace
- ▶ File -> Import -> General -> Existing project into workspace

## Import projektu który nie jest eclipsowy

- ▶ Tworzą państwo nowy projekt – bądź wybierają istniejący
- ▶ File -> Import -> General -> File system
- ▶ Następnie importują Państwo zawartość katalogu src do katalogu src projektu

# Ctrl+Spacja Twoim przyjacielem

- ▶ Ctrl+Space nie tylko uzupełnia brakujące nazwy klas i zmiennych.
- ▶ Do tego jeśli uzupełniasz nazwę klasy — ona automatycznie jest importowana
- ▶ Możesz też za pomocą uzupełniania wykorzystywać szablony

# Szablony w Eclipse

`main` funkcja main

`instanceof` Sprawdzenie czy zmienna jest odpowiedniej klasy i rzutowanie

`for` Iterowanie po tablicy, kolekcji . . .



# Tworzenie własnych szablonów

- ▶ Lista szablonów znajduje się w Window -> Preferences -> Java -> Editor -> Templates
- ▶ Tam też można tworzyć szablony

# Nawigowanie po kodzie

Ctrl + Klick na klasie

Ctrl + Shift + T Otwiera typ

Class hierarhy

# Przypianie kodu źródłowego

- ▶ Java ma tą zaletę że przychodzi razem z kodem źródłowym (bardzo często)
- ▶ Tylko że Eclipse samo owego kodu nie wykrywa
- ▶ Lista szablonów znajduje się w Window -> Preferences -> Java -> Installed JREs -> Edit -> \*.rt.jar -> Source Attachment
- ▶ Plik źródłowy jest w /jdk/src.zip

# Tworzenie wykonywalnych jarów

- ▶ Eclipse (i każde inne poważne IDE) umożliwia wygenerowanie wykonywalnego pliku jar z Państwa kodu.
- ▶ File -> Export -> Executable Jar File
- ▶ Przykład

## Inne metody

- ▶ Generalnie w dojrzałych projektach bardzo rzadko IDE rządzi budowaniem projektu — z tego prostego powodu że ludzie używają różnych IDE.
- ▶ Java ma dwa główne systemy budowania plików: `ant` i `maven`.
- ▶ `Ant` pozwala za pomocą plików XML kierować poszczególnymi krokami wykonania.
- ▶ `Maven` jest systemem który zawiera w sobie wszystkie aspekty zarządzania projektem — od definiowania jego struktury po budowę strony. Ale jest piekielnie trudny.

# JavaBeans

- ▶ Specyfikacja Java Beans to specyfikacja własności w Javie.
- ▶ Własność to taki atrybut klasy który na przykład jest tylko do doczytu dla klientów danej klasy.

## Equals i hashCode

- ▶ Do sprawdzania równości obiektów w Javie służy metoda `equals`. Domyślna jej implementacja wykonuje po prostu to samo co operator `==`.
- ▶ Różnego typu kontenery w Javie wykorzystują głównie tablicę hashów — jest to trochę wydajniejsze od drzew (które wykorzystuje C++). Wymaga jednak by każdy obiekt miał metodę `hashCode` która tak jakby sumę kontrolną dla danego obiektu,
- ▶ Zależność jest taka że jeśli `a.hashCode() != b.hashCode()` to nie `a.equals(b)`, a jeśli `a.equals(b)` to `a.hashCode() != b.hashCode()`.
- ▶ Ponieważ da się to skopać idee pozwalają na definiowanie obu tych metod na raz,

# Refactoring

- ▶ Jest to proces wprowadzania zmian w projekcie/programie, w wyniku którego zasadniczo nie zmienia się funkcjonalność. Celem refaktoryzacji jest więc nie wytwarzanie nowej funkcjonalności, ale utrzymywanie odpowiedniej, wysokiej jakości organizacji systemu. (wiki)



# Co to debugging

- ▶ Legenda głosi że któregoś dnia któryś z pierwszych lampowych komputerów zaczął popełniać błędy.
- ▶ Po długich poszukiwaniach nie wykryto przepalanej lampy.
- ▶ Okazało się jednak że na jednym z do jednej z wtyczek przysmażyła się ćma. Po wymianie wtyczki komputer zaczął działać
- ▶ Stąd pojęcie bug.

# Co to debugger

## Definicja

Debugger program komputerowy służący do dynamicznej analizy innych programów, w celu odnalezienia i identyfikacji zawartych w nich błędów, zwanych z angielskiego bugami (robakami). Proces nadzorowania wykonania programu za pomocą debuggera określa się mianem debugowania. Podstawowym zadaniem debuggera jest sprawowanie kontroli nad wykonaniem kodu, co umożliwia zlokalizowanie instrukcji odpowiedzialnych za wadliwe działanie programu. Współczesne debuggery pozwalają na efektywne śledzenie wartości poszczególnych zmiennych, wykonywanie instrukcji krok po kroku czy wstrzymywanie działania programu w określonych miejscach. Debugger jest standardowym wyposażeniem większości współczesnych środowisk programistycznych. (źródło: wikipedia)

# Co to breakpoint

## Definicja

### Breakpoint

Instrukcja w programie przed której wykonaniem debugger zatrzymuje wykonanie programu.

# Okno debugera

The screenshot shows the Eclipse IDE in a debug session. The top toolbar contains various icons for debugging, including a red box around the 'Debug' button. The 'Debug' console on the left shows the application is running at line 25 of PrimeFinder3.java. The 'Variables' window on the right displays the current state of variables: ii (4), jj (3), current (9), and isPrime (true). The code editor shows the main method with a breakpoint set at the line `if(current%jj == 0); // Breakpoint tutaj`. Red annotations explain that this is where all program execution and calls are suspended, and that the variables shown in the Variables window are those currently in scope and their values.

Name	Value
ii	4
jj	3
current	9
isPrime	true

```
public static void main(String[] args) {
    int[] primes = new int[1000];
    int ii = 0;
    int jj = 2;
    int current = 2;
    while (ii < primes.length){
        boolean isPrime = true;
        current +=1;
        for (jj = 2; jj <= Math.sqrt(current); jj++){
            if(current%jj == 0); // Breakpoint tutaj
            {
                isPrime = false;
                break;
            }
        }
        if (current%100 == 0){
            System.out.println("Checked = " + current);
        }
        if(isPrime){

```

# Przykłady

- ▶ `wyklad.wyklad3.debugger.PrimeFinder` działający wyszukiwacz liczb pierwszych.
- ▶ `wyklad.wyklad3.debugger.PrimeFinder[1-3]` wersja z błędami

## Co to jest statyczna analiza kodu

- ▶ Błędy można wykryć zasadniczo w dwóch momentach: albo przed uruchomieniem programu albo w trakcie jego uruchomienia.
- ▶ Przykładem błędu który możemy wykryć w czasie kompilacji jest na przykład literówka w nazwie klasy.
- ▶ Przykładem błędu który możemy wykryć tylko w czasie wykonania jest otwarcie pliku który nie istnieje.
- ▶ Są też błędy pośrednie — takie które ujawnią się w czasie wykonania w kompilowalnym programie, ale widać już w kodzie że ten błąd będzie.

## Czy widzicie błąd w tym kodzie

```
public static void main(String[] args) {
    JFrame frame = null;
    showFrame(frame);
    frame = new JFrame();
}

/**
 * Mniej więcej tak powinno wyglądać pozycjonowanie okienka,
 * ale tego kodu tutaj nie sprawdzałem :)
 * @param frame
 */
public static void showFrame(JFrame frame){

    frame.setSize(640, 480);
    DisplayMode mode = GraphicsEnvironment.getLocalGraphicsEnvironment()
        .getDefaultScreenDevice().getDisplayMode();
    frame.setLocation((mode.getWidth()-640)/2,
        (mode.getHeight() - 480)/2);
    frame.setVisible(true);
}
```

# A FindBugs widzi

The screenshot shows the Eclipse IDE interface with the FindBugs plugin. The main editor displays the source code for `StaticAnalysisError.java`:

```
public class StaticAnalysisError {  
  
    public static void main(String[] args) {  
        JFrame frame = null;  
        showFrame(frame);  
        frame = new JFrame();  
    }  
}
```

The Bug Explorer on the left shows a list of bugs, including "Dead store to local variable", "Non-virtual method call", and "Non-virtual method". The Bug Info window is open, displaying the following details for the selected bug:

StaticAnalysisError.java: 13  
Navigation  
Non-virtual method call in wyklad.wykklad3.findbugs.StaticAnalysisError.main(String[]) passes null for nonnull parameter. Called method wyklad.wykklad3.findbugs.StaticAnalysisError.showFrame(JFrame). Value loaded from frame. Argument 1 is definitely null but must not be null.

**Bug:** Non-virtual method call in wyklad.wykklad3.findbugs.StaticAnalysisError.main(String[]) passes null for nonnull parameter of showFrame(JFrame)

A possibly-null value is passed to a nonnull method parameter. Either the parameter is annotated as a parameter that should always be nonnull, or analysis has shown that it will always be dereferenced.

**Confidence:** High, **Rank:** Scary (6)  
**Pattern:** NP\_NULL\_PARAM\_DEREF\_NONVIRTUAL

At the bottom of the IDE, a status bar indicates "Dead store to local variable (1)".



## Kolejny przykład

- ▶ `wyklad.wyklad3.debugger` — typowy błąd który Państwo nagminnie popełniacie na laboratoriach

## Wyjaśnienie błędu

- ▶ Są dwie zmienne które mają nazwę `l1` ale egzystują w innych zakresach. Jedno `l1` jest w zakresie klasy, a drugie jest zadeklarowane tylko dla konstruktora. Referencja `new JLabel()` jest przypisana do zmiennej w zakresie konstruktora a zmienna w zakresie klasy jest `null`.

# Narzędzia do statycznej analizy kodu

**FindBugs** Na licencji z rodziny GPL. Bardzo dobre narzędzie do znajdowania potencjalnych problemów w kodzie.

**Checkstyle** Służy głównie do znajdowania błędów w formatowaniu (poważne projekty i duże firmy mają jasne zasady).

**PMD** Tutaj nacisk kładzie się też na wykrywanie nieefektywnego kodu

**Intelij Idea** To IDE ma wbudowaną rozbudowaną wykrywarke błędów w kodzie.

# Instalacja Findbugs w Eclipse

FindBugs akurat ma wtyczki też do Idei (i pewnie Netbeansa).

- ▶ Help -> Install New Software
- ▶ Dodać software site:  
`http://findbugs.cs.umd.edu/eclipse/`
- ▶ I dalej jak na obrazku

# Ramka Instalacja Findbugs w Eclipse

**Install**

**Available Software**

Check the items that you wish to install.

Work with:

Find more software by working with the "[Available Software Sites](#)" preferences.

type filter text

Name	Version
<input checked="" type="checkbox"/> FindBugs	

1 item selected

Details

Show only the latest versions of available software  Hide items that are already installed

Group items by category  What is [already installed](#)?

Show only software applicable to target environment

Contact all update sites during install to find required software