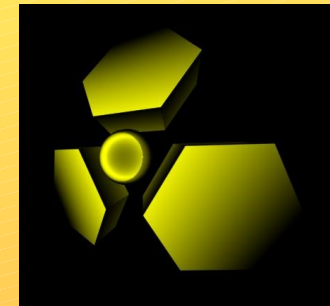


09.06.2010

Koło naukowe twórców gier,
Politechnika Warszawska



Najczęściej popełniane przez programistów błędy

A.K.A popularne anty-wzorce
 Obsługa błędów
 Wydajność

Spis rzeczy

- Anty-wzroce
 - Object Orgy
 - Magic Pushbutton
 - God object
 - Magic Strings
 - Magic Numbers
 - Blind faith
 - Golden hammer
 - Lava flow
- Obsługa błędów
 - Metody
 - Wyjątki
- Iterowanie
- Wątki

*Jedna linijka kodu
lepsza niż 1000 słów*

Object Orgy

- Niedostateczna enkapsulacja
- Obiekty bezpośrednio modyfikują zmienne innych obiektów
- Złe z wielu powodów, ale przede wszystkim – rozproszona logika
- Co by było, gdybyśmy dodawali do listy kwadraciki w trakcie działania?
- W jaki sposób zmodyfikować ten kod?

Magic Pushbutton

```
private:
void button1_Click( Object^ /*sender*/, System::EventArgs^ /*e*/ )
{
    // Get the control the Button control is located in.
    // In this case a GroupBox.
    Control^ control = button1->Parent;

    // Set the text and bgcolor of the parent control.
    control->Text = "My Groupbox";
    control->BackColor = Color::Blue;

    // Get the form that the Button control is contained within.
    Form^ myForm = button1->FindForm();

    // Set the text and color of the form containing the Button.
    myForm->Text = "The Form of My Control";
    myForm->BackColor = Color::Red;
}
```

Źródło: <http://msdn.microsoft.com/en-us/library/system.windows.forms.control.onclick.aspx>

God Object

- Obiekt, który robi „wszystko”
- Sprzeczne z koncepcją programowania strukturalnego
- Sprzeczne z koncepcją programowania obiektowego
- Nazwy: *manager, *controller, *GUI*



Magic Strings & numbers

```
double calculate(int a, double b, boolean rate) {
    if (rate) {
        return b*0.19;
    }
    return (b/a>32000)?b*0.39:b*0.34;
}
```

```
private static final double rate1 = 0.39;
private static final double rate2 = 0.34;
private static final double ratelin = 0.19;
private static final double MIN_INCOME_FOR_RATE_2 = 32000;
```

```
double calculateTaxes(int amountOfPeople, double totalIncome,
                    boolean usingLinearTaxes) {
    if (usingLinearTaxes) {
        return totalIncome*ratelin;
    }
    double averageIncome = totalIncome/amountOfPeople;
    return (averageIncome>MIN_INCOME_FOR_RATE_2)?
        averageIncome*rate2:
        averageIncome*rate1;
}
```

Blind faith

- Optymistyczne założenie, że kod działa
- Brak testów, nawet smoke testów

- Test Driven Development pozwala uniknąć tego typu problemów
- Nawet jeśli nie stosujemy TDD, testy to dobry pomysł



BLIND FAITH

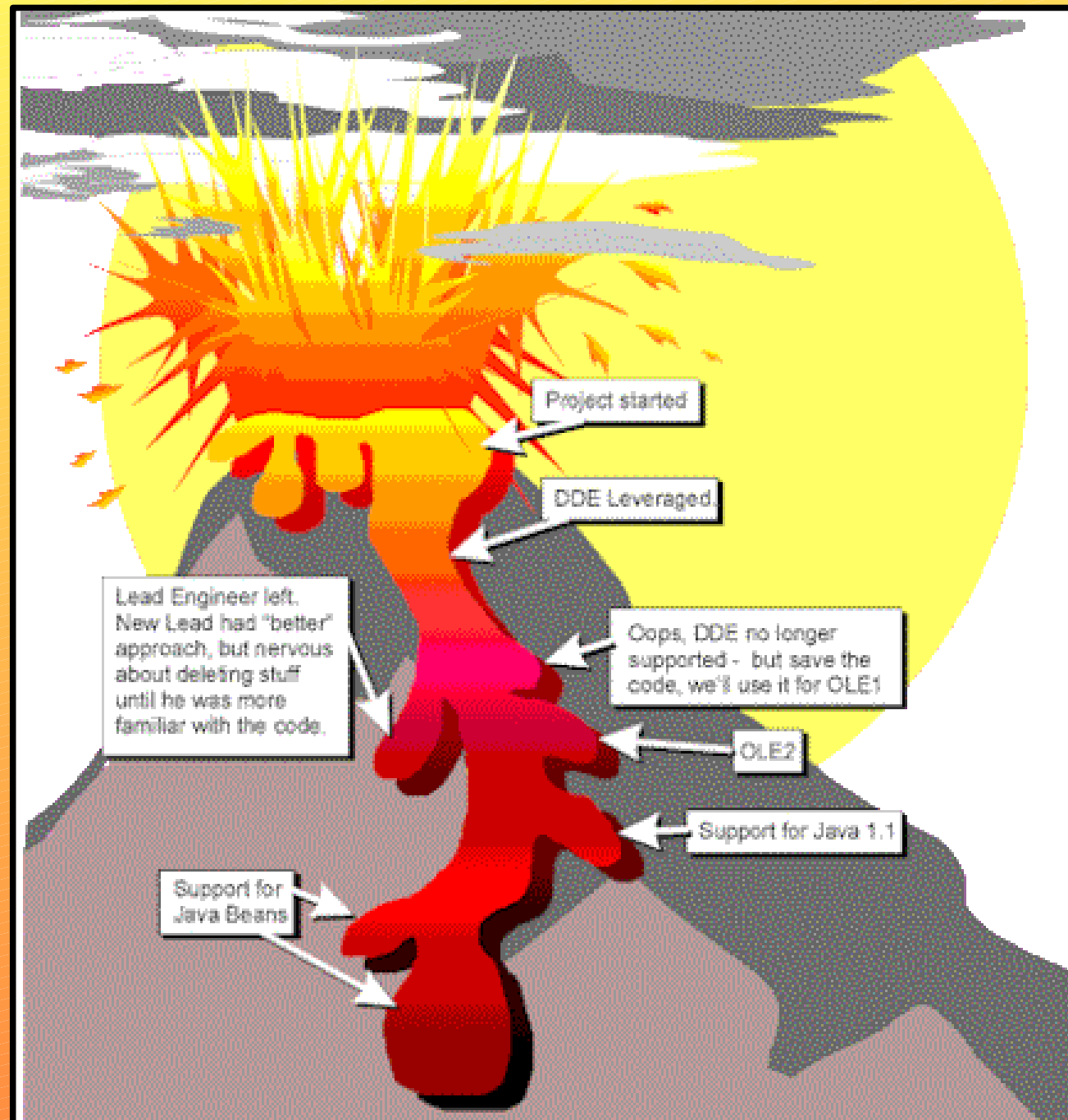
Because thinking is hard.

Golden Hammer

- Założenie, że dana technologia jest zawsze dobra do wszystkiego.
 - Zawsze.
 - Do wszystkiego.
- Przykład? Spring
- Obrona: wykorzystanie odpowiedniej technologii
 - Parsowanie tekstu – perl
 - Aplikacja wieloplatformowa – Java
 - Szybka aplikacja – C/C++
 - Łatwa w napisaniu – C# .net

Lava flow

- Nigdy nie usunięte komentarze TODO
- Zakomentowane linie kodu
- „Szybkie” workarouny



Metody zarządzania błędami

- Status działania jako zwrócona przez metodę wartość

```
int methodThatDoesSomething(Object result) {
    if (doSomething()) {
        if (!doYetAnotherThing()) {
            return 2;
        }
    } else {
        somethingHappened();
        return 1;
    }
    return 0;
}

void handleError() {
    StringBuilder resStr = new StringBuilder();
    int res = methodThatDoesSomething(resStr);
    if (res == 1) {
        System.out.println("error occured, exiting");
        return;
    } else if (res == 2) {
        System.out.println("Error 2 occured, not exiting");
    } else {
        System.out.println("method doing what it ought to do");
        System.out.println(resStr.toString());
    }
}
```

Metody zarządzania błędami

- Rzucanie wyjątkami

```
public StringBuilder methodThatDoesSomething() throws
    DoYetAnotherThingException, SomethingHappenedException
{
    if (doSomething()) {
        return doYetAnotherThing();
    } else {
        somethingHappened();
    }
    return new StringBuilder();
}
```

```
void handleError() {
    StringBuilder resStr = null;
    try {
        resStr = methodThatDoesSomething();
    } catch (DoYetAnotherThingException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (SomethingHappenedException e) {
        e.printStackTrace();
        System.exit(1);
    }
}
```

Metody zarządzania błędami

Kody błędów

- Są uciążliwe
- Nie pozwalają na wykorzystanie wartości zwracanej przez funkcję
- Wymagają bloków *case* lub *if-else if-else*
- Trzeba obsłużyć od razu po zwrocie wartości, powoduje drzewko wywołań

Wyjątki

- Są uciążliwe
- Pozwalają na wykorzystanie wartości zwracanej przez funkcję
- Wymagają, często dużego, bloku try-catch
- Część trzeba obsługiwać od razu, część można w dowolnym miejscu stosu wykonania

Zarządzanie wyjątkami

- Nigdy nie można używać `e.printStackTrace()` lub `println(„error”)`;
- Wznowienie pracy po wyjątku – w 90% bajka
- Nie wiesz co zrobić? Zrób tak:

```
try {  
    methodThatThrowsAnException();  
} catch (Exception e) {  
    throw new RuntimeException(e);  
}
```

- A potem:

```
try {  
    mainProgramMethod();  
} catch (RuntimeException e) {  
    e.printStackTrace(errWriter);  
}
```

Wydajność

- Zasady optymalizacji:
 - Nie rób tego [M.A.Jackson].
 - Nie rób tego... Na razie [M.A.Jackson].
 - Upewnij się, że potrzebujesz optymalizować
 - Sprawdź profilerem, czy optymalizacja ma sens
- *More computing sins are committed in the name of efficiency (without necessarily achieving it) than for any other single reason - including blind stupidity, W.A. Wulf*

Wydajność - przykłady

- Iteracja (iterationTest)
- Operacje na niezmiennych obiektach (stringTest)
- Koszt synchronizacji (queueTest)

Odnosiniki

- Internet:
 - http://pl.wikipedia.org/wiki/Antywzorzec_projektowy
 - <http://en.wikipedia.org/wiki/Anti-pattern>
 - <http://www.antipatterns.com/>
 - <http://java.sun.com/docs/books/tutorial/essential/exceptions/>
 - <http://c2.com/cgi/wiki?RulesOfOptimization>
- Literatura
 - *Clean Code: A Handbook of Agile Software Craftsmanship*, Robert C. Martin