

DĚLAT
DOBRÝ SOFTWARE
NÁS BAVÍ

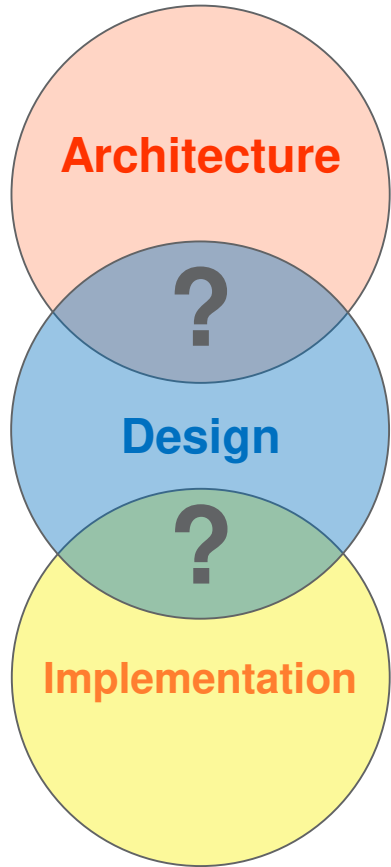
PROFINIT

Construction

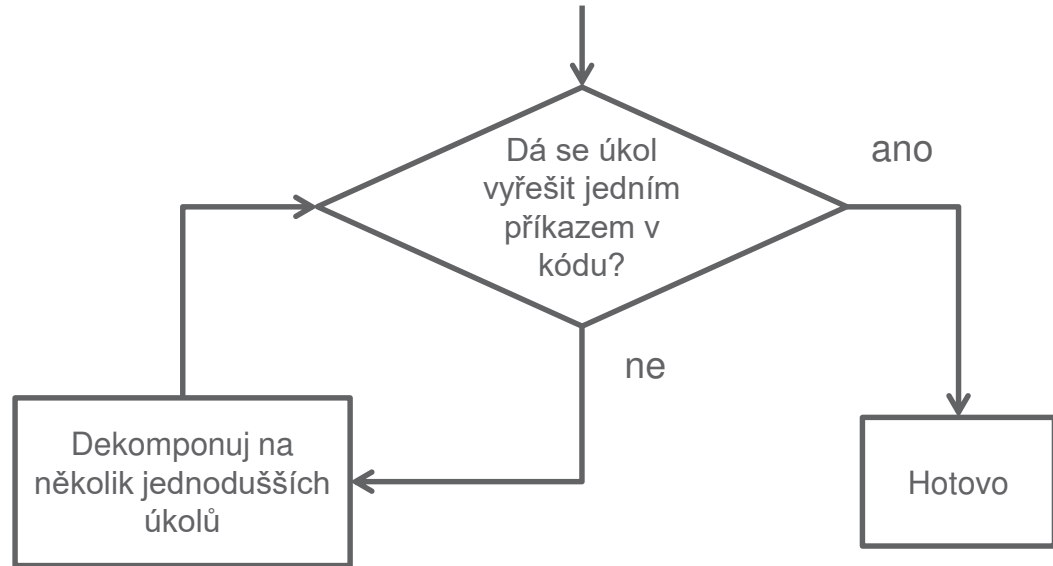
Jiří Toušek
jiri.tousek@profinet.eu
<https://profinet.eu/znalosti/>

29.10.2018

Architektura vs. návrh vs. implementace

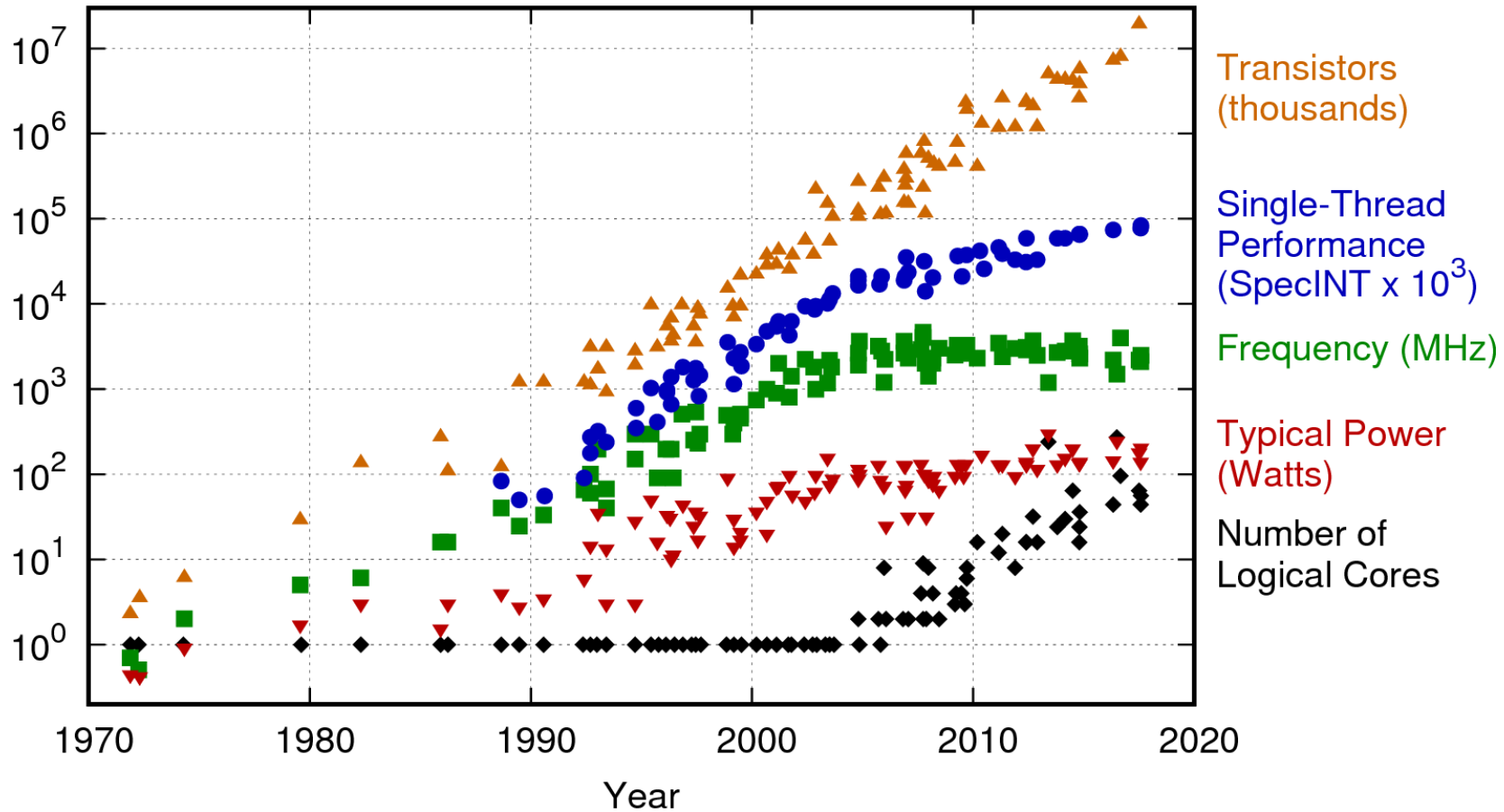


Návrh v kostce:



Historie CPU

42 Years of Microprocessor Trend Data



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

The background of the slide is a complex, abstract composition of numerous overlapping, translucent geometric shapes. These shapes, which include rectangles, parallelograms, and polygons of various sizes, are rendered in different shades of light gray. They are arranged in a way that creates a sense of depth and movement, with some shapes appearing to float above others. The overall effect is a textured, crystalline surface that changes as the viewer's perspective shifts.

Thread-safety

Programming with threads is hard

It's a useful technology – one that has the potential to improve your application's real (or perceived) performance. But it is the software equivalent of a nuclear device because if it is used incorrectly, it can blow up in your face. No – worse than that – used incorrectly, it can destroy your reputation and your business because it has nearly infinite potential to increase your testing and debugging costs.

Multithreading in VB.NET scares me more than any other new feature.

*The only way to use free threading safely is to understand it and to design your applications correctly. **Again, I stress, design your applications correctly. If your design is incorrect, it will be virtually impossible to patch up the problems later. And again, the potential cost to fix threading problems has no upper limit.***

– Dan Appleman, Moving to VB.NET: Strategies, Concepts, and Code

Programming with threads is hard

We have to have much better language-level support for concurrent programming. What we have now sucks. Rocks. Hard.

– Chris Sells, cca. 2005

Opakované použití objektů

1. Na jedno použití
 - Po použití zničte
2. Opakovaně použitelné
 - Lze použít opakovaně, ale až po ukončení předchozího použití
3. Thread-safe
 - Lze použít najednou ve více vláknech

Objekt na jedno použití

```
public class FileProcessor {  
    private InputStream is;  
    private ProcessedFile result = new ProcessedFile();  
  
    public FileProcessor(File file) {  
        is = new FileInputStream(file);  
    }  
  
    public ProcessedFile processFile() {  
        processHeaders();  
        processRows();  
        return result;  
    }  
}
```


Znovupoužitelný objekt

```
public class FileProcessor {  
    private InputStream is;  
    private ProcessedFile result = new ProcessedFile();  
  
    public FileProcessor() {  
        /* empty */  
    }  
  
    public ProcessedFile processFile(File file) {  
        is = new FileInputStream(file);  
        processHeaders();  
        processRows();  
        return result;  
    }  
}
```

Thread-safe objekt

```
public class FileProcessor {  
    public FileProcessor() {  
        /* empty */  
    }  
  
    public ProcessedFile processFile(File file) {  
        InputStream is = new FileInputStream(file);  
        ProcessedFile result = new ProcessedFile();  
        processHeaders(is, result);  
        processRows(is, result);  
        return result;  
    }  
}
```

Thread-safety a závislosti

Bezpečnost vůči použití ve více vláknech není tranzitivní.

Nebezpečnost při použití ve více vláknech ano.

Thread-safety a závislosti

Když chci použít objekt...

		Na 1 použití	Reusable	Thread-safe
V objektu, který je...	Na 1 použití	✓		✓
	Reusable	✓		✓
	Thread-safe	✓		✓

Když chci použít reusable objekt

- Vytvořit pokaždé nový
 - Tj. degraduji jej na objekt na jedno použití
- Zajistit, že dvě vlákna nebudou mít k dispozici tentýž objekt
 - Držet jeden pro každé vlákno
 - Musím si ho všude správně předat
 - V extrémním případě pro každé vlákno samostatná celá aplikace
 - Při použití z thread-safe objektu to není triviální
 - Pooling
- Upravit ho tak, aby byl thread-safe

Ukázka: Java – DB connection

```
public void foo(Connection connection) {  
    connection.query(...);  
    connection.query(...);  
    connection.query(...);  
}
```

Ukázka: Java – DB connection

```
public void foo(DataSource ds) {  
    Connection connection = ds.getConnection();  
  
    connection.query(...);  
    connection.query(...);  
    connection.query(...);  
  
    connection.close();  
}
```

Ukázka: Java – DB connection

```
public void foo(ObjectPool<Connection> pool) {  
    Connection connection = pool.borrowObject();  
  
    connection.query(...);  
    connection.query(...);  
    connection.query(...);  
  
    pool.returnObject(obj);  
}
```


Jak zajistit thread-safety

- Synchronizace
- Odstranění vnitřního stavu / immutabilita
- Rozdělení

Ukázka: Java – regulární výrazy

```
Pattern p = Pattern.compile(" +");  
Matcher m = p.matcher("My   input");  
if (m.find()) {  
    ...  
}
```

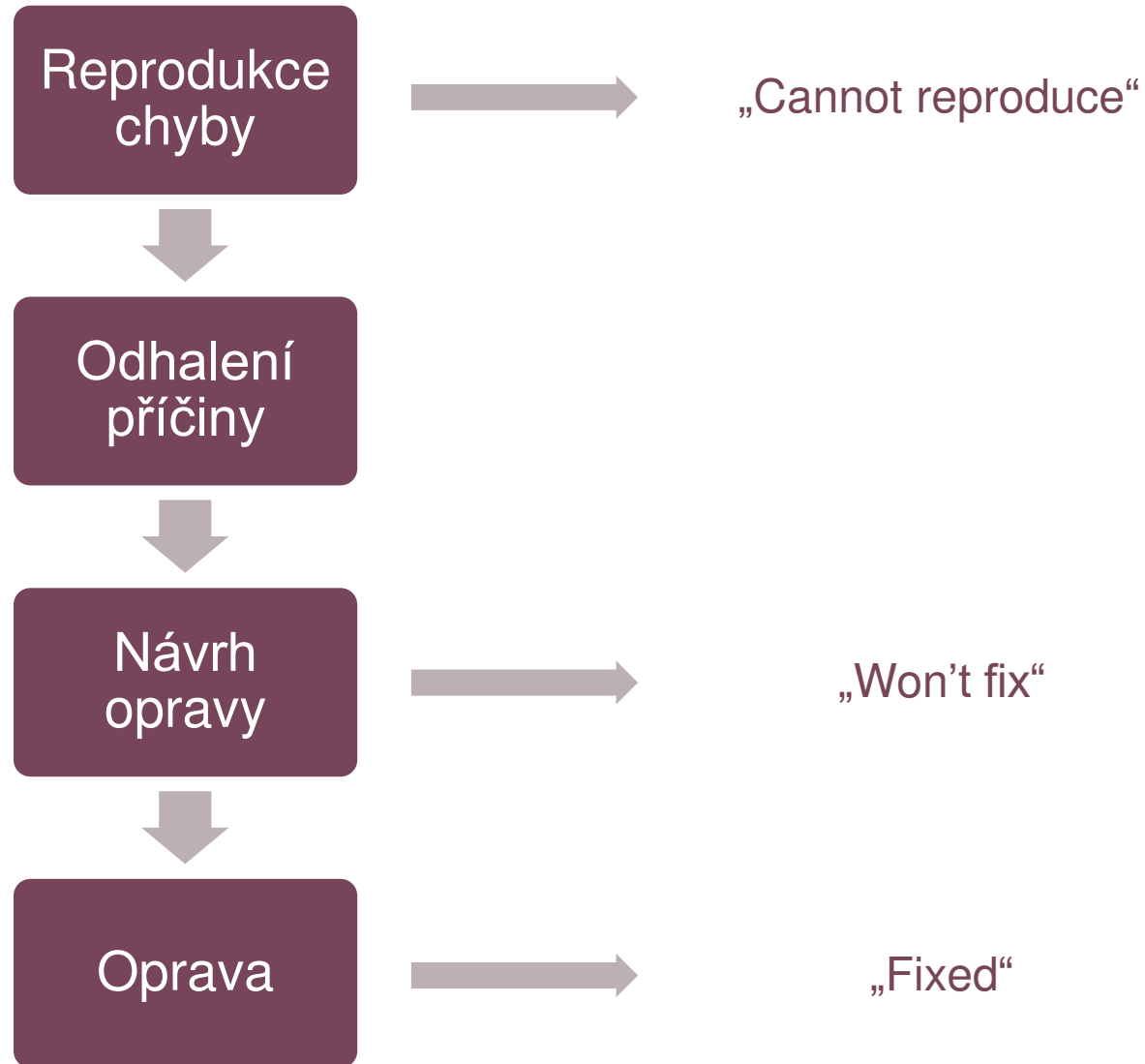
Thread-safety

- › Immutable třídy jsou inherentně thread-safe
- › Třídy bez vnitřního stavu jsou inherentně thread-safe
 - „Immutable once configured“
- › Doporučuji všude, kde to jde:
 - Služby beze stavu (bez instančních proměnných kromě odkazů na jiné služby)
 - Stav předávat v parametrech metod a držet v lokálních proměnných
- › Takto napsaný modul:
 - Nemá režii na synchronizaci (vhodný i pro aplikace nepotřebující thread-safety)
 - Není nezbytně thread-safe, ale půjde to snadno zařídit
 - Je čitelnější – je zřejmé, kudy tečou data

The background of the slide is composed of numerous overlapping, translucent, light-gray geometric shapes. These shapes, which include rectangles, trapezoids, and irregular polygons, are arranged in a way that creates a sense of depth and movement, as if they are floating or falling from the top right towards the bottom left. The text is centered in the middle of the slide.

Jak na chyby

Kroky k opravě chyby



Co potřebuji vědět k reprodukci chyby

- Co se stalo?
 - Jaký problém nastal?
 - Kde nastal?
- Za jakých okolností se to stalo?
 - Jaký byl vstup?
 - V jakém místě vstupu se chyba stala?
 - V jakém stavu byla aplikace?

Logování – Pokémon style

```
public void processFile(File file) {
    try {
        InputStream is = new FileInputStream(file);
        ProcessedFile result = new ProcessedFile();
        processHeaders(is, result);
        processRows(is, result);
        writeResult(result);
    } catch (Exception e) {
        Log.error("Došlo k chybě");
        throw e;
    }
}

private void processRows(InputStream is, ProcessedFile result) {
    ...
    for (int i = 0; i < rows.size(); i++) {
        processRow(rows[i], result);
    }
}
```

Logování – lépe

```
public void processFile(File file) {
    try {
        ...
        processHeaders(is, result);
        processRows(is, result);
        writeResult(result);
    } catch (Exception e) {
        Log.error("Došlo k chybě při čtení souboru" + file.getAbsolutePath(), e);
    }
}
```

```
private void processRows(InputStream is, ProcessedFile result) {
    ...
    for (int i = 0; i < rows.size(); i++) {
        try {
            processRow(rows[i], result);
        } catch {
            throw new RuntimeException("Chyba na řádku " + i, e);
        }
    }
}
```


Když detekuji chybu

- Pokud vím, jak situaci vyřešit (pokračovat dalším vstupem, zobrazit chybu uživateli, způsobit pád aplikace)
 - Zalogovat vše potřebné k pozdějšímu odhalení chyby
 - Pokračovat zvoleným způsobem
- Pokud nevím co s chybou, ale vím, za jakých okolností nastala
 - Vyhodit výjimku obsahující relevantní informace
 - Časté je odchycenou výjimku obalit další s přidányi informacemi a „cause“
- Pokud nevím co s chybou ani nemám co přidat
 - Nechat ji propadnout výš
 - Nakonec se najde někdo, kdo ji dokáže vyřešit
 - Systém umí řešit každou výjimku (typicky pádem)

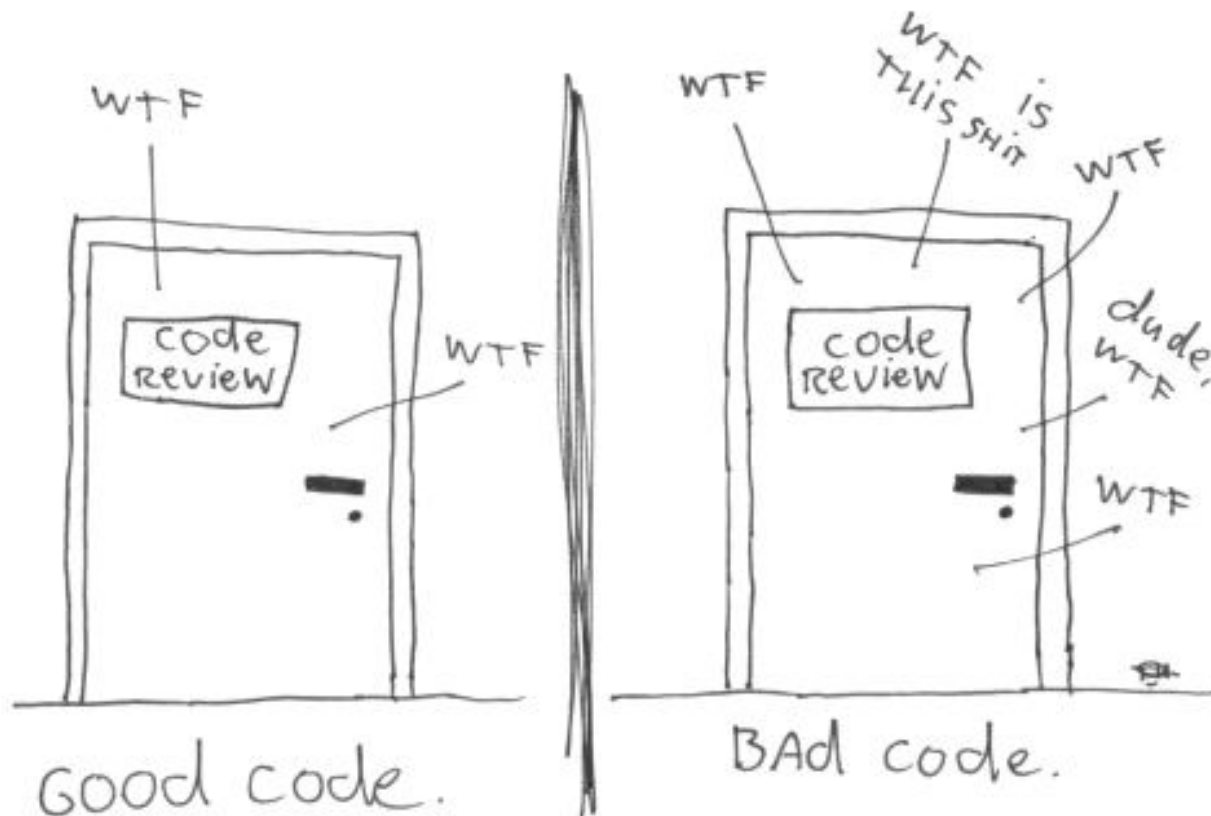
The background of the slide is a complex, abstract composition of numerous overlapping, translucent geometric shapes. These shapes, which include various polygons and rectangular prisms, are rendered in different shades of light gray and white. They are arranged in a way that creates a sense of depth and three-dimensional space, with some shapes appearing to float above others. The overall effect is a modern, architectural, and somewhat crystalline aesthetic.

Co je to kvalitní kód?

Kvalitní kód je...

- › Bezchybný
- › Rychlý / Efektivní
- › Krátký?
- › Rafinovaný?
- › **Srozumitelný**
- › **Udržitelný**

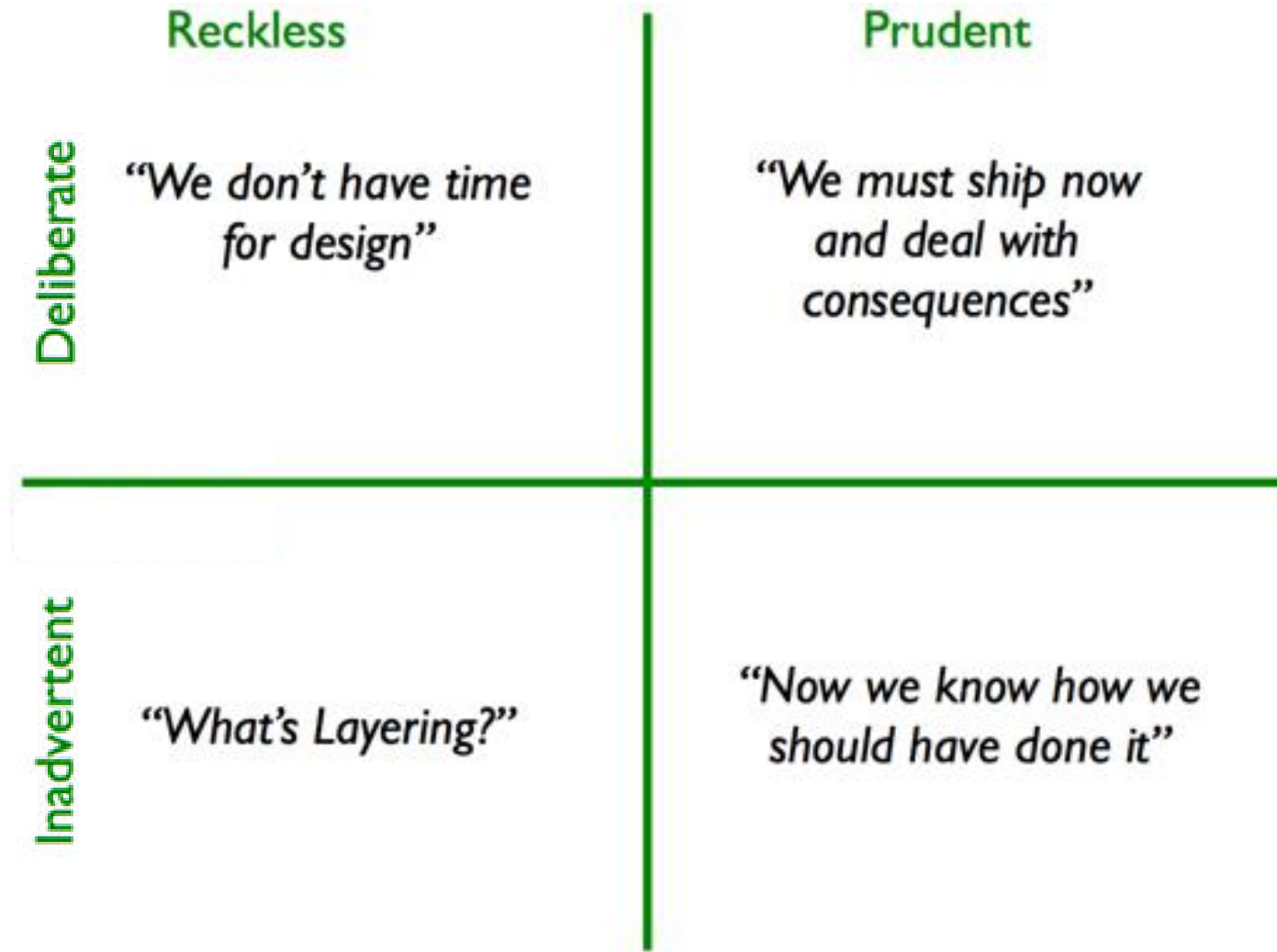
The ONLY valid measurement
of code quality: WTFs/minute



Proč psát srozumitelný kód

- › Kód píšete jednou, číst ho (vy nebo někdo jiný) budete pravděpodobně vícekrát
- › Pokud váš kód někdo nepochopí, snadno v jeho použití nebo úpravě udělá chybu
- › Nesrozumitelný kód nikdo nebude chtít používat

Kdy nepsat kvalitní kód?



The background of the slide is a complex, abstract composition of numerous overlapping, translucent geometric shapes. These shapes, which include various polygons and rectangular prisms, are rendered in different shades of light gray. They are arranged in a way that creates a sense of depth and three-dimensional space, with some shapes appearing to float above others. The overall effect is a modern, architectural aesthetic.

Jak psát srozumitelný kód

Co bude uživatel vaší metody číst

1. Jméno metody
 2. Deklarované typy parametrů a návratové hodnoty
 3. Jména deklarovaných parametrů
 4. Javadoc
 5. Samotný kód metody
- › Čím dříve se dozví vše, co potřeboval, tím dříve může přestat číst (a ušetřit čas)

Příklad: StringUtils

`join()`

Příklad: StringUtils

```
public static String join(Object[], String, int, int)
```

Příklad: StringUtils

```
public static String join(Object[] array, String separator, int startIndex, int endIndex)
```

Příklad: StringUtils

```
/**
```

Joins the elements of the provided array into a single String containing the provided list of elements.

No delimiter is added before or after the list. A null separator is the same as an empty String (""). Null objects or empty strings within the array are represented by empty strings.

```
StringUtils.join(null, *)           = null
StringUtils.join([], *)             = ""
StringUtils.join([null], *)         = ""
StringUtils.join(["a", "b", "c"], "--") = "a--b--c"
StringUtils.join(["a", "b", "c"], null) = "abc"
StringUtils.join(["a", "b", "c"], "")  = "abc"
StringUtils.join([null, "", "a"], ',') = ",,a"
```

Parameters:

array the array of values to join together, may be null

separator the separator character to use, null treated as ""

startIndex the first index to start joining from. It is an error to pass in a start index past the end of the array

endIndex the index to stop joining from (exclusive). It is an error to pass in an end index past the end of the array

Returns:

the joined String, null if null array input

```
*/
```

```
public static String join(Object[] array, String separator, int startIndex, int endIndex)
```

Příklad: StringUtils

```
/**
```

Joins the elements of the provided array into a single String containing the provided list of elements.

No delimiter is added before or after the list. A null separator is the same as an empty String (""). Null objects or empty strings within the array are represented by empty strings.

```
StringUtils.join(null, *)           = null
StringUtils.join([], *)             = ""
StringUtils.join([null], *)         = ""
StringUtils.join(["a", "b", "c"], "--") = "a--b--c"
StringUtils.join(["a", "b", "c"], null) = "abc"
StringUtils.join(["a", "b", "c"], "")  = "abc"
StringUtils.join([null, "", "a"], ',') = ",,a"
```

Parameters:

array the array of values to join together, **may be null**

separator the separator character to use, **null treated as ""**

startIndex the first index to start joining from. **It is an error to pass in a start index past the end of the array**

endIndex the index to stop joining from **(exclusive)**. **It is an error to pass in an end index past the end of the array**

Returns:

the joined String, **null if null array input**

```
*/
```

```
public static String join(Object[] array, String separator, int startIndex, int endIndex)
```

Příklad: StringUtils

```
public static String join(Object[] array, char separator, int startIndex, int endIndex) {
    if (array == null) {
        return null;
    }
    int bufSize = (endIndex - startIndex);
    if (bufSize <= 0) {
        return EMPTY;
    }

    bufSize *= ((array[startIndex] == null ? 16 : array[startIndex].toString().length()) + 1);
    StrBuilder buf = new StrBuilder(bufSize);

    for (int i = startIndex; i < endIndex; i++) {
        if (i > startIndex) {
            buf.append(separator);
        }
        if (array[i] != null) {
            buf.append(array[i]);
        }
    }
    return buf.toString();
}
```

Název metody

- › To první a často jediné, co z vaší metody bude většina lidí číst – věnujte mu pozornost
- › Měl by být stručný a přitom dostatečně popisovat, co metoda dělá
- › Pokud se vám nedaří vhodný název vymyslet, může to být špatným návrhem metody
 - Metoda dělá více různých věcí (měla by dělat vždy jen jeden svůj úkol)
 - Nemáte vlastně jasno, co má metoda dělat
 - ...
- › Pokud jste při vymýšlení jména přišli na špatný návrh metody, ušetřili jste si dost času, který byste jinak strávili později přepisováním kódu
- › Kvíz: co dělá tato metoda?
`dontStoreDataToCache()`

Název metody

`isSaveBasicDataAndTemplateAttributeValuesDeactivateDocumentReturnPossible()`

Jména a typy parametrů

- › Jméno by mělo popisovat, co daný objekt reprezentuje – nejen sám o sobě, ale i v kontextu použití parametru:

```
void merge(Graph graph1, Graph graph2)
```

vs.

```
void merge(Graph sourceGraph, Graph targetGraph)
```

- › U typovaných jazyků využívejte datové typy
 - Toto by měla být naprostá samozřejmost
 - Speciálně v Javě se ale často setkávám zejména s ignorováním enums (místo enumu se často používají booleany nebo soustavy konstant)
 - Zdá se, že velmi populární je místo „strongly typed“ používat „stringly typed“ přístup:
 - Text: "abc"
 - Číslo: "42"
 - Objekt obsahující jednu textovou a dvě číselné položky: "abc_42_666"
 - Přístup k první číselné položce objektu:
`myObjectString.split("_")[1]`

Co nepovažuji za užití datových typů

- › `Map<Vertex, Set<Boolean>>`
- › `List<List<List<String>>>`
- › `Map<List<Object>, HashSet<Pair<Integer, Integer>>>`

Parametry

- › Metoda s příliš velkým počtem parametrů je nepřehledná

```
protected PdfPTable getPrijemcePlneniTable(float documentWidth,  
    String nazevLeasingovky, String cisloLeasingoveSmlouvy, String  
    smluvniServis, String prijemceCisloUctu, String prijemceJmeno,  
    String prijemceUliceCP, String prijemceObecPSC, boolean  
    pojisteniJindeExistuje, String pojisteniJindeNazev, Boolean  
    dphAnoNeNull, String zpusobUhradyKey, String datum, boolean  
    isDPH, String sTextPodPodpisem)
```

- › Parametry typu boolean nejsou příliš vhodné

- Nutno v názvu parametru jasně deklarovat, co znamená true a co false
 - Jméno parametru ale nemusí být vždy dostupné (např. v Javě, mám-li knihovnu bez javadoc a zkompilevanou bez jmen parametrů, vidím jen typ bez jména)
- Při volání metody není na první pohled význam boolean parametru obvykle zřejmý
 - `newTable(tableName, parentDatabase, true)`
 - `newTable(tableName, parentDatabase, TableType.GLOBAL_TEMPORARY)`

Javadoc

- › Dohodněte se na jazyku a dodržujte ho
 - V angličtině může být problém se vyjádřit (zejména odborné pojmy)
 - Komentáře v češtině vyžadují neustále přepínat klávesnici a jazyk „v hlavě“
 - Komentáře v „cestine“ nevypadají dobře
- › Dokumentujte chování vůči null (všude tam, kde to není očividné)
 - U atributů objektu / třídy – zda může někdy být null a co to znamená
 - U parametrů metod – zda může být null a co se v tom případě stane
 - U návratových hodnot – zda a v jaké situaci může metoda vrátit null
- › U kolekcí a polí vnímejte rozdíl mezi prázdnou kolekcí a null
 - Většinou dává lepší smysl prázdná kolekce („seznam nalezených prvků je prázdný“) než null („seznam nalezených prvků neexistuje“)
 - Nenechte se vést leností nebo dojmem větší efektivity / úspory paměti při použití null (Java: `Collections.emptyList()`)

Javadoc

```
/**
 * Metoda cislo zaznamu milestone v cache nebo -1
 * @param firstRow
 * @param lastRow
 * @param extraRecords
 * @return
 * @throws ExpiredDataException
 */
public CachedData getTimeMilestoneCachedData(..)
```

- › Dodržujte konvence daného jazyka (jména, formátování, ...)
- › Orientace na čtenáře – kód bude čten víckrát, píšete jej jednou
 - Lenost zde není platným argumentem
- › Jména proměnných (a všeho ostatního) volte dostatečně srozumitelná
 - Příklad z jedné revize:
v cele třídě mnoho nevhodných názvů proměnných – „bi“, „is“, „bos“, „asr“, „s“, „m“, „baos“, „dos“ a můj favorit, pole s názvem „array“
 - Kvíz: co ošetřuje následující podmínka?

```
if(lv >= 0 && df >= 0 && this.hasRows() &&  
    df < this.getRows().length) { ... }
```
 - Všeho s mírou – obecně čím větší rozsah platnosti, tím důležitější je jasné jméno
 - Srozumitelnost != délka – např. význam proměnné „i“ je obecně známý
- › Nepoužívejte jednu proměnnou ke dvěma účelům
- › Ternární operátor je efektní, ale leckdy poněkud nečitelný
 - Kvíz: co vrátí následující výraz?
 $(0 < 1) ? 2 : 3 + 4$

A Roguelike in less than 512 Bytes

```
#include<stdlib.h>
#define F(n)for(j=0;j<n;j++)
#define r rand()
int main(){int x,s=46,n,i,j,z=77,l[z];char m[z*s],h[z];initscr();raw();F(z*s)j[
m]=35;F(s)for(j[l]=i=(r%4+3)*z+(n=r%17*z+r%s+z);n<=i;n+=z)for(x=n;x<=n+j/2;m[+
x]=s);F(9)l[j][m]=z,j[h]=2;m[*l]=64;*h=5;l[j][m]=62;F(z){x=n=l[i++];i%=9;if(i)!
i[h]||*l^(n+=r%3+r%3*z+~z)||--*h?0:abort();else{F(25)mvaddnstr(j,i,m+j*z,z);j=s
-getch();m[n+=j/3*z-j%3+153]^62||main();F(9)l[j+1]^n||--h[j+1]||n[m]--;}n[m]^s
||(m[l[i]=n]=x[m],x[m]=s);}}
```

This weighs in at a hefty 493 bytes of C source code. I call it the "Monster Caves".

Note that the game doesn't have enough bytes to remember (and print out) your kill total, so you'll have to do that yourself.

If you get stuck in a disconnected part of a level, try pressing some other keys... you may just be able to teleport out.

http://locklessinc.com/articles/512byte_roguelike/

The background of the slide is a complex, abstract composition of numerous overlapping, translucent geometric shapes. These shapes, which include various polygons and rectangles, are rendered in different shades of light gray and white. They are arranged in a way that creates a sense of depth and movement, as if they are floating or falling from the top right towards the bottom left. The overall effect is a textured, crystalline appearance.

Programming by Coincidence

Programming by Coincidence

```
if (typZadosti == ZALOZENI_SMLOUVY) {  
    zalozSmlouvu();  
} else {  
    zrusSmlouvu();  
}
```

Programming by Coincidence

```
if (typZadosti == ZALOZENI_SMLOUVY) {  
    zalozSmlouvu();  
} else if (typZadosti == ZRUSENI_SMLOUVY) {  
    zrusSmlouvu();  
}
```

Programming by Coincidence

```
if (typZadosti == ZALOZENI_SMLOUVY) {  
    zalozSmlouvu();  
} else if (typZadosti == ZRUSENI_SMLOUVY) {  
    zrusSmlouvu();  
} else {  
    throw new IllegalArgumentException(  
        "Nepodporovany typ zadosti: " + typZadosti);  
}
```

Programming by Coincidence

```
if (typZadosti == ZALOZENI_SMLOUVY) {  
    zalozSmlouvu();  
} else if (typZadosti == ZRUSENI_SMLOUVY) {  
    zrusSmlouvu();  
}  
// else do nothing
```

Programming by Coincidence 2

```
int pocetCelychMesicuDoKonceKvartalu = 0;
switch(mesicVRoce%3) {
    case 0: pocetCelychMesicuDoKonceKvartalu = 0;break;
    case 1: pocetCelychMesicuDoKonceKvartalu = 2;break;
    case 2: pocetCelychMesicuDoKonceKvartalu = 1;break;
}
```

Programming by Coincidence 2

```
int pocetCelychMesicuDoKonceKvartalu = 0;
switch(mesicVRoce%3) {
    case 0: pocetCelychMesicuDoKonceKvartalu = 0;break;
    case 1: pocetCelychMesicuDoKonceKvartalu = 2;break;
    case 2: pocetCelychMesicuDoKonceKvartalu = 1;break;
    default: throw new IllegalStateException(
        "Nastala situace, ke ktere nemuze dojít");
}
```

Je tento konstruktor prázdný záměrně?

```
protected MyObject() {  
  
};
```

Je tento konstruktor prázdňý záměrně?

```
protected MyObject() {  
    /* empty */  
};
```


Dřív to fungovalo...

```
entity.id = new EntityId(this.allEntities.size());
```

The background of the slide is a complex, abstract composition of numerous overlapping, translucent geometric shapes. These shapes, which include various polygons and rectangular prisms, are rendered in different shades of light gray. They are arranged in a way that creates a sense of depth and three-dimensional space, with some shapes appearing to float above others. The overall effect is a textured, crystalline surface.

Perličky

Kdy se zobrazí nabídka úvěrů?

```
/**
 * Method returns true if credit lines should be shown.
 *
 * @return true if credit lines should be shown otherwise false
 */
public boolean isShowCreditLines() {
    return
        // Showing depends of dynamic property and if user is in pilot mode (show only in pilot to
        // users in pilot mode if constant is true, if false, show to all)
        ((ebankingUser.getPilot() == null) || (!EbankingConstants.CREDIT_LINES_FUNCTIONS_PILOT) ||
        (Boolean.TRUE.equals(ebankingUser.getPilot()) && EbankingConstants.CREDIT_LINES_FUNCTIONS_PILOT))
        // and (home tab is shown and there are no campaigns)
        && ((menuBean.isHomeTab() && !bcs.isShowCampaigns())
        // or ((we are on loans tab (TAB_LOANS) on specific menu item (MNU_LST_CRE_MORA_GET)
        // or on eshop (TAB_ESHOP)) and client has no approved application)
        || (((menuBean.isWantedTab(MenuNameS24.TAB_LOANS.name())
        // THU: tento kod je zakomentovan, protoze nefuguje nastavovani actualMenuItem spravne
        /*&& (menuBean.getActualMenuItem() != null) &&
        menuBean.getActualMenuItem().name().toUpperCase().equals(
        MenuNameS24.MNU_LST_CRE_MORA_GET.name())*/

        )
        || menuBean.isWantedTab(MenuNameS24.TAB_ESHOP.name())) &&
        bcs.getApprovedApplicationID() == null))

        // and there exist some credit lines
        && !isItemListEmpty();
}
```

Map není nikdy dost

```
public interface IValidator {  
  
    public void validate(  
        Object[] objects,  
        Map<Object, Object> globalParams,  
        Map<Object, Object> localParams,  
        Map<Object, Object> rowParams,  
        Locale locale);  
  
}
```

DB procedura

```
...  
**  Nazev ulozene procedury:  esipo_s_prehl_rotv  
**  
**  Funcionalita:  Procedura slouzi  
**  
**  Vstupni parametry:  
...
```

Trocha logiky

```
public void setPlatnostDoOdvolani(boolean platnostDoOdvolani) {  
    if (platnostKontaktniAdresy == null) {  
        platnostKontaktniAdresy = new IntervalOdDo();  
    }  
    else {  
        platnostKontaktniAdresy = new IntervalOdDo(  
            platnostKontaktniAdresy.getZacatek(), null);  
    }  
}
```

Čím se od sebe metody liší?

```
/**
 * Directory interface poskytovany modulum skrze Webservice webovou
 * aplikaci datovych schranek.
 */
public interface IDSDirectoryService {

    public ListResponse<Issuer> getIssuer(
        IssuerSearchCriteria searchCriteria, String personId);

    public ListResponse<Issuer> findIssuer(
        IssuerSearchCriteria searchCriteria, String personId);
}
```

Handle stuff

```
dataProcessor.processData(elementsOnPages);
```



```
#DEAR FUTURE SELF,  
#  
# YOU'RE LOOKING AT THIS FILE BECAUSE  
# THE PARSE FUNCTION FINALLY BROKE.  
#  
# IT'S NOT FIXABLE. YOU HAVE TO REWRITE IT.  
# SINCERELY, PAST SELF
```

DEAR PAST SELF, IT'S KINDA
CREEPY HOW YOU DO THAT.

```
#ALSO, IT'S PROBABLY AT LEAST  
# 2013. DID YOU EVER TAKE  
# THAT TRIP TO ICELAND?
```

STOP JUDGING ME!

