

Objektové modelování, AD7B36OMO

Návrhy rozhraní

Tomáš Mayer

Katedra počítačů
Fakulta elektrotechnická
České vysoké učení technické v Praze

<https://edux.feld.cvut.cz/courses/AD7B36OMO>

Přednáška je vytvořena na základě původních přednášek autorů: Karel Richta;Pavel Strnad, 2014

MOTIVACE

- PROČ SPECIFIKACE?
- ČASTOU PŘÍČINOU MNOHO CHYB V CHOVÁNÍ SYSTÉMU JE NEDOROZUMNĚNÍ NA ROZHRANÍ DVOU ČÁSTÍ KÓDU
- NEDOKUMENTOVANÉ ROZHRANÍ JE V MYSLI VÝVOJÁŘE, VÝVOJÁŘ DRUHÉ ČÁSTI KÓDU MŮŽE ROZHRANÍ VNÍMAT JINAK
- SPECIFIKACE ROZHRANÍ
 - DOKUMENTACE PRO KLIENTA, KTERÝ ROZHRANÍ VYUŽÍVÁ
 - NEMUSÍ ZKOUMAT KÓD
 - Odstínění pro vývojáře kódu
 - VÝVOJÁŘ MÁ MOŽNOST MĚNIT KÓD ANIŽ BY MUSEL PŘEDÁVAT JAKO DOKUMENTACI NOVÝ KÓD KLIENTŮM

MOTIVACE – PROČ SPECIFIKACE?

```
public BigDecimal divide(BigDecimal val, int newScale, int roundingMode)
    throws ArithmeticException, IllegalArgumentException
{
    if (roundingMode < 0 || roundingMode > 7)
        throw
        new IllegalArgumentException("illegal rounding mode: " + roundingMode);

    if (intVal.signum () == 0) // handle special case of 0.0/0.0
        return newScale == 0 ? ZERO : new BigDecimal (ZERO.intVal, newScale);

    // Ensure that pow gets a non-negative value.
    BigInteger valIntVal = val.intVal;
    int power = newScale - (scale - val.scale);
    if (power < 0)
    {
        // Effectively increase the scale of val to avoid an
        // ArithmeticException for a negative power.
        valIntVal = valIntVal.multiply (BigInteger.TEN.pow (-power));
        power = 0;
    }

    BigInteger dividend = intVal.multiply (BigInteger.TEN.pow (power));

    BigInteger parts[] = dividend.divideAndRemainder (valIntVal);

    BigInteger unrounded = parts[0];
    if (parts[1].signum () == 0) // no remainder, no rounding necessary
        return new BigDecimal (unrounded, newScale);

    if (roundingMode == ROUND_UNNECESSARY)
        throw new ArithmeticException ("Rounding necessary");

    int sign = intVal.signum () * valIntVal.signum ();

    if (roundingMode == ROUND_CEILING)
```

divide

```
public BigDecimal divide(BigDecimal divisor,
                        int scale,
                        int roundingMode)
```

Returns a BigDecimal whose value is (this / divisor), and whose scale is as specified. If rounding must be performed to generate a result with the specified scale, the specified rounding mode is applied.

The new divide(BigDecimal, int, RoundingMode) method should be used in preference to this legacy method.

Parameters:

divisor - value by which this BigDecimal is to be divided.
scale - scale of the BigDecimal quotient to be returned.
roundingMode - rounding mode to apply.

Returns:

this / divisor

Throws:

ArithmeticException - if divisor is zero, roundingMode==ROUND_UNNECESSARY and the specified scale is insufficient to represent the result of the division exactly.

IllegalArgumentException - if roundingMode does not represent a valid rounding mode.

MOTIVACE

- SPECIFIKACE ROZHRANÍ UMOŽNÍ DECOUPLING
- PŘEDSTAVUJE SPOLEČNÝ KONTRAKT, KTERÝ MUSÍ OĚ STRANY DODRŽET
- OBĚ ČÁSTI KLIENT VOLAJÍCÍ KÓD PŘES ROZHRANÍ A VÝVOJÁŘ KÓDU MOHOU VYVÍJET NEZÁVISLE

BEHAVIORÁLNÍ EKVIVALENCE

- OTÁZKA ZDA DVA ZDROJOVÉ KÓDY SE CHOVÁJÍ STEJNĚ
- MOŽNOST VÝMĚNY KÓDU ZA JINÝ BEZE ZMĚNY CHOVÁNÍ SYSTÉMU
- ROLI MOHOU HRÁT
 - MNOŽINA VSTUPNÍCH PARAMETRŮ
 - KONTEXT, VE KTERÉM KÓD BĚŽÍ
 - SPECIFIKACE BY MĚLA OBSAHOVAT POPIS TĚCHTO VLIVŮ
 - PRECONDITIONS – ZÁVAZEK PRO KLIENTA
 - POSTCONDITIONS – ZÁVAZEK PRO IMPLEMENTÁTORA
- EKVIVALENCE SIMULAČNÍCH MODELŮ
- BEHAVIORÁLNÍ EKVIVALENCE - NENÍ OBECNĚ TESTOVATELNÁ

BEHAVIORÁLNÍ EKVIVALENCE

- REFLEXIVNÍ:
 - $E \sim E$
- SYMETRICKÁ:
 - IF $E1 \sim E2$ THEN $E2 \sim E1$
- TRANSITIVNÍ:
 - IF $E1 \sim E2$ AND $E2 \sim E3$ THEN $E1 \sim E3$
- PŘEDEVŠÍM TRANSITIVNOST DŮLEŽITÁ PRO SIMULACE

PRECONDITIONS A POSTCONDITION

- PRECONDITIONS – POPISUJÍ PODMÍNKY PRO VSTUPNÍ HODNOTY
- POSTCONDITIONS – POPISUJÍ, JAKÝM ZPŮSOBEM MÁ BÝT VYMEZENA IMPLEMENTACE

```
static int find(int[] arr, int val)
```

requires: val occurs exactly once in arr

effects: returns index i such that arr[i] = val

PRECONDITION A POSTCONDITION V JAVADOC

```
static int find(int[] arr, int val)
    requires: val occurs exactly once in arr
    effects: returns index i such that arr[i] = val
```

```
/**
 * Find a value in an array.
 * @param arr array to search, requires that val occurs exactly once
 *           in arr
 * @param val value to search for
 * @return index i such that arr[i] = val
 */
static int find(int[] arr, int val)
```


SILNĚJŠÍ VERSUS SLABŠÍ SPECIFIKACE

- V PŘÍPADĚ ZMĚNY IMPLEMENTACE NEBO PŘÍMO VLASTNÍ SPECIFIKACE POTŘEBUJEME URČIT, JAK BEZPEČNÉ JE PROVÉST ZMĚNU V IMPLEMENTACI VZHLEDEM K JIŽ EXISTUJÍCÍM KLIENTŮM
- DVĚ SPECIFIKACE $S1$, $S2$
- $S2$ JE SILNĚJŠÍ NEBO STEJNÁ JAKO $S1$ KDYŽ
 - PRECONDITION PRO $S2$ JSOU SLABŠÍ NEŽ NEBO STEJNÉ JAKO PRO $S1$
 - POSTCONDITION PRO $S2$ JSOU SILNĚJŠÍ NEBO STEJNÉ JAKO PRO $S1$, PRO STAVY, KTERÉ USPOKOJÍ PRECONDITION PRO $S1$
- V TAKOVÉM PŘÍPADĚ IMPLEMENTACE $S2$ DOKÁŽE USPOKOJIT TAKÉ POTŘEBY KLIENTŮ VYUŽÍVAJÍCÍCH $S1$. LZE TEDY $S1$ NAHRADIT $S2$.

SILNĚJŠÍ A SLABŠÍ SPECIFIKACE

- PŘÍKLAD

```
static int findExactlyOne(int[] a, int val)  
requires: val occurs exactly once in a  
effects: returns index i such that a[i] = val
```

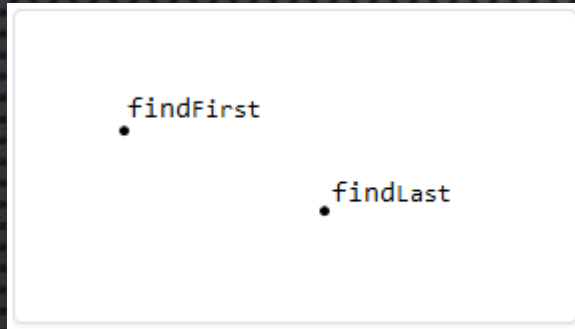
- MŮŽE BÝT NAHRAZENA - MÁ SLABŠÍ PRECONDITION

```
static int findOneOrMore,AnyIndex(int[] a, int val)  
requires: val occurs at least once in a  
effects: returns index i such that a[i] = val
```

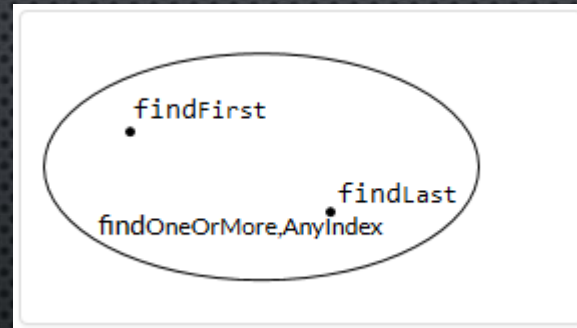
- A TA MŮŽE BÝT NAHRAZENA - MÁ SILNĚJŠÍ POSTCONDITION

```
static int findOneOrMore,FirstIndex(int[] a, int val)  
requires: val occurs at least once in a  
effects: returns lowest index i such that a[i] = val
```

GRAFICKÉ ZNÁZORNĚNÍ

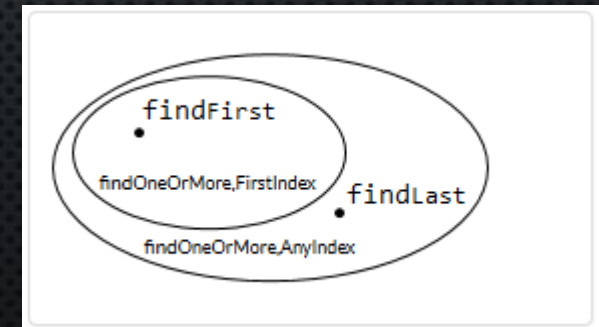


`findFirst`, `findLast` - implementace



`findOneOrMore`, any index - specifikace

`findOneOrMore`, `firstIndex` – silnější specifikace



NULL REFERENCE

- REFERENCE NA OBJEKTY A POLE MOHOU MÍT SPECIÁLNÍ HODNOTU – NULL
- NEUKAZUJE NA ŽÁDNÝ OBJEKT
- NELZE PRO PRIMITIVNÍ OBJEKTY

- POUŽITÍ NULL – POTENCIÁLNÍ VYJÍMKY `NullPointerException` V RUNTIME
- POKUD MOŽNO SE JIM VYHÝBAT, PŘEDEVŠÍM NA ROZHRANÍ
- DOBRÁ PRAXE PRO KAŽDOU OPERACI IMPLICITNĚ NEPOVOLOVAT NULL V PARAMETRECH ANI NÁVRATOVÝCH HODNOTÁCH
- V PŘÍPADĚ, ŽE NĚKTERÝ PARAMETR POVOLUJE NULL, PAK VE SPECIFIKACI EXPLICITNĚ ZMÍNIT – TOTÉŽ PLATÍ I PRO NÁVRATOVÉ HODNOTY

TESTOVÁNÍ A SPECIFIKACE

- TESTY MUSÍ SPLŇOVAT PODMÍNKY SPECIFIKACE
- ZNALOST IMPLEMENTACE MŮŽE BÝT VHODNÝM DOPLŇKEM PRO TESTOVÁNÍ
 - MŮŽE NAPŘÍKLAD POMOCI S TESTOVÁNÍM SPECIFICKÝCH PODMÍNEK, KTERÉ NEJSOU VE SPECIFIKACI UVEDENY
 - TESTY NESMÍ BÝT PODŘÍZENY IMPLEMENTACI, ALE VÝHRADNĚ SPECIFIKACI ROZHRAŇÍ

```
static int find(int[] arr, int val)  
    requires: val occurs in arr e  
    ffects: returns index i such that arr[i] = val
```

```
int[] array = new int[] { 7, 7, 7 };  
assertEquals(0, find(array, 0)); // bad test case: violates the spec  
assertEquals(7, array[find(array, 7)]); // correct
```

SPECIFIKACE VYJÍMEK

- ANTIPATTERN – PŘEDÁVAT INFORMACE O CHYBĚ NEBO OKRAJOVÉM STAVU POMOCÍ NÁVRATOVÉ HODNOTY

```
class BirthdayBook { LocalDate lookup(String name) { ... } }
```

- ATIPATTERN - POKUD V BIRTHDAYBOOK NEEEXISTUJE DANÉ JMÉNO VRACÍ MÍSTO LOCALDATE NULL V HORŠÍM PŘÍPADĚ NĚJAKÉ DOMLUVENÉ DATUM, KTERÉ PO DOBU ŽIVOTA SYSTÉMU NEPŘEDPOKLÁDÁME, ŽE NASTANE 9/9/9999
- LÉPE, ABY METODA VYHAZOVALA VYJÍMKY

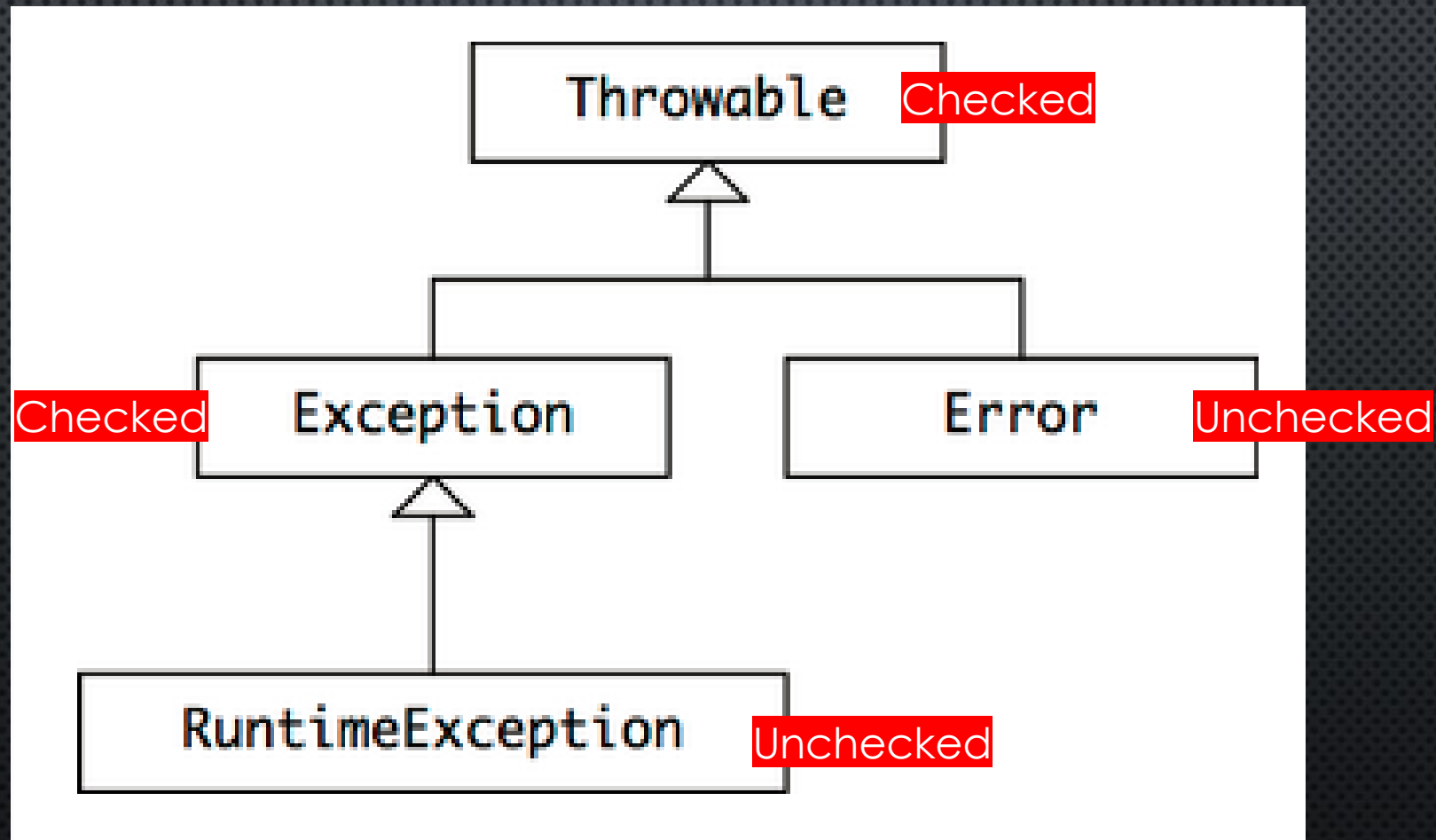
SPECIFIKACE VYJÍMEK

- LÉPE, ABY METODA VRACELA VYJÍMKY

```
LocalDate lookup(String name) throws NotFoundException {  
    ...  
    if ( ...not found... ) throw new NotFoundException();  
    ...  
}
```

- VYJÍMKY SE MOHOU VYSKYTNOUT PŘI CHYBĚ NEBO JAKO SPECIÁLNÍ VÝSLEDEK
- V PRVNÍM PŘIBLÍŽENÍ
 - CHECKED VYJÍMKY – PRO SPECIÁLNÍ VÝSLEDKY
 - UNCHECKED VYJÍMKY – PRO CHYBOVÉ STAVY PŘÍP. TAM, KDE KLIENT BUDE PŘEDCHÁZET SPECIÁLNÍM VÝSLEDKŮM OŠETŘENÍM PŘED VOLÁNÍM METODY

HIERARCHIE VYJÍMEK V JAVĚ



POUŽITÍ VYJÍMEK ANTIPATTERN

- NEPOUŽÍVAT VYJÍMKY TAM, KDE SE NEJEDNÁ O VYJÍMEČNÉ STAVY

```
try {  
    int i = 0;  
    while (true)  
        a[i++].f();  
} catch ArrayIndexOutOfBoundsException e) { }
```

Převzato z Effective Java Joshua Bloch

- NENÍ VHODNÉ ZDE POUŽÍT VYJÍMKU, JEPŠÍ NAPŘÍKLAD FOR CYKLUS PŘES PRVKY POLE
- ARRAYINDEXOUTOFBOUNDSEXCEPTION MŮŽE NAVÍC NASTAT NAPŘÍKLAD JINAK NEŽ V KÓDU OČEKÁVÁME -> TĚŽKO ODHALITELNÁ CHYBA

SPEFIKACE METOD MUTUJÍCÍCH DATA

- METODA, KTERÁ MĚNÍ VSTUPNÍ DATA

```
static boolean addAll(List<T> list1, List<T> list2)
```

requires: list1 != list2

effects: modifies list1 by adding the elements of list2 to the end of it, and returns true if list1 changed as a result of call

- PRO SPECIFIKACI ROZHRANÍ JE VELMI VHODNÁ KONVENCE JAKO PRO NULL – POKUD NENÍ EXPLICITNĚ UVEDENO V ROZHRANÍ, PAK METODA NEPROVÁDÍ ZMĚNU VE VSTUPNÍCH DATECH
- POPIS MUTACE VSTUPNÍCH PARAMETRŮ BY SE VŽDY MĚL EXPLICITNĚ OBJEVIT VE SPECIFIKACI ROZHRANÍ

VÝZNAM SPECIFIKACE

- PŘEDCHÁZÍ CHYBÁM – DOBRÁ SPECIFIKACE JASNĚ DOKUMENTUJE PŘEDPOKLADY, NA KTERÝCH STAVÍ KLIENT I IMPLEMENTÁTOR. CHYBY ČASTO VZNIKAJÍ Z NESPRÁVNĚ POCHOPENÉHO ROZHŘANÍ A SPECIFIKACE TOMUTO PŘECHÁZÍ. POUŽÍVÁNÍ STROJEM KOTROLOVANÉHO POPISU VE SPECIFIKACI JAKO TYPOVÁ KONTROLA A POŽITÍ VYJÍMEK MŮŽE JEŠTĚ VÍCE REDUKOVAT CHYBY
- SROZUMITELNOST - KRÁTKÁ JEDNODUCHÁ SPECIFIKACE JE MNOHEM VÍCE SROZUMITELNÁ NEŽ VLASTNÍ IMPLEMENTACE. OSTATNÍ VÝVOJÁŘI NEMUSÍ STUDOVAT KÓD, ABY MOHLI ROZHŘANÍ VYUŽÍT
- PŘIPRAVENOST NA ZMĚNY - SPECIFIKACE ZAVÁDÍ KONTRAKT MEZI RŮZNÝMI ČÁSTMI KÓDU. TYTO ČÁSTI SE MOHOU, POKUD DORŽÍ KONTRAKT ROZHŘANÍ ROZVÍJET NEZÁVISLE

PŘESNOST POPISU SPECIFIKACE

- POPISUJE SPECIFIKACE POUZE JEDEN MOŽNÝ VÝSTUP PRO DANÝ VSTUP NEBO DÁVÁ VOLNOST VÝVOJÁŘI VYBRAT Z MNOŽINY MOŽNÝCH HODNOT
- DETERMINISTIC – JEDNOZNAČNĚ URČENÝ VÝSTUP

```
static int findExactlyOne(int[] arr, int val)  
requires: val occurs exactly once in arr effects:  
returns index i such that arr[i] = val
```

- UNDETERMINED – NENÍ JEDNOZNAČNĚ URČENO, VÝVOJÁŘ MÁ JISTOU VOLNOST IMPLEMENTACE

```
static int findOneOrMore,AnyIndex(int[] arr, int val)  
requires: val occurs in arr  
effects: returns index i such that arr[i] = val
```

DEKLARATIVNÍ VERSUS OPERATIVNÍ SPECIFIKACE

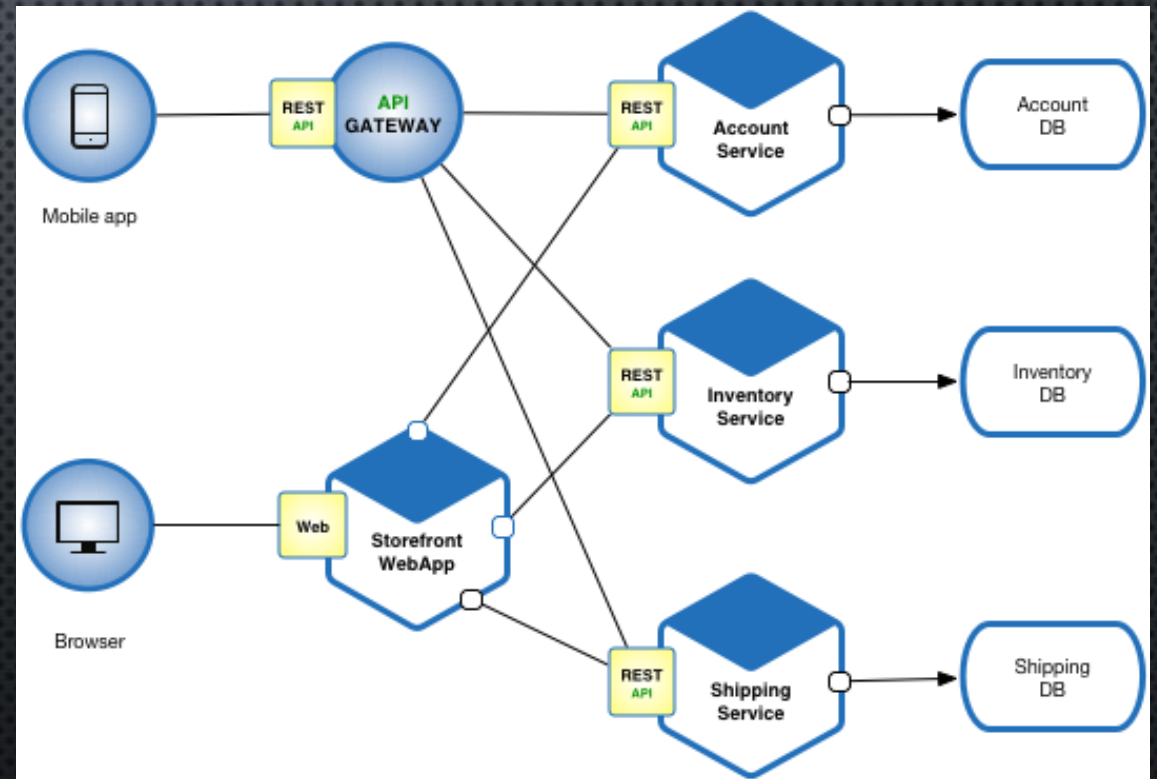
- OPERATIVNÍ – SPECIFIKACE POPISUJE KROKY IMPLEMENTACE A DÁVÁ PŘEDPIS PRO VÝVOJÁŘE IMPLEMENTOVAT
- DEKLARATIVNÍ – SPECIFIKACE NEPOPISUJE DETAILS INTERNÍCH KROKŮ IMPLEMENTACE, POPISUJE VÝSTUP NA ZÁKLADĚ VSTUPŮ DO METODY
- VE VĚTŠINĚ PŘÍPADŮ JE PREFEROVANÝ DEKLARATIVNÍ PŘÍSTUP
- SPECIFIKACE – DEKLARATIVNÍ PŘÍSTUP
 - OBYČEJNĚ KRATŠÍ
 - LÉPE SROZUMITELNÁ
 - NEODHALUJE VNITŘNÍ IMPLEMENTACI, NA KTERÉ BY MOHL PAK KLIENTSKÝ KÓD ZÁVISET A BYL TAK VÍCE CITLIVÝ NA ZMĚNY V IMPLEMENTACI

NÁVRH DOBRÉ SPECIFIKACE

- SPECIFIKACE JE KLÍČOVÝ KONTRAKT MEZI IMPLEMENTÁREM A KLIENTEM
- MĚLA BY BÝT KOHERENTNÍ – METODY BY MĚLY MÍT VŽDY JEDNU JASNĚ DEFINOVANOU ZODPOVĚDNOST
- NÁVRATOVÁ HODNOTA BY MĚLA NĚST JASNOU A JEDNOZNAČNOU INFORMACI
- POKUD MOŽNO, SPECIFIKACE BY MĚLA VYUŽÍVAT ABSTRAKTNÍ TYPY
- ZABRAŇUJE CHYBÁM PŘI VOLÁNÍ ROZHRAŇÍ ZPŮSOBENÝCH RŮZNÝM VÁKLADEM ROZHRAŇÍ
- MĚLA BY BÝT SROZUMITELNÁ
- PŘIPRAVENA NA ZMĚNY

MICROSERVICES

- ARCHITEKTONICKÝ STYL ZNÁMÝ JAKO MICROSERVICE ARCHITECTURE
 - SADA LOOSELY COUPLED SLUŽEB
 - ČÁSTI KÓDU, KOMPONENTY BĚŽÍCÍ V SAMOSTATNÝCH PROCESECH Z HLEDISKA OS
 - KOMPONENTY MEZI SEBOU KOMUNIKUJÍ SÍŤOVÝM PROTOKOLEM NAPŘ. REST NEBO POUŽÍVAJÍ MESAGING
 - KOMPONENTY SAMOSTATNĚ NASAZOVANÉ, MOŽNO MĚNIT POUZE ČÁSTI SYSTÉMU



Převzato z <http://microservices.io/patterns/microservices.html>

MICROSERVICES VÝHODY

- KAŽDÁ MICOSERVICE JE RELATIVNĚ MALÁ
 - SNADNĚJI LZE POCHOPIT JEJÍ IMPLEMENTACE
 - IDE JE RYCHLEJŠÍ
 - APLIKACE STARTUJE RYCHLEJI
- KAŽDÁ MICROSERVICE MŮŽE BÝT NAsAZENA DO RUNTIME NEZÁVISLE
- LEPŠÍ ŠKÁLOVATELNOST ŘEŠENÍ
- LEPŠÍ IZOLACE PROTI VÝPADKŮM (NAPŘÍKLAD OUT OF MEMORY ERRORS)
- LÉPE SE PRACUJE S NAsAZENÍM NOVÝCH KNIHOVEM, FRAMEWORK

MICROSERVICES - NEVÝHODY

- VÝVOJÁŘ MUSÍ PRACOVAT S VĚTŠÍ SLOŽITOSTÍ DISTRIBUOVANÉHO SYSTÉMU
- NÁROČNĚJŠÍ TESTOVÁNÍ
- MUSÍ BÝT IMPLEMENTOVANÝ MEZI SERVISNÍ KOMUNIKAČNÍ MECHANISMUS
- OBTÍŽNÁ IMPLEMENTACE TRANSAKČÍ, ZEJMÉNA POKUD NEJSOU K DISPOZICI DISTRIBUOVANÉ TRANSAKCE
- OBTÍŽNÁ IMPLEMENTACE PŘÍPADŮ UŽITÍ, POKUD SE ROZPROSTÍRAJÍ PŘES VÍCE MICROSERVICE
- VĚTŠÍ KOMPLEXITA NASAZENÍ SYSTÉMU
- ZVÝŠENÉ NÁROKY NA OPERAČNÍ PAMĚŤ

SPECIFIKACE A MICROSERVICES

- SPECIFIKACE JE KLÍČOVÁ V MICROSERVICES ARCHITEKTURE
- IMPLEMENTACE SLUŽEB JSOU OBYČEJNĚ NAVRHOVÁNY RŮZNÝMI TÝMI, PROTO MUSÍ BÝT KLADENÝ VELKÝ DŮRAZ NA SPECIFIKACE ROZHRANÍ
- API MUSÍ MÍT VELMI DOBŘE DEFINOVANOU SÉMANTIKU A SCHÉMA VERZÍ
- V PŘÍPADĚ MICROSERVICES JE DŮLEŽITÉ ODLIŠOVAT DVA DRUHY API
- VEŘEJNÉ API – VOLÁNÍ KLIENTEM
- BACKEND API – MEZI SERVISNÍ VOLÁNÍ

WEB API

- WEB API JE APPLICATION PROGRAMMING INTERFACE NA WEBOVÉM SERVERU NEBO WEBOVÉM KLIENTU
- SERVER-SIDE WEB API SESTAVÁ Z JEDNOHO NEBO VÍCE ENDPOINTŮ
- REQUEST-RESPONSE KONCEPT
- DATOVÝ FORMÁT JSON NEBO XML
- VE VĚTŠINĚ ŘEŠENÍ JE POSTAVENÝ NA HTTP(S), VYUŽÍVÁ HTTP RESPONSE KÓDŮ
- MNOHO ŘEŠENÍ VÍCE NEŽ SOAP WEB SERVICES VYUŽÍVÁ RESTFUL WEB RESOURCES

STANDARD OPENAPI

- DEFINUJE STANDARDNÍ, NA JAZYKU NEZÁVISLÉ ROZHRANÍ
- REST API
- UMOŽŇUJE JAK STROJI, TAK ČLOVĚKU PROHLÍŽET A POROZUMĚT SLUŽBĚ A JEJÍMU VÝZNAMU ANIŽ BY MUSEL ZKOUMAT ZDROJOVÝ KÓD NEBO DOKONCE SLEDOVAT SÍŤOVÝ PROVOZ
- DOBŘE NAPSANÁ SPECIFIKACE UMOŽNÍ KONZUMENTOVI SLUŽBY POROZUMĚT A INTERAGOVAT SE VZDÁLENOU SLUŽBOU S MINIMÁLNÍM MNOŽSTVÍM IMPLEMENTAČNÍ LOGIKY
- OPENAPI DEFINICE MŮŽE BÝT POUŽITA
 - PRO GENEROVÁNÍ KLIENTSKÉHO I SERVEROVÉHO KÓDU PRO RŮZNÉ PROGRAMOVACÍ JAZYKY
 - GENEROVÁNÍ TESTOVACÍCH NÁSTROJŮ A TESTOVACÍCH PŘÍPADŮ
 - GENEROVÁNÍ DOKUMENTACE

SWAGGER

- FRAMEWORK PRO DEFINICI ROZHRANI PODLE SPECIFIKACE OPENAPI
- POKRÝVÁ CELÝ ŽIVOTNÍ CYKLUS
 - NÁVRH
 - DOKUMENTACE
 - TESTOVÁNÍ
 - DEPLOYMENT
- SWAGGER EDITOR - NÁVRH
- SWAGGER CODEGEN – BUILD
- SWAGGER INSPECTOR - TESTOVÁNÍ
- SWAGGER UI - DOKUMENTACE

SWAGGER EDITOR

- STRUKTUROVANÁ DEFINICE ROZHRAŇÍ
- JSON
- YAML
- EDITAČNÍ OKNO, OKNO S PRŮBĚŽNÝM ZOBRAZENÍM GENEROVANÉ DOKUMENTACE
- MOŽNOST TESTOVÁNÍ SPECIFIKACE
- MOŽNOST GENEROVÁNÍ KÓDU SERVEROVÉ ČÁSTI
- MOŽNOST GENEROVÁNÍ KÓDU KLIENTSKÉ ČÁSTI
- GENEROVÁNÍ DO VELKÉHO MNOŽSTVÍ PROGRAMOVACÍCH JAZYKŮ A FRAMEWORKŮ

APIARY

- APIARY – OBDOBNÝ NÁSTROJ PRO DEFINICI ROZHRAŇÍ
- PODOBNĚ JAKO SWAGGER – EDITOR STRUKTUROVANĚHO POPISU ROZHRAŇÍ
- DOKUMENTACE
- TESTOVÁNÍ

The screenshot displays the APIary editor interface in a web browser. The browser address bar shows the URL `https://app.apiary.io/tomasmayer/editor`. The user profile is `tomasmayer` (TomasMayer • tomasmayer). The interface includes a navigation bar with tabs for `Documentation`, `Inspector`, `Editor` (active), `Tests`, `People`, and `Settings`. Below the navigation bar, there are utility buttons: `Link this Project to GitHub`, a settings icon, a button with `A`, a `Valid document` status indicator, a `Preview` button, a toggle switch labeled `On`, and a `Save` button.

The main editor area shows a REST API definition for a `Questions Collection` endpoint. The definition includes the following details:

- `FORMAT: 1A`
- `HOST: http://polls.apibluprint.org/`
- `# tomasmayer`
- Description: `Polls is a simple API allowing consumers to view polls and vote in`
- Endpoint: `## Questions Collection [/questions]`
- Method: `### List All Questions [GET]`
- Response: `+ Response 200 (application/json)`

The response body is a JSON array of question objects:

```
[
  {
    "question": "Favourite programming language?",
    "published_at": "2015-08-05T08:40:51.620Z",
    "choices": [
      {
        "choice": "Swift",
        "votes": 2048
      },
      {
        "choice": "Python",
        "votes": 1024
      },
      {
        "choice": "Objective-C",
```

REFERENCE

- [HTTPS://OCW.MIT.EDU/ANS7870/6/6.005/s16/CLASSES/06-SPECIFICATIONS/SPECS/](https://ocw.mit.edu/ans7870/6/6.005/s16/classes/06-specifications/specs/)
- [HTTPS://OCW.MIT.EDU/ANS7870/6/6.005/s16/CLASSES/07-DESIGNING-SPECS/](https://ocw.mit.edu/ans7870/6/6.005/s16/classes/07-designing-specs/)
- BEHAVIORÁLNÍ EKVIVALENCE
 - [HTTP://WWW.CS.MCGILL.CA/~HV/ARTICLES/DISCRETEEVENT/P82-YUCESAN.PDF](http://www.cs.mcgill.ca/~hv/articles/discreteevent/p82-yucesan.pdf)
- [HTTP://MICROSERVICES.IO/PATTERNS/MICROSERVICES.HTML](http://microservices.io/patterns/microservices.html)