Dnes

bude plno věcí

- Objektový návrh,
- UML,
- hygiena programování,
- návrhové principy.

Objektový návrh

... aneb jak bychom to dělali, kdybychom to uměli

- Důležitý u větších projektů, souvisí s dekompozicí.
- Práci je vhodné rozdělit do co nejmenších tříd a objektů (které implementujeme s méně chybami).
- Popíšeme, jak budou třídy a objekty vypadat. Ve které třídě bude jaký atribut, jaká metoda, jak spolu prvky budou komunikovat,...
- pak už zbývá to jen implementovat.
- Objektový návrh můžeme pojmout třeba jako hlavičky tříd a metod a definice atributů.
- Neměl by být těžkopádný a měl by dle možností snadno umožňovat různá přirozená rozšíření!

Příklad

Auta s pískem

- Diskrétní simulace operovala s kalendářem událostí, jádrem, procesy,...
- ... toto jsou vhodné entity pro dekompozici a přirozené kandidáty na třídy.
- Kalendář událostí osadíme metodami `pridej`, `uber`, `prerozvrhni`.
- Přidáme třídy `auto` a `stavbyvedouci`, které budou vyřizovat události. Objekt typu `auto` se bude starat o události související s autem a jízdou, metody `odjed`, `prijed`, kterým parametrem řekneme kam.
- `stavbyvedouci` bude mít informace o dělnících na místě a metody `naloz`, `vyloz`.

Auta s pískem

objektový návrh pokr.

- `udalost` bude třída popisující událost. Bude popisovat `cas`, `ucastnik` a `parametry`. Druhé dva parametry uděláme typem `object`, aby bylo možné do nich předat cokoliv.
- `jadro` bude další třída, která se bude starat o komunikaci kalendáře s procesy. Tedy zjistí, o jakou událost jde, a zavolá toho, kdo se o ni má postarat.
- Bude-li ve hře zúžení (vozovky), vyřešíme je třídou `zuzeni` s frontami aut na obou stranách, které bude `auto` volat metody `prijizdim`, `odjizdim`.

Objektový návrh – poznámky

proč a jak jej budeme cvičit

- My budeme objektové návrhy cvičit na snadnějších problémech (na těžší nemáme čas).
- Budeme zkoušet problém rozdělit na co nejmenší součásti, které dávají smysl.
- První netriviální využití uvidíte u zápočtového programu.
- Budeme se zabývat výhodami a nevýhodami jednotlivých objektových návrhů (rozšiřitelnost, snadnost implementace,...).
- Dosud jsme zapisovali heuristicky, nyní budou vhodné formalismy:
- UML a Data Flow Diagramy.

UML

Unified Modeling Language

- Krabičky představují objekty/třídy,
- jejich prvky atributy a metody,
- šipky značí komunikaci (někoho s někým).

`https://www.lucidchart.com/pages/uml-class-diagram`

Data Flow Diagram

říká, jak data programem putují

- Reprezentuje se olabelovaným orientovaným grafem.
- Vrcholy představují operace (konané úkony),
- hrany jsou olabelovány putujícími daty (mezi vrcholy).

`https://am7s.com/what-is-data-flow-diagram/`

Hygiena programování

je dost široký pojem

- Dva aspekty: Zdravotní a výkonnostní
- Zdravotní: tělesná a duševní.
- Výkonnostní: složitější (bude za chvíli).

Zdravotní hygiena

aneb BOZP

- Je třeba dodržovat BOZP,
- ... a neskončit u kolegů z oboru medicina.
- I psychiatrie se vyučuje na lékařské fakultě.

Somatická rizika

Nemoci z Matfyzáctví

- Sedavá práce $\Rightarrow$ vznik některých rizikových faktorů CoViD 19, konkrétně hypertenze ($\Leftarrow$ stres, nedostatek pohybu a pro Matfyzáky velmi typické nadměrné solení), hypercholesterolemie ($\Leftarrow$ nedostatek pohybu, stres), obezita (DTTO).
- Nevhodné pracovní podmínky (špatné držení těla, nevhodné pracovní pomůcky) $\Rightarrow$ bolesti hlavy, migrény, syndrom karpálního tunelu, záněty šlach,...
- psychosomatické poruchy (dlouhodobě špatná nálada vám zdraví nezlepší, naopak se objevují různé nevysvětlitelné potíže, například bolesti, nevolnost, zažívací potíže,...).

Duševní rizika

Nemoci z Matfyzáctví II

- Prudké výkyvy nálad (souvisí s úspěšností ladění a nejde tedy o deprese),
- různé formy šílenství (obvykle od přepracování) – nutno dávat pozor (na pocit chaosu – který naštěstí obvykle člověku znemožní pokračovat v programování),
- somatoformní poruchy (je-li vám špatně, radost vám to neudělá),
- insomnia, noční děsy – obvykle od stresu. Přiměřený stres je nutné se naučit zvládat (bez stresu se ublahobytníme). Je-li nadměrný, ničí produktivitu naší práce.

Vyhoření, vyhasnutí

alias Z73.0 v katalogu

- Patří do famílie Z73 – Potíže spojené s vedením života. Znáte nejspíše ze základní školy.
- Přichází pomalu a záludně, hrozí vysoce motivovaným pracovníkům.
- Významná posila se časem stává v týmu přítěží (vedoucí mají zájem na dobrém fungování týmu, je tudíž vhodné se na ně obrátit, ale je to nutné udělat včas).
- Pracovník v terminální fázi vyhasnutí se často není schopen k práci vrátit (nikdy). To by byla škoda.
- U nás přichází pomalu (v horizontu let, na Linkách bezpečí prý už v řádu týdnů).

Co s tím?

Všichni jsme vlastně přírodovědci...

- Jsme informatici, tak umíme ladit.
- Člověk nemá tak pěkný debugger jako Visual Studio, ale posílá nám různé zajímavé informace (které je vhodné sledovat).
- Při programování dělat přestávky, nenechávat si práci na poslední chvíli, pracovat průběžně, rovnoměrně a ve vhodných podmínkách.
- Zkrátka to, co vám říkáme od začátku roku. :-)
- Vhodné je vytipovat si nenáročný sport a ten pravidelně provádět, minimalizovat riziko výmluv. Například procházky (třeba do práce a zpět), yoga (nenáročná na prostor), anaerobní cvičení (nenáročné na čas).
- Věnčitý virus je za dveřmi a ptá se, v jaké jste kondici... (váš život, vaše rozhodnutí).

Hygiena programování

výkonový pohled

- Při programování proti sobě máme nepřítele stejně důvtipného, jako jsme sami. (c) Rudolf Kryl <= 1997
- Napíšeme-li kód jak čuně, nikdo se v něm za čas nevyzná (časem ani my – přednášející o tom něco ví :-)).
- Programátora, v jehož kódu se nikdo (další) nevyzná, těžko někdo zaměstná.
- Kód je třeba vhodně dekomponovat (do modulů, tedy souborů, tříd a objektů a funkcí).
- Kód musíme psát efektivně, například eliminovat společné podvýrazy (tedy zbytečně neopisovat totéž mnohokrát), opakovaně vykonávaný kód vylifrovat do funkce,...

Hygiena programování II

jak programovat, abychom se v tom vyznali

- Proměnné je potřeba vhodně pojmenovávat. Pro pojmenování jsou různé konvence. Buďto slova_oddělujeme_podtržítky nebo jeOznačujemeVelkýmiPísmeny. Měli bychom dělat právě jedno z toho (v celém projektu), C# používá to druhé.
- Indentace: V Pythonu povinná, jinde silně doporučena.
- Komentáře: Mají říkat, co není vidět (tedy myšlenky, nikoliv to, co je zjevné z kódu).
- Vhodný komentář: Spočítáme aritmetický průměr hodnot v poli.
- Nevhodný komentář: Do cyklicí proměnné přiřadíme 0 a prolézáme pole (to každý blbec vidí v kódu).
- Půl roku po odevzdání zimního zápočtového programu zkuste tento modifikovat (a něco uvidíte).

SOLID

Pět návrhových principů jak psát kód

- Single responsibility – každá entita má zodpovídat za jednu věc (například sečti čísla v poli, nikoliv sečti čísla v poli, polož slupku od banánu na chodník a zakokrhej),
- Open-closed – entity (tedy prvky) by měly být otevřené pro rozšíření, ale uzavřené změnám (odolné vůči pokusům o změny). Souvisí s dědičností a zapouzdřením.
- Liskov substitution – Místo rodiče lze použít jakéhokoliv (jeho) potomka.
- Interface segregation – Více specifických interfaců je lepších, nežli jeden univerzální (například nastavování atributů v Tkinter v zimě vs. nyní Windows Forms Applications).
- Dependency inversion – Závislost by měla být na abstraktním, ne na konkrétním.

Verzovací systémy

tedy systémy pro správu verzí

- RCS, CVS, SVN, GIT,...
- slouží pro archivaci, sdílení a správu verzí především při vývoji většího projektu.
- Založíte repozitář, nahrajete do něj data.
- Ta si mohou vhodní lidé stáhnout (downloadovat), modifikovat a nahrát zpět do repozitáře (uploadovat).
- Při nahrání změn se přidávají komentáře (co jste udělali).
- Můžete použít například pro letní zápočtový program.
- Budete mít k dispozici všechny verze všech souborů, které tam nahrajete (všechny verze, které commitnete).

Verzovací systémy – pokračování

- Nastudujte minimální instrukční repertoir pro `git`, tedy příkazy `init`, `clone`, `pull`, `commit`, `push`, `add`, `status`.
- Důležité je zálohovat na jiný disk, než na kterém pracujete!
- Například na `gitlab.mff.cuni.cz` najdete tzv. gitlab, tedy git-server i s webovým rozhraním.
- Každý z Fakulty by tam měl mít účet.
- Před uploadem si přečtěte podmínky použití (ať se nevzdáte práv, která si chcete ponechat).

To je všechno!

Dneska už nic dalšího nebude

Děkuji za pozornost.