Hra „Hádání čísla“

Poznámky k ALG2

Cílem předmětu je seznámení s dalšími prostředky Java – třídy, objekty a související témata – Java Collection Framework, Java IO, Java NIO. Tedy snahou je postoupit alespoň částečně k něčemu čemu se říká „objektové programování“, navrhovat vlastní, obecně použitelné, na kontextu nezávislé třídy, používat existující prostředky (třídy) Javy.

Aplikovat odlišný přístup k členění kódu

- návrh dílčích dále použitelných celků, předem jasně definovaný způsob použití daného celku;
- oddělení uživatelského rozhraní (UI) od dat a algoritmů nad těmito daty (DC, DA);
- základní otázka: jak by mělo libovolné UI komunikovat s DC vrstvou?
- návrh dílčích, použitelných celků – tím je myšlený návrh tříd,
- uživatelské rozhraní, většinou komunikace uživatele a programu v konzolovém okně, řádkový vstup a výstup, často řádkové menu.

Základní postup

- **Úloha**, její definice. Co bude na vstupu a co na výstupu programu.
- **Specifikace programu** – toto může být na úrovni návrhu komunikace programu s uživatele, jaké bude schéma menu a co by se mělo stát při jednotlivých volbách – jaká data kdy získá UI (uživatelské rozhraní) od uživatele a co poskytne dále DC vrstvě, co by od ní měl naopak získat. Toto bez vazby na konkrétní datové typy.
- **Návrh vhodných celků tj. tříd pro uchování dat a realizaci algoritmů programu** (DC – data, řízení, algoritmy). Jaké třídy, jaké by měly mít kompetence, jaké funkcionality by měla třída poskytovat – **jak má probíhat komunikace** (třídy nebo její instance) s okolím – určíme metody „rozhraní třídy“, tj. navrhneme veřejné metody včetně konstruktoru/ů – statické/instanční, parametry, návratový typ.
- Nyní je možné doplnit/**implementovat komunikaci UI (uživatelské rozhraní) s DC (data, algoritmy)** – doplnit jednotlivé části hlavního programu.
- Nyní je na řadě vlastní **implementace třídy** – privátní data, veřejné metody, případný kód členit a využit privátní metody (členění kódu; část kódu, kterou potřebujeme provádět na různých místech kódu třídy třeba i s jinými daty apod.), kód včetně kontroly a ošetření případných chybových stavů. Algoritmy vůbec nemusí být optimální, nutně ale musí být jednoduché, dobře čitelné tak, aby byly v případě potřeby snadno modifikovatelné.
- V průběhu implementace jednotlivých metod třídy **testovat**, zda jednotlivé dílčí metody dělají to, co a jak mají (poskytují správný výsledek) – pro testování lze použít různé přístupy. Testování dílčích jednotek – unit testy.
- Je možné zkusit **testovat celek**. Alfa verze – alfa testování pouze vývojáři – odstranění hlavních chyb.
- Na základě dosavadní verze **oprava chyb**, úprava nevhodného kódu. Nyní je na řadě podívat se, zda jsme naplnili zadání, je potřeba něco doplnit, možná doplnit ošetření chybových stavů – je nutné rozhodnout co ošetřovat a jak, z čeho se má program vzpamatovat a umožnit další použití a kdy má informovat o nastalých problémech a skončit (protože se z dané chyby nemůže dost dobře zotavit).
- Máme k dispozici **beta verzi**. Podstupujeme program k beta testování.
- Na základě testování, provedeme opravy chyb, úpravy a vytvoříme první **produkční verzi** našeho SW.

Od první funkční verze je možné se podívat na případnou optimalizaci potřebných částí kódu, provést refaktoring. Pokud jsme dobře provedli návrh, měníme zpravidla pouze vnitřní implementaci, nemusíme od okamžiku návrhu ale nijak měnit to, co bylo navržené jako základní „rozhraní našich tříd“ případně, je-li to účelné, ho pouze doplňujeme.

Zadání úloh

1. Naši cílem je vytvořit prostředky pro program, který bude uživateli umožňovat hrát hru „hádání čísla“. Program jsme realizovali již v ALP1 – program sestával z jediné třídy, komunikace s uživatelem formou řádkového menu, možnost volby pozice hráče, vhodné členění do metod.

   Nyní obdobnou úlohu vyřešíme znovu v souladu s výše uvedeným postupem ve snaze uplatnit alespoň základní objektový přístup. V konečné fázi tak vytvoříme program s třídou hlavního programu (pojmenované např. `HadaniCislaApp`) a se dvěma třídami (pojmenovaných např. `HadamCislo`, `MyslimSiCislo`), které uchovávají data konkrétních her a poskytují prostředky pro jejich hraní (poskytují akce nad/s těmito daty).

   Základní zásady: Vhodně navrhnout třídy `HadamCislo`, `MyslimSiCislo` tak, aby byly zcela nezávislé na uživatelském rozhraní a poskytovaly všechny prostředky pro hraní příslušných her. Návrh metod včetně dokumentačních komentářů a ošetření neplatných hodnot parametrů, tj. v případě, že jsou metodě předané neplatné hodnoty parametrů, vytvoříme a „vyhodíme“ objekt vhodné třídy výjimek například třídy `IllegalArgumentException` (včetně popisu chyby a informací o hodnotě chybného argumentu).

2. Program, který bude nabízet hru „šibenice“ tj. postupné odkrývání tajenky. Schéma by mohlo být obdobné jako u „hádání čísla“.