Zápočtová práce z předmětu „Algoritmizace a programování“

Vaším úkolem je v jazyce Java implementovat program orientovaný na manipulaci s daty uchovanými v diskových souborech: *systém pro sportovní akce definovaného zaměření (týmy, hráči, plán vzájemných utkání, průběžné zpracování výsledků, hodnocení jednotlivých hráčů, podpora atletických závodů ve víceboji apod.); třídní kniha (třídy, žáci, předměty, průběžné a výsledné hodnocení); zpracování vymezené části studijní agendy (evidence přihlášených studentů, dílčích testů a celých přijímacích zkoušek; výsledky studentů a výpočet výše stipendia dle stanovených kritérií); objednávkový systém (obědy, časopisy apod.); výpůjční systém (knihovna, půjčovna nářadí, aut apod.); rezervační systém (ubytování, doprava, restaurační služby); evidenční systém pro různé oblasti (účty, skladové hospodářství, zpracování přihlášených studentů a výsledků zkoušek …); podpora dílčí části výrobního procesu firmy; výpočty vybraných typů daní; meteorologická data a jejich zpracování; generování testů dle zadané struktury (podpora opravy testů, hodnocení, …)*. **Program by měl řešit dílčí část běžné reality, měl by být smysluplný a to jak co do vymezení dané části reality, tak co do jejího řešení.**

Jak postupovat, co všechno odevzdat?

- Práci budete odevzdávat prostřednictvím **GitHub.**
- Pro realizaci zápočtové práce si vytvořte prázdný projekt, inicializujte git a stáhněte si obsah vzdáleného repozitáře atd., obdobně jako to děláte u jednotlivých cvičení v tomto semestru.
- Po stažení obsahu vzdáleného repozitáře máte v adresáři projektu mimo jiné i podadresáře `src`, `doc`, `data`. V každém z nich je soubor `*.info` s informací, co zde má být.
- Vámi realizovaná práce by pak měla být ve třech základních adresářích (stažené soubory neměňte):
  - `src` – zde budou zdrojové soubory (soubory tříd) vašeho programu; ve svém projektu opět, jako u jednotlivých cvičení, vytvořte balík pojmenovaný svým příjmením (malými písmeny bez diakritiky), zde dva „podbalíky“ jede pro třídu/třídy UI, ve druhém by pak měl být systém tříd řešících zadanou úlohu;
  - `data` – zde by měly být datové soubory použitelné pro vyzkoušení funkcionalit vašeho programu a to minimálně dvě sady (popřípadě více, pokud je to vhodné pro testování programu) s různými daty;
  - `doc` – v tomto adresáři bude doprovodná dokumentace vašeho programu.
- Pokud navíc použijete nějakou další knihovnu, vytvořte v adresáři projektu i podadresář `lib`, do kterého tuto knihovnu umístíte. Obsah uvedených tří (`src`, `data`, `doc`) nebo čtyř (`src`, `data`, `doc`, `lib`) podadresářů pak nahrajete do repozitáře na github.
- **Realizujte program** tak, aby splňoval dále uvedené požadavky (viz druhá strana tohoto dokumentu).
- Třídy řešící zadanou úlohu (tj. kromě UI) by měly být doplněné o dokumentační komentáře. Z těchto dokumentačních komentářů **vytvořte „programátorskou dokumentaci“**. Pro její vytvoření byste měli mít podporu ve vývojovém prostředí. Tuto dokumentaci zabalte do jediného ZIP souboru a zkopírujte do adresáře `doc`. Do názvu souboru promítněte své příjmení a název programu; já bych svůj dokument nazvala například `kralovcova_zavod_javadoc.zip`.
- Dále **vytvořte dokument, ve kterém popíšete** vaši realizaci. Zde by mělo být: titulní list s názvem práce a indentifikací autora apod., popis úlohy (zadání, co je cílem/smyslem), jaké konkrétní akce podporuje výsledný program, s jakými soubory program pracuje, popis formátu jednotlivých souborů, implementovaný systém tříd a jejich závislosti (graficky), popis kompetencí jednotlivých tříd. Vše vhodně formátované, pokud možno bez stylistických a pravopisných chyb. Dokument uložte ve formátu PDF do adresáře `doc`. Do názvu souboru promítněte své příjmení a název programu; já bych svůj dokument nazvala například `kralovcova_zavod_popis.pdf`.
- Pokud realizujete dle pokynů, budete mít vše připravené k odevzdání. **Nahrajte požadované soubory do vzdáleného repozitáře na github.com vytvořeného pro zápočtovou práci.**

Co má obsahovat výsledný program?

- Výsledný kód musí být kompatibilní s Java17.
- **Oddělení uživatelského rozhraní od vlastní logiky programu** (tj. dat a algoritmů) – oddělit na úrovni nejenom tříd, ale i balíků.
- Textové uživatelské rozhraní formou řádkového menu. Menu by mělo vždy nabízet (a program umožnit vykonávat) pouze v daném okamžiku relevantní akce (tj. realizujte kontextové / kontextově závislé menu). Pokud má někdo zájem zabývat se tvorbou grafického uživatelského rozhraní, potom by to nemělo být na úkor ostatních požadavků.
- Program by měl nabízet **komunikaci s uživatelem minimálně ve dvou různých jazycích**. Pro vícejazyčný program využijte prostředky, které jsou součástí Javy – tj. na jedné straně vytvořte soubor „properties“ se všemi různými komunikačními texty a jeho varianty v podporovaných jazycích, na druhé straně v programu (v kódu) získávejte příslušné textové řetězce prostřednictvím instance `ResourceBundle`.
- Vhodně **členění programu** (balíky, třídy, rozhraní). Jednotlivé části programu by měli být minimálně ve dvou balících. V jednom z těchto balíků by měla být třída uživatelského rozhraní. Dobře promyslet a rozvrhnout kompetence jednotlivých tříd a jejich vzájemné souvislosti.
- Využití možností **objektového návrhu**, vhodné rozdělení do tříd. Dobře promyslet a rozvrhnout kompetence
- **Použití některé/některých kontejnerových tříd jazyka Java** (`LinkedList`, `ArrayList`, `HashSet`, `HashMap`, …).
- **Korektní a smysluplná práce se soubory a adresáři**.
- V programu by se neměly vyskytovat konkrétní jména souborů či adresářů pevně zavedená v kódu. V některých případech ale může program pracovat pouze se soubory určitých typů, nebo nějak systematicky pojmenovaných (jako tomu bylo v programu Banka) potom by tyto pevně dané části měly být v zavedených/pojmenovaných konstantách apod. – respektive ve finálních atributech na vhodném místě.
- Zobrazení jisté množiny dat (zobrazení uživateli, popřípadě ukládání do souborů) – nabídnout/realizovat **různá kritéria uspořádání**, popřípadě filtrování dat. Pro uspořádání dle různých kritérií využijte integrované metody řazení (sort) spolu s interface `Comparable` a `Comparator`). Pokud je to smysluplné, potom umožněte řazení i dle textové informace – v tomto případě využijte vhodně inicializovanou instanci `Collator` pro uspořádání dle pravidel příslušného jazyka (dle českých zvyklostí).
- Doplňte **dokumentační komentáře** – dokumentační komentáře jednotlivých tříd, atributů, metod. Součástí dokumentačního komentáře každé třídy musí být jméno autora, základní význam a kompetence třídy. V dokumentačním komentáři metod uvést co metoda dělá, jaké má parametry a jejich význam, co a za jakých podmínek vrací, jaké a v jakých případech „vyhazuje“ výjimky.
- **Ošetření chybových/problémových/nestandardních stavů programu.** Vhodná/korektní práce s výjimkami. Na straně algoritmů/logiky úlohy identifikujte potenciální chybové stavy s případných vyhozením výjimky vhodného typu.
- Na straně **UI by mělo být zachycení, ošetření výjimek**, zotavení programu po nastalé chybě (data programu by po chybě neměla zůstat v nekonzistentním tvaru). Pomocí systému výjimek neošetřujeme logické chyby programu – ty je potřeba odladit/odstranit. Zvažte možnost logování programu: zaznamenávání jednotlivých základních aktivit programu, v případě nestandardní situace (výjimky) zapsat do logu o jaký problém se jedná, v jaké fázi provádění kódu byl program (metoda `printStackTrace()` nastalé výjimky), aktuální obsah dat programu.
- Využití regulárního výrazu (popřípadě regulárních výrazů) pro vhodné účely vítáno.