Záverečný test
Zadanie

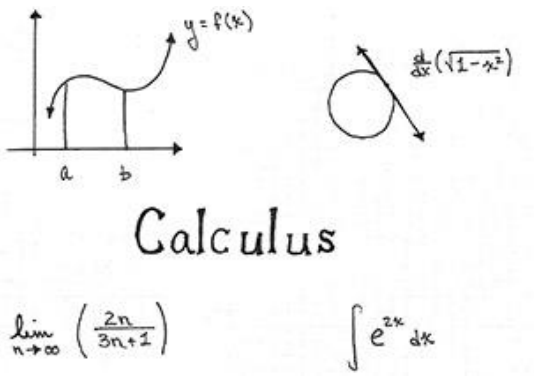
Ústav informatiky
Prírodovedecká fakulta
UPJŠ v Košiciach

Dvakrát meraj (rozmýšľaj), raz rež (programuj)

Dôležité pravidlá a informácie (viac na stránke predmetu)**:**

- čas na riešenie úloh je **240 minút**,
- nie je dovolená žiadna (elektronická aj neelektronická) komunikácia s kýmkoľvek okrem dozoru
- v prípade akýchkoľvek problémov alebo z dôvodu ohodnotenia riešenia kontaktujte dozor,
- riešenia je možné nechať si ohodnotiť aj priebežne,
- **funkčnosť každej metódy musí byť preukázaná spustením na vami vytvorenom testovacom vstupe, nespustiteľné metódy neumožňujú zisk príslušných bodov,**
- všetky inštančné premenné musia byť neverejné.

Matematická analýza



Motivácia: Pre mnohých študentov informatiky je (nielen) predmet *Matematická analýza* poriadny strašiak – veď kto by sa nezľakol, keď prvý raz uvidí ozajstnú matematiku. Prvák Jožko nie je výnimkou. Rozhodol sa ale pristúpiť k štúdiu obzvlášť zodpovedne. Na *YouTube* začal hľadať videá o *prokrastinácii* a *časovom manažmente*. Z rôznych metód si nakoniec vybral metódu ToDo zoznamov. Založil si ToDo zápisník, v ktorom si zaznamenáva študijné úlohy vo forme ToDo položiek. Z každej prednášky a cvičenia si do ToDo zápisníka zapíše niekoľko študijných úloh ako nesplnené ToDo položky – napríklad naštudovať *Lagrangeovu vetu o strednej hodnote*, samostatne si preriešiť *Príklad 4.6.21*, atď. Teda každej študijnej úlohe na začiatku prislúcha jedna nesplnená ToDo položka v zápisníku. Keď potom ide samostatne študovať, z ToDo zápisníka si vyberie nejakú nesplnenú ToDo položku. Jej prislúcha nejaká študijná úloha. Potom na úlohe pracuje a do zápisníka si k príslušnej ToDo položke poznačí, kedy začal a ako dlho pracoval. Ak sa mu úlohu nepodarí splniť celú resp. s výsledkom nie je spokojný, napíše si ju opäť ako novú ToDo položku **na koniec** zoznamu - aby nezabudol, že sa má na ňu ešte pozrieť. Jednej študijnej úlohe tak môže v zápisníku prislúchať viacero ToDo položiek, pričom najviac jedna z nich je nesplnená. Keď už má to PAZko, rozhodol sa, že by z klasického zápisníka mohol prejsť na niečo modernejšie a vytvoriť si app-ku.

Pohľad analytika: Pri implementácii budeme potrebovať:

- triedu `ToDoPolozka`, ktorá reprezentuje (splnenú alebo nesplnenú) ToDo položku a uchováva informácie o prislúchajúcej študijnej úlohe ako aj o plnení samotnej ToDo položky,
- triedu `ToDoZapisnik`, ktorá bude uchovávať zoznam ToDo položiek.

Zadanie: V balíku `sk.upjs.finalTerm` vytvorte triedu `ToDoPolozka` obsahujúcu dátové položky prístupné cez gettre (a podľa uváženia aj modifikovateľné cez settre):

- **`tyzden`** (týždeň semestra, ktorého sa týka študijná úloha prislúchajúca k položke),
- **`tema`** (názov témy, ktorej sa týka študijná úloha prislúchajúca k položke, napr. *„Inflexné body“, „Silne neohraničené funkcie“*, …),
- **`typ`** (typ úlohy, resp. toho, čomu sa venuje študijná úloha prislúchajúca položke, napr. *„veta“, „dôkaz“, „definícia“, „príklad“, „úloha“*, ...),
- **`nazov`** (názov prislúchajúcej študijnej úlohy, napr. *„Veta 1.5.8.“*, *„Definícia 1.4.4.“*, …) ,
- **`datum`** (dátum v tvare `dd.mm.rrrr`, kedy sa začala plniť ToDo položka),
- **`cas`** (čas v tvare `hh:mm`, kedy sa začala plniť ToDo položka),
- **`trvanie`** (trvanie plnenia ToDo položky v minútach).

Upozornenie: Zadanie pre triedu `ToDoPolozka` predpisuje dátové položky prístupné cez gettre. Aké privátne inštančné premenné použijete na uloženie týchto dátových položiek je na vašom rozhodnutí.

Ďalej vytvorte aj triedu `sk.upjs.finalTerm.ToDoZapisnik`, ktorá bude uchovávať zoznam ToDo položiek súvisiacich s predmetom (v našom prípade s predmetom *Matematická analýza*).

Konštruktory a pridávanie ToDo položiek (3 body dokopy – povinné):

- `public ToDoPolozka(int tyzden, String tema, String typ, String nazov)` – použije sa na vytvorenie nesplnenej ToDo položky,
- `public ToDoPolozka(int tyzden, String tema, String typ, String nazov, String datum, String cas, int trvanie)` – použije sa na vytvorenie už splnenej ToDo položky,
- `public void pridaj(ToDoPolozka polozka)` – inštančná metóda v triede `ToDoZapisnik`, ktorá pridá položku do ToDo zápisníka.

Práca so súbormi (povinné):

V triede `ToDoPolozka`:

- `public static ToDoPolozka zoStringu(String popis)` – statická metóda, ktorá vráti referenciu na novovytvorený objekt triedy `ToDoPolozka`. Parameter je `String` v tvare "tyzden `\t` tema `\t` typ `\t` nazov", resp. "tyzden `\t` tema `\t` typ `\t` nazov `\t` datum `\t` cas `\t` trvanie", ak ToDo položka už bola splnená (3 body);
  Poznámka: Znak `\t` je neviditeľný znak tabulátora. `Scanner`-u môžete povedať, že oddeľovač má byť tabulátor zavolaním jeho metódy `useDelimiter("\t").`
- `public String toString()` – vráti reťazec vhodne reprezentujúci údaje v ToDo položke (1 bod).

V triede `ToDoZapisnik`:

- `public static ToDoZapisnik zoSuboru(String nazovSuboru)` – statická metóda, ktorá z uvedeného súboru prečíta zoznam ToDo položiek, pričom v každom riadku bude popis jednej ToDo položky (4 body).
- `public void uloz(String nazovSuboru)` – uloží všetky ToDo položky v zápisníku do súboru v tvare, ktorý vie spracovať metóda `zoSuboru(String nazovSuboru)` (3 body).
- `public String toString()` – vráti reťazec vhodne reprezentujúci obsah zápisníka (1 bod).

Inštančné metódy triedy `ToDoZapisnik`:

Ak niektorá z metód nevie vrátiť referenciu na objekt s požadovanými vlastnosťami, metóda nech vráti `null`.

- `public int vratCasStudia()` - vráti celkový čas v minútach, ktorý bol venovaný štúdiu – plneniu ToDo položiek (1 bod).
- `public List<ToDoPolozka> vratNesplnenePolozky()` – vráti zoznam nesplnených študijných úloh ako zoznam im prislúchajúcich nesplnených ToDo položiek (1 bod).
- `public List<String> vratStudijneTemy()` – vráti zoznam tém, ktoré sú v jednotlivých študijných úlohách, pričom každá z tém je v zozname len raz (3 body).
- `public ToDoZapisnik vratZapisnikZTyzdna(int tyzden)` – vráti referenciu na novovytvorený ToDo zápisník obsahujúci len ToDo položky týkajúce sa zadaného týždňa semestra – vzájomné poradie ToDo položiek musí ostať zachované (3 body).
- `public ToDoPolozka najdiNajdlhsiuPolozku()` – vráti referenciu na splnenú ToDo položku, ktorej plnenie trvalo najväčší čas (3 body).
- `public boolean overPripravenostNaTest(int[] tyzdne)` – vráti, či je príprava na písomku z obsahu zadaných týždňov semestra ukončená, t.j. nemáme žiadne nesplnené ToDo položky súvisiace s týmito týždňami semestra (4 body).

- `public List<ToDoPolozka> vratRozpracovaneUlohy()`– vráti zoznam nesplnených študijných úloh (ako zoznam im prislúchajúcich nesplnených ToDo položiek), ktoré už ale boli (čiastočne) plnené – študijná úloha je identifikovaná týždňom, témou, typom a názvom (5 bodov).
- `public Map<String, Integer> analyzujCasPodlaTypu()` – pre každý typ úlohy vráti celkový čas v zaokrúhlených celých percentách strávený riešením úloh tohto typu (6 bodov).
- `public boolean indikujNocnehoStudenta()` – vráti `true` práve vtedy, ak aspoň 50% ToDo položiek sa začalo riešiť v čase medzi 22:30 a 5:30 (6 bodov).
- `public String najdiNajtazsiDen()` – vráti dátum, v ktorom sa strávilo najviac času plnením študijných úloh (6 bodov). Za zohľadnenie trvania ToDo položiek, ktoré prechádzajú cez polnoc plus 5 bodov.
- `public ToDoPolozka najdiNajrozpracovanejsiaUlohu()` – vráti referenciu na nesplnenú ToDo položku, ktorá prislúcha nesplnenej študijnej úlohe, ktorej plnením sa zatiaľ celkovo strávilo najviac času – úloha je identifikovaná týždňom, témou, typom a názvom (9 bodov).
- `public List<ToDoPolozka> optimalnyStudijnyPlan()` – vráti usporiadaný zoznam nesplnených ToDo položiek v poradí, v akom by mali byť plnené so zohľadnením prerekvizít. Pri tejto úlohe predpokladáme, že ToDo položky študijných úloh boli po prvý raz zapisované do zoznamu v poradí zohľadňujúcom prerekvizity a teda, ak nejaká študijná úloha vyžaduje nejaké prerekvizity (znalosti), tie sú predmetom už skôr po prvý raz zapísanej študijnej úlohy. Zároveň predpokladáme, že ToDo položka prislúchajúca študijnej úlohe, ktorá sa začala plniť, no nebola splnená, sa pripisuje stále na koniec zoznamu (9 bodov).

Triedenie a komparátor (dokopy 5 bodov):

Vytvorte triedu `ChronoKomparator` implementujúcu `java.util.Comparator<ToDoPolozka>` s metódou (3 body):

- `public int compare(ToDoPolozka polozka1, ToDoPolozka polozka2)` – porovná ToDo položky chronologicky podľa začiatku plnenia.

V triede `ToDoZapisnik` implementujte inštančnú metódu (2 body):

- `public List<ToDoPolozka> vratPolozkyPodlaCasu()` – vráti zoznam splnených ToDo položiek usporiadaných chronologicky podľa času začiatku plnenia.

Výnimky (3 body)

Vytvorte nekontrolovanú výnimku `NeexistujuciTyzdenException` a vhodne ju použite aspoň v jednej metóde.