Záverečný test
Zadanie

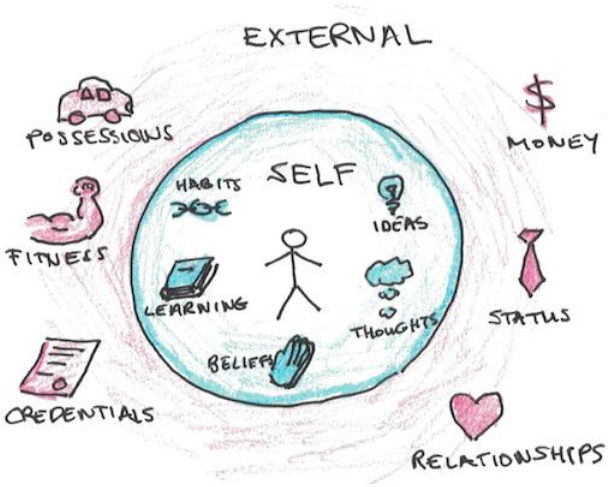
Ústav informatiky
Prírodovedecká fakulta
UPJŠ v Košiciach

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 (nie až v závere testu),
- **funkčnosť každej metódy musí byť preukázaná spustením na vami vytvorenom testovacom vstupe, nespustiteľné metódy neumožňujú zisk príslušných bodov,**
- všetky inštančné premenné musia byť neverejné.

Novoročné predsavzatia & self improvement



zdroj: https://www.scotthyoung.com/

Motivácia: Nový rok je za nami a Elemír si už teraz uvedomil, že sa vykašľal na všetky novoročné predsavzatia. Naštudoval si ako sa v živote posúvať aj po malých krokoch. Na to aby si vybudoval návyk potrebuje 80 až 100 dní, mnoho činností môže kombinovať. Pri umývaní zubov môže cvičiť nohy alebo balans alebo iné aktivity. Pri ceste do a z obchodu je škála samozdokonaľovacích aktivít tiež široká môže sa učiť ďalší jazyk, cvičiť s nákupom alebo počúvať vzdelávací podcast. Počas čítania skript môže jazdiť na rotopede. Ale aktivity ako silové cvičenie alebo naučiť sa variť, krátky poobedný spánok (power nap) sa kombinujú ťažšie. Existuje mnoho aplikácii ktoré mu vedia pomôcť ale vždy sú zamerané iba na jeden druh aktivít, cvičenie, kurz varenia, cudzí jazyk. Aby svoj progres vo všetkých smeroch mohol sledovať rozhodol sa naprogramovať si pomocníka, kde by si zapisoval všetky svoje drobné činnosti k sebazdokonaľovaniu.

Pohľad analytika: Pri implementácii aplikácie budeme potrebovať:

- triedu `Aktivita`, ktorá reprezentuje jednu vykonanú aktivitu,
- triedu `PlanRozvoja`, ktorá bude uchovávať zoznam vykonaných aktivít.

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

- **`nazov`** (názov aktivitý ktorej sa venoval napr. cvičenie lýtok, učenie sa Španielčiny, jazda na rotopede alebo iné)
- **`datum`** (dátum, kedy sa aktivite venoval, vo formáte DD.MM.RRRR, napr. "05.12.2021" – v dátume sú vždy úvodné nuly doplnené tak, aby deň aj mesiac boli dvojciferné),
- **`casZaciatku`** (čas, kedy začal aktivitu, vo formáte HH:MM, napr. "13:05", v čase sú vždy úvodné nuly doplnené tak, aby sa každá časť časového reťazca skladala z 2 cifier, resp. bola dvojciferná),
- **`dlzka`** (čas, koľko minút sa venoval aktivite),
- **`pridruzenie`** (činnosť ku ktorej pridal samozdokonaľovaciu aktivitu napr. umývanie zubov, cesta do obchodu alebo „žiadna“ ak sa venoval iba aktivite)
- **`typ`** ("fyzicka", "mentalna", "oddychova", "strava" alebo iné podľa toho o aký typ ide)
- **`aplikacia`** (názov mobilnej aplikácie ktorú použil napr. na cvičenie, alebo kurz varenia)
- **`detail`** (detail z aplikácie napríklad názov cvičenia, level cvičenia, recept ktorý varil alebo iné)

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

Nie na všetky aktivity sa využívajú mobilné aplikácie a preto nemajú vyplnené položky aplikácia a detail z aplikácie.

Ďalej vytvorte aj triedu `sk.upjs.finalTerm.PlanRozvoja`, ktorá bude uchovávať zoznam aktivít.

Konštruktory a pridávanie aktivit do zoznamu (3 body dokopy – povinné):

- `public Aktivita(String nazov, String datum, String casZaciatku, int dlzka, String pridruzenie, String typ, String aplikacia, String detail)` – použije sa na vytvorenie záznamu o aktivite s použitím mobilnej aplikácie.
- `public Aktivita(String nazov, String datum, String casZaciatku, int dlzka, String pridruzenie, String typ)` – použije sa na vytvorenie záznamu bez aplikácie.
- `public void pridaj(Aktivita aktivita)` – inštančná metóda v triede `PlanRozvoja`, ktorá pridá vykonanú aktivitu do zoznamu aktivít.

Práca so súbormi (povinné):

V triede `Aktivita`:

- `public static Aktivita zoStringu(String popis)` – statická metóda, ktorá vráti referenciu na novovytvorený objekt triedy `Aktivita`. Parameter je `String` v tvare "`nazov \t datum \t casZaciatku \t dlzka \t pridruzenie \t typ \t aplikacia \t detail`", resp. "` nazov \t datum \t casZaciatku \t dlzka \t pridruzenie \t typ`", ak ide o aktivitu bez použitia aplikácie (3 body);
  Poznámka: Znak `\t` je neviditeľný znak tabulátora. `Scanner`-u môžete povedať, že oddeľovač má byť tabulátor zavolaním jeho metódy `useDelimiter("\t").` Medzera pre a za `\t` sú len kvôli zlepšeniu čitateľnosti zadania, v reťazci reálne nie sú.
- `public String toString()` – vráti reťazec vhodne reprezentujúci údaje o aktivite (1 bod).

V triede `PlanRozvoja`:

- `public static PlanRozvoja zoSuboru(String nazovSuboru)` – statická metóda, ktorá z uvedeného súboru prečíta zoznam aktivít, pričom v každom riadku bude popis jednej aktivity (4 body).
- `public void uloz(String nazovSuboru)` – uloží všetky aktivity zo zoznamu aktivit do súboru v tvare, ktorý vie spracovať metóda `zoSuboru(String nazovSuboru)` (3 body).
- `public String toString()` – vráti reťazec vhodne reprezentujúci všetky aktivity uchované v zozname aktivít (1 bod).

Nasledujúce úlohy môžete riešiť v ľubovoľnom poradí. Ale riešenie jednej môže zjednodušiť nasledujúce.

Inštančné metódy triedy `Aktivita`:

- `public String vratCasKonca()` – vráti čas kedy skončila aktivita v rovnakom tvare ako čas začiatku (2 body).

Inštančné metódy triedy `PlanRozvoja`:

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

- `public int vratCas()` – vráti koľko celých hodín sa strávilo aktivitami (1 bod).
- `public int najdlhsiaAktivita()` – vráti koľko minút trvala najdlhšia aktivita (1 bod).

- `public List<String> vratPouzivaneAplikacie()` – vráti názvy všetkých aplikácii, ktoré boli použité, vo vrátenom zozname sa môže každá aplikácia nachádzať najviac raz (2 body).
- `public List<String> vratAktivityPodlaTypu(String typ)` – vráti názvy všetkých aktivít, ktorým sa venoval, vo vrátenom zozname sa môže každý názov vyskytovať najviac raz (1 body).
- `public int casAktivitPocasDna(String datum)` – vráti čas koľko minút sa venoval aktivitám počas dňa zadaného parametrom. Predpokladajte, že žiadna aktivita nepresahuje polnoc. (2 body).
- `public boolean skontrolujAplikaciu(String aplikacia, String nazovAktivity)` – vráti či aplikácia, sa používa pri vykonávaní aktivity zadanej parametrom (2 body).
- `public List<String> nepopularneAktivity()` – vráti zoznam názvov aktivít ktoré sa vykonali práve raz (3 body).
- `public List<String> vratZoznamPridruzenychAktivit()` – vráti zoznam, v ktorom sú všetky názvy aktivít ktoré vždy boli vykonávané ako pridružené. Každý názov sa nachádza v liste nanajvýš raz (4 body).
- `public Map<String, Double> percentoTypov()` – vráti mapu, kde je každému typu aktivity "fyzicka", "mentalna" alebo ďalšie typy priradené koľko percent celkového času sa ňou strávilo (4 body). +1 bod za zaokrúhlenie percent na 2 desatinné miesta.
- `public int[] histogramPoHodinach()` – vráti pole veľkosti 24, každý index zodpovedá hodine počas dňa. Vo vrátenom poli na pozícii i je koľko minút sa dokopy venoval aktivitám počas danej hodiny, pre jednoduchosť celú aktivitu zapíšte do hodiny kedy začala. (4 body) +3 body ak aktivite ktorá je na prelome hodín rozdelíte minúty správne k jednotlivým hodinám. Predpokladajte, že aktivity netrvajú dlhšie ako hodinu a aktivity nepresahujú polnoc. +1 bod ak aktivity presahujú polnoc a správne sa ich čas rozpočíta.
- `public PlanRozvoja vratPlanRozvojaZaObdobie(String odDatumu, String poDatum)` – metóda vráti plán rozvoja za obdobie určené parametrami (vrátane dní určených parametrami) (3 body).
- `public Map<String, Integer> dosiahnuteLevely()` – metóda vráti mapu aktivity a najväčšieho dosiahnutého levelu, pre každý druh aktivity (napr. cvičenie brucha) ktorá využíva aplikáciu založenú na leveloch. Predpokladajte, že v detailoch je uvedené slovo "level" nasledované číslom táto dvojica je oddelená medzerou, reťazec detail môže obsahovať aj ďalšie detaily napr. "Gratulujeme absolvovali ste level 7". (5 bodov).
- `public Map<String, Integer> najpouzivanejsieAplikacie()` – vráti mapu, kde je každej aplikácii priradené aké celkové množstvo času v minútach bola použitá (4 body).
- `public List<String> top5Aplikacii()` – vráti zoznam piatich aplikácii ktoré sa využívali najviac minút pri rôznych aktivitách. (3 bodov). Pozn. Bez korektného riešenia predchádzajúcej úlohy nemôžete riešiť túto.
- `public boolean dennaVyukovaAplikacia(String nazovAplikacie)` – aplikáciu nazveme "dennou výukovou" ak na každý deň nám dá inú činnosť. Napr. každý deň nám vyberie iný recept ktorý budeme variť. Metóda vráti `true` ak aplikácia zadaná parametrom má vždy iný detail (3 body).
- `public Map<String, Integer> univerzalnostZakladnychCinnosti()` – vytvorte mapu, kde je každej základnej činnosti (napr. umývanie zubov) priradené koľko rôznych aktivít je možné počas nej vykonávať (napr. spievať, jódlovať, cvičiť lýtka sú 3). (6 bodov).

Triedenie (4 body)

Vytvorte triedu PorovnavacAktivit, ktorá implementuje rozhranie java.util.Comparator<Aktivita> s príslušnou metódou compare, aby po aplikovaní triediaceho algoritmu boli zotriedené aktivity podľa dátumu a času začiatku (4 body).

Poznámka k riešeniu: Ak chcete ako oddeľovač (delimiter) tokenov pre `Scanner` nastaviť bodku, použite `useDelimiter("\\.")`