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.

Turistický denník

Motivácia: V zdravom tele zdravý duch! Byť turista a chodiť na túry v každom ročnom období je moderné a prospešné pre psychické aj fyzické zdravie (aspoň tak sa tvrdí...) Mimochodom turistika je medzi [prírodo]vedcami veľmi populárna (samozrejme výnimky potvrdzujú pravidlo). Zároveň dnešná informačná doba priam nabáda každého turistu, aby si svoje turistické výlety (túry) manažoval a zdieľal v nejakej appke – turistickom denníku. Turistický denník slúži nielen na evidenciu uskutočnených túr, ale aj plánovanie budúcich. Pri plánovaní túr nám môžu pomôcť portály, ktoré poskytujú mnoho informácii o rôznych turistických trasách. Keď nás nejaká trasa zaujme, tak si ju do turistického denníka poznačíme (spolu s údajmi) ako neabsolvovanú, resp. plánovanú túru. Po absolvovaní túry si potom do denníka k túre dopíšeme ďalšie informácie ako dátum absolvovania, čas začiatku a konca túry. Dobrá appka by mala poskytovať funkcionalitu na vyhľadávanie, výber, usporadúvanie túr ako aj informácie o jednotlivých dňoch, rekordoch a ďalšie potrebné a zaujímavé štatistické údaje, ktoré motivujú k ďalším túram a prekonávaniu vlastných hraníc. Kvôli jednoduchosti nepredpokladáme nočné alebo viacdňové túry. Každá túra začína a končí v ten istý deň. V jeden deň môžeme uskutočniť aj viac túr.

Pohľad analytika: Pri implementácii budeme potrebovať:

- triedu `Tura`, ktorá uchováva informácie o plánovej alebo už uskutočnenej túre,
- triedu `Dennik`, ktorá bude uchovávať zoznam túr, a
- triedu `PomocneMetody`, ktorá bude obsahovať pomocné metódy.

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

- **`trasa`** (popis trasy, napr. „*Obručné – Kráľova studňa – Lenartov*“ alebo „*Turčianske Teplice – Rakša – Mača – sedlo za Drieňkom – Drieňok a späť*“; trasa sa skladá z pomlčkami oddelených názvov miest na trase, pričom ak je na konci popisu trasy „*a späť*“, trasa pokračuje navštívenými miestami v opačnom poradí),
- **`vzdialenost`** (vzdialenosť medzi začiatočným a koncovým miestom trasy v metroch, napr. 10200 m),
- **`stupanie`** (celkové stúpanie na trase v metroch, napr. 313),
- **`klesanie`** (celkové klesanie na trase v metroch, napr. 503),
- **`predpokladanyCas`** (predpokladaný čas prejdenia trasy v minútach, napr. 200),
- **`datum`** (dátum v tvare `dd.mm.rrrr`, kedy sme absolvovali túru),
- **`zaciatok`** (čas v tvare `HH:MM`, kedy sme vyrazili na túru, napr. „10:20“),
- **`koniec`** (čas v tvare `HH:MM`, kedy sme túru ukončili, napr. „13:10“).

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

Ďalej vytvorte triedu `sk.upjs.finalTerm.Dennik`, ktorá bude uchovávať zoznam túr.

Konštruktory a pridávanie túr (3 body dokopy – povinné):

- `public Tura(String trasa, int vzdialenost, int stupanie, int klesanie, int predpokladanyCas)` – použije sa na vytvorenie ešte neabsolvovanej (plánovanej) túry.
- `public Tura(String trasa, int vzdialenost, int stupanie, int klesanie, int predpokladanyCas, String datum, String zaciatok, String koniec)` – použije sa na vytvorenie už absolovanej túry.
- `public void pridaj(Tura tura)` – inštančná metóda v triede `Dennik`, ktorá pridá túru do denníka.

Práca so súbormi (povinné):

V triede `Tura`:

- `public static Tura zoStringu(String popis)` – statická metóda, ktorá vráti referenciu na novovytvorený objekt triedy `Tura`. Parameter je reťazec v tvare `"trasa\tvzdialenost\tstupanie\tklesanie\tpredpokladanyCas\tdatum\tzaciatok\tkoniec"`, resp. `"trasa\tvzdialenost\tstupanie\tklesanie\tpredpokladanyCas"` ak túra ešte nebola absolvovaná (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 o túre (1 bod).

V triede `Dennik`:

- `public static Dennik zoSuboru(String nazovSuboru)` – statická metóda, ktorá z uvedeného súboru prečíta turistický denník (zoznam túr), pričom v každom riadku bude popis jednej túry (4 body).
- `public void uloz(String nazovSuboru)` – uloží všetky túry z denníka do súboru v tvare, ktorý vie spracovať metóda `zoSuboru(String nazovSuboru)` (3 body).
- `public String toString()` – vráti reťazec vhodne reprezentujúci obsah denníka (1 bod).

Statické pomocné metódy v triede `PomocneMetody` (náročnejšie metódy, riešte ich neskôr)**:**

Vytvorte triedu `PomocneMetody` iba s privátnym konštruktorom (1b), ktorá bude poskytovať statickú metódu:

- `public static int prevedCasNaMinuty(String hhmm)` – statická metóda, ktorá časový reťazec vo formáte „HH:MM“ prevedie na počet minút od polnoci („00:00“). Pre reťazec „02:45”, resp. „2:45“ má metóda vrátiť číslo 165 = **2** * 60 + **45** (3 body). Ak reťazec nie je možné previesť na minúty alebo ak počet hodín, resp. minút nezodpovedná reálnej hodnote (napr. ak ide o časy „03:70“, „03:-10“, „25:00“, atď.) metóda nech vyhodí nekontrolovanú výnimku triedy `NespravnyCasException` (4 body).

Inštančné metódy triedy `Tura` (náročnejšie metódy, riešte ich neskôr):

- `public List<String> vratZoznamMiest()` - vráti miesta na trase ako zoznam reťazcov. Ak popis trasy končí „a späť” zoznam bude obsahovať príslušné miesta v opačnom poradí. Môžete predpokladať, že názov žiadneho miesta neobsahuje pomlčku (7 bodov). Bonus (3b): Vyriešte prípady, kedy názov miesta môže obsahovať aj pomlčku (napr. Šaštín-Stráže, Frýdek-Místek, …). Všimnite si, že pomlčka v názve na rozdiel od pomlčky v oddeľovači miest na trase neobsahuje naľavo a napravo žiadne medzery.

- `public Tura vytvorOpacnuTuru()` - vráti novú neabsolvovanú túru, ktorá vznikne z existujúcej, ak sa vyberieme opačným smerom. Nezabudnite vhodne nastaviť (invertovať) popis trasy, zameniť stúpanie a klesanie. Kvôli jednoduchosti predpokladaný čas na prejdenie meniť nebudeme (6 bodov).

Inštančné metódy triedy `Dennik`:

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

- `public int vratPrejdenuVzdialenostVKm()` - vráti celkovú vzdialenosť prejdenú na absolvovaných túrach v kilometroch zaokrúhlených nadol (1 bod).
- `public List<Tura> vratPlanovaneTury()` – vráti zoznam túr, ktoré sú naplánované v denníku, ale ešte neboli absolvované (1 bod).
- `public List<Tura> vytvorOdporucanie(int minVzdialenost, int maxVzdialenost)` – vráti referenciu na zoznam neabsolvovaných (plánovaných) túr, ktorých vzdialenosť je medzi hodnotami minimum a maximum (2 body).
- `public Tura najdiNajnarocnejsiuTuru()` – vráti referenciu na túru, ktorý má najväčšiu náročnosť, pričom $náročnosť = \frac{vzdialenost*\left(\frac{(stupanie+klesanie)*50}{vzdialenost}+1\right)}{1000}$, zdroj: `http://www.nwhiker.com/` (3 body).
- `public double vypocitajNarocnostDna(String datum)` – adaptovaním vzorca na výpočet náročnosti túry vypočítajte náročnosť dňa – t.j. vzdialenosti, stúpania a klesania túr uskutočnených počas zadaného dňa sa budú sčítavať; ak sa v daný deň žiadna túra neuskutočnila, metóda nech vráti hodnotu `Double.NaN` (4 body).
- `public List<String> vratOpakovaneTrasy()` – vráti zoznam trás, ktoré sme absolvovali viac ako raz (teda 2 razy a viac), pričom každá z týchto trás nech je v zozname práve raz (5 bodov). Môžete predpokladať, že každá trasa je zapísaná jednotným spôsobom. Za riešenie, ktoré sa vysporiada aj s trasami tvaru („X – Y – Z – Y – X“ a „X – Y – Z a späť“) je bonus 3 body.
- `public String najdiNajuchodenejsiDatum()` – vráti dátum (deň), počas ktorého sme celkovo prešli najviac metrov (7 bodov).
- `public Map<String, Integer> vytvorMesacneSumarizacie()` - ku mesiacu a roku v tvare `mm.rrrr` vráti vzdialenosť v celých kilometroch, ktorá bola prejdená v danom období; mesiace, v ktorých sme nespravili žiadnu túru vynechávame (6 bodov).
- `public List<Tura> najdiOkruzneTury()` - vráti zoznam okružných túr, t.j. túr, ktorých trasa začína a končí na tom istom mieste (3 body).
- `public Tura najdiNajproblemovejsiuTuru()` – vráti najproblémovejšiu túru, t.j. takú uskutočnenú túru, ktorej prejdenie trvalo najviac v pomere k odhadovanému času. napr. dvakrát tak dlho alebo trikrát tak dlho (5 bodov).
- `public String zistiTrvanieVyletu(String datum)` - vráti čas v tvare `HH:MM`, koľko ubehlo od začiatku prvej túry po koniec poslednej túry v daný deň (7 bodov).
- `public Map<String, Integer> vratPoctyNavstiveni()` - vráti mapovanie, ktoré každému miestu objavujúcemu sa na trasách uskutočnených túr priradí, koľko krát sme ho navštívili. Ak v rámci jednej túry nejaké miesto navštívime viac ráz, započítavame ho len raz (7 bodov).
- `public List<Tura> vyberTopTury(int n)` – vráti referenciu na zoznam prvých (nanajvýš) `n` neabsolovaných túr usporiadaných podľa vzdialenosti počnúc najdlhšou (8 bodov).

Triedenie a komparátor (dokopy 5 bodov):

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

- `public int compare(Tura tura1, Tura tura2)` – porovná túry podľa dátumu a času ich začiatku.

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

- `public List<Tura> vratTuryPodlaCasu()` – vráti zoznam absolvovaných túr usporiadaných podľa dátumu a času ich začiatku.