Záverečný test
Zadanie

Ústav informatiky
Prírodovedecká fakulta
UPJŠ v Košiciach

Dvakrát meraj (rozmýšľaj), raz rež (programuj)

Pravidlá a informácie:

- 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.

PAZJet

Motivácia: Cestovanie vlakom na Slovensku je teraz „in“. Aj etablovaná spoločnosť PAZJet - prevádzkovateľ zelených korytnačích vláčikov sa rozhodla vstúpiť na tento segment trhu – trhu veľkých vlakov. Na leasing získala nové vlaky, zaškolila personál, vytvorila informačný systém na rezerváciu a kúpu lístkov. PAZJet sa rozhodol staviť na povinne miestenkové vlaky pre zabezpečenie tej najvyššej kvality cestovania (vrátane fľaše kvalitného žabieho slizu od Majstra N). Po prvých mesiacoch prevádzky sa rozhodli analyzovať údaje s cieľom vylepšiť svoje produktové portfólio. Nuž a na to budú potrebovať analytický podporný softvér. Keďže prioritou PAZJet-u je kvalita vo všetkom, oslovili študentov PF UPJŠ...

Ako fungujú vlaky: Povinne miestenkový vlak znamená, že každý pasažier si kupuje lístok na konkrétne miesto vo vlaku (miesto je určené číslom vozňa a číslom sedadla v tomto vozni). Pod pojmom vlak nechápeme konkrétne ťahané vozne a lokomotívu, ale dopravné spojenie podľa nejakého grafikonu na nejakej trase. Napr. „IC 504 Wüstenrot“ je vlak, ktorý vychádza každý deň okrem nedele z Košíc o 15:26 do Bratislavy. Počas tejto cesty stojí v staniciach: Košice, Kysak, Spišská Nová Ves, Poprad-Tatry, Liptovský Mikuláš, Žilina, Trenčín, Trnava a Bratislava hl.st. Zoznam týchto staníc predstavuje trasu vlaku „IC504 Wüstenrot“. Iným spojením je vlak „IC502 Lyoness“, ktorý jazdí po rovnakej trase, no z Košíc vychádza o 9:24. Vlaky jazdiace opačným smerom z Bratislavy do Košíc už majú iné označenie (aj keď fyzicky to môžu byť rovnaké lokomotívy a vozne). Napr. existuje vlak „IC507 Lyoness“ idúci po trase s opačným poradím, ktorý odchádza z Bratislavy o 17:37. Označenie vlaku tak jedinečným spôsobom identifikuje trasu vlaku vrátane časov odchodov z jednotlivých staníc.

Pohľad analytika: Pri implementácii budeme potrebovať:

- triedu `Listok`, ktorá bude uchovávať údaje o jednom cestovnom lístku,
- triedu `ZoznamListkov`, ktorá bude uchovávať zoznam predaných cestovných lístkov,
- triedu `TrasyVlakov`, ktorá bude uchovávať mapovanie označení vlakov na ich trasy (zoznam staníc).

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

- **datum** (dátum, v ktorý lístok platí, napr. „15.1.2015“)
- **nastup** (názov nástupnej stanice, napr. „Košice“)
- **vystup** (názov cieľovej stanice, napr. „Žilina“)
- **vlak** (označenie vlaku, napr. „IC502 Lyoness“)
- **vozen** (číselné označenie vozňa vo vlaku, napr. 1, 2, 3, ….. maximálny počet vozňov je 8)
- **miesto** (číslo identifikujúce konkrétne miesto vo vozni, napr. 1, 2, 3, …)
- **typ** (typ cestovného lístka, napr. „Obyčajný“, „Študent“, „ZŤP“, ...)
- **cena** (cena za lístok v EUR)

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

Poznámka: Kvôli zjednodušeniu zadania neuchovávame ďalšie informácie, ktoré možno nájsť na bežnom miestenkovom lístku (napr. kód lístka, spôsob platby, dátum zakúpenia, čas odchodu, ...).

Ďalej vytvorte aj triedu `sk.upjs.finalTerm.ZoznamListkov`, ktorá bude uchovávať nejaký zoznam cestovných lístkov, a triedu `sk.upjs.finalTerm.TrasyVlakov`, ktorá bude uchovávať zoznam trás jednotlivých vlakov.

Konštruktory a pridávanie lístkov (2 body dokopy – povinné):

- `public Listok(String datum, String nastup, String vystup, String vlak, int vozen, int miesto, String typ, double cena)` – použije sa na vytvorenie záznamu o jednom cestovnom lístku.
- `public void pridaj(Listok listok)` – inštančná metóda v triede `ZoznamListkov`, ktorá pridá záznam o cestovnom lístku do zoznamu.

Práca so súbormi (povinné):

V triede `Listok`:

- `public static Listok zoStringu(String popis)` – statická metóda, ktorá vráti referenciu na novovytvorený objekt triedy `Listok`. Parameter je `String` v tvare "dátum `\t` nástup `\t` výstup `\t` vlak `\t` vozeň `\t` miesto `\t` typ `\t` cena" (2 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 lístku (1 bod).

V triede `ZoznamListkov`:

- `public static ZoznamListkov zoSuboru(File f)` – statická metóda, ktorá z uvedeného súboru prečíta zoznam lístkov, pričom v každom riadku bude popis jedného lístka (4 body).
- `public void uloz(File subor)` – uloží záznamy o všetkých lístkoch v zozname do súboru v tvare, ktorý vie spracovať metóda `zoSuboru(File f)` (3 body).
- `public String toString()` – vráti reťazec vhodne reprezentujúci všetky lístky v zozname (1 bod).

Inštančné metódy triedy `ZoznamListkov`:

- `public double celkovaTrzba()` – vráti celkovú tržbu z predaných lístkov (1 bod).
- `public int pocetPasazierov(String datum, String vlak)` – vráti celkový počet pasažierov prepravených zadaným vlakom v daný deň (2 body).
- `public Set<String> zastavkyVlaku(String datum, String vlak)` – pre zadaný dátum a vlak vráti zoznam staníc, na ktorých vlak musí v tento deň zastaviť, t.j. staníc, kde niekto nastupuje alebo vystupuje (3 body) – ak vlaky PAZJetu majú meškanie, nestoja tam, kde netreba.
- `public Map<String, Double> trzbyVlakov()` – vráti mapovanie, ktoré každému vlaku priradí celkovú tržbu z lístkov na tento vlak (4 body).
- `public int naVyhriateSedadlo()` – vráti koľko pasažierov si sadlo na sedadlo „vyhriate“ po predošlom pasažierovi (PAZJet v rámci odporúčacieho systému odporúča „vyhriate“ sedadlá). Sedadlo ostáva vyhriate po predošlom pasažierovi len na stanici, na ktorej tento pasažier vystúpil (4 body).
- `public int najobsadenejsiVozen(String datum, String vlak)` - pre zadaný dátum a vlak vráti číslo vozňa, v ktorom sa prepravilo najviac pasažierov. Počet vozňov vlaku je najviac 8, t.j. čísla vozňov sú z množiny {1, 2, ..., 8} (6 bodov).
- `public String novaPobocka(List<String> existujucePobocky)` – vráti najlepšiu stanicu pre vytvorenie novej pobočky. PAZJet chce otvoriť nové predajno-klientské miesto (pobočku) na

vlakovej stanici. Z dôvodu efektivity chce PAZJet nájsť takú stanicu, kde najviac pasažierov PAZJetu nastupuje/vystupuje, t.j. je tam najväčší súčet nastupujúcich a vystupujúcich pasažierov, a zároveň tam PAZJet ešte nemá svoju pobočku (6 bodov).

- `public Set<String> studentskeVlaky()` – vráti vlaky, v ktorých sa celkovo prepravilo viac študentov ako iných cestujúcich. Študentov rozpoznáte tak, že ich lístok je typu „Študent“ (8 bodov).
- `public Map<String, Double> kvartalneTrzby(int odRoku, int poRok)` – vráti mapovanie, ktoré každému kvartálu v zadanom rozmedzí rokov (vrátane) priradí tržbu v tomto kvartáli. Volanie `kvartalneTrzby(2013, 2015)` teda vytvorí mapovanie s kľúčmi „I.2013”, „II.2013”, „III.2013”, „IV.2013”, ..., „III.2015”, „IV.2015” (12 bodov).

Triedenie a komparátor (dokopy 2+1 bod):

Vytvorte triedu `KomparatorPodlaCeny` implementujúcu `java.util.Comparator<Listok>` s metódou:

- `public int compare(Listok l1, Listok l2)` – porovná cestovné lístky podľa ceny.

V triede `ZoznamListkov` implementujte inštančnú metódu:

- `public void zoradPodlaCeny()`– usporiada lístky v zozname podľa ceny.

Výnimky (3 body)

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

Inštančné metódy triedy `TrasyVlakov`:

- `public void pridajTrasu(String vlak, List<String> trasa)` – pridá (alebo zmení, ak už bola definovaná) popis trasy vlaku so zadaným označením. Popisom trasy rozumieme zoznam názvov staníc na trase vlaku v poradí od východiskovej po cieľovú stanicu (2 body).
- `public List<String> vratTrasu(String vlak)` – vráti trasu zadaného vlaku alebo `null`, ak nebola definovaná (1 bod).
- `public Set<String> vlakyVProtismere(String vlak)` – vráti označenia tých vlakov, ktoré premávajú na opačnej trase, t.j. majú rovnakú postupnosť zastávok, ale v opačnom poradí (4 body).
- `public static TrasyVlakov zoSuboru(File f)` – statická metóda, ktorá z uvedeného súboru prečíta zoznam trás vlakov (5 bodov).
- `public void uloz(File subor)` – uloží záznamy o trasách vlakov do súboru v tvare, ktorý vie spracovať metóda `zoSuboru(File f)` (3 body).

Inštančné metódy triedy `Listok` (vyžadujúce prvé dve metódy triedy `TrasyVlakov`):

- `public int ostavaZastavok(String aktualnaStanica, TrasyVlakov trasyVlakov)` – vráti o koľko zastávok končí platnosť lístka, ak sa aktuálne pasažier nachádza na stanici `aktualnaStanica` (môžete predpokladať korektný vstup, 4 body).

Inštančné metódy triedy `ZoznamListkov` (vyžadujúce prvé dve metódy triedy `TrasyVlakov`):

- `public Map<String, Integer> pocetPasazierovVoVlaku(String datum, String vlak, TrasyVlakov trasyVlakov)` – pre zadaný dátum, vlak a usporiadaný zoznam staníc na jeho trase vráti mapovanie, ktoré každej stanici na trase priradí počet pasažierov nachádzajúcich sa vo vlaku v dobe odchodu vlaku z tejto stanice (10 bodov).
- `public int pocetDobrychListkov(TrasyVlakov trasyVlakov)` – vráti počet dobrých lístkov. Lístok považujeme za dobrý, ak je predaný na aspoň 50% z trasy vlaku. Kvôli jednoduchosti predpokladáme, že vzdialenosť medzi nasledujúcimi stanicami na trase je rovnaká (10 bodov).

Hinty:

- Pri načítavaní čísel s desatinnou časťou nezabudnite použiť metódu `useLocale(Locale.US)` objektov triedy `Scanner`.
- V triede `Listok` vytvorte metódu `public boolean patriVlaku(String datum, String vlak)`, ktorá vráti, či lístok je vydaný na zadaný vlak v danom dátume. Pomôže to zjednodušiť zápis viacerých metód.
- Pri riešení `naVyhriateSedadlo` odporúčame napísať si najprv podmienku (rozdelenú do viacerých podpodmienok), ktorá charakterizuje „želaný“ vzťah medzi nastupujúcim a vystupujúcim pasažierom (resp. ich lístkami). Jedna z prirodzených (nutných, no nie postačujúcich) podpodmienok môže byť tá, že títo pasažieri cestujú v tom istom vlaku a v ten istý deň.
- Pri riešení `kvartalneTrzby` odporúčame vytvoriť si pomocnú metódu, ktorá k zadanému dátumu vytvorí reťazec s kvartálom, do ktorého tento dátum patrí.
- Ak chcete od oddeľovač tokenov („slov“) v objekte triedy `Scanner` použiť bodku, zavolajte metódu `useDelimiter("\\.");`