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 komunikácia s kýmkoľvek okrem dozoru,
- v prípade akýchkoľvek problémov alebo z dôvodu ohodnotenia riešenia kontaktujte dozor,
- riešenia je možné nechať si ohodnotiť aj priebežne, povinná časť musí byť ohodnotená do 150 minút od začiatku
- **funkčnosť každej metódy musí byť preukázaná spustením na vami vytvorenom testovacom vstupe, nespustiteľné metódy neumožňujú zisk príslušných bodov,**
- všetky inštančné premenné musia byť neverejné.

Vlaková stanica

Motivácia: Železničná doprava zohráva kľúčovú úlohu v prepojení miest, krajín aj celých kontinentov. Od prvých parných lokomotív až po moderné vysokorýchlostné vlaky prešla dlhým vývojom. Každá vlaková stanica eviduje prichádzajúce aj odchádzajúce vlaky, či už ide o osobné spojenia alebo nákladné súpravy. Vašou úlohou je pripraviť podklad pre informačný systém vlakovej stanice, ktorý bude evidovať všetky prichádzajúce a ochádzajúce vlaky.

Pohľad analytika: Pri implementácii budeme potrebovať:

- triedu `Train`, ktorá uchováva informácie o jednom vlaku,
- triedu `Station`, ktorá bude uchovávať zoznam všetkých vlakov v jednom dni.

Zadanie: V (ľubovoľnom) balíčku vytvorte triedu `Train` obsahujúcu dátové položky prístupné cez gettre (a podľa uváženia aj modifikovateľné cez settre):

- **`number`** – označenie čísla vlaku (napr. EX42, REX1954, Os7814, NEx74201)
- **`source`** – názov stanice, z ktorej vlak prichádza,
- **`destination`** – názov cieľovej stanice, kam vlak smeruje,
- **`scheduledTime`** – plánovaný čas odchodu (v tvare HH:MM),
- **`departureTime`** – skutočný čas odchodu (v tvare HH:MM),
- **`carCount`** – počet vozňov vo vlaku,
- **`passengerTrain`** – ozačuje, či ide o osobný vlak (true), alebo nákladný (false).

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

Evidujeme vlaky pre jeden vybraný deň. Čas bude uvedený od 00:00 do 23:59. Pre jednoduchosť predpokladáme, že skutočný čas odchodu bude v rovnaký deň.

Osobné vlaky sú napríklad Os7814 (osobný vlak), REX1954 (regionálny expres) alebo EX42 (expres). Neosobné, teda nákladné vlaky, môžu mať označenia ako Pn64732 (nákladný vlak), Lv55014 (lokomotíva

bez súpravy) alebo NEx74201 (nákladný expres). Označenie vlaku je vždy tvorené písmenami a číslicami, pričom písmená môžu byť aj malé.

Ďalej vytvorte triedu `Station`, ktorá bude uchovávať zoznam vlakov.

Konštruktory a evidovanie vlakov (povinné):

- `public Train(String number, String source, String destination, String scheduledTime, String departureTime, int carCount, boolean passengerTrain)` – použije sa na evidovanie vypraveného vlaku,
- `public Train(String number, String source, String destination, String scheduledTime, int carCount, boolean passengerTrain)` – použije sa na evidovanie vlaku, ktorý ešte neodišiel,
- `public void addTrain(Train train)` – metóda v triede `Station`, ktorá zaeviduje údaje o vlaku.

Práca so súbormi (povinné):

V triede `Train`:

- `public static Train fromString(String input)` – statická metóda, ktorá vráti referenciu na novovytvorený objekt triedy `Train`. Parameter je reťazec v tvare `"number;source;destination;scheduledTime;departureTime;carCount;passengerTrain"`, resp. `"number;source;destination;scheduledTime;carCount;passengerTrain"` ak vlak ešte nebol vypravený.
  Poznámka: `Scanner`-u môžete povedať, že oddeľovač má byť bodkočiarka zavolaním jeho metódy `useDelimiter(";")`.
- `public String toString()` – vráti reťazec vhodne reprezentujúci údaje o vlaku.

V triede `Station`:

- `public static Station loadTrains(String fileName)` – statická metóda, ktorá z uvedeného súboru prečíta informácie o stanici (zoznam vlakoví pričom v každom riadku bude popis jedného vlaku.
- `public void saveTrains(String fileName)` – uloží všetky zaevidované vlaky do súboru v tvare, ktorý vie spracovať metóda `loadTrains`.
- `public String toString()` – vráti reťazec vhodne reprezentujúci kompletný popis vlakov.

Za povinnú časť bude udelených **15 bodov**. Nasledovné úlohy môžete riešiť v ľubovoľnom poradí:

Metódy triedy `Train` (úlohy sú zoradené podľa počtu bodov):

- [2b] `public int delay()` - vráti počet minút meškania vlaku

Metódy triedy `Station` (úlohy sú zoradené podľa počtu bodov):

- [1b] `public int departedTrains()` - vráti počet osobných vlakov, ktoré už odišli.
- [1b] `public Train connection(String stationA, String stationB)` - vráti referenciu na vlak, ktorý jazdí medzi dvoma zadanými stanicami (v ľubovoľnom smere). Ak je takýchto vlakov viac, vráti ľubovoľný z nich.
- [2b] `public Train firstDeparture(String destination)` - vráti vlak, ktorý odchádza prvý do cieľovej stanice.
- [2b] `public List<String> destinations(String source)` - vráti zoznam cieľových staníc, kam jazdí vlak zo stanice zadanej parametrom source. Tento zoznam nech je zoradený podľa abecedy.

- [3b] `public List<String> stationNames()` - vráti zoznam názvov staníc, z ktorých vlak prichádza alebo kam smeruje, zoradený podľa abecedy, kde sa každý názov nachádza najviac jedenkrát.
- [3b] `public Map<Integer, Integer> cars()` - metóda vypočíta početnosť nákladných vlakov podľa počtu vozňov. Vráti referenciu na mapu, kde kľúčom sú počty vozňov (v nákladných vlakoch) a hodnotou počet výskytov.
- [3b] `public double passengerCarsMedian()` - vráti medián počtu vozňov v osobných vlakoch. Medián je hodnota, pre ktorú platí, že polovica hodnôt je väčších alebo rovných ako medián a polovica je menších alebo rovných. Ak je počet hodnôt párny, medián sa počíta ako priemer dvoch hodnôt "v strede".
- [3b] `public double ratioOfPassengerTrains()` - vráti koľko percent vlakov je osobných. Výsledné percento je zaokrúhlené na dve desatinné miesta. Riešenie bez zaokrúhlenia je za 2 body.
- [3b] `public Map<String, Integer> typeCount()` - metóda vráti referenciu na mapu, kde kľúčom sú označenia vlaku (EX, Os a pod. - teda písmená bez čísel z označenia) a hodnotou je počet týchto vlakov v danom dni.
- [4b] `public String topDestination()` - vráti názov stanice, do ktorej smeruje najviac vlakov (destination). Pri rovnosti počtu staníc rozhoduje celkový počet (súčet) vozňov. Riešenie, ktoré bude prechádzať zoznamom vlakov viackrát získa iba 2 body.
- [4b] `public List<String> delayedTrains(String currentTime)` - vráti zoznam vlakov, ktoré mali meškanie. Berieme do úvahy aj vlaky, ktoré ešte nevyrazili. Aktuálny čas je v parametri `currentTime` v rovnakom formáte ako parameter `scheduledTime` (HH:MM).
- [5b] `public int hourRange()` - vráti rozdiel medzi najväčším počtom vlakov vypraveným počas jednej hodiny a najmenším počtom vlakov vypravených v rámci jednej hodiny. Hodinu počítame od HH:00 do HH:59.
- [6b] `public void reliableTrains(List<Station> previous)` - metóda dostane na vstupe záznamy z predošlých dní, ktoré popisujú stanicu. Metóda vráti zoznam označení vlakov, ktoré boli vypravené každý deň a nemali meškanie (vrátane aktuálneho dňa).
- [6b] `public void topKDelayed(int k)` - vypíše *k* vlakov, ktoré majú najväčšie meškanie. Vlaky sú usporiadané podľa dĺžky meškania od najväčšieho. Ak je *k* väčšie ako počet vlakov, ktoré odišli, vypíše všetky vlaky.

Ďalšie úlohy:

- [4b] Vytvorte metódu, ktorá usporiada všetky vlaky chronologicky podľa plánovaného času odchodu.
- [3b] Vytvorte nekontrolovanú výnimku `InvalidTimeFormatException` a použite ju na vhodnom mieste.