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 hodnotiteľov,
- 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.

Diaľničné úseky

Cestná infraštruktúra štátu je tvorená rôznymi typmi ciest. Obzvlášť medzinárodná preprava je vedená po diaľniciach a rýchlostných cestách. Takéto cesty sú (nie len na Slovensku) tvorené jednotlivými úsekmi. Napríklad slovenská diaľnica D1 pozostáva zo 44 úsekov, pričom niektoré sú v prevádzke, iné vo výstavbe a niektoré iba v pláne.
V zadaní nebudeme robiť rozdiely medzi rýchlostnou cestou a diaľnicou. Úlohou je pripraviť základnú platformu pre analýzu diaľničnej siete, ktorá pozostáva z mnohých úsekov tvoriacich jednotlivé diaľnice.

Pohľad analytika: Pri implementácii budeme potrebovať:

- triedu `Usek`, ktorá uchováva informácie o jednom diaľničnom úseku,
- triedu `DialnicnaSiet`, ktorá bude uchovávať zoznam diaľničných úsekov. Úseky sú v tomto zozname uložené v ľubovoľnom poradí.

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

- **`nazov`** – oficiálny názov úseku tvorený dvoma miestami oddelenými pomlčkou (napr. Prešov, Juh - Ličartovce;Važec - Mengusovce),
- **`oznacenie`** – uvádza názov diaľnice alebo rýchlostnej cesty, ku ktorej prislúcha daný úsek (napr. D1, R4),
- **`dlzka`** – počet kilometrov tvoriacich diaľničný úsek (napr. 7.8km pre úsek Prešov, Západ - Prešov, Juh),
- **`tunel`** – označuje názov tunela, ktorý je na danom úseku. Môžeme predpokladať, že úsek obsahuje najviac jeden tunel (ak by ich bolo viac, boli by rozdelené na menšie úseky),
- **`intenzita`** – údaj o počte prejdených vozidiel za jednotku času podľa pravidelného celoštátneho sčítania dopravy (napr. 14 837 pre úsek Štrba - Mengusovce). V prípade nového úseku alebo úseku vo výstavbe bude uvedená hodnota 0.
- **`datumVystavby`** – dátum, ktorý označuje začiatok výstavby úseku vo formáte MM/RRRR (napr. 06/2021),
- **`pocetMesiacovVystavby`** – súvislý počet mesiacov vrátane mesiaca, v ktorom sa začala výstavba a vrátane mesiaca odovzdania úseku do používania,
- **`cena`** – obstarávacia cena v eurách podľa zmluvy.

Upozornenie: Zadanie pre triedu `Usek` predpisuje dátové položky prístupné cez gettre. Aké privátne inštančné premenné použijete na uloženie týchto dátových položiek je na vašom rozhodnutí. Poradie premenných v triede aj v konštruktore si môžete zvoliť podľa vlastného uváženia.

Ďalej vytvorte triedu `sk.upjs.finalterm.DialnicnaSiet`, ktorá bude uchovávať zoznam úsekov.

Konštruktory a evidovanie úsekov (3 body dokopy – povinné):

- `public Usek(String nazov, String oznacenie, double dlzka, String tunel, int intenzita, String datumVystavby, int pocetMesiacovVystavby)` – použije sa na evidovanie úseku v prevádzke.
- `public Usek(String nazov, String oznacenie, double dlzka, String tunel, int intenzita, String datumVystavby, int pocetMesiacovVystavby, int cena)` – použije sa na evidovanie úseku vo výstavbe.
- `public void eviduj(Usek usek)` – inštančná metóda v triede `DialnicnaSiet`, ktorá zaeviduje údaje o jednom úseku.

Práca so súbormi (povinné):

V triede `Usek`:

- `public static Usek zoStringu(String popis)` – statická metóda, ktorá vráti referenciu na novovytvorený objekt triedy `Usek`. Parameter je reťazec v tvare `"nazov \t oznacenie \t dlzka \t tunel \t intenzita \t datumVystavby \t pocetMesiacovVystavby \t cena"`, resp. bez posledného údaju o cene ak ide o úsek v prevádzke (3 body);
  Poznámka: Znak `\t` je neviditeľný znak tabulátora. `Scanner`-u môžete povedať, že oddeľovač má byť tabulátor zavolaním jeho metódy `useDelimiter("\t").` Medzera pred a za `\t` sú len kvôli zlepšeniu čitateľnosti zadania, v reťazci reálne nie sú. Na zahrnutie informácie o neprítomnosti tunela si môžete zvoliť ľubovoľnú formu (ľubovoľný reťazec).
- `public String toString()` – vráti reťazec vhodne reprezentujúci údaje o jednom úseku (1 bod).

V triede `DialnicnaSiet`:

- `public static DialnicnaSiet nacitajSiet(String nazovSuboru)` – statická metóda, ktorá z uvedeného súboru prečíta diaľničnú sieť (zoznam úsekov), pričom v každom riadku bude popis jedného úseku (4 body).
- `public void ulozSiet(String nazovSuboru)` – uloží všetky zaevidované úseky do súboru v tvare, ktorý vie spracovať metóda `nacitajSiet(String nazovSuboru)` (3 body).
- `public String toString()` – vráti reťazec vhodne reprezentujúci kompletnú evidenciu pre danú diaľničnú sieť (1 bod).

  Upozornenie: Súbory použité v metódach `nacitajSiet` a `ulozSiet` musia mať rovnaký formát, Teda musí platiť, že súbory A a B majú rovnaký obsah, ak prebehlo načítanie siete zo súboru A a uloženie do súboru B pomocou vyššie uvedených metód.

V triede `Usek` (úlohy môžete riešiť v ľubovoľnom poradí):

- Vytvorte metódy `public boolean voVystavbe()` a `public boolean maTunel()`, ktoré vrátia true ak je úsek vo výstavbe, resp. obsahuje tunel. Tieto úlohy nie sú bodované, ale pomôžu lepšej čitateľnosti v ďalších metódach.
- Vytvorte metódu `public String datumUkonceniaVystavby()`, ktorá na základe hodnôt premenných `datumVystavby` a `pocetMesiacovVystavby` vypočíta a vráti mesiac, v ktorom bola odovzdaná stavba do prevádzky. Výstup nech je v rovnakom formáte ako `datumVystavby`, čiže MM/RRRR (4 body).
- Trieda `Usek` nech implementuje rozhranie `Comparable<Usek>`. Implementujte príslušnú metódu takým spôsobom, aby po zotriedení boli najprv uvedené úseky s väčším počtom kilometrov. (1 bod).

Inštančné metódy triedy `DialnicnaSiet` (úlohy môžete riešiť v ľubovoľnom poradí):

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

- `public int dlzkaDialnice(String oznacenie)` – vráti celkovú dĺžku diaľničných úsekov v metroch pre konkrétnu diaľnicu zadanú parametrom `oznacenie`. (2 body).
- `public String najviacStavieb()` – vráti označenie diaľnice, na ktorej je najviac úsekov vo výstavbe. Pri rovnosti počtu úsekov vo výstavbe vráti akúkoľvek diaľnicu s týmto počtom (3 body).
- `public void noveScitanie(Map<String, Integer> scitanie)` – na základe nového sčítania upraví hodnoty intenzity dopravy v aktuálnom zozname úsekov. Metóda modifikuje zoznam úsekov (2 body).
- `public Set<String> dialniceNaScitanie(String aktualnyDatum)` – vráti referenciu na množinu označení diaľníc, na ktorých je potrebné urobiť nové sčítanie. Takéto sčítanie je potrebné realizovať, ak sa na diaľnici nachádza aspoň jeden úsek, ktorý je už v prevádzke aspoň rok a zatiaľ nemá údaj o intenzite dopravy. Aktuálny dátum vo formáte MM/RRRR je parametrom metódy. Zvážte použitie pomocnej metódy na overenie, či ubehol aspoň rok od uvedenia do prevádzky a implementujte metódu `datumUkonceniaVystavby` v triede `Usek`. (5 bodov).
- `public int priemernaIntenzita(String oznacenie)` – vráti priemernú intenzitu dopravy na vybranej diaľnici (zadanej parametrom `oznacenie`). Výsledok nech je zaokrúhlený smerom nadol (dolná celá časť) (2 body).
- `public boolean asponTretinaTunelov()` – vráti true, ak aspoň tretina všetkých úsekov má tunel (2 body).
- `public List<String> miesta()` – vráti všetky miesta, ktoré definujú nejaký úsek (sú v názve úseku), vo forme zoznamu utriedeného podľa abecedy (3 body).
- `public Usek najvacsiaCenaZaKilometer()` – vráti referenciu na úsek vo výstavbe s najväčšou cenou za 1 kilometer (2 body).
- `public boolean suvislaDialnica(List<String> miesta)` – na základe zoznamu miest tvoriacich koncové body úsekov metóda overí, či existuje súvislá diaľnica prechádzajúca cez všetky miesta v zozname - teda, či medzi každými dvoma nasledovnými miestami existuje úsek v prevádzke. Môžete predpokladať, že miesta sú uvedené v správnom poradí a jednotlivé úseky majú označenia názvov v príslušnom smere (napr. pre miesta: Trnava, Hlohovec, Piešťany sa overí, či existujú úseky Trnava-Hlohovec a Hlohovec-Piešťany). (5 bodov + 2 body za verziu, ktorá prejde zoznamom úsekov iba raz + 1 bod za verziu s ľubovoľným poradím miest v názve úseku).
- `public List<Usek> voVystavbe(int rok)` – vráti novovytvorený zoznam úsekov, ktorý obsahuje všetky úseky, ktoré boli v zadaný rok vo výstavbe. Do zoznamu nech sú zahrnuté aj úseky, ktoré sa v danom roku začali stavať aj tie, čo boli uvedené do prevádzky v ten rok (5 bodov).

- `public Map<String, Double> poctyKilometrov()` – vráti mapu, kde kľúčom je označenie diaľnice a pre každú diaľnicu sa vypočíta podiel počtu kilometrov na celkovom počte kilometrov v diaľničnej sieti. (5 bodov + 1bod za zaokrúhlenie výsledných percent na dve desatinné miesta).
- `public int rozdelenie(double limit)` – na základe nového nariadenia je potrebné rozdeliť dlhé úseky na menšie časti. Všetky úseky, ktoré nemajú tunel a majú viac kilometrov ako je zadaný limit budú v evidencii rozdelené na dve menšie časti. Metóda modifikuje aktuálny zoznam úsekov. Úsek nech sa rozdelí približne na polovicu tak, aby jeden úsek mal celočíselný počet kilometrov. Miesto, kde sa bude úsek deliť nech má názov pozostávajúci z oboch názvov koncových miest (napr. Hybe-Važec s dĺžkou 10.4km sa rozdelí na úsek Hybe-Hybe/Važec s dĺžkou 5km a Hybe/Važec-Važec s dĺžkou 5.4km) (4 body).

Výnimka, usporiadanie:

Implementujte rozhranie `Comparator<Usek>` takým spôsobom, aby pri zoradení úsekov boli najprv uvedené úseky v prevádzke zoradené podľa intenzity dopravy a následne úseky vo výstavbe zoradené podľa dátumu začiatku výstavby. (4 body)

Vytvorte nekontrolovanú výnimku `NekorektnyFormatDatumu` a použite ju na všetkých miestach, kde nie je vstupným parametrom (resp. načítavaným údajom) dátum s očakávaným formátom MM/RRRR. (4 body).