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,
- **funkčnosť každej metódy musí byť preukázaná spustením na vami vytvorenom testovacom vstupe, nespustiteľné metódy neumožňujú zisk príslušných bodov,**
- všetky inštančné premenné musia byť neverejné.

Mestský helpdesk

Motivácia: Mnoho obyvateľov miest je nespokojných s fungovaním mesta, v ktorom žijú. Veľkou výzvou je komunikácia medzi občanmi a mestom. Preto sa prvák Jožko rozhodol pomôcť svojmu mestu a pripraviť mu mestský helpdesk. Úlohou helpdesku je poskytnúť jednoduchý prostriedok, ktorý umožní obyvateľom zadávať požiadavky, ktorým by sa malo mesto venovať. Aby nezadávali obyvatelia viackrát tú istú požiadavku a zároveň zvýšili prioritu požiadavky, majú možnosť za požiadavky hlasovať (formou like-ov) na webstránke mesta. Ihneď po zadaní požiadavky sa požiadavka zverejní na stránke mesta. Niektoré požiadavky sú akútne a treba ich riešiť hneď. Iné požiadavky zas môžu počkať (napr. mesto na nich nemá peniaze). Preto sa vymyslel nasledovný postup spracovania zadaných požiadaviek: Každá novo zadaná požiadavka je preskúmaná zamestnancom mesta. Zamestnanec mesta môže požiadavku zamietnuť (napríklad, ak jej riešenie nie je v kompetencii mesta). Ak ju nezamietne, musí určiť predpokladané náklady na jej riešenie. Podľa akútnosti požiadavky sa požiadavka buď rieši ihneď (stáva sa riešenou), alebo mesto v snahe oddialiť jej riešenie nechá občanov o nej hlasovať (aj s predpokladanými nákladmi) na webstránke mesta. Až podľa výsledkov hlasovania sa mesto rozhodne, či ju bude riešiť. Ak sa požiadavka vyrieši, stane sa vyriešenou a mesto na webstránke zverejní, koľko peňazí sa minulo na vyriešenie tejto požiadavky.

Pohľad analytika: Pri implementácii budeme potrebovať:

- triedu `Poziadavka`, ktorá reprezentuje požiadavku, uchováva informácie o nej zadané obyvateľom mesta a stav spracovania požiadavky,
- triedu `HelpDesk`, ktorá bude uchováva zoznam požiadaviek.

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

- **`kategoria`** (cesty, chodníky, voda, plyn, elektrika, čistota, mestská zeleň, atď.),
- **`upresnenie`** (text upresňujúci ktorý pomôže upresní požiadavku, napríklad na verejných wc netečie voda, netečie teplá voda, tečie hrdzavá voda, voda smrdí, vo vode je piesok, atď.),
- **`zadavatel`** (meno zadávateľa problému),
- **`miesto`** (ulica, ulica a číslo domu alebo viacero ulíc oddelených čiarkami, ktoré identifikujú miesto, ktorého sa požiadavka týka),
- **`datumZadania`** (dátum v tvare dd.mm.rrrr, kedy bol problém zadaný),
- **`pocetLikeov`** (počet „like-ov“ ktoré získala požiadavka od ostatných obyvateľov),

- **`stav`** (stav požiadavky po spracovaní mestom: hlasovanie, riesena, ukoncena, zamietnuta),
- **`datumZmenyStavu`** (dátum v tvare `dd.mm.rrrr`, kedy bol naposledy zmenený stav požiadavky),
- **`predpokladaneNaklady`** (predpokladané náklady na vyriešenie požiadavky),
- **`náklady`** (množstvo peňazí, ktoré sa zatiaľ minulo na riešenie požiadavky).

Upozornenie: Zadanie pre triedu `Poziadavka` 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.HelpDesk`, ktorá bude uchovávať zoznam problémov.

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

- `public Poziadavka(String kategoria, String upresnenie, String zadavatel, String miesto, String datumZadania, int pocetLikeov)` – použije sa na vytvorenie nespracovanej požiadavky,
- `public Poziadavka(String kategoria, String upresnenie, String zadavatel, String miesto, String datumZadania, int pocetLikeov, String stav, String datumZmenyStavu, double predpokladaneNaklady, double naklady)` – použije sa na vytvorenie požiadavky spracovanej mestom,
- `public void pridaj(Poziadavka poziadavka)` – inštančná metóda v triede `HelpDesk`, ktorá pridá požiadavku do HelpDesk-u.

Práca so súbormi (povinné):

V triede `Poziadavka`:

- `public static Poziadavka zoStringu(String popis)` – statická metóda, ktorá vráti referenciu na novovytvorený objekt triedy `Poziadavka`. Parameter je `String` v tvare "kategoria \t upresnenie \t zadavatel \t miesto \t datumZadania\t pocetLikeov", resp. " kategoria \t upresnenie \t zadavatel \t miesto \t datumZadania \t pocetLikeov \t stav \t datumZmenyStavu \t predpokladaneNaklady \t naklady ", ak požiadavka už bola spracovaná (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 v položke (1 bod).

V triede `HelpDesk`:

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

Inštančné metódy triedy `HelpDesk`:

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

- `public int vratMinutePeniaze()` - vráti celkovú sumu peňazí, ktoré sa už **použili** na financovanie položiek (1 bod).
- `public List<Poziadavka> vratNespracovanePolozky()` – vráti zoznam nespracovaných požiadaviek (1 bod).
- `public List<String> vratZadavatelov()` – vráti zoznam zadávateľov, pričom každý zadávateľ je v zozname len raz (3 body).

- `public HelpDesk vratHelpDeskPreKategoriu(String kategoria)` – vráti referenciu na novovytvorený HelpDesk obsahujúci len položky týkajúce sa zadanej kategórie (3 body).
- `public Poziadavka najdiNajziadanejsiuPolozku()` – vráti referenciu na položku, ktorá ešte nebola spracovaná alebo sa o nej hlasuje a má najviac like-ov (3 body), ak ich je viac a aspoň jedna je spracovaná, tak vráti spracovanú položku s najmenšími predpokladanými nákladmi (+2 body).
- `public boolean overUkoncenePolozkyVKategoriach(String[] kategorie)` – vráti, či všetky spracované položky z daných kategórii sú ukončené alebo zamietnuté, t.j. nemáme žiadne položky, ktoré môžeme zadať mestským službám (4 body).
- `public List<Poziadavka> vratOpakovanePoziadavky()` – vráti zoznam všetkých opakovaných požiadaviek (ako zoznam im prislúchajúcich položiek), pričom požiadavku vyhlásime za opakovanú, ak v helpdesk-u existuje aspoň jedna ďalšia požiadavka, ktorá má rovnakú kategóriu, zadávateľa a miesto (4 bodov).
- `public Map<String, Double> peniazeNaKategorie()` – pre každú kategóriu vráti celkové náklady, ktoré už boli použité na splnenie požiadaviek z danej kategórie (5 bodov).
- `public Map<Integer, String> topKategoriePocasRoka()` – pre každý mesiac (bez ohľadu na rok) vráti, z ktorej kategórie bolo zadaných najviac požiadaviek v danom mesiaci (7 bodov).
- `public String najproblemovejsiaUlica()` – najproblémovejšia ulica je taká, ktorá sa nachádza na najväčšom množstve požiadaviek (7 bodov). Pozn.: v dátovej položke `miesto` sa môže nachádzať ulica, ulica a číslo domu alebo viacero ulíc oddelených čiarkami.
- `public List<Poziadavka> topHlasovanePoziadavky(int n)` – vráti referenciu na zoznam `n` požiadaviek, ktoré majú najviac like-ov a ešte neboli spracované alebo sa o nich hlasuje (7 bodov).
- `public double vazenaPriemernaPercentualnaChybaOdhadov(String kategoria)` – pre danú kategóriu spočíta váženú chybu percentuálnej odchýlky ukončených požiadaviek. Percentuálna chyba odhadu je rozdiel medzi predpokladom nákladov a reálnymi nákladmi uvedený v percentách (ak je odhad 100 € a náklady sú 80 € alebo 120 €, tak v oboch prípadoch je chyba 20%). Každá chyba má v priemere váhu svojho odhadu (máme položku s odhadom 20 € a reálne náklady 40 €, tak jej chyba je 100% a položku s odhadom aj reálnymi nákladmi 180 €, tak vážená percentuálna chyba odhadu týchto položiek je 10%) (6 bodov).
- `public List<Poziadavka> prehodnotenePoziadavky()` – vráti referenciu na zoznam zamietnutých požiadaviek, ktoré sa dostali opäť do helpdesk-u po ich zamietnutí a neboli znova zamietnuté (to znamená že existuje aspoň jedna ďalšia požiadavka, ktorá má rovnakú kategóriu, zadávateľa, miesto a navyše má iný stav ako zamietnutá a dátum jej zadania je až po dátume zamietnutia zamietnutej požiadavky) (5 bodov).
- `public List<Poziadavka> akcnyPlan(double suma)` – vráti referenciu na zoznam spracovaných požiadaviek, pričom množstvo peňazí, ktoré musíme ešte investovať na ich dokončenie je nanajvýš *suma*, žiadna požiadavka nie je predražená a požiadaviek je maximálny možný počet (10 bodov).

Triedenie a komparátor (dokopy 3 body):

Vytvorte triedu `CenovyKomparator` implementujúcu `java.util.Comparator<Poziadavka>` s metódou (2 body):

- `public int compare(Poziadavka poziadavka1, Poziadavka poziadavka2)` – porovná požiadavky podľa doterajších nákladov minutých mestom v súvislosti z danými požiadavkami.

V triede `HelpDesk` implementujte inštančnú metódu (1 body):

- `public List<Poziadavka> vratPoziadavkyPodlaNakladov()` – vráti zoznam vyriešených požiadaviek usporiadaný podľa celkových nákladov na vyriešenie požiadavky.

Výnimky (3 body)

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