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.

SAD: Cestovné lístky

Motivácia: Spomínate si na cestu medzimestským autobusom? Jedna z prvých vecí, ktorú spravíte pri nastúpení do autobusu, je kúpa cestovného lístka zo zastávky, na ktorej práve teraz nastupujete, do cieľovej zastávky, kde vystúpite. Pri kúpe lístka oznámite vodičovi autobusu názov cieľovej zastávky, aký typ lístka kupujete („celý“, „študentský”, „ZŤP“, ...) a spôsob platby (hotovosť alebo elektronická peňaženka – karta). Po prejdení celej trasy sa v pokladni u vodiča nachádzajú záznamy o predaných lístkoch, ktoré sa v rámci uzávierky zosumarizujú. A práve táto sumarizácia bude cieľom tohto zadania.

Pohľad analytika: Pri implementácii budeme potrebovať:

- triedu `Listok`, ktorá bude uchovávať údaje o jednom predanom cestovnom lístku,
- triedu `ListkyNaTrase`, ktorá bude uchovávať zoznam všetkých predaných cestovných lístkov na trase (lístky, ktoré boli stornované, sa v tomto zozname nenachádzajú).

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

- **casPredaja** (čas predaja lístka ako čas, ktorý uplynul od okamihu, kedy autobus vyrazil na trasu zo svojho stanovišťa, napr. **„**`00:10:15`**“**).
- **nastup** (názov zastávky, kde sa lístok zakúpil, napr. **„**`Prešov, AS`**“**),
- **vystup** (názov zastávky, kde má cestujúci vystúpiť, napr. **„**`Košice, AS`**“**),
- **typ** (typ cestovného lístka, napr. **„**`ZŤP`**“**, **„**`OBYČAJNÝ`**“**, **„**`ŠTUDENT`**”**, …)
- **cena** (cena cestovného lístka v eurách, napr. `3.75`)
- **idKarty** (identifikátor karty, ak bol lístok zakúpený prostredníctvom elektronickej peňaženky, napr. **„**`ISIC12345678`**“**, **„**`DPMK31415`**”**, …)

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

Ďalej vytvorte aj triedu `sk.upjs.finalTerm.ListkyNaTrase`, ktorá bude uchovávať zoznam lístkov predaných na trase a tiež názvy zastávok tvoriacich trasu.

Konštruktory a pridávanie lístkov (4 body dokopy – povinné):

- `public Listok(String casPredaja, String nastup, String vystup, String typ, double cena, String idKarty)` – použije sa na vytvorenie záznamu o predaji cestovného lístka zaplateného bezhotovostne (kartou),

- `public Listok(String casPredaja, String nastup, String vystup, String typ, double cena)` použije sa na vytvorenie záznamu o predaji cestovného lístka zaplateného v hotovosti,
- `public ListkyNaTrase(List<String> trasa)` – použije sa na vytvorenie zoznamu cestovných lístkov predaných na trase. Trasa je popísaná ako zoznam zastávok (podľa poradia na trase), ktoré tvoria túto trasu.
- `public void pridaj(Listok listok)` – inštančná metóda v triede `ListkyNaTrase`, ktorá pridá záznam o predaji cestovného lístka na trase do zoznamu predaných lístkov.

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 "čas predaja `\t` nástup `\t` výstup `\t` typ `\t` cena `\t` id karty", resp. v tvare "čas predaja `\t` nástup `\t` výstup `\t` typ `\t` cena" ak bol lístok predaný v hotovosti (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 lístku (1 bod).

V triede `ListkyNaTrase`:

- `public static ListkyNaTrase zoSuboru(File f)` – statická metóda, ktorá z uvedeného súboru prečíta zoznam zastávok na trase a zoznam predaných lístkov. Odporúčaný formát je uložiť v prvom riadku súboru tabulátormi oddelený zoznam zastávok a v každom z nasledujúcich riadkov popis (záznam) jedného lístka (4 body).
- `public void uloz(File subor)` – uloží záznamy o zastávkach na trase a všetkých lístkoch do súboru v tvare, ktorý vie spracovať metóda `zoSuboru(File f)` (3 body).
- `public String toString()` – vráti reťazec vhodne reprezentujúci všetky predané lístky na trase (1 bod).

Inštančné metódy triedy `ListkyNaTrase` (za predpokladu, že lístky sa pridávajú chronologicky)**:**

- `public double trzba()` – vráti celkovú tržbu z predaných lístkov (1 bod).
- `public double trzbaVHotovosti()` – vráti celkovú tržbu z lístkov predaných v hotovosti (1 bod).
- `public double najdrahsiListok(String typ)` – vráti cenu najdrahšieho kúpeného lístka zadaného typu (4 body).
- `public List<String> zastavky()` – vráti zoznam všetkých zastávok, na ktorých stál autobus počas cesty (4 body + 3 body za usporiadanie vrátených zastávok podľa poradia na trase).
- `public Map<String, Integer> nastupy()` – vráti mapu, ktorá každej zastávke priradí, koľko cestujúcich na tejto zastávke nastúpilo (6 bodov).
- `public Set<String> bezhotovostneZastavky()` – vráti množinu všetkých zastávok (kde niekto nastúpil), na ktorých všetci nastupujúci cestujúci platili bezhotovostne (5 bodov).
- `public String najoblubenejsiaDestinacia()` – vráti zastávku, na ktorej vystúpilo najviac ľudí. Ak je takých viac, vráťte ktorúkoľvek z nich (5 bodov).
- `public String najvzdialenejsiCiel(String zastavka)` – vráti najvzdialenejšiu zastávku, kam cestoval cestujúci, ktorý nastúpil v zadanej zastávke (7 bodov).
- `public int pocetPlatitelov()` – vráti celkový počet platiteľov na trase. Platiteľom rozumieme osobu, ktorá platí za jeden alebo viac lístkov. Napríklad rodič s 2 deťmi kupuje 3 lístky, no platiteľom týchto 3 lístkov je len jedna osoba. Lístky, ktoré majú jedného platiteľa, môžeme rozoznať tak, že boli vydané v rovnaký čas (8 bodov).
- `public Map<String, Integer> pocetCestujucichVAutobuse()` – vráti mapu, ktorá každej zastávke priradí, koľko cestujúcich bolo v autobuse, keď autobus z tejto zastávky odchádzal (9 bodov).
- `public int najdlhsieZastavenie()` – vráti trvanie najdlhšieho zastavenia autobusu v sekundách na zastávke kvôli predaju lístkov. Trvanie zastavenia na zastávke vypočítajte ako čas medzi predajom prvého a posledného lístka na tejto zastávke. (7 bodov).

Triedenie a Comparable (dokopy 4 body):

Niekedy sa stane, že pokladňa v autobuse vytvorí súbor s predanými lístkami, ktoré nie sú usporiadané chronologicky. Vytvorte riešenie, ktoré umožní usporiadať lístky chronologicky.

V triede `Listok` implementujte rozhranie `java.util.Comparable<Listok>` s metódou:

- `public int compareTo(Listok l)` – porovná lístky podľa času predaja.

V triede `ListkyNaTrase` implementujte inštančnú metódu:

- `public void zoradChronologicky()`– usporiada lístky podľa času predaja.

Výnimky (3 body)

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

Rady a nápoveda:

Čas: Pri riešení úloh môže byť vhodné čas vo forme reťazca vo formáte "H:M:S" previesť na číslo vyjadrujúce počet sekúnd od začiatku merania času: 3600*H+60*M+S.

```
public static int casovyRetazecNaSekundy(String casovyRetazec) {
	Scanner citac = new Scanner(casovyRetazec);
	citac.useDelimiter(":");
	int hh = citac.nextInt();
	int mm = citac.nextInt();
	int ss = citac.nextInt();
	citac.close();
	return hh * 3600 + mm * 60 + ss;
}
```

Načítanie, resp. zapísanie trasy:

```
public static List<String> citajTrasu(String riadokSTrasou) {
	List<String> trasa = new ArrayList<String>();
	Scanner citac = new Scanner(riadokSTrasou);
	citac.useDelimiter("\t");
	while (citac.hasNext()) {
		trasa.add(citac.next());
	}
	citac.close();
	return trasa;
}

public static String trasaNaRetazec(List<String> trasa) {
	if (trasa.isEmpty())
		return "";
	StringBuilder sb = new StringBuilder();
	sb.append(trasa.get(0));
	for (int i=1; i<trasa.size(); i++)
		sb.append("\t" + trasa.get(i));
	return sb.toString();
}
```