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.

HC Turtles

Motivácia: Toto tu ešte nebolo. Jožko Turtlák dostal za úlohu vybudovať od nuly hokejový start-up *HC Turtles*. Všetky tímy v Extralige, KHL i NHL sú na pozore. Najdôležitejšia vec je zostaviť čo najvyváženejšie mužstvo s kvalitnými hráčmi na každom poste. Ako to už v start-upoch býva, budget (rozpočet) je limitovaný. Keďže trh s hráčmi je veľmi široký a aj výkonnosť samotných hráčov sa čas od času mení, potrebuje sa v „ponuke“ hráčov i celkovej situácii zorientovať. Ako človek, ktorý má k IT blízko, vie, že bez smart systému, ktorý by spravoval a analyzoval databázu registrovaných hráčov, to nepôjde.

Pohľad analytika: Pri implementácii budeme potrebovať:

- triedu `Hrac`, ktorá uchováva informácie o jednom registrovanom hráčovi,
- triedu `TrhHracov`, ktorá uchováva zoznam hráčov a poskytuje základné metódy „business intelligence“.

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

- **meno** (napr. `„Zdeno Char”, „Richard Murár”`, …),
- **datumNarodenia** (napr. `„26.1.1996“`) – dátum narodenia,
- **post** (jedno z `brankár`, `útočník` alebo `obranca`),
- **vykonnost** - u útočníkov a obrancov je to počet kanadských bodov, u brankárov percentuálna úspešnosť zákrokov (0-100); pri nových hráčoch má hodnotu 0,
- **plat** (napr. `999`) – očakávaný základný mesačný plat v celých eurách,
- **klub** (napr. `„Rýchle korytnačky“`) – aktuálny klub, za ktorý hráč hrá – môže byť aj `null`, ak je hráč aktuálne bez zmluvy (hovorím tiež, že hráč je voľný),
- **zmluvaDo** (napr. `„5.2.2016“`) – dátum, do ktorého je platná zmluva – môže byť aj `null`, ak je hráč aktuálne bez zmluvy.

Upozornenie: Zadanie triedy `Hrac` 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.TrhHracov`, ktorá bude uchovávať nejaký zoznam hráčov. Nepredpokladáme, že hráči majú v zozname nejaké konkrétne usporiadanie. Zároveň tento zoznam uchováva údaje o hráčoch k aktuálnemu dátumu (ak má hráč nastavený nejaký klub, tak ku aktuálnemu dátumu je hráč hráčom tohto klubu).

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

- `public Hrac(String meno, String datumNarodenia, String post, double vykonnost, int plat, String klub, String zmluvaDo)` – použije sa na vytvorenie záznamu o hráčovi s platnou zmluvou.
- `public Hrac(String meno, String datumNarodenia, String post, double vykonnost, int plat)` – použije sa na vytvorenie záznamu o hráčovi bez zmluvy.
- `public void pridaj(Hrac hrac)` – inštančná metóda v triede `TrhHracov`, ktorá pridá záznam hráča do zoznamu hráčov.

Práca so súbormi (povinné):

V triede `Hrac`:

- `public static Hrac zoStringu(String popis)` – statická metóda, ktorá vráti referenciu na novovytvorený objekt triedy `Hrac`. Parameter je `String` v tvare "meno `\t` datumNarodenia `\t` post `\t` vykonnost `\t` plat", resp. "meno `\t` datumNarodenia `\t` post `\t` vykonnost `\t` plat `\t` klub `\t` zmluvaDo", ak má hráč platnú zmluvu (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 hráčovi (1 bod).

V triede `TrhHracov`:

- `public static TrhHracov zoSuboru(String nazovSuboru)` – statická metóda, ktorá z uvedeného súboru prečíta zoznam hráčov, pričom v každom riadku bude popis jedného hráča (4 body).
- `public void uloz(String nazovSuboru)` – uloží všetky záznamy o všetkých hráčoch v zozname do súboru v tvare, ktorý vie spracovať metóda `zoSuboru` (3 body).
- `public String toString()` – vráti reťazec vhodne reprezentujúci všetkých hráčov v zozname (1 bod).

Inštančné metódy triedy `TrhHracov`:

- `public double priemernyPlatAktivnych()` – vráti priemerný očakávaný plat tých hráčov, ktorí majú platnú zmluvu (1 bod).
- `public boolean spoluhraci(String meno1, String meno2)` – vráti, či uvedení hráči sú aktuálne spoluhráči, t.j. hrajú v rovnakom klube (2 body).
- `public List<String> kluby()` – vráti mená (bez opakovania) všetkých klubov v zozname (2 body).
- `public double medianPlatovAktivnych()` – vráti medián (prostrednú hodnotu) očakávaných platov tých hráčov, ktorí majú platnú zmluvu (3 body).
- `public String liderKlubu(String klub)` – vráti meno najvýkonnejšieho hráča v zadanom klube, ktorý nie je brankárom (3 body).
- `public Map<String, Integer> velkostiKlubov()` – vráti map, v ktorom každému klubu bude priradený aktuálny počet jeho hráčov (4 body).
- `public String najbohatsiKlub()` – vráti meno klubu, ktorý dáva dokopy najviac na výplaty svojich hráčov, ak je takých viac, vráťte ľubovoľný z nich (5 bodov).
- `public List<Hrac> volniHraci(String datum)` – vráti zoznam hráčov, ktorí by mali byť voľní k zadanému dátumu za predpokladu, že sa v medzičase neuskutočnia nákupy hráčov (6 bodov).
- `public void zoradPodlaKlubovAVykonu()` – zaradí hráčov abecedne podľa klubov, pričom v rámci klubu sú najprv uvedení brankári a potom ďalší hráči. Brankári aj hráči sú zoradení podľa výkonnosti – čím je hráč lepší, tým je v rámci klubu uvedený skôr. Hráči bez klubu sú na konci zoznamu a usporiadaní sú podľa výkonnosti počnúc najlepším. Aj v tomto prípade najprv uvádzame brankárov a potom hráčov na iných postoch (6 bodov).

- `public Map<String, Hrac> brankarskeJednotky()` - vráti map, v ktorom bude každému klubu priradená brankárska jednotka – brankár s najväčšou úspešnosťou (výkonnosťou). Ak klub brankára nemá (napr. sa ešte len formuje), bude mu priradená hodnota `null` (7 bodov).
- `public String najskusenejsiTim()` - vráti meno tímu, ktorý má vo svojich radoch najstaršieho hráča, ak je najstarších hráčov viac, uprednostní toho s najvyššou produktivitou (7 bodov)
- `public List<Hrac> zostavTim(int limit)` – zostaví tím z voľných hráčov, ktorý bude spĺňať:
    - tím sa skladá z brankára, troch útočníkov a dvoch obrancov,
    - súčet očakávaných platov hráčov v tíme je menší ako zadaný limit,
    - celkový súčet výkonnosti hráčov (nebrankárov) prenásobený percentuálnou úspešnosťou brankára je najväčší možný.

  Ak sa požadovaný tím nedá zostaviť, metóda nech vráti `null`. Hodnotenie: 10 bodov
    - **Bonus**: Upravte metódu tak, aby okrem vrátenia `null` vypísala aj informáciu, prečo sa tím nepodarilo zostaviť. Ak je príčinou nedostatok voľných hráčov na určitých pozíciách, vypíšte na akých a koľko. Ak je príčinou nízky finančný limit, vypíšte s akým finančným limitom by sa tím podarilo zostaviť (+4 body).
- `public Map<String, Integer> utocneSilyKlubov()` – vráti map, v ktorom každému klubu bude priradená útočná sila klubu. Útočná sila klubu je súčet výkonnosti (kanadských hodnotení) najlepších troch útočníkov klubu. Ak má klub menej ako 3 útočníkov (t.j. nevie zostaviť ani jednu úplnú útočnú formáciu), v mape sa nebude nachádzať (10 bodov).

Výnimky (3 body)

Vytvorte nekontrolovanú výnimku `NeznamyPostException` a vhodne ju použite v konštruktoroch triedy `Hrac`.