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, 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.

Amatérska badmintonová liga

zdroj: raphaelai.org

Motivácia: Študenti UPJŠ sa rozhodli, že počas letného semestra 2025/2026 chcú mať sezónu amatérskej badmintonovej ligy. Hrá sa 14 týždňov, teda od pondelka 9.2. do nedele 17.5. hráči vždy hrajú jeden proti jednému. Badminton hrajú podľa nasledujúcich pravidiel zápas sa hrá na 2 víťazné hry, víťaz hry musí mať aspoň 21 bodov a vyhrať aspoň o 2 body. Zápas môže skončiť 21:13, 22:20 ak prvý hráč vyhral už po dvoch hrách alebo 17:21, 24:22, 21:19 ak sa hralo na 3 hry.

Pohľad analytika: Pri implementácii aplikácie budeme potrebovať:

- triedu `Match`, ktorá reprezentuje jeden zápas,
- triedu `Season`, ktorá bude uchovávať zoznam všetkých zápasov v sezóne.

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

- **playerA** – meno prvého hráča (napr. Jožko Mrkvička, alebo Mária Nováková),
- **playerB** – meno druhého hráča,
- **score** – v tvare A1:B1, A2:B2 ak sa hrali 2 hry alebo A1:B1, A2:B2, A3:B3 ak sa hrali 3 hry (napr. 21:18, 23:21 alebo 21:15, 21:23, 17:21),
- **day** – poradové číslo dňa sezóny (napr. 8), pondelok 9.2. je deň 0,
- **startTime** – čas začiatku prvého setu v tvare HH:MM (napr. 20:30),
- **endTime** – čas konca posledného setu v rovnakom tvare ako startTime, môžete predpokladať, že sa žiaden zápas nehral cez polnoc,
- **spectators** – zoznam hráčov ligy ktorí sledovali zápas ako diváci, (napr. „Ján Novák, Mária Nováková“), ak je divákov viac sú oddelení čiarkami.

Upozornenie: Zadanie pre triedu `Match` 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.Season`, ktorá bude uchovávať zoznam zápasov (Match).

Konštruktory a pridávanie prezentácií do plánu podujatia (povinné):

- `public Match(String playerA, String playerB, String score, int day, String startTime, String endTime, String spectators)` – použije sa na vytvorenie zápasu s divákmi.
- `public Match(String playerA, String playerB, String score, int day, String startTime, String endTime)` – použije sa na vytvorenie zápasu bez divákov.
- `public void pridaj(Match match)` – inštančná metóda v triede `Season`, ktorá pridá zápas do sezóny.

Práca s reťazcami a súbormi (povinné):

V triede `Match`:

- `public static Match fromString(String input)` – statická metóda, ktorá vráti referenciu na novovytvorený objekt triedy `Match`. Parameter je `String` v tvare "`playerA;playerB;score;day;startTime;endTime;spectators`", resp. "`playerA;playerB;score;day;startTime;endTime`", pre zápas bez divákov. *Poznámka:* Poznámka: Scanner-u môžete povedať, že oddeľovač má byť bodkočiarka zavolaním jeho metódy useDelimiter(";").
- `public String toString()` – vráti reťazec vhodne reprezentujúci údaje o zápase.

V triede `Season`:

- `public static Season loadSeason(String fileName)` – statická metóda, ktorá z uvedeného súboru prečíta informácie o všetkých zápasoch sezóny, pričom v každom riadku bude popis jedného zápasu.
- `public void saveSeason(String fileName)` – uloží všetky zápasy v sezóne do súboru v tvare, ktorý vie spracovať metóda `loadSeason`.
- `public String toString()` – vráti reťazec vhodne reprezentujúci všetky zápasy sezóny.

Za povinnú časť bude udelených 15 bodov. Nasledujúce (nepovinné) úlohy môžete riešiť v ľubovoľnom poradí. Ale riešenie jednej môže zjednodušiť nasledujúce.

Odporúčané (nepovinné) metódy triedy `Match`:

- Odporúčame vytvoriť si pomocnú metódou, ktoré vrátia `score` vo formáte, ktoré je vhodný na spracovanie v ďalších metódach.
- Odporúčame vytvorenie metódy, ktorá vráti čas jedného zápasu v minútach.
- Odporúčame vytvorenie metódy, ktorá vráti zoznam divákov vo vhodnom formáte.
- Odporúčame vytvorenie metódy, ktoré vrátia víťaza a/alebo porazeného.
- Ďalšie pomocné metódy podľa potreby.

Inštančné metódy triedy `Season`:

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

- [1b] `public int numberOfMatches()` – vráti koľko zápasov sa odohralo v sezóne.
- [1b] `public int playTime()` – koľko minút trvali dokopy všetky zápasy sezóny.
- [2b] `public int longestMatch()` – vráti dĺžku najdlhšieho zápasu v minútach.
- [2b] `public List<String> players()` – vráti zoznam všetkých divákov, bez duplicít.
- [2b] `public Match topMatch()` – vráti zápas, ktorý mal najviac divákov.
- [2+1b] `public double twoGameMatches()` – vráti koľko % zápasov sa hralo iba na 2 hry, výsledok zaokrúhlite na 2 desatinné miesta.
- [3b] `public int victoriesOf(String player)` – vráti koľko zápasov vyhral hráč zadaný parametrom `player`.

- [3b] `public Map<String, Integer> matchesPerPlayer()` – vráti mapu, kde každému hráčovi je priradené koľko zápasov odohral.
- [3b] `public String topSpectator()` – vráti meno diváka, ktorý videl najviac zápasov.
- [4b] `public List<String> nightPlayers(String time)` – vráti zoznam všetkých hráčov, bez duplicít, ktorí **každú** svoju hru začali po čase zadanom parametrom `time`.
- [3b] `public int[] dailyHistogram()` – metóda vráti histogram počtu odohraných hier počas dňa v týždni (pondelok 0, utorok 1, streda 2, ..., nedeľa 6), napr. na indexe 2 je počet hier odohraných v stredu. Môžete predpokladať, že sezóna začala v pondelok.
- [4b] Napíšte metódu `public Map<String, Integer> eloAfterSeason()`, ktorá vráti hodnotenie „ELO“ pre každého hráča na konci sezóny. Na začiatku sezóny začne každý hráč s ELO 1000. Po každom zápase sa mení ELO víťaza aj porazeného. Pred hrou mali víťaz a porazený nasledujúce hodnotenie, *eloWinnerBefore* a *eloLoserBefore*, z nich sa spočíta *winnerChance* pre každý zápas:

$$winnerChance = \frac{1}{1+10^{x}}, kde\ x = \frac{(\text{eloLoserBefore} - \text{eloWinnerBefore})}{400}$$

  Hodnotenie po hre je $eloWinnerAfter = eloWinnerBefore + 32(1 - winnerChance)$,
  $eloLoserAfter = eloLoserBefore - 32(1 - winnerChance)$.
  Pozn.: Táto metóda vyžaduje aby sa vstup spracovával chronologicky. Môžete predpokladať, že zápasy sú usporiadané chronologicky. Alebo ich usporiadajte, viď posledná úloha.

- [4b] `public int longestPersonalStreak(String player)` – vráti najviac koľko dní za sebou daný hráč odohral bez prestávky.
- [4b] `public boolean watchAndPlay(String player)` – vráti či daný hráč videl aspoň jeden zápas každého svojho súpera.
- [6b] `public void topKWinratePlayers(int k)` – metóda vypíše *k* hráčov ktorí majú najlepší pomer víťazstiev ku počtu ich hier. Hráči sú usporiadaný podľa pomeru. Ak je k väčšie ako počet hráčov, tak metóda vypíše všetkých.
- [6b] `public boolean everyonePlaysEveryone()` – metóda vráti či každý hráč hral aspoň raz s každým iným hráčom počas sezóny.

Nezaradené úlohy

[3b] Pridajte nekontrolovanú výnimku InvalidScoreFormatException, ktorá sa bude vhodne týkať prípadu, ak je *score* v nekorektné. To znamená, že je zlý počet hier alebo počet bodov nespĺňa, že víťaz musí mať aspoň 21 bodov a víťaz musí vyhrať o 2 body. Výnimku použite na vhodnom mieste.
[3b] Vytvorte metódu, ktorá usporiada všetky zápasy v sezóne chronologicky (podľa dňa začiatku a času začiatku).