Haladó Java gyakorlat 2023.06.30.

Határidő Nincs megadva határidő **Pont** 40 **Kérdések** 1 **Elérhető** jún 30, 17:45 - jún 30, 19:50 körülbelül 2 óra
Időkorlát Nincs **Engedélyezett próbálkozások** Korlátlan

Ezt a kvízt ekkor zárolták: jún 30, 19:50 .

Próbálkozások naplója

	Próbálkozás	Idő	Eredmény
LEGUTOLSÓ	1. próbálkozás	104 perc	0 az összesen elérhető 40 pontból *

* Néhány kérdés még nem lett értékelve

Ezen próbálkozás eredménye: **0** az összesen elérhető 40 pontból *
Beadva ekkor: jún 30, 19:31
Ez a próbálkozás ennyi időt vett igénybe: 104 perc

1. kérdés **Még nincs értékelve / 40 pont**

Haladó Java ZH, 2023.06.30.

Feltételek

- A feladat megoldását önállóan, más segítsége nélkül kell elkészíteni.

- Kommunikáció csak az oktatókkal megengedett.
  - Az elkészített megoldást nemcsak a ZH végéig, hanem egészen a ZH napjának végéig nem szabad megosztani mással (pl. fórumba vagy publikus verziókezelő rendszerbe felöltés).
  - A megoldás elkészítéséhez használható a Java API és a JUnit dokumentációja. Ezek a Canvasből letölthetők, kicsomagolhatók.
- Az elkészített megoldást **zip** formátumba csomagolva kell feltölteni a Canvasbe.
  - A `zip` tartalmazza a forrásfájlokat és a járulékos `txt` fájlokat.
  - A **ZH végén kb. 10 percet érdemes fenntartani** a kód tisztázására, fordíthatóvá tételére, tömörítésére, beküldésére.

A program elkészítéséről

A program minden része legyen a lehető legjobb minőségű.

Ahol "lambda" szerepel egy típus megnevezésében, a legalkalmasabb funkcionális interfészt kell kiválasztani a megvalósításhoz.

A program mindegyik részét, amennyire csak lehetséges,

`Stream`ek használatával kell elkészíteni.

Ha a szöveg külön nem említi, feltételezhető, hogy az adatok a megfelelő formátumúak és tartalmúak.

Innen letölthetők tesztfájlok.

ValidatedBy (2 pont)

Készíts egy `zh.ValidatedBy` annotációt, amit csak típusokra lehet feltenni. Az annotációnak egy String adattagja van, ami annak a függvénynek a neve, ami eldönti egy objektumról, hogy az érvényes-e.

TimeStamp (10 pont)

Készítsd el a `zh.TimeStamp` osztályt. Az osztálynak egyetlen `int` adattagja van, mely az éjfél óta eltelt perceket jelképezi. Ne lehessen közvetlen létrehozni. Készítsd el a következ függvényeket:

- `fromString(String)` egy statikus metódus, ami egy `TimeStamp` -et hoz létre. A paraméter formátuma: `hh:mm` azaz 2 számjegy ami az órát, 2 számjegy, ami a percet jelöli, kettősponttal elválasztva. (Valid értékek: `08:20` , `11:02` , `17:30` , stb.)
- `fromTime(int)` egy statikus metódus, ami egy `TimeStamp` -et hoz létre, a paraméter az éjfél óta eltelt percek száma.
- `toString()` objektum szintű metódus, ami `hh:mm` formátumban adja vissza a TimeStamp értékét
- `diff(TimeStamp)` megmondja hány perc telt el eközött (`this`) és a paraméterül kapott `TimeStamp` között.
- `isValid()` objketum szintű adattag, ami akkor tér vissza igazzal, ha a rögzített percek száma legalább 1.
- `minutes` megadja, hány percet jelöl az adott `TimeStamp`
- Helyezd el a `TimeStamp` osztályon a `ValidatedBy` annotációt és add neki értékül a `filteringCheck` értéket. A `filteringCheck()` legyen egy boolean-nel visszatérő objektum szintű metódus, ami az `isValid()` értékével tér vissza.

ShipLog (10 pont)

A `zh.ShipLog` egy hajó adatatait tartalmazza egy versenyen. 4 adattagja van:

- `String name` a hajó neve
- `String shipClass` a hajó verseny osztálya
- `TimeStamp start` az indulás ideje
- `TimeStamp finish` a célbaérés ideje

Helyezd el a `ShipLog` osztályon a `ValidatedBy` annotációt és add neki értékül az `allFilledOut` értéket. Az `allFilledOut()` legyen egy boolean-nel visszatérő objektum szintű metódus, ami akkor tér vissza igazzal, ha a `ShipLog` objektum midnen mezője ki van töltve (nincs benne null, vagy üres, csak `whitespace` -ekből álló rész), és minden `TimeStamp` valid.

Minden adattag legyen megváltoztathatatlan. Készíts hozzájuk lekérdező metódust.

Készts egy statuikus gyártó függvényt `readShip(String)` néven, ami egy String-et kap paraméterül. A Stringben 4 mező szerepel `,` -vel elválasztva, pl: `Thetisz,Sudar Sport,12:20,14:30`. Az első mező a hajó neve, a második a verseny osztály neve, a harmadik a kezdés időpontja, a negyedik a célbaérés időpontja.

- Ha a kezdés időpontjánál (3. adattag) `DNS` (did not start) szerepel, akkor a hajó nem indult el. Ezt jelölje érvénytelen `TimeStamp`.
- Ha a célbaérés időpontjánál (4. adattag) `DNF` (did not finish) szerepel, azt jelenti, hogy a hajó nem ért célba. Ezt jelölje érvénytelen `TimeStamp`.

Készítsd el az alábbi metódusokat:

- `started()` egy `boolean`-nal visszatérő objektum szintű függvény, ami akkor igaz, ha a start `TimeStamp` valid.
- `finished()` egy `boolean`-nal visszatérő, objektum szintű metódus, ami akkor igaz, ha a finish `TimeStamp` valid.
- `time()` egy `int`-tel visszatérő metódus, ami megadja, hogy mennyi ideig tartott a hajónka a futam.

Comparator (2 pont)

Készíts egy publikus, statikus, megváltoztathatatlan `RUN_COMPARATOR` nevű, `Comparator<ShipLog>` típusú adattagot a `ShipLog` osztályba, mely növekvő sorba tudja tenni a `ShipLog`-okat aszerint, hogy mennyi idő telt el az indulás és a célbaérkezés között.

Győztesek (5 pont)

Készítsd el a `winnersByClass(Stream<ShipLog>)` statikus metódust a `ShipLog` osztályba. Ez egy olyan Stream-et kap, amiben `ShipLog`-ok találhatók, a visszatérési értéke pedig `Map<String, Optional<ShipLog>>` ahol a kulcs a hajóosztály neve, az érték pedig a leggyorsabb hajó verseny logja az osztályban. A függvény végezze el ezeket a műveleteket, ebben a sorrendben:

- figyelmenkívül hagyja az olyan logokat, ahol a hajó nem indult el
- figyelmenkívül hagyja az olyan logokat, ahol a hajó nem ért célba
- Figyelmenkívül hagyja az olyan hajókat, akiknél az érkezés ideje kissebb, vagy egyenlő az indulási idővel
- kigyűjti egy `Map`-be osztályonként melyik hajó nyert. (Azt a `ShipLog` értéket, ami az adott versenyosztály legjobb idejét írja le.) (Tipp: használhatod a `Collectors.groupingBy` és a `Collectors.minBy` metódusokat.)

Category (2 pont)

Készítsd el a `zh.Category` nevezetű **felsorolási típust** , aminek 4 értéke értéke lehet:

- `RACER`
- `LAZY`
- `WEAK`
- `CHEATER`

Minden adattagnak legyen egy `accept(ShipLog)` metódusa, ami egy `boolean` értékkel tér vissza; a:

- `LAZY` esetén igazzal, ha a hajó nem indult el
- `WEAK` esetén igazzal, ha a hajó elindult, de nem ért célba
- `RACER` esetén igazzal, ha a hajó elindult, célbaért, és az indulási idő előbb van, mint az érkezési idő
- `CHEATER` esetén igazzal, ha a hajó elindult, célba ért, és az indulási idő több, vagy egyenlő, mint az érkezési idő

Készíts egy `categorize(ShipLog)` nevű statikus függvényt a `Category` típusba, ami azt a `Category`-t adja vissza, ami elfogadja a paraméterül kapott `ShipLog` objektumot.

Utils (9 pont)

Készíts egy `zh.Utils` nevezetű osztályt. Az osztályna ne lehessen példányát létrehozni. Legyen rajta 3 statikus, publikus metódus.

- `<T> void addFiltered(List<T>, List<T>, Predicate<T>)` olyan generikus metódus, ami az első listából az összes olyan elemet beszúrja a második listába, amire a predikátum igazzal tér vissza.
- `<T> void saveTo(String, Function<T, String>, Stream<T>)` olyan generikus metódus, ami veszi az elemeket a Stream-ből, meghívja rajtuk a `Function`-t, és az így kapott `String`-eket mint külön sorok lementi az első paraméterül kapott névvel megjelölt file-ba. Ha még nem létezik a file, akkor hozza létre. Ha létezik, akkor az új adatokat fűzze hozzá. Az új adatok és az előző adatok között legyen sorvége jel (`\n`). A file kezeléséhez a **NIO** eszközkészletét használd.
- `<T> List<T> filterByValidity(List<T>)` készíts egy generikus metódust, ami egy listából egy új listát állít elő, olyan módon, hogy a lista minden elemére ellenőrzi, hogy adott elemen van-e `ValidatedBy` annotáció:
- ha nincs, akkor az elemet elfogadja
- ha van, akkor azzal a metódussal ellenőrizze, hogy az adott objektum megfelel-e a fletételnek (a metódus `true`-val tér vissza), és csak azokat tartsa meg, amiket sikerült is validálni.

↓ **src.zip (https://canvas.elte.hu/files/2324649/download)**

Kvízeredmény: **0** az összesen elérhető 40 pontból