Adatszerkezetek – gyakorlati jegy

Hallgatói nyilvántartás

Időtartam: 90 perc · **Nyelv:** Java vagy Go (választható) · **Gép:** a terem számítógépe
Használható: a kurzus példaprogramjai és diái, saját jegyzet · **Nem használható:** internet, mesterséges intelligencia (MI) alapú eszközök, más hallgató segítsége

A feladat

Egy egyetem hallgatóiról nyilvántartást vezetünk. Minden hallgatónak van **azonosítója** (`id` , egész szám), **neve** és **pontszáma** (0–100). Valósítsa meg a kiinduló kódban **TODO**-val jelölt műveleteket **saját adatszerkezettel**!

	Java	Go
Kiinduló kód	`gyakorlati/java/StudentRegistry.java`	`gyakorlati/go/main.go`
Futtatás	`java StudentRegistry.java`	`go run .` (a `go` mappában)
Tiltott a megoldásban	`java.util` gyűjtemények (`ArrayList` , `LinkedList` , `HashMap` , `TreeMap` …)	`container/list` , `map` , beépített rendezés

A program a végén **szintenként** kiírja, mely ellenőrzések sikeresek (**OK**) és melyek nem (**HIBA**). Az ellenőrző részt **ne módosítsa**!

A szintek

A szinteket **sorban** kell teljesíteni: egy szint csak akkor számít, ha az összes előző is hibátlan.

1. szint – elégséges (2): egyirányú láncolt lista

- `add(s)` – új hallgató a lista **végére**
- `size()` – a hallgatók száma
- `find(id)` – keresés azonosító szerint (nincs találat: `null` / `nil`)
- `toString()` / `String()` – pl. `[1003 Li Wei 87, 1001 Wang Fang 92]` , üres lista: `[]`

2. szint – közepes (3)

- `remove(id)` – a hallgató törlése; igaz, ha volt ilyen (az **első** elem törlésére is figyeljen!)
- `averageScore()` – a pontszámok átlaga (üres listánál 0)
- `best()` – a legnagyobb pontszámú hallgató; egyenlőség esetén a listában előbb álló

3. szint – jó (4)

- `addSortedByScore(s)` – beszúrás úgy, hogy a lista pontszám szerint **csökkenő** sorrendű maradjon; egyenlő pontszám esetén az új hallgató a már bent lévők **mögé** kerüljön (stabil beszúrás)
- `reverse()` – a lista megfordítása **helyben**, új csomópontok létrehozása nélkül

4. szint – jeles (5): bináris keresőfa (kulcs: az azonosító)

- `insert(s)` – beszúrás; ismétlődő azonosítót **nem** szúr be újra
- `find(id)` – keresés azonosító szerint
- `height()` – a fa magassága (üres fa: −1, egyetlen csomópont: 0)
- `inorderIds()` – az azonosítók inorder sorrendben, pl. `[1001, 1002, 1003]`

A 4. szint ellenőrzése egy **mérést** is futtat (20 000 keresés listában és keresőfában). Figyelje meg az eredményt – a szóbeli részben kérdezni fogunk róla!

Értékelés

1. **Automatikus ellenőrzés:** a program által kiírt szint a **legmagasabb elérhető** jegy.
2. **Szóbeli ellenőrzés** (kb. 5 perc): el kell tudnia magyarázni a saját kódját, és válaszolnia kell 2–3 kérdésre (pl. *Mi a lépésszáma a `find` műveletnek? Mi történik az első elem törlésekor?*).
   Ha a magyarázat nem meggyőző, a jegy **csökkenhet**.
3. **Nem fogadható el** a megoldás, ha nem fordul le, ha tiltott gyűjteményt használ, vagy ha az ellenőrző részt módosították.

Tippek

- Haladjon **szintenként**, és minden metódus után futtassa a programot!
- Gondoljon a **speciális esetekre**: üres lista, egyelemű lista, az első és az utolsó elem.
- A 9., 10., 14. és 15. hét példaprogramjai sokat segítenek.
- Mentse gyakran a munkáját!

Név: ..... **Azonosító:** ..... **Választott nyelv:** Java / Go