Practical grade – grading guide

Preparation

- Copy to the students' computers: the `practical/java/` and `practical/go/` folders (starter code), and the slides and the `code/` demo programs of the course.
- Must **not** be put on the students' computers: the `practical/solution/` folder (reference solution in Java and Go).
- Check on the computers: `java -version` (17+) and `go version` (1.23+). If possible, disable internet access.
- The starter code is generated from the reference solution: `python3 eszkozok/kiindulo_kod.py <solution> <output> java|go` – this guarantees that the checker part is identical.

How to assess (about 5–7 minutes per student)

1. **Run:** `java StudentRegistry.java` or `go run .` – the program prints the level reached automatically.
2. **Quick code review:**
   - does it use a forbidden collection (`ArrayList` , `LinkedList` , `HashMap` , `container/list` , `map` …)?
   - has the checker part (the `main` and `check` functions) been modified?
   - is it “cheating” (e.g. `toString` returns a fixed text, or `find` does not search in the student's own list)?
3. **Oral questions:** 2–3 questions from the list below, according to the student's level.
4. **Grade:** the automatic level, which **may be lowered by one grade** if the student cannot explain their own code; in serious cases (obviously not their own work) fail.

Oral questions

Levels 1–2

- How many steps does the `add` operation take? How could it be O(1)? *(End pointer / tail)*
- What happens in `remove` when the first element is deleted? Why must the previous element be remembered too?
- What does `find` return if the list is empty? Why does the program not crash?
- Why is `find` O(n) in a linked list, and why can binary search not be used?

Level 3

- How do you make `addSortedByScore` stable? *(`>=` in the condition – it steps behind the equal ones)*
- Show how `reverse` works on a three-element list! Why must the `next` pointer be saved?
- How many steps does sorted insertion take? *(O(n))*

Level 4

- What does the measurement show? Why is the search tree faster? *(O(log n) on average vs O(n))*
- What would the height of the tree be if the ids were inserted in **increasing** order? What would follow from that?
- Why does the inorder traversal give the sorted order?
- How would you delete a node with two children? *(inorder successor)*

Common mistakes to watch for

Mistake	Symptom in the check
`add` does not handle the empty list	`NullPointerException` / nil pointer panic at level 1
`remove` does not update the head for the first element	“deleting the first element” FAIL
`best()` returns the later equal one because of `>=`	“Best() -> Wang Fang” FAIL
`addSortedByScore` with a `>` condition – not stable	“stable order” FAIL
`reverse` creates new nodes	the check passes – ask about it **orally**!
the BST does not handle duplicates	“a duplicate id is not inserted” FAIL

Alternative: as a take-home assignment

The task can also be given as a **take-home assignment** (as in the old course): the student receives it one week before week 16 and **defends it orally** in week 16. In this case the oral part has more weight (10 minutes, at least 4 questions), and the student must be able to explain every line of the solution.