CMPT 280

Topic 11: Ordered Binary Trees

Mark G. Eramian

University of Saskatchewan

References

- Textbook, Chapter 11

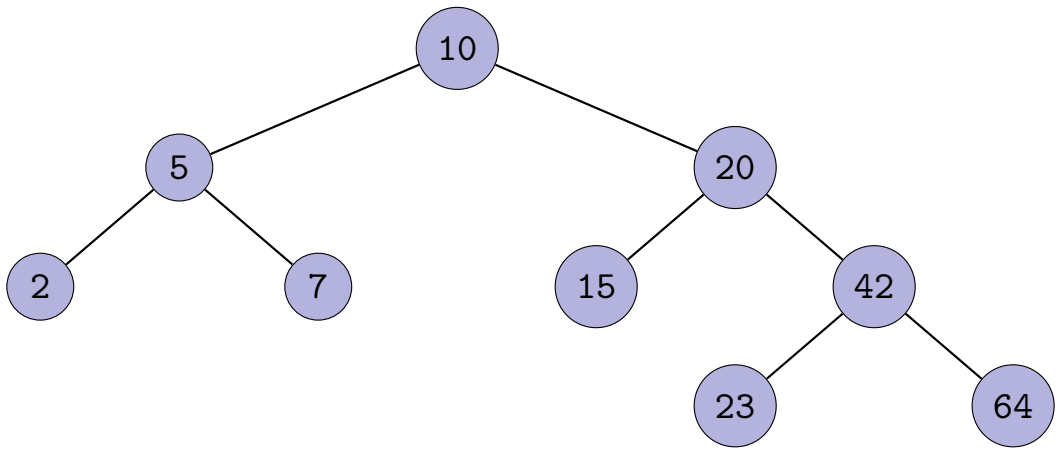
Searchable Dispensers

Reading Review

We learned in the readings that ordered binary trees are *searchable dispensers*.

- What are the properties of *searchable dispensers*?

Exercise 1



- Which nodes are visited when searching for 7? 20? 23?
- Where would 9 get inserted? What about 17? 21? 3? 42?

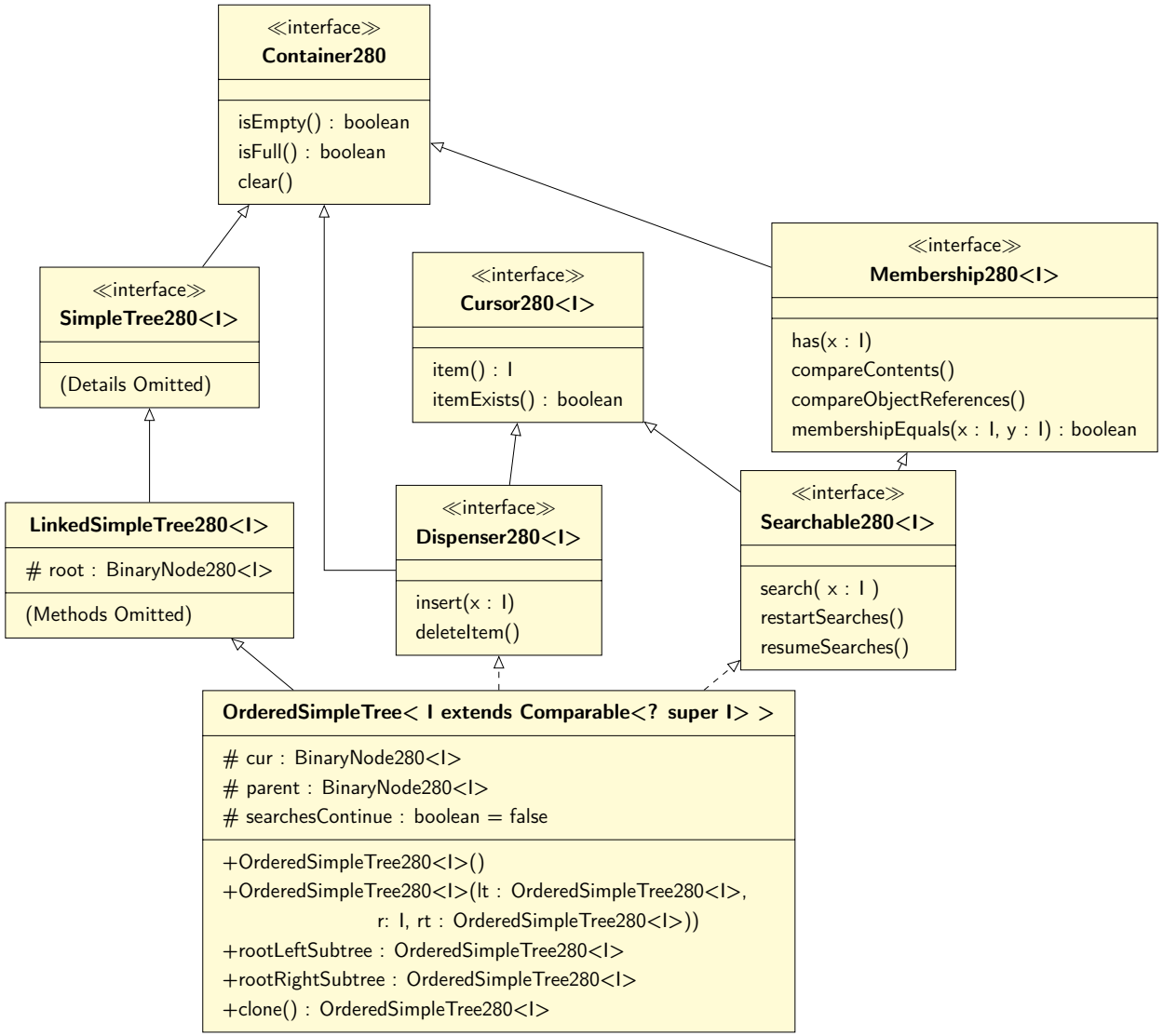
Ordered Binary Trees

Operations on Ordered Binary Trees consist of all the operations of a LinkedSimpleTree280, plus:

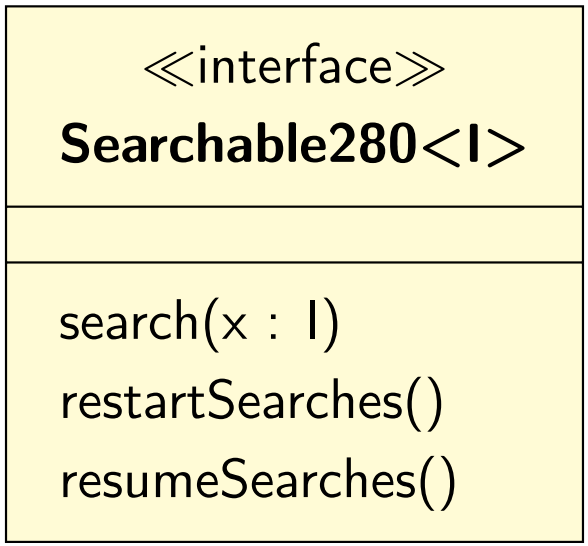
- `item`, `itemExists` – container with cursor!
- `insert`, `has`, `deleteItem` – a dispenser!
- `search`, resume/restart search – a searchable dispenser!

But... user can't insert wherever they want and arbitrary manipulation of the cursor is not allowed.

Inheritance Hierarchy of `OrderedSimpleTree280`



Additional methods in Searchable280<I>



- In the reading we omitted discussion of `restartSearches()` and `resumeSearches()`.
- These methods set an internal state that determines whether searches always start from the beginning or resume from the current cursor position.

Exercise 2

a) What should the class header be for `OrderedSimpleTree280<I>`?

b) What instance variables should it have?

Exercise 3

Searching in an ordered binary tree involves moving the cursor between tree levels. It will help if we have some cursor methods specialized for trees.

a) Write a method called `above` which returns true if the cursor is positioned above the root, and false otherwise.

b) Write a method called `below` which returns true if the cursor is positioned below the last level of the tree, and false otherwise.

c) Write the `itemExists` method.

d) Write the `item` method.

Exercise 4

- Write the `search()` method which, given an element, positions the tree's cursor at that element (if it exists in the tree).

Reminder: we need to respect the state of the `searchesContinue` variable.

Exercise 5

- Write the `has()` method which, given an element, returns true if the tree contains the element, and returns false otherwise.
- What is the worst-case time complexity of our method?

Reminder: previous searches must still be resumable, so a postcondition of this method must be that the cursor position after the method is executed is the same as the cursor position when the method is called.

Hint: can we use any methods we have already written to make this one easy?

Exercise 6

- Write the `insert()` method which, given an element, inserts it into the ordered binary tree at the appropriate position.
- What is the worst-case time complexity of our method?

Hint: this method is easier to write if you don't use the cursor.

Next Class

- Next class reading: Chapter 12: Ordered Binary Tree Deletion