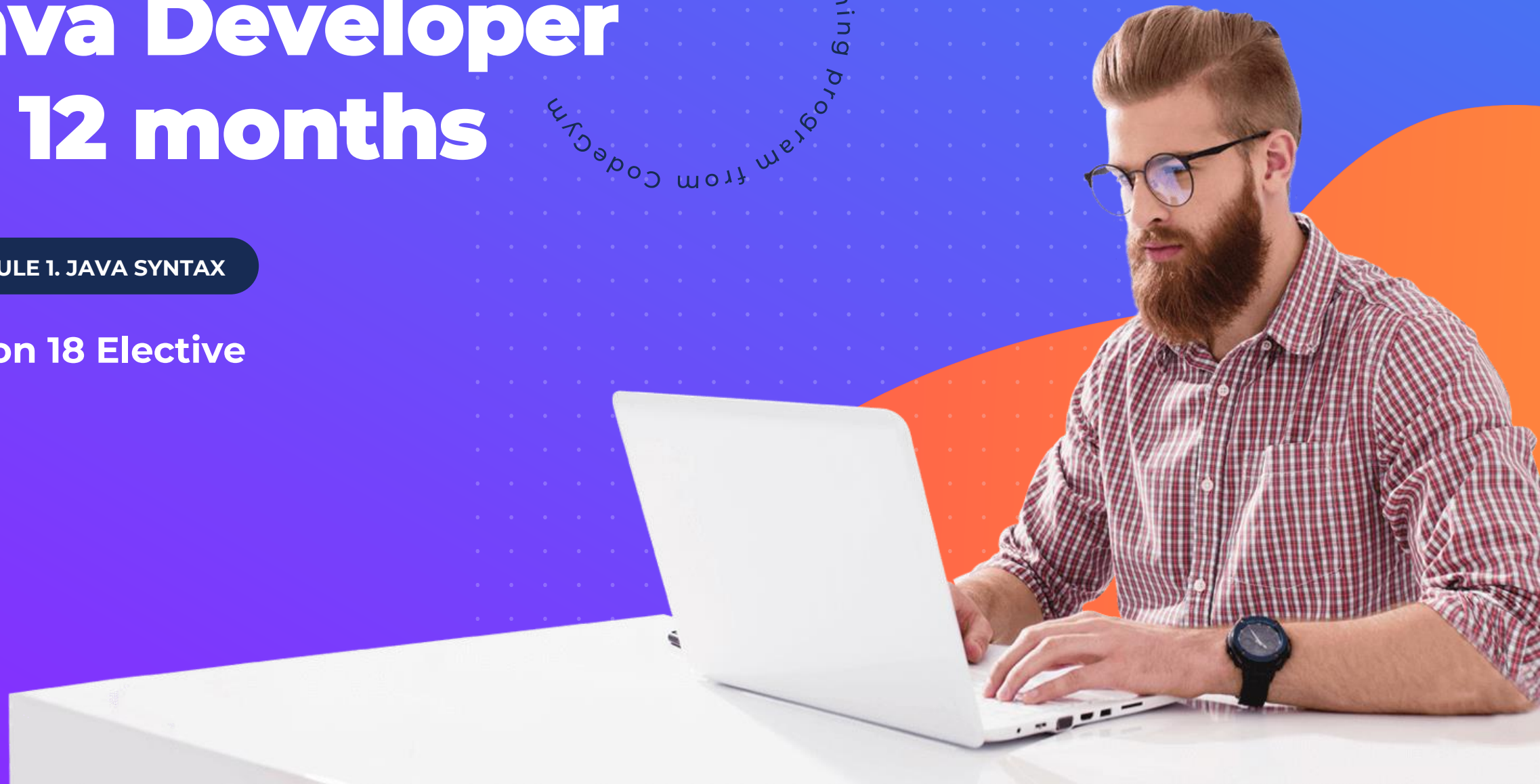
CODEGYM

Mentor-supported training program from CodeGym

Java Developer in 12 months

MODULE 1. JAVA SYNTAX

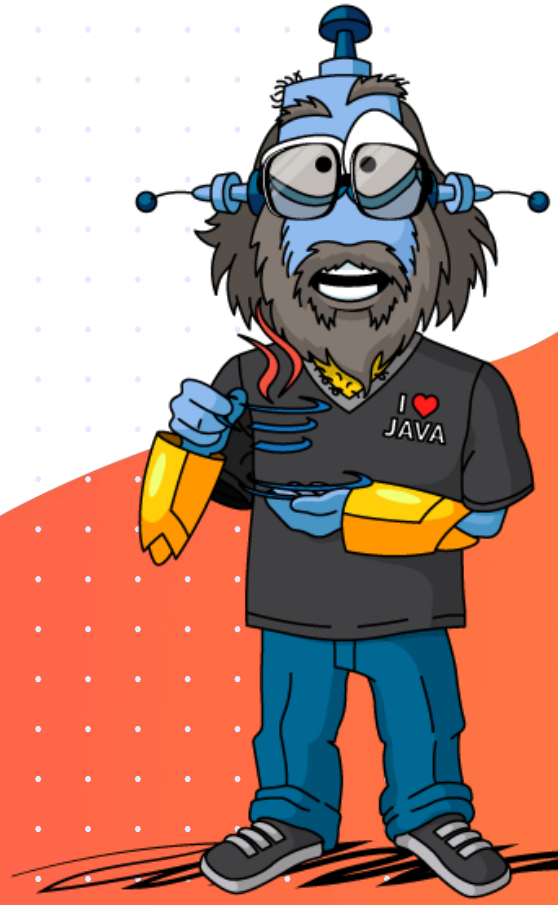
Lesson 18 Elective



CODEGYM

Lesson plan

- LinkedList collection
- Queue
- SortedMap



CODEGYM

Introducing the LinkedList collection

Outwardly, a LinkedList appears to be the same as an ArrayList.

It is implemented as a linked list: a set of individual elements, each storing a link (or reference) to the next and previous elements. To insert an element in the middle of the list, you only have to change the references of its immediate neighbors.

But to get the element whose index is 30, you have to move through all the objects from 0 to 30. In other words, the set and get operations have a very slow implementation here.

Operation	Method	ArrayList	LinkedList
Add an element	add(value)	Fast	Very fast
Insert an element	add(index, value)	Slow	Very fast
Get an element	get(index)	Very fast	Slow
Set an element	set(index, value)	Very fast	Slow
Remove an element	remove(index)	Slow	Very fast

CODEGYM

Nobody uses LinkedList

The author of the LinkedList class recently tweeted:

"Does anyone actually use LinkedList?
I wrote it, and I never use it."

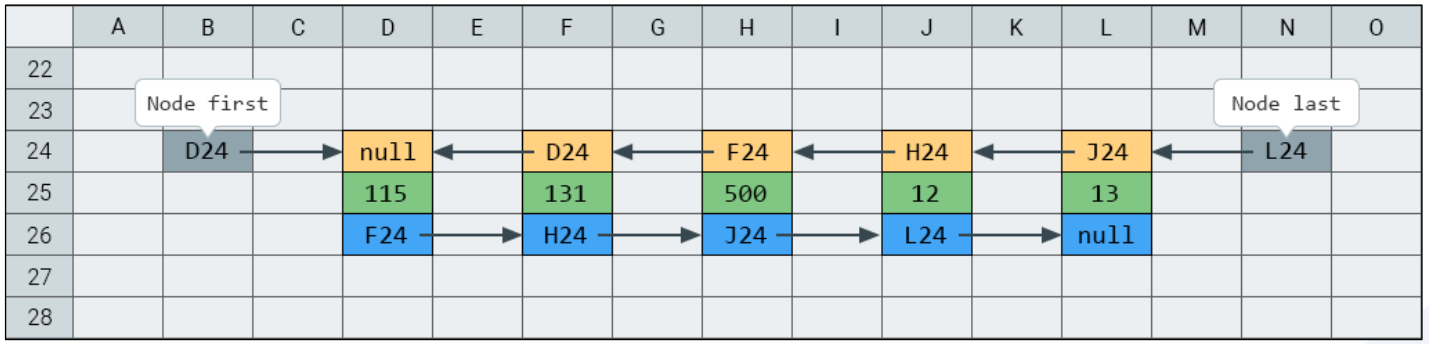
The ArrayList class began to be able to insert elements into the middle of the list very quickly. When inserting an element in the middle of the list, you have to shift all the elements after the insertion point by 1 towards the end of the list. This used to take time.

But now everything has changed. All the elements of an array are near one another in the same block of memory, so the operation to shift the elements is performed by a very fast low-level command: System.arraycopy().

CODEGYM

How LinkedList is structured

LinkedList uses a data structure called a doubly inked list.

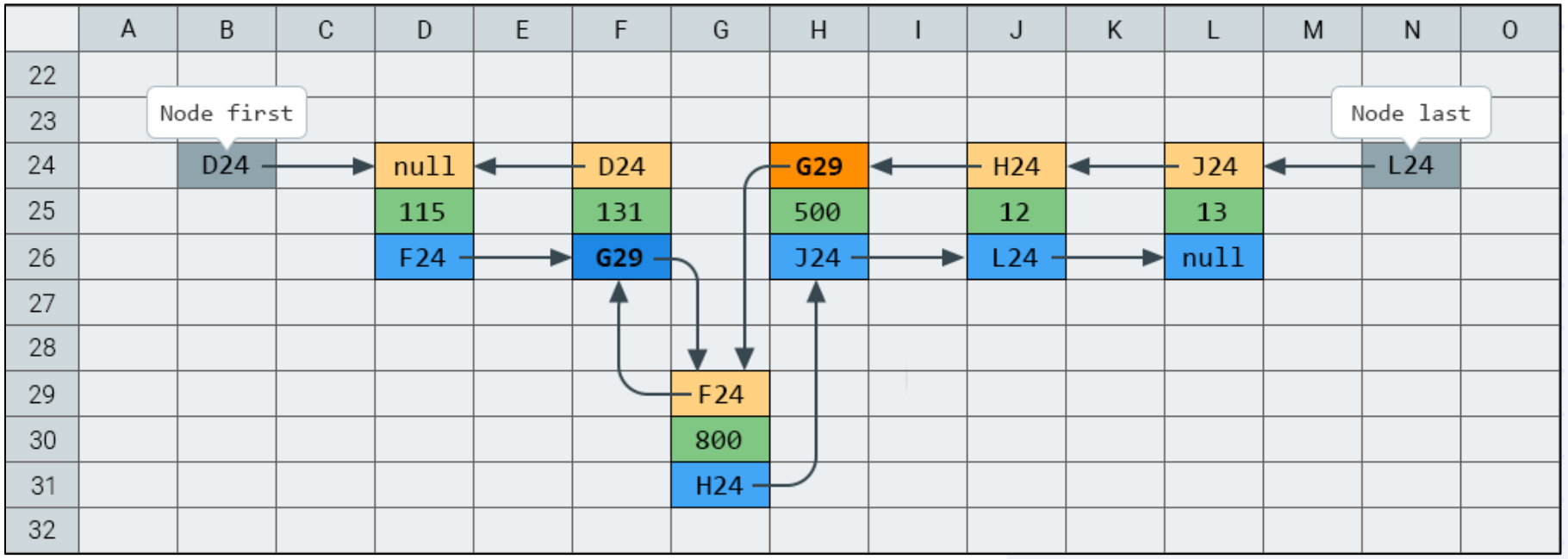


In the middle, you have a chain of Node objects (objects, not variables). Each of them consists of three fields:

- prev — stores a **reference** (link) to the previous Node object (cells with a yellow background).
- value — stores the **value** of this element of the list (cells with a green background).
- next — stores a **reference** (link) to the next Node object (cells with a blue background)

CODEGYM

Insert an element to a linked list

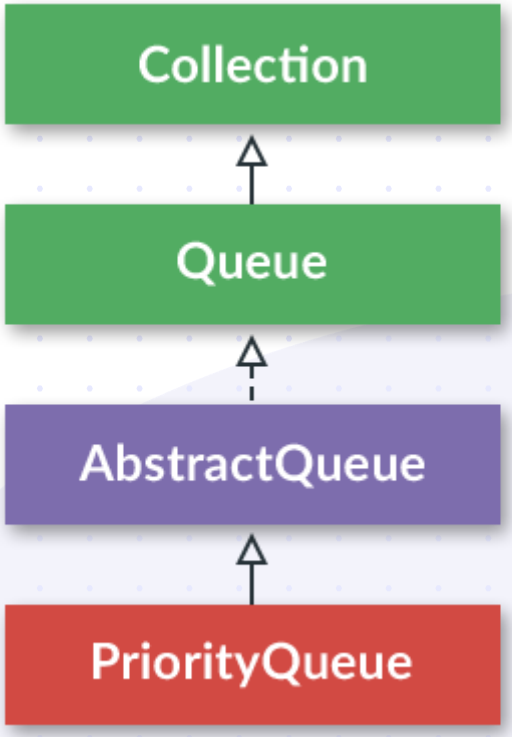


CODEGYM

Queue

Queue is a queue. Elements are added to the end of the queue, but are taken from the beginning.

PriorityQueue is essentially the only standard implementation of the **Queue** interface, if we don't count **LinkedList**, which is technically also a queue.



CODEGYM

SortedMap

The SortedMap interface extends Map and creates a mapping where all elements are sorted in ascending order of their keys, or according to the provided Comparator.

The Comparator is usually provided when a SortedMap object is created.

SortedMap adds several methods:

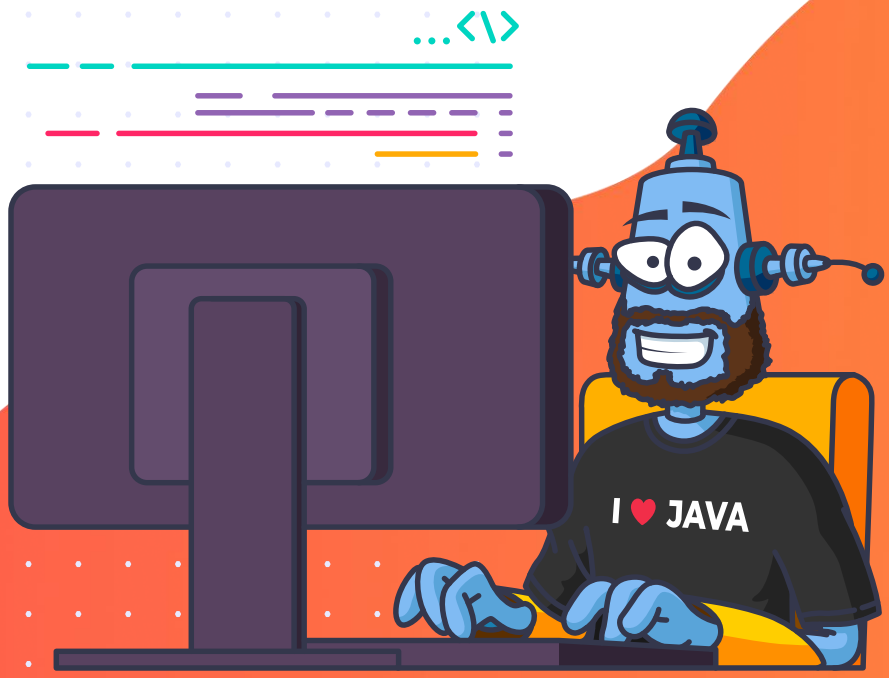
Method	Description
TKey firstKey()	Returns the key of the first element in the map
TKey lastKey()	Returns the key of the last element in the map
SortedMap<TKey, TValue> headMap(TKey end)	Returns a ***SortedMap*** that contains all the elements of the original ***SortedMap***, from the beginning up to (but not including) the element with the key end
SortedMap<TKey, TValue> tailMap(K start)	Returns a ***SortedMap*** that contains all elements of the original ***SortedMap***, starting at the element with the key start (inclusive)
SortedMap<TKey, TValue> subMap(TKey start, TKey end)	Returns a ***SortedMap*** that contains all the elements of the original ***SortedMap***, from the element with key start up to (but not including) the element with key end

CODEGYM

Homework

MODULE 1. JAVA SYNTAX

Complete Level 19



CODEGYM

Answers to questions

MODULE 1. JAVA SYNTAX