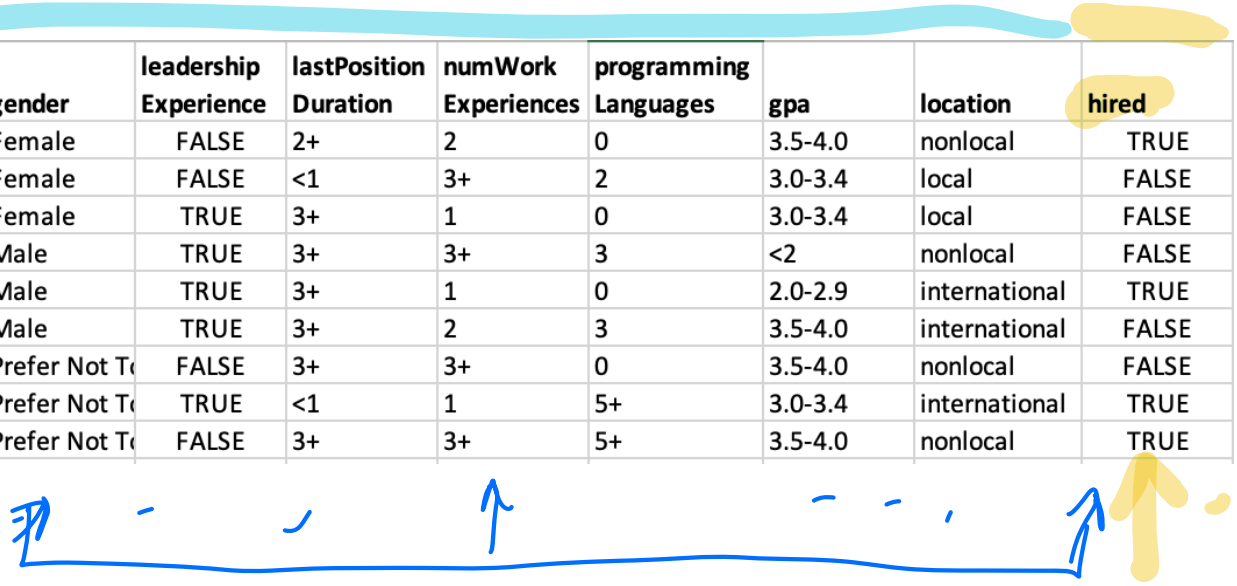
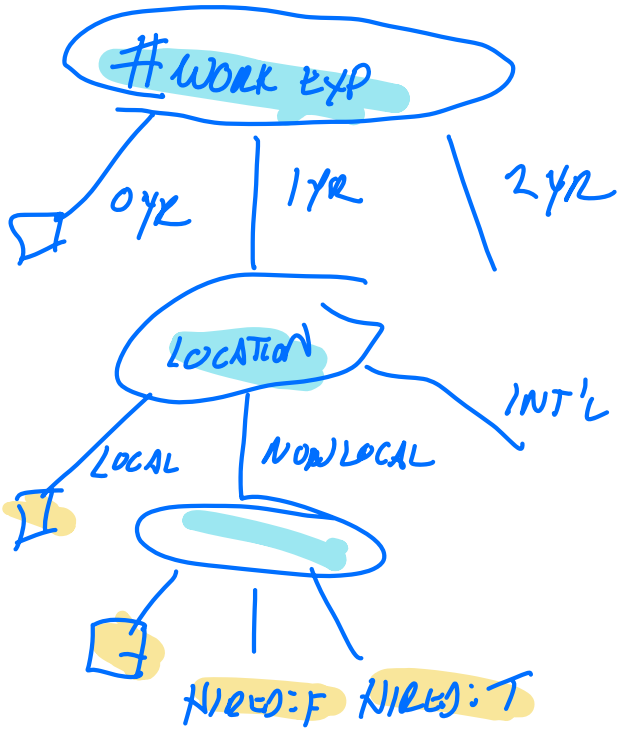
Project Preview : THINKING ABOUT DECISION TREES

gender	leadership Experience	lastPosition Duration	numWork Experiences	programming Languages	gpa	location	hired
Female	FALSE	2+	2	0	3.5-4.0	nonlocal	TRUE
Female	FALSE	<1	3+	2	3.0-3.4	local	FALSE
Female	TRUE	3+	1	0	3.0-3.4	local	FALSE
Male	TRUE	3+	3+	3	<2	nonlocal	FALSE
Male	TRUE	3+	1	0	2.0-2.9	international	TRUE
Male	TRUE	3+	2	3	3.5-4.0	international	FALSE
Prefer Not T	FALSE	3+	3+	0	3.5-4.0	nonlocal	FALSE
Prefer Not T	TRUE	<1	1	5+	3.0-3.4	international	TRUE
Prefer Not T	FALSE	3+	3+	5+	3.5-4.0	nonlocal	TRUE





=> Idea: given a set of of data in rows, how to build a tree that can predict decisions for new rows of data?

Lecture 11: Arrays and ArrayLists

(from last time) Consider the following layouts for the list [8, 3, 6, 4] – what program might generate this heap layout?

Activity: Now imagine the list had the following layout in memory (all the items consecutive and in order). What sequence of memory objects would get visited to compute `L.get(2)`?

	@1012	**ConsecList**
0	@1013	8
1	@1014	3
2	@1015	6
	@1016	4
	@1017	
	@1018	

BASED ON PICTURE, ELEMENT 2 IS AT

@1015 = @1012 + 2 + 1

@1015: ELEMENT 2 (+1) — @1012: START OF LIST (+1) — 2 (+1)

CONSTANT TIME. ⇒ O(1)

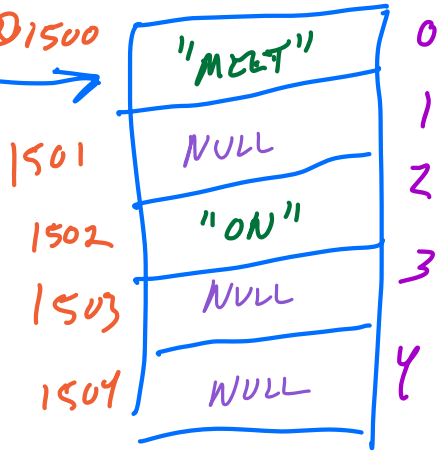
⇒ THIS IS CALLED AN ARRAY,

FORMS THE BASIS FOR JAVA'S ARRAYLIST (AMONG OTHER THINGS)

AN EXAMPLE:

WORDS[0] = "MEET"

WORDS[2] = "ON"



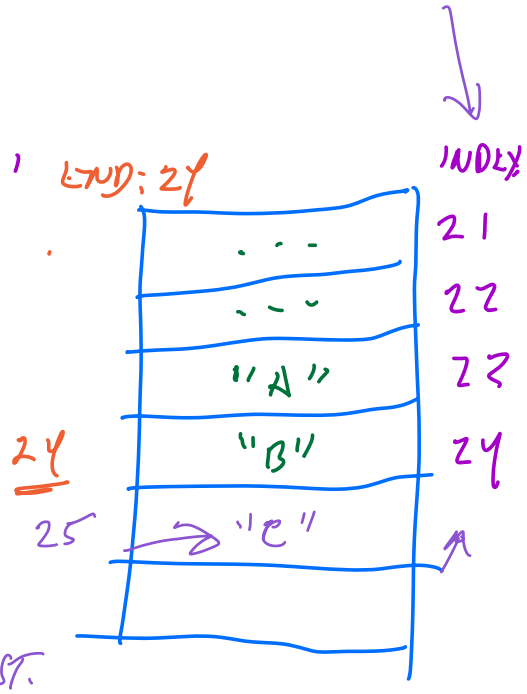
THIS IS CALLED AN INDEX INTO THE ARRAY

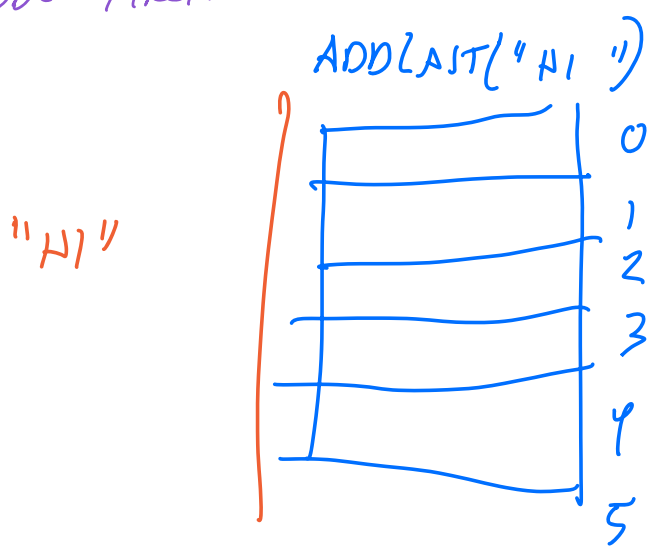
⇒ USED TO GET/SET A SPECIFIC SLOT, RELIES ON ADDRESSES

With an array, Java makes all the slots for us ahead of time,
We decide how to use them
Don't need to make nodes to hold objects every time we add something

NOW WOULD WE MAKE ADDFIRST/ADDLAST?

=> CONSIDER WHAT WOULD HAPPEN TO ADD TO THE MIDDLE FIRST.





Adding to a full ArrList

ArrList AL = new ArrList(3)

When we created ArrList, have fixed number of slots

@1012	**ArrList** theArray: @1013 end: 0 eltcount: 2
@1013	"hello"
@1014	"there"
@1015	"brown"
@1016	

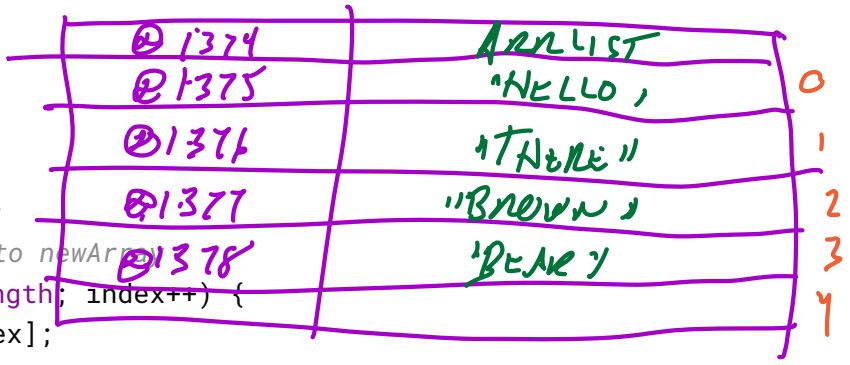
Assume this ArrList is named AL.

Now run `AL.addLast("bear")`

AL.ADDLAST("HELLO")
AL.ADDLAST("THERE")
AL.ADDLAST("BROWN")
AL.ADDLAST("BEAR")

If we want more space, we need to "resize" by getting a new array of larger size, copy everything over, then add new item

WHAT HAPPENS??




```
private void resize(int newSize) {
    // make the new array
    String[] newArray = new String[newSize];
    // copy items from the current theArray to newArray
    for (int index = 0; index < theArray.length; index++) {
        newArray[index] = this.theArray[index];
    }
    // change this.theArray to refer to the new, larger array
    this.theArray = newArray;
}

public void addLast(String newItem) {
    if (this.isFull()) {
        // add capacity to the array
        this.resize(this.theArray.length + 1);
        // now that the array has room, add the item
        this.addLast(newItem);
    } else {
        if (!(this.isEmpty())) {
            this.end = this.end + 1;
        }
        this.eltcount = this.eltcount + 1;
        this.theArray[end] = newItem;
    }
}
```

IF FULL RESIZE

ADDLAST FROM BEFORE

(You don't need to understand all the code here, we just want you to see the shape of it.)