Arrays

In this lecture, we will cover a fundamental structure in programming **Arrays**.

Your future in CS

I used to include this on my slides, but since these slides have changed - going to just leave it up here for every notebook. I get a lot of questions about more programming courses, the concentrations, and minors in computer science. Here is a brief reminder.

CS 165 – Next Course In Sequence, also consider CS 220 (math and stats especially)

- CO Jobs Report 2021 – 77% of *all* new jobs in Colorado require programming
- 60% of all STEM jobs requires *advanced* (200-300 level)
- 31% of all Bachelor of Arts degree titled jobs also required coding skills
- 2016 Report found on average jobs that require coding skills paid $22,000 more

- Concentrations in CS:
    - Computer science has a number of concentrations.
        - General concentration is the most flexible, and even allows students to double major or minor pretty easily.
        - Software Engineering
        - Computing Systems
        - Human Centered Computing
        - Networks and Security
        - Artificial Intelligence
        - Computer Science Education.
    - Minors:
        - Minor in Computer Science - choose your own adventure minor
        - Minor in Machine Learning - popular with stats/math, and engineering
        - Minor in Bioinformatics - Biology + Computer Science

Recalling the Past

- For every value you want to store
- You need a variable
    - What if you want to store 100 values? 10,000 values?
- Introducing Arrays
    - Reserving memory for storing values, in order from the 0 index
- Sound Familiar - Recall String
    - The String object contains

- chars in order!
- It is a character array!

Index	Character
0	k
1	i
2	n
3	n
4	i
5	k
6	i
7	n
8	n
9	i
10	k

In [2]:
```
char[] palindrome = {'k', 'i', 'n', 'n', 'i', 'k'};
```

What is happening?

Stack

variable	value
palindrome	@arrayC

@arrayC

k	i	n	n	i	k
0	1	2	3	4	5

- Arrays are stored on the "heap" (same location new objects are stored)
- Current memory stores a reference to that array
- To access individual elements of the array, we use brackets + index

In [5]:
```
char[] palindrome =  new char[6];  // declare the size of the array (more common)
palindrome[0] = 'k';
palindrome[1] = 'i';
palindrome[2] = 'n';
palindrome[3] = 'n';
palindrome[4] = 'i';
palindrome[5] = 'k';

System.out.println(palindrome[0]);
```

```
System.out.println(palindrome[3]);

System.out.println(Arrays.toString(palindrome)); // allows me to print the contents of
// Arrays are MUTABLE!
palindrome[0] = 'C';

System.out.println(palindrome[0]);
System.out.println(Arrays.toString(palindrome)); // allows me to print the contents of
```

```
k
n
[k, i, n, n, i, k]
C
[C, i, n, n, i, k]
```

Arrays are not Objects (sort of)

- They do not have methods!
- They are simply ways to
  - reserve N number of locations of a certain TYPE
    - Yes, any type!
  - you can then access the location by the index
- Everything is stored **in order**
- Locations are always reserved, even if nothing is stored!
  - numeric values default to 0
  - Objects default to `null`
- .length gives the total amount of space allocated!
  - notice no parens. This is not a method, but a command

In [7]:
```
String[] cast = new String[10];

System.out.println(cast); // just a memory address
System.out.println(cast.length);

cast[0] = "brad";
cast[1] = "janet";
cast[2] = "magenta";
cast[3] = "columbia";
cast[4] = "riff-raff";
cast[5] = "eddie";
cast[6] = "scott";
cast[7] = "frankie";
cast[5] = "rocky";

System.out.println(Arrays.toString(cast));
```

```
[Ljava.lang.String;@35542fa6
10
[brad, janet, magenta, columbia, riff-raff, rocky, scott, frankie, null, null]
```

Notice the two `null` items at the end, as we only filled spots 0-7!

Student Practice

- Create a new file for practice
- In that file, create a String array of length 4
- Agree that at the table, your seats are labeled 0-3
  - Add a person based on the seat you are sitting in
- Print out the final array using the Arrays.toString()
- Print out each person followed by their seat number

> Discussion
> What are seats with no one in there? It is possible to tell how many seats are at the table? Could you accomplish the same thing with an ArrayList? Yes, no, maybe a bit more difficult.

In [12]:
```
String[] seats = new String[4];

seats[0] = "Amy";
seats[2] = "Rory";

System.out.println(Arrays.toString(seats));

for(int i = 0; i < seats.length; i++) {
    System.out.printf("%s is in seat %d%n", seats[i] != null ? seats[i] : "No one", i+
}
```

```
[Amy, null, Rory, null]
Amy is in seat 1
No one is in seat 2
Rory is in seat 3
No one is in seat 4
```

Arrays vs. ArrayList

- ArrayLists are lists that use Array as the underlining structure
- Arrays are just how you declare a group of objects in order
- ArrayList is an individual object someone wrote

feature	array	ArrayList
can contain primitives	X	only with wrapper classes and boxing/unboxing
fixed size	X	
variable size		X
how to access elements	direct access	through methods (.get, .set, .add)
speed	faster	slower due to method overhead

- When to use arrays over ArrayList?
  - When your size is **fixed**, arrays are much faster to use!
  - When you need to keep order on sparsely populated datasets (that are often fixed sizes)
    - [value, null, null, null, value, null, value]

- They are used about equally, just depends on what you are doing.

In [53]:
```
import java.time.Instant;

long[] values = new long[1000000];

Instant start = java.time.Instant.now();
for(int i = 0; i < values.length; i++) {
    values[i] = i * 10l;
}
Instant end = java.time.Instant.now();
System.out.println("Array Loop Done: " + java.time.Duration.between(start, end).toMill

List<Long> valList = new ArrayList<>(); // remember polymorphism use List
start = java.time.Instant.now();
for(int i = 0; i < 1000000; i++) {
    valList.add(i*10l);
}
end = java.time.Instant.now();
System.out.println("List Loop Done: " + java.time.Duration.between(start, end).toMilli

// note this is not really the best way to determine the time between algorithms but i
// us an idea
```

```
Array Loop Done: 50
List Loop Done: 124
```

Student Challenge

- Create an array that can store 10 numbers
  - You can use the short hand of {} when you initialize the array
  - Print out the result
- Copy those elements into a new array of length 20
  - Use a loop
  - Print out the result

> Discussion
>
> How would this activity relate to how you imagine an ArrayList is implemented behind the scenes?

In [54]:
```
int[] vals = {12, 15, 16, 2982, 22, 42, 181, 13, 9, 10};
int[] vals2 = new int[20];
System.out.println(Arrays.toString(vals));

for(int i = 0; i < vals.length; i++) {
    vals2[i] = vals[i];
}

System.out.println(Arrays.toString(vals2));
```

```
[12, 15, 16, 2982, 22, 42, 181, 13, 9, 10]
[12, 15, 16, 2982, 22, 42, 181, 13, 9, 10, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

Overview

- Arrays are mutable
- Arrays declare N number of primitives or objects in order
  - They are stored on the heap
- .length is a command to get how many variables are declared
- brackets [] give direct access to each value - store or get
- Use Arrays.toString(array) to print the array contents
  - Else you will end up just printing the memory address of the location reserved
- Use Arrays when your size is fixed!
- Use ArrayLists when your size is unknown
  - at a cost to speed!