Day 2

Tuesday, September 12, 2017 9:07 AM

Array - An *array* is a container object that holds a fixed number of values of a single type. The length of an array is established when the array is created. After creation, its length is fixed.

Each item in an array is called an *element*, and each element is accessed by its numerical *index*. The first element has an index of 0.

Package = directory: the file structure of your project. This is where you put your java classes and interfaces.

Import - allows you to specify classes you want to use from other packages.

Access modifiers

- Private - only accessible within the class
- No access modifier (default) - class and package
- Protected - class, package, and children
- Public - everywhere

String API -

- The String class represents character strings.
- Immutable - cannot be changed, not to be confused with reassigning the variable.
- All string literals in Java programs, such as "abc" belong to this class.
- There are several methods for working with strings, such as substring

String builder/buffer - Both are used for creating mutable strings. Which is more efficient that just using String if you are doing string manipulations. Buffer is synchronized, builder is not. Which means that builder is faster.

Exception handling -

- Try-catch-finally
    - When you have a try block you must have either a catch block or a finally block but both are not required.
    - You can have multiple catch blocks but order matters
    - Try with resources, allows the try block to close resources without us having to close them in our finally block
- Checked /compile time exceptions - exceptions that are not due to errors in our logic, must surround with try catch block.
- Unchecked/runtime exceptions - exceptions that can be resolved purely through logic, no try catch is required

Garbage collection - Java's built in memory management system. Automatically removes objects from memory that are no longer being referenced. Calls the .finalize() method on the object before removing it.

Wrapper Classes

- Object representation of each primitive.
- Auto boxing - Java's way of automatically converting a primitive into it's wrapper class without us having to do anything.