Day 3

Wednesday, November 1, 2017 8:58 AM

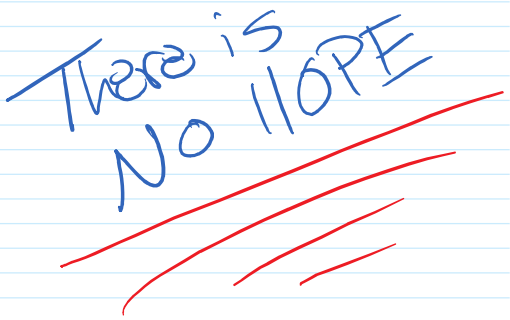
Collection Interface : Group of data
Collections Class : Unity Class for Collection
Array Class : Utility class

Collection Interface
- List - ordered, dynamically sized array
- Set - no duplicates, not strictly ordered, no access by index
- Queue - Ordered, can only access push to back, can only read/pop from the front, for the most part FIFO.

List cont
- Array List : Very fast access, slow if we need to add a lot.
- Linked List : Efficient for adding to the end. Efficient for inserting new data basically anywhere.

Set cont
- Hash Set - Faster due to hashes
- Tree Set - creates a binary tree (guarantees order and auto balances)

Iterator Class : hasNext() returns true if there is another upcoming token available .
next() returns the next upcoming available token.



When writing code, Blake's top priorities are maintainability and scalability.

Generics<> - A useful way to help add stability to our code and to keep things clean. It helps by making more of the bugs which will eventually find a way into our code detectable at compile time.

Map: Basically a dictionary. Key & Value. Our value could be an object and the key refers to that object. If you have a map and know the key, you can ask the map to return the object or data correlated to the key.

Hash Table and Hash Map: Hash map is not synchronized or thread safe, allows null key. hash table is synchronized and thread safe, no null keys.

CTRL + SHIFT + F = Eclipse auto format
CTRL + SHIFT + O = Eclipse auto organize our imports
ALT + SHIFT + R = Eclipse refactor a name

I/O

Inputs:
- File Input Stream - reads one byte at a time
- File Reader - Reads 2 bytes at a time (Java equivalent of a character)
- Buffered Reader - Reads entire lines at a time
- Scanner - Has a smaller buffer than buffered reader and buffered writer and has additional functionality (can use Regex?)
- Object Input Stream - reads byte code (serialized objects?) and creates actual objects. **NEED FOR BANK APP**

- Serialization: The conversion between object and byte code.
- Scanner: is used to get user input for banking console application
- Try with Resources: let's you put a closeable within parenthesis of the try-block. Whether it succeeds or fails, it closes out the try-catch for us.

- Right click potential errors in Eclipse and add the recommended try-catch's. FREE CODE IS FOR THE WIN!!!
- If you want your object to be serialized, it MUST implement serializable!!!

- Marker interface: has no methods
- Functional interface: has one abstract method
- Lambda: anonymous function

- Stream: A flow of data. (collection of data is stored - array). A stream on the other hand can potentially be infinite because data goes in and data flows out.

- Sequential streams: operate on one thread.
- Parallel streams: operate on multiple threads.

- Filter() - We receive large amount of data. 1st step is to filter out anything we don't want out of that large amount of data
- Map() - Change the type of data to meet my model.
- Reduce() - compute a single value out of all the data

- Optional type : can contain a null value.