**Java 4: Research**

Strings

The first thing that needs to happen here is to define a method. A method is a block of code that only acts once it is called upon1. For instance, in the Labs for week 4, an example was to count the number of times letters were used at the beginning of a word. If the letter was used more than once the count would increase by one. Instead of writing a block of code each time a new word was added all that would need to be done is create a method. Each time the method would be called that block of code in the method would run. This makes the program, in a whole run faster and smoother.

Now that a method has been defined, the method signature needs to be defined. A method signature is the name of the method and its parameters. A method signature is designed by the programmer, it contains a name and the parameters of the method2. An example of a method signature is

getArea(int x, int y)

The name of the method is getArea, with two parameters int x and int y. Altogether is the method signature.

The first method is **length3**. The method signature is public int length(). It returns an integer of the length of a string. A good use of this method would be to count characters if you want to specify a limit. This method could be used in a form to limit the amount of characters in a description.

The second method is **concat**3. The method signature is public String concat(String str). This adds a character or a string to the end of another character or string. This would be useful to concatenate multiple fields together to create a full name in an address block for a letter. The fields could be first name, *space*, last name.

The third method is **contains**3. The method signature is public boolean contains(CharSequence s). This method returns true of false for a character sequence. This method could be used for creating passwords or entering passwords. Checking on the different sequences specified (upper case, lower case, number, special character).

The fourth method is **join**3. The method signature is public String join(CharSequence delimiter, CharSequence elements). This method could be used to add commas between words in a collection of names.

The fifth method is **isEmpty**3. The method signature is public boolean isEmpty(). This method returns a true value if there is no string, or false if there is a string. This could be useful to check on required fields. If the field is empty and the program shows a true, an error message is shown. If the field is not empty, then the program does not show an error.

Collections

The differences between lists, sets, and maps are how they store data. Lists are ordered and can have duplicate data4. Sets are unordered and does not allow duplicate data to be stored4. Maps use a key and value to store data, there are two parameters needed for map collections. Maps do not allow duplicate keys but do allow duplicate values5.

Two ways a list collection can be implemented are ArrayList and LinkedList. ArrayList creates a dynamic array, that can be altered5. LinkedList is when data in the list is stored in different locations in memory5. LinkedList is easier to work with and alter. Making it a preferred implementation of the list collection. Use LinkedList to do a lot of insertion and deletion of information. Use ArrayList for random access of collections.

Two was a set collection can be implemented are HashSet and LinkedHashSet. The HashSet implementation creates a hash table. Since set collections do not allow duplicates, the hash table will check all entries to make sure there is not a duplicate. LinkedHashSet keeps the data linked together. The LinkedHashSet maintains an order whereas the HashSet does not maintain an order5.

Two ways a map collection is implemented are HashMap and LinkedHashMap. The HashMap maintains the order of storing data by a key and value. Where the key is the address and the value is the information being stored. The LinkedHashMap is similar to the HashMap. However, the LinkedHashMap maintains and order for searching, whereas the HashMap does not maintain order. If quick searching is desired the LinkedHashMap is more desirable5.

Each implementation has its uses. It depends on what is best for the program. The ArrayList, HashSet, and HashMap are all good for manipulations(adding, deleting). Where LinkedList, LinkedHashSet, LinkedHashMap are able to do the adding and deleting but they also maintain an order for searching. To say one is better than the other all depends on the expectations.

To instantiate list collection:

List<String> list = new ArrayList<String>();

To instantiate HashSet for StringBuilder:

Set<StringBuilder> sb = new HashSet<StringBuilder>();

To instantiate HashMap of String, String:

Map<String, String> dictionary = new HashMap<String, String>();

1 An Introduction to Methods in Java with Examples, <https://www.simplilearn.com/tutorials/java-tutorial/methods-in-java#:~:text=of%20Java%20programming.-,What%20are%20Methods%20in%20Java%3F,only%20be%20executed%20when%20called>.

2 Method Signature in Java, <https://www.scaler.com/topics/method-signature-in-java/>

3 String Java Documentation, https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/String.html#method-summary

4 Collections in Java - Javapoint, <https://www.javatpoint.com/collections-in-java>

5 Collections in Java – GeeksforGeeks, <https://www.geeksforgeeks.org/collections-in-java-2/>