ArrayList<>

The following are some of the methods used in creating a list using ArrayList<>:

1. .add() – adds an element to the list
2. .remove() – removes an element from the list
3. .removeAll() – removes based on a predicate, as long as it is met, all elements are removed.
4. .removeIf() – remove from list if predicate is met.
5. .removeRange() – remove elements for a given index.
6. .addall() - adds all elements of a given collection to the list
7. .clear() – remove all elements from a list
8. .get() – fetch a list or a particular element within the list.
9. .isEmpty() – check to see if the list is empty, returns Boolean.
10. .lastIndex() – return the last index in the list.
11. Object[].toArray() – returns all elements in the list to an array in the correct order.
12. Object.clone() – provides a shallow clone of the items in the ArrayList.
13. .contains() – returns Boolean if the list contains the searchParam.
14. .replaceAll() – used to replace all elements in a list with another list.
15. .retainAll() – used to retain all elements of a given list.
16. .set() – used to replace a given element within a list.
17. .sort() – used to sort the list based on a comparator.
18. List<E> subList – used to fetch a part of a list using given range (index).
19. .size() – gives the number of elements within the list.
20. .trimToSize() – trim capacity of the ArrayList to its current size.

Within the current milestone I have used the .add(), .get() and .remove() methods for the list. I have skeleton code to implement a sorting method that I could see being used in conjunction with a subList or finding the index of a given element.

Java T Point (n.d.) Java ArrayList class. Retrieved from <https://www.javatpoint.com/java-arraylist>

The benefit for using Java generics to determine errors at compile time. This is helpful for picking out issues in your code early before you actually run the project and find out errors. If you define something as a String and then pass it an Integer, you will get a compile error as the IDE knows that you are trying to send one time to another.

Another benefit is the autoboxing of primitive types and the lack of needing to cast the output of a generic back to its native form. If you create an ArrayList<String> list = new ArrayList() and add a few values to that, you can simply use System.out.println(list.get(0)). It knows it is a string and will print it accordingly. You don’t have to cast (String) to the element of the list.

I’ve used genereics in the creation of my PersonContac, BusinessContact and AddressBook array lists. I also use my FileAccessService to access the mock data in the “addressbook” list currently.