# Week 15: Final Review

# Final Review

## Topics

1. **Arraylist**
2. **String Methods**
3. **File I/O**

Why do we use an ArrayList?

## ArrayList

- An ArrayList is a resizable array.
- An ArrayList is dynamic where an array is static.
- Elements can be added and removed from an ArrayList and the size of the ArrayList will change accordingly.
- An ArrayList can only store objects, not primitive data types.
- An ArrayList is a class in Java that is part of the `java.util` package.

Declaring an ArrayList

```
ArrayList<Type> list = new ArrayList<Type>();
```

or

```
ArrayList<Type> list = new ArrayList<>();
```

ArrayLists only store objects. You must use a wrapper class for primitive data types

```
ArrayList<Integer> list = new ArrayList<Integer>();
ArrayList<Double> list = new ArrayList<Double>();
ArrayList<Character> list = new ArrayList<Character>();
ArrayList<Boolean> list = new ArrayList<Boolean>();
```

String is a class, so you can use it without a wrapper class

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

## ArrayList Methods

ArrayLists have many methods that can be used to manipulate the list.

- `add()`
- `get()`
- `set()`
- `remove()`
- `size()`

Declare an ArrayList of Strings, add elements, and print the elements

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

list.add("Hello");
list.add("World");

for (int i = 0; i < list.size(); i++) {
    System.out.println(list.get(i));
}
```

`add()` adds an element to the end of the list

`get()` returns the element at the specified index

You can use a for-each loop to iterate through the list

# String Methods

We use `String` in Java store an array of characters. `String` is a class in Java and has many methods that can be used to manipulate the string.

See the slides from week 13 for more information on `String` methods.

Let's print each character of a string on a new line

```
String str = "Hello World";

for (int i = 0; i < str.length(); i++) {
    System.out.println(str.charAt(i));
}
```

Let's compare two strings

```
String str1 = "Hello";

String str2 = "Hello";
```

We can compare each character of the string.

```
boolean equal = true;

if (str1.length() == str2.length()) {
    for (int i = 0; i < str1.length(); i++) {
        if (str1.charAt(i) != str2.charAt(i)) {
            equal = false;
            break;
        }
    }
} else {
    equal = false;
}

---

Or we can compare both strings using the `equals()` method.

```java

if (str1.equals(str2)) {
    System.out.println("The strings are equal");
} else {
    System.out.println("The strings are not equal");
}
```
```

The first method gives us character by character comparison.

The second method gives us a comparison of the entire string. Returning a boolean value.

# File I/O

Let's write to a file using `PrintWriter`

```
import java.io.*;

public class WriteToFile {

    public static void main(String[] args) throws IOException {
            File file = new File("output.txt");
            PrintWriter writer = new PrintWriter(file);
            writer.println("Hello World");
            writer.close();

    }
}
```

Let's read from a `.txt` file using `Scanner` and the contents to an ArrayList.

```
import java.io.*;

public class ReadFromFile {

    public static void main(String[] args) throws IOException {
        File file = new File("output.txt");
        Scanner scanner = new Scanner(file);
        ArrayList<String> list = new ArrayList<String>();

        while (scanner.hasNext()) {
            list.add(scanner.nextLine());
        }

    }
}
```

Let's read a `.csv` file, split at the comma, and add the elements to an ArrayList.

```
import java.io.*;

public class ReadCSV {

    public static void main(String[] args) throws IOException {
        File file = new File("data.csv");
        Scanner scanner = new Scanner(file);
        ArrayList<String> list = new ArrayList<String>();

        while (scanner.hasNext()) {
            String line = scanner.nextLine();
            String[] data = line.split(",");
            for (int i = 0; i < data.length; i++) {
                list.add(data[i]);
            }
        }

    }
}
```

We can parse data to the appropriate data type using the wrapper classes for primitive data types.

We can use the `parseInt()` and `parseDouble()` methods to convert a string to an integer or double.

```
Integer.parseInt(data);
Double.parseDouble(data);
```

# Practice

- Create a `.csv` file with multiple three values lines containing a string, an integer, and a double.

```
Hello, 5, 3.5
```

- Then read the file into a Java program and save the data to three ArrayLists of the appropriate data type.
- Find the average of the integers and the maximum double value in the ArrayLists.
- Save the string followed by the average of the integers and maximum double to a new file.