Java Quick Reference repl
Due: Friday 4/12/24

The Java Quick Reference is a piece of paper you will have as a reference during the AP exam (They'll give you a copy of it). To help you familiarize yourself with it and what it means, I'm making you use it for a creative assignment.

The assignment: create a repl that uses every single one of the methods and commands listed in the Java Quick Reference (see paper copy, or next page) in order.

Please use them in groups, in the order listed in the document. Use a comment and a print call to clearly indicate which group is which. Use print calls to say which method you are using as you use it (see my examples below). There will be lines you need to add in order to use some of the calls (for example, you need to add import java.util.ArrayList; to the top before you can use the ArrayList calls near the bottom, or you may want to add a Scanner).

For example:

```
//String class
System.out.println("Starting the String group");
String myString = new String("Making a new String this way");
System.out.println("A String: "+myString);
System.out.println("Using .length(): "+myString.length());
System.out.println("Using substring with two numbers: "+myString.substring(9,12));
```

I just showed you a few lines above. You would need several more lines to finish with the String group.

Then move on to the Integer group, etc.

Everyone's repl needs to be different. Please do your own work.

There is no help video for this. If you are stumped, ask for help. Request my tutorial. None of this is new material. Look at your notes. Look at old repls. You can do this!

See the paper handout I gave you, or if you prefer, see the reference on the next page.

Java Quick Reference

Accessible methods from the Java library that may be included in the exam

Class Constructors and Methods	Explanation
String Class	
`String(String str)`	Constructs a new `String` object that represents the same sequence of characters as `str`
`int length()`	Returns the number of characters in a `String` object
`String substring(int from, int to)`	Returns the substring beginning at index `from` and ending at index `to - 1`
`String substring(int from)`	Returns `substring(from, length())`
`int indexOf(String str)`	Returns the index of the first occurrence of `str`; returns `-1` if not found
`boolean equals(String other)`	Returns `true` if `this` is equal to `other`; returns `false` otherwise
`int compareTo(String other)`	Returns a value `<0` if `this` is less than `other`; returns zero if `this` is equal to `other`; returns a value `>0` if `this` is greater than `other`
Integer Class	
`Integer(int value)`	Constructs a new `Integer` object that represents the specified `int` value
`Integer.MIN_VALUE`	The minimum value represented by an `int` or `Integer`
`Integer.MAX_VALUE`	The maximum value represented by an `int` or `Integer`
`int intValue()`	Returns the value of this `Integer` as an `int`
Double Class	
`Double(double value)`	Constructs a new `Double` object that represents the specified `double` value
`double doubleValue()`	Returns the value of this `Double` as a `double`
Math Class	
`static int abs(int x)`	Returns the absolute value of an `int` value
`static double abs(double x)`	Returns the absolute value of a `double` value
`static double pow(double base, double exponent)`	Returns the value of the first parameter raised to the power of the second parameter
`static double sqrt(double x)`	Returns the positive square root of a `double` value
`static double random()`	Returns a `double` value greater than or equal to `0.0` and less than `1.0`
ArrayList Class	
`int size()`	Returns the number of elements in the list
`boolean add(E obj)`	Appends `obj` to end of list; returns `true`
`void add(int index, E obj)`	Inserts `obj` at position `index` (`0 <= index <= size`), moving elements at position `index` and higher to the right (adds 1 to their indices) and adds 1 to size
`E get(int index)`	Returns the element at position `index` in the list
`E set(int index, E obj)`	Replaces the element at position `index` with `obj`; returns the element formerly at position `index`
`E remove(int index)`	Removes element from position `index`, moving elements at position `index + 1` and higher to the left (subtracts 1 from their indices) and subtracts 1 from size; returns the element formerly at position `index`
Object Class	
`boolean equals(Object other)`	
`String toString()`	