Classes

and Arrays

Nov 15

equality

and memory

- Strings and all instances of classes have two ways to compare to each other
  - ==
    - compares pointers!
  - .equals()
    - compares strings

```java
public class Equality {
    public static void main(String[] args) {
        String s = new String("this");
        String t = new String("that");
        System.out.println("s == t " +  (s == t));
        System.out.println("s.equals(t) " + s.equals(t));

        String ss = s;
        System.out.println("s==ss " + (s == ss));
        System.out.println("s.equals(ss) " + s.equals(ss));

        ss = new String("this");
        System.out.println("s==ss " + (s == ss));
        System.out.println("s.equals(ss) " + s.equals(ss));
    }
}
```

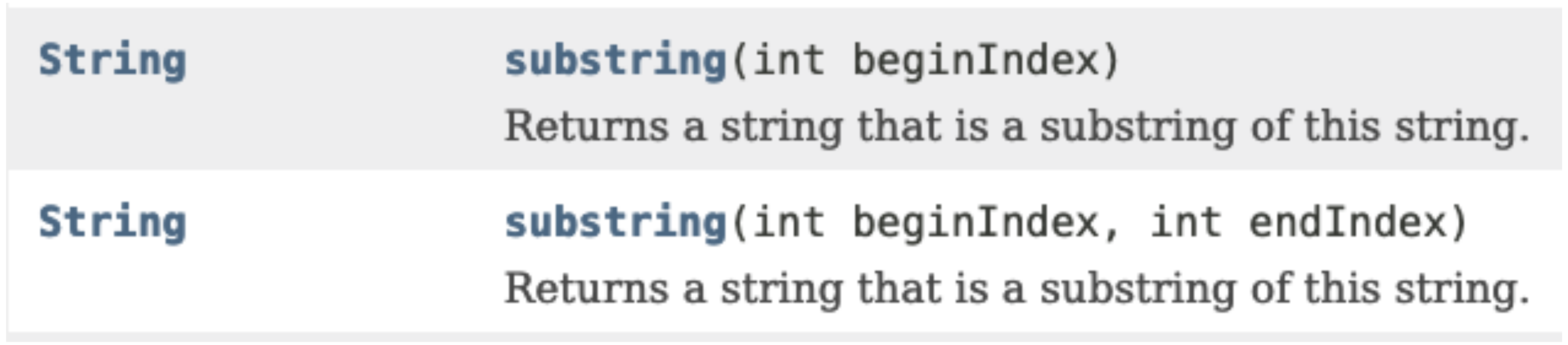
Aliases

- Alias: When 2 things point at the same thing

  -
    ```
    String ss = new String("this is");
    String tt = ss;
    ```
- Strings are immutable
  - The internal memory (state) is not allowed to change
  - so aliases are not very obvious
- Arrays are a lot like object instances

  -
    ```
    int[] ss = new int[3];  // [0,0,0]
    int[] tt = ss;          // tt points to same memory as ss
    tt[0]=42;
    ss[1]=99;               // [42,99,0] for both ss and tt
    ```

substring()



String	substring(int beginIndex) Returns a string that is a substring of this string.
String	substring(int beginIndex, int endIndex) Returns a string that is a substring of this string.

- There are lots more methods on String

```java
public class FunWithStrings2 {
    public static void main(String[] args) {
        String ss ="The quick brown fox jumps.";
        System.out.println(ss.substring(4, 9));
        System.out.println(ss.substring(ss.indexOf('f'), ss.indexOf('f')+3));
        System.out.println(ss.substring(ss.indexOf('j')));
    }
}
```

Improvable, worth it??

Activity

- What is the longest prefix shared by any two strings on the command line?
- Simpler: what is the longest prefix shared by the first string on the command line with any other string?
  - Does the first string have the same first letter as any other?
  - Does the first string have the same first 2 letters?

Look at the methods on String.
There is a handy one: startsWith

```
java Activity17 abcdef sdfg adfgh absdfg abcfff sdfghy
Just first: 3
ALL: 4
```

Class Instances and Arrays

```
public class CArray {
    public static void main(String[] args) {
        double dd;
        double dd0 = 0.0; // does this mean "I do not know yet?"
        System.out.println("dd" + dd);

        double[] dA = new double[3];
        for (int i = 0; i < dA.length; i++) {
            System.out.println("dA + " + i + " " + dA[i]);
        }

        String ss;
        String ssb = "";  // does this mean "I do not know yet
        System.out.println("ss <<" + ss + ">>");
        String[] ssA = new String[3];
        for (int i = 0; i < ssA.length; i++) {
            System.out.println("ssA " + i + " " + ssA[i]);
        }
    }
}
```

error: variable dd might not have been initialized

error: variable ss might not have been initialized

Java: definite assignment

```
dd0.0
dA  0 0.0
dA  1 0.0
dA  2 0.0
ss
ssA 0 null
ssA 1 null
ssA 2 null
```

Null

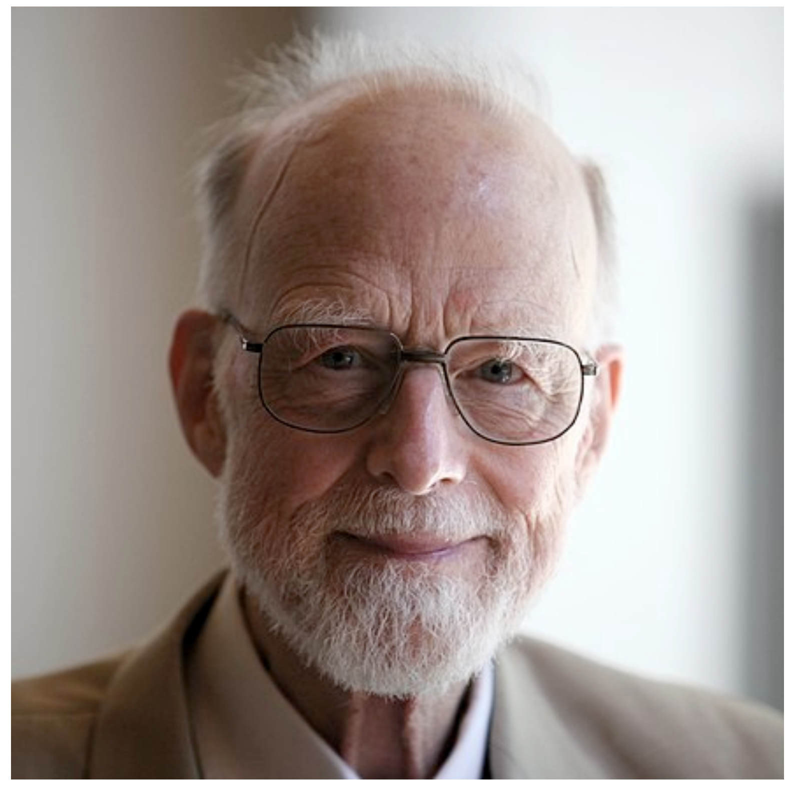
the default

- Null is
  - a default value for class instances
  - String aaa = null;
  - Any variable holding an instance of a class can be set to "null"
- Why??
  - temporary value for a variable before it's initialized
  - to indicate that the object does not exist (yet)
  - a method can return null to signal that there was no result from the operation
  - we can pass null to a method that takes an object as a parameter to indicate "no object"

Null

is evil??

- But null is literally Nothing
  - You cannot do anything with it
  - if you try you will get a "Null Pointer Exception"
  - One of the most common runtime errors.
  - Tony Hoare originated the idea (and named it null) in 1964
    - "my billion dollar mistake"
      - underestimate



Static and non Static

the demise of the blueprint analogy

- static methods belong to the class itself
  - you do not need an instance to run them!
    - for instance every method you have written for this class
    - Math.pow(2,3)
- Non static methods must be run on an instance of a class
  - FileReader fr = new FileReader("file.txt");
    while (fr.ready()) { ....

It only makes sense to ask if a file is ready to be read if the FileReader knows what file is being asked about

a non-static method