Classes
organizing code in Java

Why do programs start "public class XXX {}"?
Class

- A class is a definition of a concept
  - a blueprint of a house
    - In Java "public class BlahBlah"
    - pretty much everything in Java is a class
  - An "object" is an instance of a class
    - a physical house
    - in Java "new BlahBlah"
Library Class vs Concept classes

- Library
  - Someone made it for you
    - often the people who wrote the Java language
      - this week
  - Concept
    - you make it yourself to meet your needs
      - after Thanksgiving
Classes have

• memory -- sometimes known as "state"
  • for instance, the FileReader class remembers
    • what file you opened
    • where it is in the reading of that file
  • state may be hidden (private) or visible (public)

• methods
  • allow you to interact with the class and its state
    • . notation
    • FileReader.ready(), FileReader.read() ...
public class FunWithStrings {
    public static void main(String[] args) {
        int x;
        x = 5;
        int xx = 55;

        String s;
        s = new String("5");
        String ss = new String("55");
        String sss = "555";
    }
}
Methods on String

• is VSC type the name of the instance followed by a period ("."), and you get a long list of the method names
• top hit in google to query "java string class methods"
  • https://docs.oracle.com/javase/8/docs/api/java/lang/String.html
• oracle.com is the authoritative source for Java information
  • Oracle "owns" java
• in VSC hover over method and get the same documentation as from Oracle web site!
  • I still use web site a lot to find the method I want
  • Then use VSC for documentation so I use the method correctly
charAt method of String

```java
public class FunWithStrings2 {
    public static void main(String[] args) {
        String ss = new String("This is an example");
        char c = ss.charAt(8);
        System.out.println("The eighth char of " + ss + " is " + c + " and its ASCII value is " + (int)c);
    }
}
```

**charAt**

`java.lang.String.charAt(int index)`

Returns the `char` value at the specified index. An index ranges from 0 to `length() - 1`. The first `char` value of the sequence is at index 0, the next at index 1, and so on, as for array indexing.

If the `char` value specified by the index is a surrogate, the surrogate value is returned.

**Parameters:**
- `index` the index of the `char` value.

**Returns:**
- the `char` value at the specified index of this string. The first `char` value is at index 0.
public class FunWithStrings2 {
    public static void main(String[] args) {
        String ss = new String("This is an example");
        char c = ss.charAt(8);
        System.out.println("The eighth char of " + ss + " is '" + c + "' and its ASCII value is " + (int) c);

        ss = "The quick brown fox jumps.";
        System.out.println("The eighth char of " + ss + " is '" + c + "' and its ASCII value is " + (int) c);
    }
}
More String methods

- length()
  - note the () as opposed to array.length!
  - annoys me!
  - even worse other things use size()
  - the number of characters in the string

- indexOf(char)
  - the first location of char in the string
  - -1 if not there

```java
public class FunWithStrings2 {
    public static void main(String[] args) {
        String ss = new String("This is an example");
        char c = ss.charAt(8);
        ss = "The quick brown fox jumps.";

        System.out.println(ss.length());
        System.out.println(ss.indexOf('o'));
        System.out.println(ss.indexOf(c));
    }
}
```
Using indexOf

find all indices of a character

- indexOf is "Overloaded"
  - there is a one parameter and a two parameter version
  - We will use both!!

```java
public class AllIndices {
    public static void main(String[] args) {
        String preamble = "We the People ...");
        int currentIndex = preamble.indexOf("e");
        int count = 1;
        while (currentIndex > 0) {
            System.out.println(count + " " + currentIndex);
            currentIndex = preamble.indexOf("e", currentIndex + 1);
            count++;
        }
    }
}
```
Activity -- working with Strings

• for each string in the command line,
  • print the string
  • then print each char in string on separate line
    • if the character is 'q' or the same as the first character in the string also print "!!!!" on the line
  • leave a blank line between words
• This will look a lot like you are working with a 2d array
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')));
    }
}
```

substring() method:
- `substring(int beginIndex)`
  - Returns a string that is a substring of this string.
- `substring(int beginIndex, int endIndex)`
  - Returns a string that is a substring of this string.

Improvable, worth it??
• Strings and all instances of classes have two ways to compare to each other
  • ==
    • compares pointers!
  • .equals()
    • compares strings

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));
    }
}
Classes and Non-classes
or Java is weird

• Pretty much everything you do in java is with a class

• All semester
  • "Classes should start with an initial capital letter"

• But int, float, double, long, char, boolean, short, byte start with a lower case letter
  • what is with that??