Strings and characters

- A string is simply an array of characters
- Characters are letters, symbols, spaces, punctuation, ...
- Characters are stored as the data type char
- A char is 2 bytes long

Characters

- We use single quotes to surround characters
  ```java
  char ch1 = 'a';
  char ch2 = 'A';
  char ch3 = '!';
  char ch4 = '7';
  ```
Representing characters numerically

- char variables do not store characters directly
- char values are converted to a numeric value
  - ASCII is used to represent Western languages (like English): 128 codes
  - Unicode is used to represent non-Western languages with special characters (as well as Western languages): 34,168 codes
- Java: uses Unicode

More about ASCII

- “Printable” characters have ASCII codes between 32 and 126
  - the rest are control characters, like TAB and ESC
- Some common ASCII codes:
  - d = 100
  - D = 68
  - @ = 64
  - space = 32

Using `char` and ASCII

- We can compare 2 characters directly, because they are represented numerically:
  - e.g., 'T' < '6' is false ('T' = 84, '6' = 54)
- Convert character to ASCII code:
  ```java
  char c1 = 'f';
  System.out.println(c1 + " = "+(int)c1);
  ```
- Convert ASCII code to character:
  ```java
  System.out.println("Character 80 is "+(char)80);
  ```

Strings

- So far, we've been treating String as a primitive type
- `String s = “Blah blah blah”;` is equivalent to
  ```java
  String s = new String("Blah blah blah");
  ```
  - sort of!
Example

```java
String s = "Blah blah blah"; // new String stored in memory
String s2 = s;
// s and s2 point to the same location in memory now
String s3 = new String("Blah blah blah");
// contents of s3 are the same as s, s2 BUT s3 is in a
// different spot in memory!
String s4 = new String("Blah blah blah"); // ditto
boolean areEqual;
areEqual = s1 == s2;
areEqual = s1.equals(s2);
areEqual = s3 == s4;
areEqual = s3.equals(s4);
areEqual = s1 == s3;
```

StringBuffer

- Cannot change strings once they are created!
  - string methods return a new string
- String manipulation: use a StringBuffer
  - can replace characters, delete characters, append characters, insert characters....directly to the string

Question

Say we want to count the number of words in a sentence. How would we go about doing this?

Another useful string manipulation tool: StringTokenizer

- Splits a string into pieces
- Can use any marker we want
  - typically spaces, punctuation marks, etc.
- In book: pg 769-771 (697-699 in old version)
- In java.util