### **Character Data Type**

* single character
* char letter = 'A'; char numChar = '4';
* NOTE: "A" is a String, 'A' is a character
* A String "Hello World" is made up of 11 characters

### **[ASCII](https://en.wikipedia.org/wiki/ASCII)**

### **[(Links to an external site.)](https://en.wikipedia.org/wiki/ASCII)**

* Characters are stored to numerical values using ASCII encoding
* '0' to '9' are mapped to 48 to 57
* 'A' to 'Z' are mapped to 65 to 90
* 'a' to 'z' are mapped to 97 to 122
* So, you can do letter++ or numChar < letter.
* Matt Damon uses ASCII in *[The Martian](https://en.wikipedia.org/wiki/The_Martian_(film))*
* *[(Links to an external site.)](https://en.wikipedia.org/wiki/The_Martian_(film))*
* to talk to Earth. Read about it [here](http://www.businessinsider.com/the-martian-hexidecimal-language-2015-9)
* [(Links to an external site.)](http://www.businessinsider.com/the-martian-hexidecimal-language-2015-9)
* .

### **Special Characters**

* \b - backspace
* \t - tab
* \n - linefeed
* \\ - \ (back slash)
* \" - " (double quote)
* Example:

System.out.println("He said \"Java is fun\"");

### **Character Methods**

* Character.isDigit(c)
* Character.isLetter(c)
* Character.isLetterOrDigit(c)
* Character.isLowerCase(c)
* Character.isUpperCase(c)
* Character.toUpperCase(c)
* Character.toLowerCase(c)

### **[String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html)**

### **[(Links to an external site.)](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html)**

### **Data Type**

* Sequence to Characters
* String is a class like [Scanner](http://docs.oracle.com/javase/7/docs/api/java/util/Scanner.html)
* [(Links to an external site.)](http://docs.oracle.com/javase/7/docs/api/java/util/Scanner.html)
* or [System](https://docs.oracle.com/javase/7/docs/api/java/lang/System.html)
* [(Links to an external site.)](https://docs.oracle.com/javase/7/docs/api/java/lang/System.html)
* Example

String message = "Hello World";

### **String Methods**

* length() - number of characters in string
* charAt(x) - returns the char at position x (**NOTE**: positions start at 0)
* concat(s) - concatenate strings
* toUpperCase() - convert to all caps
* toLowerCase() - convert to all lower case
* trim() - removes whitespace from front and end of string
* Example

String message = "Hello World"

String newMessage = message.concat(" to Jane");

int len = message.length(); // 11

char c = message.charAt(6); // 'W'

* NOTE: Can also use + to cancatenate two strings

### **Reading String from Keyboard**

* Use Scanner
* input.next() - gets the next word
* input.nextLine() - gets the whole line
* Reading a character; use nextLine() followed by charAt(0)

### **Comparing String**

* equals(s)
* equalsIgnoreCase(s)
* startsWith(s)
* endsWith(s)
* contains(s)

### **Other Methods**

* substring(5) // substring starting at position 5
* substring(5,7) // substring starting a position 5 and ending at position 7
* indexOf(c) // position of first occurrence of character c

### **Converting Strings**

* Use Integer.parseInt() to covert string to int

String iStr = "123";

int i = Integer.parseInt(intStr); // 123

* Use Double.parseDouble() to convert string to double

String dStr = "12.3";

double d = Double.parseDouble(dStr); // 12.3

### **Testing Input**

* Can test format of input data before trying to assign it to a variable using hasNextInt() and hasNextDouble()
* For example

int i;

if (input.hasNextInt())

i = input.nextInt();

else System.out.println("Not an int");

### **Formatting Output**

* Use System.out.printf for more control of output
* Of the form: System.out.printf(format, item1, item2, ...)
* Format can have the following specifiers:
  + %d - for integers
  + %f - of decimals
  + %c - for characters
  + %s - for strings
  + %b - for booleans
* For example

String name = "Jane";

int temp = 65;

System.out.printf("Hello %s. Today's temperature is %d degrees.\n", name, temp);

### **Format Width and Precision Specifiers**

Illustrated with some examples:

* %5c - prints 4 spaces and then a character
* %5b - prints 0 space and false or 1 spaces and true
* %5d - prints an int right-justified in 5 spaces, e.g. " 123", " 1234"
* %-5d - prints an int left-justified in 5 spaces, e.g. "123 ", or "1234 "
* %6.2f - prints a decimal right-justified in 6 spaces with 2 digits after the decimal point, e.g. " 12.34"
* %12s - print a string right-justified in 12 spaces (or more if string is greater than length 12).