Primitive Data Types

After learning about variable initialization and assignment, you should be aware that data types are serious business. They can determine the success or failure of your project. Therefore, you should know them extremely well. This document should serve as a quick reference guide for the data types we will be using most often in this class. Research each of the terms below and write their definitions in the boxes below

int : Use the Integer class to use `int` data type as an unsigned integer.

Double: The `double` data type is a double-precision 64-bit IEEE 754 floating point. For decimal values, this data type is generally the default choice. As mentioned above, this data type should never be used for precise values, such as currency.

Boolean: The `boolean` data type has only two possible values: `true` and `false`. Use this data type for simple flags that track true/false conditions. This data type represents one bit of information, but its "size" isn't something that's precisely defined.

float: As with the recommendations for `byte` and `short`, use a `float` (instead of `double`) if you need to save memory in large arrays of floating point numbers. This data type should never be used for precise values, such as currency.

char: The `char` data type is a single 16-bit Unicode character. It has a minimum value of `'\u0000'` (or 0) and a maximum value of `'\uffff'` (or 65,535 inclusive).

short: The `short` data type is a 16-bit signed two's complement integer. It has a minimum value of -32,768 and a maximum value of 32,767 (inclusive). As with `byte`, the same guidelines apply: you can use a `short` to save memory in large arrays, in situations where the memory savings actually matters.

long: he `long` data type is a 64-bit two's complement integer. The signed long has a minimum value of -2^{63} and a maximum value of $2^{63}-1$. In Java SE 8 and later, you can use the `long` data type to represent an unsigned 64-bit long, which has a minimum value of 0 and a maximum value of $2^{64}-1$. Use this data type when you need a range of values wider than those provided by `int`. The `Long`

class also contains methods like `compareUnsigned`,`divideUnsigned` etc to support arithmetic operations for unsigned long.