Data types and Variables

Mangat.

Variables

- A variable, similar to math class, is a holder for data
- In math we commonly use ‘x’ to hold some value. 99% of the time we are trying to find out what the value is.
- In programming we can use words instead of just a single letter and we can store more than just numbers

Example code

```
class AddTip {
   public static void main(String args[]) {
         double amount;
         amount = 19.95;
         amount = amount + 3.00;
         System.out.print(“We will pay $”);
         System.out.print(amount);
         System.out.println(“ for the pizza delivery .”);
   }
}
```

Variables

double amount;

- This code creates a **variable** called amount
- The variable is capable of storing a **double** value (real number)
- At this point in the code amount does not have any value **assigned** to it.

Assignment

amount = 19.95;

- The variable *amount* has been **assigned** a value of 19.95
- This means that the number 19.95 is now stored inside of the variable *amount*

Operators

amount = amount + 3.00;

- The code above takes the current value of *amount* (19.95), adds 3.00 to it, and then assigns the new value to *amount*
- After this statement, *amount now stores 22.95*
- The + is an one of many **operations** you can use in Java

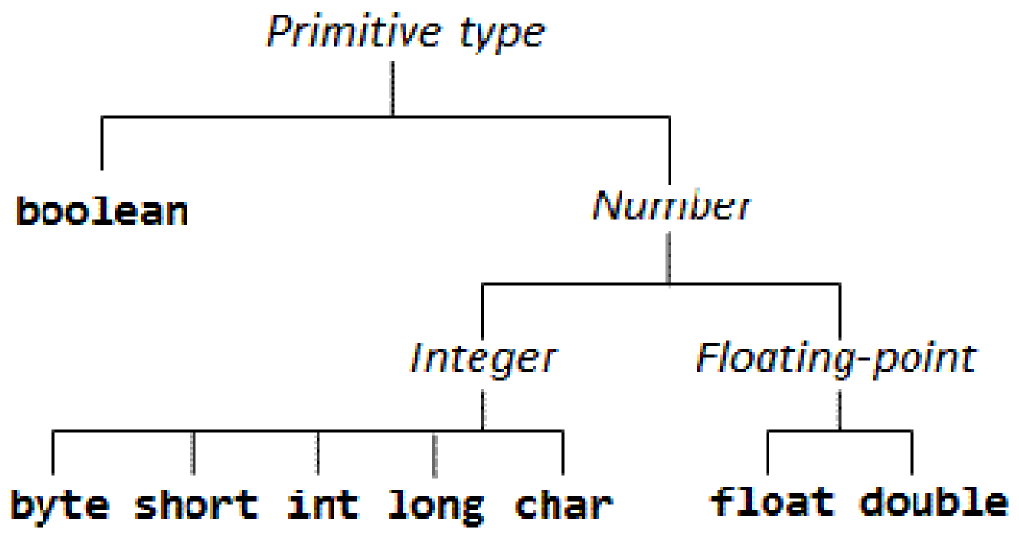
Outputting a variable

```
System.out.print(“We will pay $”);
System.out.print(amount);
```

- Notice that there are no quotation marks around *amount*
- Quotations specify that a specific **string** (letters/words) is to be outputted
- No quote means that the stored value inside of *amount* is to be outputted (22.95)

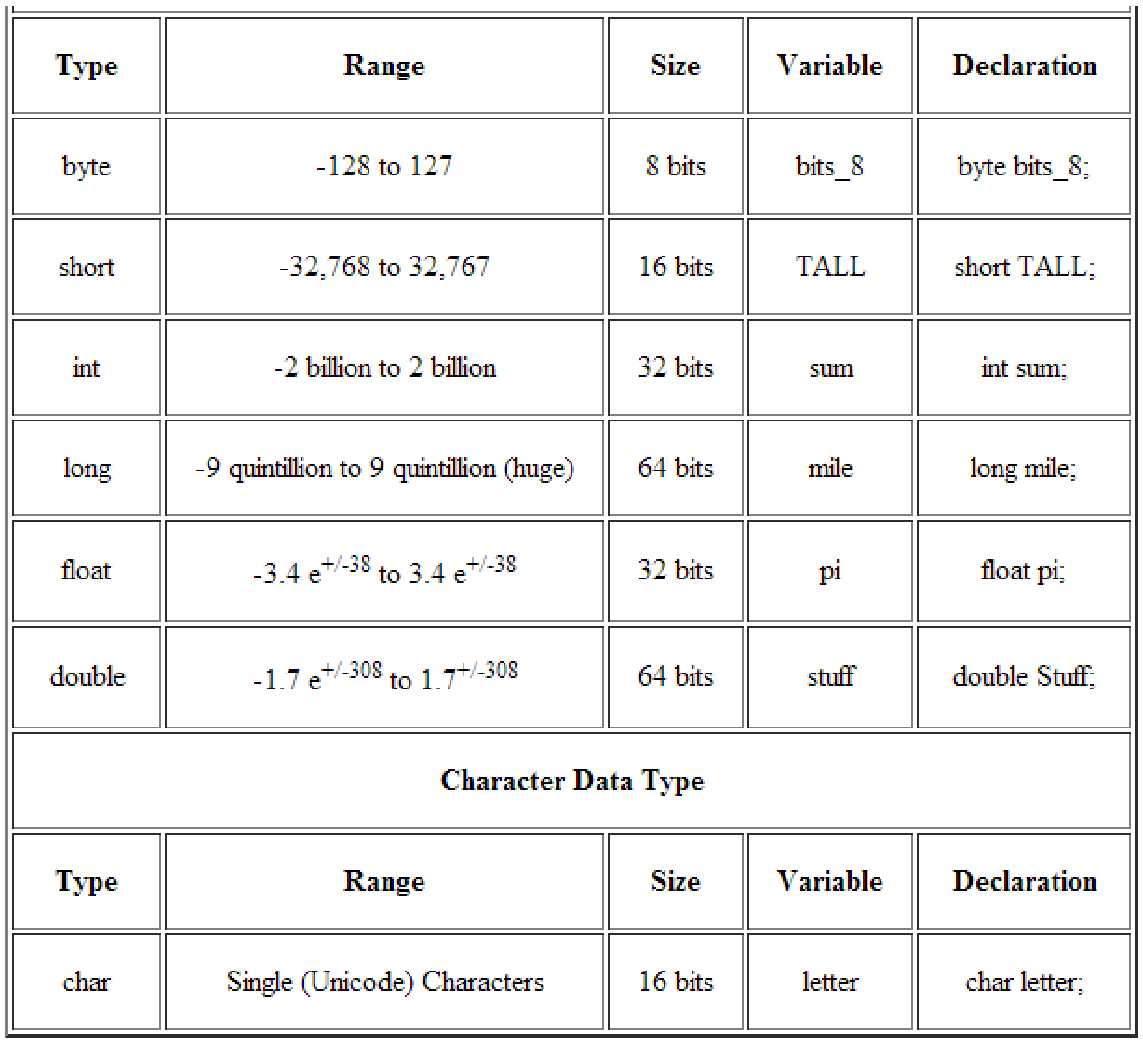
Java – Variable types

- Below is a diagram of Java **primitive** types
- A primitive is a data type that is built into java. (Notice it is not like System.out.println)



Java Primitives

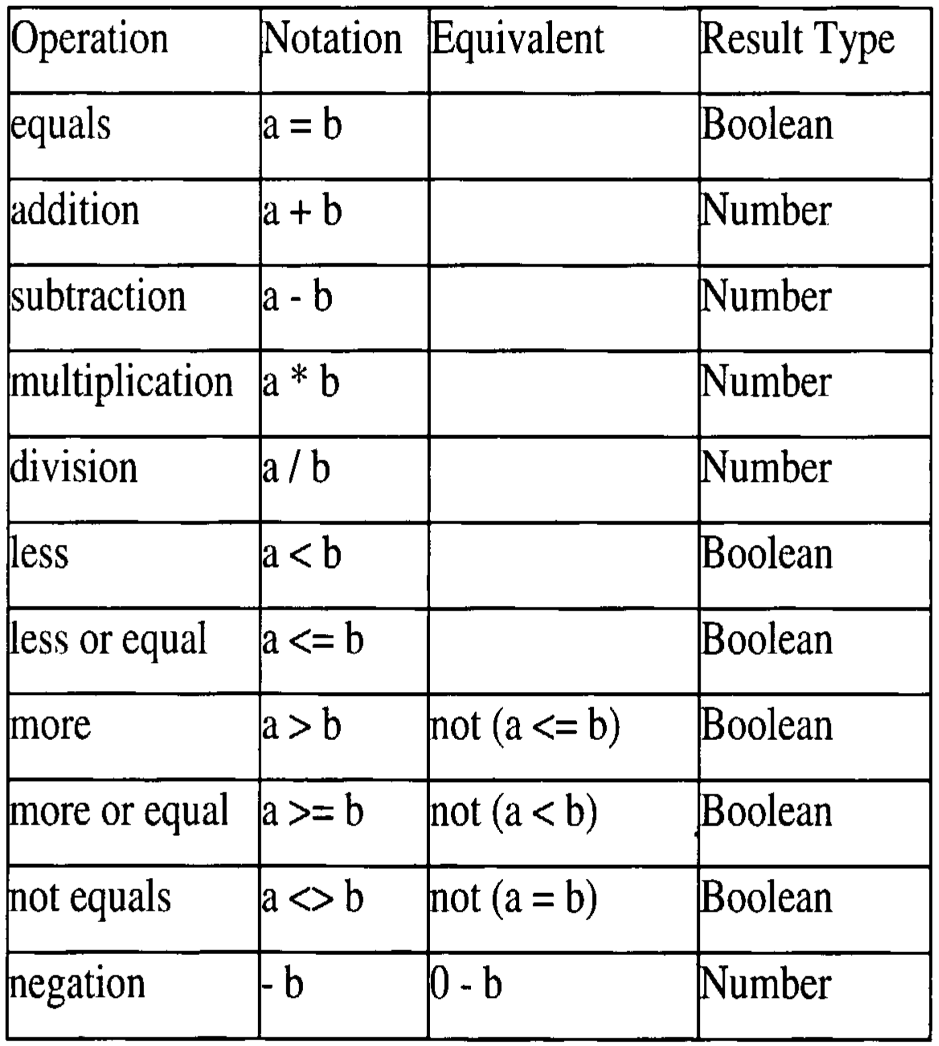
- Variables can be created for any of the data types listed
- The chart shows example of how to create a new variable for each primitive



Type	Range	Size	Variable	Declaration
byte	-128 to 127	8 bits	bits_8	byte bits_8;
short	-32,768 to 32,767	16 bits	TALL	short TALL;
int	-2 billion to 2 billion	32 bits	sum	int sum;
long	-9 quintillion to 9 quintillion (huge)	64 bits	mile	long mile;
float	$-3.4\ e^{+/-38}$ to $3.4\ e^{+/-38}$	32 bits	pi	float pi;
double	$-1.7\ e^{+/-308}$ to $1.7^{+/-308}$	64 bits	stuff	double Stuff;
Character Data Type				
Type	**Range**	**Size**	**Variable**	**Declaration**
char	Single (Unicode) Characters	16 bits	letter	char letter;

Java - Operations

- These are operations that can be performed
- The 'answer' to the expression is called a result
- BEDMAS applies



Operation	Notation	Equivalent	Result Type
equals	a = b		Boolean
addition	a + b		Number
subtraction	a - b		Number
multiplication	a * b		Number
division	a / b		Number
less	a < b		Boolean
less or equal	a <= b		Boolean
more	a > b	not (a <= b)	Boolean
more or equal	a >= b	not (a < b)	Boolean
not equals	a <> b	not (a = b)	Boolean
negation	- b	0 - b	Number

You turn

```
class AddTip {
   public static void main(String args[]) {
          double amount;
          amount = 19.95;
          amount = amount + 3.00;
          System.out.print(“We will pay $”);
          System.out.print(amount);
          System.out.println(“ for the pizza delivery .”);
   }
}
```

Exercise – carpet.java

- Design a program with three variables: a, *w*, and *l*
- Assign a value of 16 to w
- Assign a value of 10 to l
- Multiply the two variables and store the result into a
- Output the value of a to the screen

- What does the program do?
  - Rename the variables to make them more appropriate to the scenario
  - Output an appropriate message when giving the result back to the user
  - Make sure to include a header and some comments.

Questions?

- Next – Interaction with the user