# Types and Math

Java supports two main types of data: ***primitive types*** (like `int` and `double`) that represent simple data, and ***reference types*** (like `String` and `Scanner`) that represent more complex data.

Manager: Recorder:

Presenter: Reflector:

## Content Learning Objectives

*After completing this activity, students should be able to:*

- Explain what values can be assigned to primitive type variables.
- Describe what it means for a variable to store an object reference.
- Apply methods from the Math class based on the documentation.

## Process Skill Goals

*During the activity, students should make progress toward:*

- Tracing assignment statements to update contents of memory. (Information Processing)

# Model 1 Primitive Types

| Keyword | Size | Min Value | Max Value | Example |
|---|---|---|---|---|
| `byte` | 1 byte | $-128$ | 127 | `(byte) 123` |
| `short` | 2 bytes | $-32,768$ | $32,767$ | `(short) 12345` |
| `int` | 4 bytes | $-2^{31}$ | $2^{31}-1$ | `1234567890` |
| `long` | 8 bytes | $-2^{63}$ | $2^{63}-1$ | `123456789012345L` |
| `float` | 4 bytes | $-3.4 \times 10^{38}$ | $3.4 \times 10^{38}$ | `3.14159F` |
| `double` | 8 bytes | $-1.8 \times 10^{308}$ | $1.8 \times 10^{308}$ | `3.141592653589793` |
| `boolean` | 1 byte | N/A | N/A | `true` |
| `char` | 2 bytes | 0 | 65,535 | `'A'` |

Note that 1 byte is 8 bits, i.e., eight "ones and zeros" in computer memory. Since there are only two possible values for each bit, you can represent $2^8 = 256$ possible values with 1 byte.

## Questions (15 min) Start time:

**1**. Which of the primitive types are integers? Which are floating-point (i.e. have a fractional part)?

**2**. Why do primitive types have ranges of values? What determines the range of the data type?

**3**. Why can't computers represent every possible number in mathematics? Will they ever be able to do so?

**4**. What is the data type for each of the following values? (Note: F stands for float and L for long. The default types are double and int.)

| | | |
|---|---|---|
| 1.14159 | 7.2E-4 | -128 |
| 0 | 0.0 | '0' |
| -1.0F | -13L | false |
| 123 | 'H' | true |

**5**. Based on the examples below, when does Java allow you to assign one type of primitive variable to another? Note the underscore makes the type word a variable name.

```
Assigning a literal to a variable
int int_ = 3;
long long_ = 3L;
float float_ = 3.0F;
double double_ = 3.0;

Assigning to an int variable
int_ = int_;
int_ = long_;      // illegal
int_ = float_;     // illegal
int_ = double_;    // illegal

Assigning to a long variable
long_ = int_;
long_ = long_;
long_ = float_;    // illegal
long_ = double_;   // illegal
```

```
Assigning to a float variable
float_ = int_;
float_ = long_;
float_ = float_;
float_ = double_;  // illegal

Assiging to a double variable
double_ = int_;
double_ = long_;
double_ = float_;
double_ = double_;

Assigning between different types
int_ = '0';
int_ = false;      // illegal
double_ = '0';
double_ = false;   // illegal
boolean = 2.0;   // illegal
```

**6**. Given the following variable declarations, which of the assignments are not allowed?

```
byte miles;
short minutes;
int checking;
long days;
float total;
double sum;
boolean flag;
char letter;
```

```
checking = 56000;
total = 0;
sum = total;
total = sum;
checking = miles;
sum = checking;
flag = minutes;
days = '0';
```

# Model 2 Reference Types

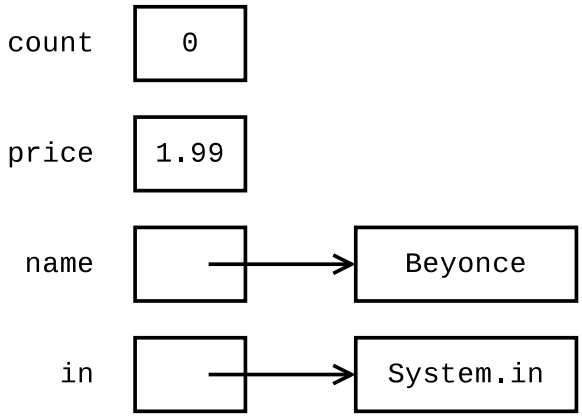
Java has eight primitive types: `byte`, `short`, `int`, `long`, `float`, `double`, `boolean`, `char`.

All other data types are objects, which use references. A ***reference*** is an integer that "points to" a location in memory. This is often called an address. For example, a `String` variable stores a reference to a string object.

When drawing memory diagrams, we use an arrow to refer to other memory locations (rather than make up integer values for the actual addresses).

```
int count;
double price;
String name;
Scanner in;

count = 0;
price = 1.99;
name = "Beyonce";
in = new Scanner(System.in);
```




## Questions (20 min) Start time:

**7**. What are the names of the reference types in the example above?

**8**. Notice the pattern Java uses for type names like `int` and `String`:

a) Are reference type names all lowercase or capitalized?

b) Are primitive type names all lowercase or capitalized?

**9**. Variables in Java can use at most eight bytes of memory. Explain why the values `"Beyonce"` and `System.in` cannot be stored directly in the memory locations for `name` and `in`.

**10**. What is the value of a primitive type variable like `count` or `price`?

**11**. What is the value of a reference type variable like `name` or `in`?

**12**. Carefully explain what it means to assign one variable to another. For example, what does the statement `price = count;` do in terms of memory?

**13**. Draw a memory diagram for the following code. Make sure your answer is consistent with what you wrote for #12.

```
int width;
double score;
Scanner input;
String first;
String other;

width = 20;
score = 0.94;
input = new Scanner(System.in);
first = "Taylor";
score = width;
other = first;
```

**14**. What is the output of the following statements after running the code above? Explain your answer using the diagram.

```
first = "Swift";
System.out.println(other);
```

# Model 3 Math Methods

Consider the following methods defined in the `Math` class. (This list isn't exhaustive; the `Math` class has over 90 methods in total!)

| Modifier and Type | Method and Description |
|---|---|
| `static int` | **`abs`**`(int a)`<br>Returns the absolute value of an `int` value. |
| `static double` | **`log`**`(double a)`<br>Returns the natural logarithm (base *e*) of a `double` value. |
| `static double` | **`pow`**`(double a, double b)`<br>Returns the value of the first argument raised to the power of the second argument. |
| `static double` | **`random`**`()`<br>Returns a `double` value with a positive sign, greater than or equal to `0.0` and less than `1.0`. |
| `static int` | **`subtractExact`**`(int x, int y)`<br>Returns the difference of the arguments, throwing an exception if the result overflows an `int`. |

The code for these methods is written in a file named *Math.java*. Here is what the definition of the `abs` method looks like:

```
public static int abs(int a) {
    // code omitted
}
```

To use a method from another source file (like *Math.java*), you must first specify the class name:

```
value = abs(-5);       // Error: cannot find symbol
value = Math.abs(-5);  // correct
```

## Questions (15 min) Start time:

**15**. What type of value does `Math.random()` return? Give an example of what a random value might look like.

**16**. When *defining* a method (like `abs` or `log`), what do you need to specify before the method name and after the method name?

**17**. Define a method named `average` that takes two integers named `x` and `y` and returns a `double`. Don't write any semicolons or braces.

**18**. When *using* a method, what do you need to specify before the method name and after the method name?

**19**. For each method in Model 3, write a Java statement that uses the method and assigns the result to a variable.

*What you wrote for Question #17 is called the method's **signature**. The variables declared inside the parentheses are called **parameters**. When invoking a method, the values you provide are called **arguments**. Since arguments will be assigned in order to parameters, their types must be compatible and there must be the same number of arguments as parameters.*