What did we talk about last time?

- Software development
- Lab 1
Questions?
Data Representation
What if you want to write a Java program that can...

- Edit music files
- Play a DVD
- Organize your photo album

Each of these tasks manipulates a lot of data

MP3’s, DVD’s, and jpegs are complicated kinds of data

Java must build these kinds of data out of much simpler kinds of data
You have heard people (me, probably) talking about all the 1’s and 0’s inside of a computer. What does that all really mean?

Using semiconductor physics, we can make a tiny little piece of a microchip be in one of two states, say, OFF and ON, like a switch. If we say that OFF is 0 and ON is 1, then, by using a lot of these switches, we can represent a lot of 1’s and 0’s.
What do we do with those 1’s and 0’s?
To begin with, we represent numbers.
How many of you have heard of base 10?
How many of you have heard of base 2?
What’s the definition of a number system with a given base?
Our normal number system is base 10.

This means that our digits are: 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9.

Base 10 means that you need 2 digits to represent ten, namely 1 and 0.

Each place in the number as you move left corresponds to an increase by a factor of 10.
Base 10 Example

3,482,931

- Millions
- Hundred thousands
- Thousands
- Hundreds
- Tens
- Ones
The binary number system is base 2
This means that its digits are: 0 and 1
Base 2 means that you need 2 digits to represent two, namely 1 and 0
Each place in the number as you move left corresponds to an increase by a factor of 2 instead of 10
Base 2 Example

1024's -> 1111111000001 -> Ones

256's -> 256's

Sixty fours

Sixteens

256’s

Fours

128's

Eights

512's

Thirty twos

Twos
So, what’s the value?

11111100001 =

$1 \cdot 2^{10} + 1 \cdot 2^9 + 1 \cdot 2^8 + 1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5$

$+ 0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 =$

$1024 + 512 + 256 + 128 + 64 + 32 + 1 =$

2017
The good news

- You don’t actually have to worry about doing binary conversions when you are coding Java.
- You should know this information because it explains how Java is different from math.
- In math, you can talk about an arbitrarily large number.
- In Java, each number is stored with a specific number of binary digits.
- There are limits on how big (or small) a given number can be.
Built-in Types of Data
We are going to focus on five basic types of data in Java. These are:

- **int**: For whole numbers
- **double**: For rational numbers
- **boolean**: For true or false values
- **char**: For single characters
- **String**: For words

*String* is a little different from the rest, but we’ll get into that later.
The int Type
The `int` type

- As I just said, the `int` type is used to store integers (positive and negative whole numbers and zero)
- Examples:
  - 54
  - -893992
  - 0
- Inside the computer, an `int` takes up 4 bytes of space, which is 32 bits (1’s and 0’s)
With 32 bits, an int can hold integers from about -2 billion up to 2 billion

Positive numbers are represented just like we’ve seen

Negative numbers need an extra trick that we aren’t going to go into

The actual maximum value is: 2147483647
Let’s say you add 100 to the maximum `int` value 2147483647
You do **not** get 2147483747
Instead, it becomes a very negative number: -2147483549
This phenomenon is called **overflow**
The opposite thing happens if you have a very negative number and you subtract a number that makes it too negative
This phenomenon is called **underflow**
Each type has a number of literals associated with it

A literal is a concrete value within a given type

Literals for the int type are numbers exactly like what you would expect:

- 115
- -9837461
- 2
Variables

- It is also possible to create **variables** for each data type
- Think of a variable as a “box” that you can put values into
- The name of a variable is an **identifier**
- We can **declare** a variable of type `int` with **identifier** `i` using the following line of code:

```c
int i;
```
Storage for an `int`

```java
int i;
```

- This line of code creates a box named `i` that is designed only to hold `ints`
- That is, it is 32 bits in size

What’s inside this box?
Assignment into an int

- By default, the declaration of an int puts the literal value 0 inside the box

- Remember, you must declare a variable before using it
Java variables are not like variables in math which have a fixed (but unknown) value.

Instead, a Java variable can be changed by a line of code.

We use the assignment operator (=) to change the value of a variable as follows:

```
i = 5;
```
Changing the value of a variable

- This line of code \textbf{stores} 5 into \texttt{i}
- Think of the \texttt{=} operator as an arrow pointing left

\begin{itemize}
  \item Let’s see this in Eclipse
\end{itemize}
You will use the `int` type very often

Sometimes, however, you need to represent numbers with a fractional part

The `double` type is well suited to this purpose

Declaration of a `double` variable is just like an `int` variable:

```c
double x;
```
This line of code creates a box named `x` that is designed only to hold `doubles`.

It has a different size from an `int x`.
This line of code **stores 3.14159 into x**

Remember that the = operator is like an arrow pointing left

Let’s see this in Eclipse
The boolean Type
Numbers are great
But, sometimes you only need to keep track of whether or not something is true or false
This is what the boolean type is for
You will understand it better when we cover conditionals in a couple of weeks
Declaration of a boolean variable is like so:

```plaintext
boolean value;
```
This line of code creates a box named `value` that is designed only to hold `booleans`. It cannot be used to store numbers.
Assignment for a boolean

value = false;

- This line of code **stores false** into **value**
- Remember that the = operator is like an arrow pointing left

false

false
The char Type
Sometimes you need to deal with characters
This is what the `char` type is for
The `char` type only allows you to store a single character like ' $ ' or ' q '
Declaration of a `char` variable is like so:

```cpp
char c;
```
Storage for a `char`

```cpp
char c;
```

- This line of code creates a box named `c` that is designed only to hold `chars`.
- It is used to store characters from `most` of the different scripts in the world.
This line of code **stores** the letter 'a' into the variable **c**.

We must use the single quotes so that Java knows we are talking about the character 'a' and not a variable named `a`.

Try this on your own in Eclipse.
The String Type
The **String** type is different from the other types in several ways:

- The **important** thing for you to focus on now is that it can hold a large number of **chars**, not just a single value.
- A **String** literal is what we used in the Hello, World program.

```java
String word;
```
String word;

- This line of code creates a box named `word` that is designed only to hold `Strings`.
- It is used to store text of any length from `most` of the different scripts in the world.
Assignment for a String

word = "Mad flavor";

- This line of code stores the String "Mad flavor" into word
- We must use the double quotes so that Java knows we are talking about the text "Mad flavor"

```
word

"Mad flavor"

"Mad flavor"
```
Recap
## Summary of types

<table>
<thead>
<tr>
<th>Type</th>
<th>Kind of values</th>
<th>Sample Literals</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>Integers</td>
<td>-5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>900031</td>
</tr>
<tr>
<td>double</td>
<td>Floating-point Numbers</td>
<td>3.14</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-0.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.02e23</td>
</tr>
<tr>
<td>boolean</td>
<td>Boolean values</td>
<td>true</td>
</tr>
<tr>
<td></td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>char</td>
<td>Single characters</td>
<td>'A'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'Z'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'&amp;'</td>
</tr>
<tr>
<td>String</td>
<td>Sequences of characters</td>
<td>&quot;If you dis Dr. Dre&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;10 Sesquipedalians&quot;</td>
</tr>
</tbody>
</table>
Upcoming
Next time...

- Getting input from the command line
- Basic math operations
- Lab 2
Reminders

- Keep reading Chapter 3 of the textbook