Panel 1

Last Time

Framework of every Java program

```java
public class MyProgram {
    public static void main(String[] args) {
        /* Code goes here */
    }
}
```

3 steps to create a program

1. Edit
2. Compile
3. Execute

Compiling and executing with BlueJ

₃

HW: Create a snowman or Olympic Rings

Panel 2

Min HW:

- Create a folder CSAS1111
  in "My Documents"
- Every BlueJ project from now on should
  live inside that CSAS1111 folder
- Turn file extension viewing on!
  - Tools \ Folder Options \ View
    - Open any Folder
      □ Hide File Ext.
Panel 3

**Why Java?**

- Modern yet tried language (by SUN)
- Open & Free
- Object-Oriented
- Great documentation
- Machine-independent (create once, run everywhere)
  
e.g. Mac or Linux or Sun

Panel 4

**Definition 1.06: Java Virtual Machine (JVM)**

The Java Virtual Machine (JVM) is a platform-independent engine used to run Java applets and applications. The JVM knows nothing of the Java programming language, but it does understand the particular file format of the platform and implementation independent class file produced by a Java compiler. Therefore, class files produced by a Java compiler on one system can execute without change on any system that can invoke a Java Virtual Machine.

When invoked with a particular class file, the JVM loads the file, goes through a verification process to ensure system security, and executes the instructions in that class file.
Panel 5

Data

Java understands 4 "atomic" or basic data types:

- int - integer, or whole #, pos. or neg
- char - character, single character
- double - real or decimal #
- boolean - logical value true or false

(String) - list of characters (or text)

Panel 6

A little more details:

Definition 1.08: Primitive Java Data Types

Java supports the following primitive, or basic, data types:

- int, long, or short to represent integer numbers
- byte or char to represent decimal numbers
- float to represent character values
- double to represent logical values
- short to represent "no type"

Each numeric type has a largest and smallest possible value, as indicated in table 1.10.

Definition 1.09: Literals

Literals are constant values for the basic data types. Java supports the following literals:

- int, short: digits only, with possible leading plus (+) or minus (-) sign
- long: like int literals, but must end with an "L"
- doubles: digits including possible periodic point or leading plus (+) or minus (-) sign, or numbers in scientific notations, e.g. 123.456E78, where each s represents a digit
- char: Single Unicode characters enclosed in single quotes, including the special control sequences described in table 1.11
- boolean: true or false

In addition, Java has an object literal called null for object references.

Examples: int: 1, 42, long: 123L
float: 5.2f, boolean: true
Panel 7

**Binary Representation**

Ultimately, every computer must store all data as a sequence of 0’s and 1’s inside switches! A 0 or 1 is called 1 bit (not switch)

100101 is a number - which one?

First binary number:  
1st decimal number: 0

```
<table>
<thead>
<tr>
<th>Bin</th>
<th>Dec</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>11</td>
<td>3</td>
</tr>
<tr>
<td>100</td>
<td>4</td>
</tr>
<tr>
<td>101</td>
<td>5</td>
</tr>
<tr>
<td>110</td>
<td>6</td>
</tr>
<tr>
<td>111</td>
<td>7</td>
</tr>
<tr>
<td>1000</td>
<td>8</td>
</tr>
</tbody>
</table>
```

Panel 8

Convert **11001** from binary to decimal

1. Make a table of powers of 2, right to left

<table>
<thead>
<tr>
<th>2^5</th>
<th>2^4</th>
<th>2^3</th>
<th>2^2</th>
<th>2^1</th>
<th>2^0</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>16</td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ \frac{11001}{2} = 1100 \ldots 1 \]

2. Fill in binary number, right to left

3. Add the "powers with a 1": 16 + 8 + 1 = 25

Try **1000**
Convert 13 to binary

1. Make table of powers of 2

<table>
<thead>
<tr>
<th></th>
<th>32</th>
<th>16</th>
<th>8</th>
<th>4</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

2. Find largest power of 2 that fits your number, enter a 1 in that column

3. Find remainder and repeat

\[ \Rightarrow 13 = (1101)_2 \]
Panel 11

Convert \((\text{10001001})_2\) to decimal

\[
\begin{align*}
256 & \quad 128 \quad 64 \quad 32 \quad 8 \quad 4 \quad 2 \quad 1 \\
1 & \quad 0 & \quad 0 & \quad 1 & \quad 0 & \quad 0 & \quad 1 & \quad 1 \\
\end{align*}
\]

\[
289
\]

Convert \((\text{11011011})_2\) to decimal

\[
\begin{align*}
256 & \quad 128 \quad 64 \quad 32 \quad 8 \quad 4 \quad 2 \quad 1 \\
1 & \quad 1 & \quad 0 & \quad 1 & \quad 0 & \quad 0 & \quad 1 & \quad 1 \\
\end{align*}
\]

\[
128
\]

\[
\frac{256 + 128 + 32 + 8 + 2 + 1}{32} = \frac{32}{8} = \frac{2}{1} = 31
\]

Panel 12

Convert 31 to binary:

\[
\begin{align*}
32 & \quad 16 \quad 8 \quad 4 \quad 2 \quad 1 \\
1 & \quad 1 & \quad 1 & \quad 1 & \quad 1 & \quad 1
\end{align*}
\]

Convert 321 to binary

\[
\text{HW}
\]
Panel 13

Counting in binary:

\[ 2^5 = 32 \]

on one hand - 5 slots \( \Rightarrow \) [xxxxx]

largest

\[ 2^6 = 64 \] count to 1023 on both

hands

Is \((1101001101)_2\) even or odd?

Panel 14

**Definition 1.11: Bits and Bytes**

Computer memory is measured in bits and bytes. One bit is a unit that can store either a 0 or a 1. A group of 8 bits is called one byte.

How many bits or bytes are required to store one boolean?

If I used \(4\) bytes to store integers, what is the largest integer possible? How about plus/minus?

\[ 2^{32} - 1 = 4,294,967,296 - 1 \]

Plus + minus: \( 2^{31} - 1 \) or \( \times 2^{52} \)
Panel 15

Suppose my PC has 512 MB of memory.

a) What is the largest integer it could store?

\[ 512 \text{ MB} = 512 \cdot 1000000 \text{ bytes} = \frac{512000000 \text{ bytes}}{8 \text{ bytes\ per\ bit}} = 2^{20} \]

b) If one integer takes 4 bytes, how many integers could be stored?

\[ \frac{512000000 \text{ bytes}}{4 \text{ bytes\ per\ integer}} = 128000000 \text{ integers} \]

Panel 16

All data types stored on a computer have a smallest and largest value. In Java,

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>double</td>
<td>largest positive/negative value: ±1.7976931348623157E308</td>
</tr>
<tr>
<td></td>
<td>smallest non-zero value: ±4.9E-324</td>
</tr>
<tr>
<td></td>
<td>significant digits: 16 digits after decimal point</td>
</tr>
<tr>
<td>float</td>
<td>largest positive/negative value: ±3.4028235E38</td>
</tr>
<tr>
<td></td>
<td>smallest non-zero value: ±1.4E-45</td>
</tr>
<tr>
<td></td>
<td>significant digits: 8 digits after decimal point</td>
</tr>
<tr>
<td>int</td>
<td>largest value: 2147483647</td>
</tr>
<tr>
<td></td>
<td>smallest value: -2147483648</td>
</tr>
<tr>
<td>short</td>
<td>largest value: 32767</td>
</tr>
<tr>
<td></td>
<td>smallest value: -32768</td>
</tr>
<tr>
<td>long</td>
<td>largest value: 9223372036854775807</td>
</tr>
<tr>
<td></td>
<td>smallest value: -9223372036854775808</td>
</tr>
</tbody>
</table>
Panel 17

**Declare Variables**

on Wed.

Quiz Wed.