Types of Data

Lecture 3

- Finally!
  - Hey, that background is really important!
  - Lots of Computer Scientists don't know that stuff
  - We will see how it all relates
    - Starting today!

Looking at Code

Outline

- Data types
  - How data is manipulated
  - Variables and constants
  - What ranges of values are possible?
  - What expressions are possible?

Types

- Data is represented by a value and a type.
- The **type** is the category of value
  - It represents what range of values are possible
- There are **primitive types** and **complex types**
  - Primitive types are simple values
  - Complex types are made up of primitive types

- These types accept basic values we are all used to using
  - Integer numbers
  - Real numbers
  - Letters
  - Words
  - Truth

The Primitive Types

Integers

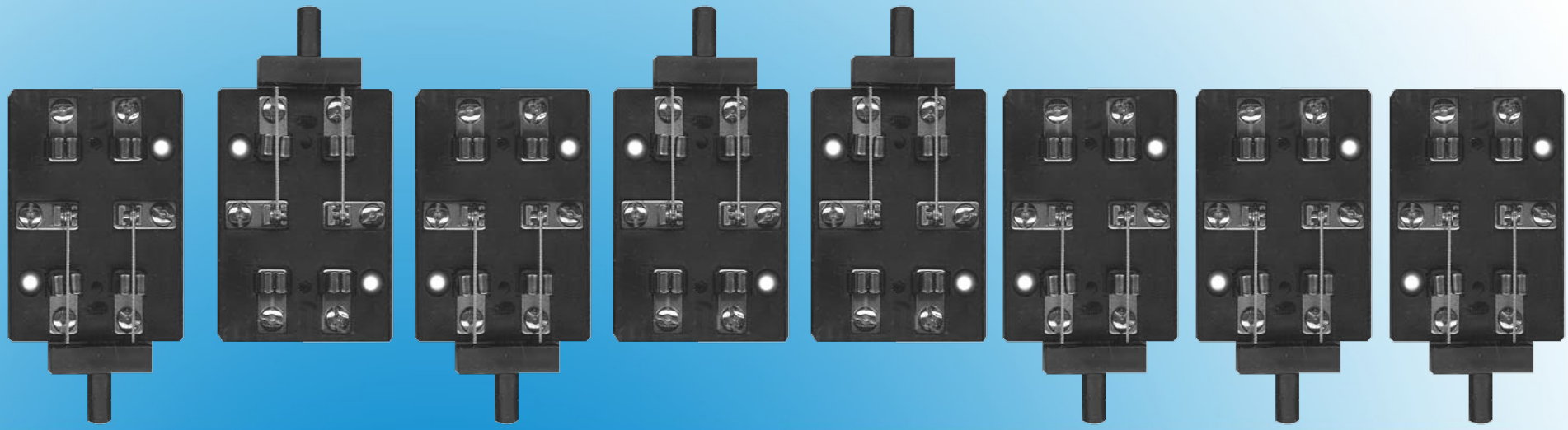
- An **integer** is a whole number.
  - Ex: 1, -5, 42, 3 * 5, 2 + 2, 2 / (4 * 2) + 5
- Mathematically, these values represent the set of numbers called $\mathbb{Z}$
- $\mathbb{Z}$ is infinite, so we usually *restrict the type to a range of numbers*.
- These are primarily used for counting

" ... 7 bits! 8 bits! 8 bits in a byte! Bwahahahahaha! "

– Count von Count

Java's Integers

- In Java, there are several *Integer* types:
  - **byte**, **short**, **int**, and **long**
- Each have a different range of values
  - Remember: we can only store finite values!



Java's Integer Numbers

- **byte** – 8 bits
  - -2^7 to 2^7-1 (-128 to 127)

- **short** – 16 bits
  - -2^{15} to $2^{15}-1$ (−32,768 to 32,767)

- **int** – 32 bits
  - -2^{31} to $2^{31}-1$ (−2,147,483,648 to 2,147,483,647)

- **long** – 64 bits
  - -2^{63} to $2^{63}-1$ (−9,223,372,036,854,775,808 to 9,223,372,036,854,775,807)

- **Real numbers** are those that may have a fractional component.
  - Ex: 3.1415, -1.5, 3, 1.5 + 1.5, 1.3 * 2
- Mathematically, these values represent the set of numbers called $\mathbb{R}$
- These are used for a wide variety of things
  - Currency, measurement, percentages, etc

Real Numbers

Java's Real Numbers

- Much like integers, Java has a couple of real types
  - They also differ in the ranges of values
- We have to encode these in binary
  - There are several ways to do this
- Java makes use of **floating point** representation
  - **IEEE 754** to be exact
  - It is actually über complicated
    - It makes my head hurt

Java's Real Numbers

- **float** – 32 bits
  - Stores 2^{32} values (1.401e-45 to 3.402e38)
- **double** – 64 bits
  - Stores 2^{64} values (4.941e-324 to 1.798e308)

- That range is far more than 2^{64}
  - In fact, it is *infinite*
  - What gives?

Floating Point Approximation

- Floating point values are approximations
  - Like all infinite things on a computer
- That means, some values cannot be stored
  - For instance, 3.42 might not fit!
  - It may be stored as 3.4200000000000001
- You will see this behavior in your programs

Expressions

- An **expression** is a group of symbols that makes a mathematical statement
- Interestingly, these expressions have a value and they have a type
- Examples:
  - 2 + 2
  - 1.5 * 2
  - 4.0 – 2
  - 7 / 2

Java's Expressions

- In Java, expressions look very much like familiar mathematical symbols

+	Addition
–	Subtraction
*	Multiplication
/	Division
%	Modulo (remainder)

Java's Expressions

- Java uses the tried and true My Dear Aunt Sally precedence
  - Parentheses will be done first

2 + 4 * 3	14
(2 + 4) * 3	18
3 * 2 + 4	10
3 * (2 + 4)	18

Constants and Variables

- **Constants** are values that cannot change during the execution of the program
- **Variables** are values that may change during the execution of the program
  - These are given a name
  - When you refer to a variable, you use the value it represents

(2 + a) * 3

((3 * b) + c) / d

Java's Variables

- To use a variable, you have to **declare** it first
  - You must describe the **type** and give it a **name**
  - The type and name do not change

- For instance, I need a variable to count the number of apples I have

```
int apples;
```

- I need to measure the area of my living room

```
double area;
```

Assignment in Java

- Once you have a variable declared, you can update its value at any time
- You do so with the assignment operator =

```
int apples;
apples = 42;
apples = 3 * apples;
apples = apples;
```

Assignment in Java

- Remember variables have a type
  - A type tells us what values can be assigned
- Some assignments are invalid:

```
int apples;
apples = 3.4;
apples = 3.4 * apples;
apples = 1.5 + 1.5;
```

Other Data Types: char

- We can store letters (*characters*) in a variable

```
char mander;
mander = 'a';
mander = 'a' + 1;
mander = mander + 1;
```

Other Data Types: boolean

- We can also represent *truth values*
  - These are values that reflect a value of true or false (much like a bit)

```
boolean truthiness;
truthiness = 3 > 4;
truthiness = 3 <= 4;
truthiness = apples == 42;
```

- Booleans are useful for making decisions
  - We might react differently depending on whether or not a statement is true
  - Like how to pronounce 'char' in conversation

<

```
char bear;
```

```
char izard;
```

Asking Questions

Asking Questions

- Some other questions people or code might ask:
  - Is a number even?
  - Am I done reading this file?
  - Is this password I received the correct one?
  - Has picobot cleared the maze?
  - Do students really get Monty Python references?
  - How devastating is the answer to the previous question?

Note: These are all either yes or no

Logical Operators

- Logical Operators are mathematical operators that compute truth values from a comparison
  - They are given two inputs
  - They result in a value of *true* or *false*

- There are several comparison operators:

> < >= <= == !=

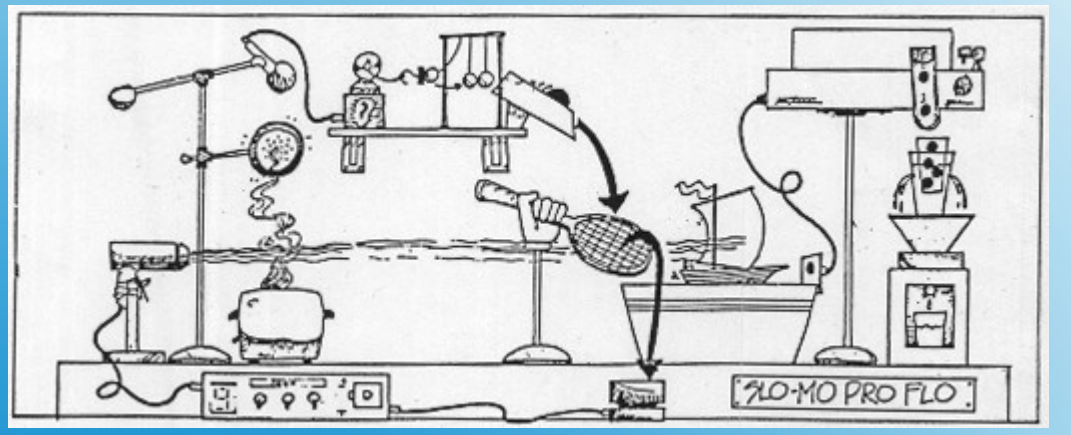
Boolean Expressions

- The result of a comparison has a *boolean* type
  - Just like 2 + 2 yields an *int* and 3.2 * 2 a *double*

```
boolean truthiness;
int apples;
apples = 42;
truthiness = apples == 42;
apples = 41;
truthiness = apples == 42;
```

Complex Types

- **Complex types** are those that are composed of primitive types
- We noted that we can represent complicated things with a simple mechanism
  - This is how we express this with programming



Other Data Types: String

- A **String** is a *complex data type* that represents a *string* of letters
  - It can represent words, street addresses, etc
- It is composed of many *char* variables, but you don't have to worry about how they are used
  - This is *abstracted* away!

```
String words;
words = "Hello";
words = words + ", Rosa!";
```

String is Different

- Because it is a complex data type, String operates differently
  - Because the actual values (chars) are hidden deep inside an abstraction
  - We will learn much more about this later
- As such, comparing Strings is awkward

```
String words = "Hello";
boolean truth =
  words.equals("Hello");
```

First Program

- Let's use some variables to write a simple program.
- We have to start with some boilerplate code:

```
public class MyFirstProgram {
	public static void main(String[] args) {
		// This is a comment!
		// Our code goes here!
	}
}
```

```
public class MyFirstProgram {

        public static void main(String[] args) {

                int number;
                number = 42;

                int remainder;
                remainder = number % 2;

                boolean truth = remainder == 0;

                // This will print out to the screen:
                System.out.println(truth);
        }
}
```

A Finished Program

- Save program as: `MyFirstProgram.java`
- First, we need to **compile** the program
  - Translates our code into machine language
  - Produces a `MyFirstProgram.class` file
  - Type: `javac MyFirstProgram.java`
- Now, we can run!
  - Type: `java MyFirstProgram`

To Execute