Using Data Types
Data Types

Data type. Set of values and operations on those values.

Primitive types. Ops directly translate to machine instructions.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Set of Values</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>true, false</td>
<td>not, and, or, xor</td>
</tr>
<tr>
<td>int</td>
<td>$-2^{31}$ to $2^{31} - 1$</td>
<td>add, subtract, multiply</td>
</tr>
<tr>
<td>double</td>
<td>any of $2^{64}$ possible reals</td>
<td>add, subtract, multiply</td>
</tr>
</tbody>
</table>

We want to write programs that process other types of data.
- Colors, pictures, strings, input streams, ...
- Complex numbers, vectors, matrices, polynomials, ...
- Points, polygons, charged particles, celestial bodies, ...
**Objects**

**Object.** Holds a data type value; variable name refers to object.

**Impact.** Enables us to create our own data types; define operations on them; and integrate into our programs.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Set of Values</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color</td>
<td>24 bits</td>
<td>get red component, brighten</td>
</tr>
<tr>
<td>Picture</td>
<td>2D array of colors</td>
<td>get/set color of pixel (i, j)</td>
</tr>
<tr>
<td>String</td>
<td>sequence of characters</td>
<td>length, substring, compare</td>
</tr>
</tbody>
</table>
Constructors and Methods

**To construct a new object:** Use keyword `new` and name of data type.

**To apply an operation:** Use name of object, the dot operator, and the name of the method.

```java
String s;
s = new String("Hello, World");
System.out.println(s.substring(0, 5));
```
Image Processing
**Color Data Type**

**Color.** A sensation in the eye from electromagnetic radiation.

**Set of values.** [RGB representation] $256^3$ possible values, which quantify the amount of red, green, and blue, each on a scale of 0 to 255.

<table>
<thead>
<tr>
<th>R</th>
<th>G</th>
<th>B</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>255</td>
<td>0</td>
<td>0</td>
<td><img src="image" alt="Red" /></td>
</tr>
<tr>
<td>0</td>
<td>255</td>
<td>0</td>
<td><img src="image" alt="Green" /></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>255</td>
<td><img src="image" alt="Blue" /></td>
</tr>
<tr>
<td>255</td>
<td>255</td>
<td>255</td>
<td><img src="image" alt="White" /></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td><img src="image" alt="Black" /></td>
</tr>
<tr>
<td>255</td>
<td>0</td>
<td>255</td>
<td><img src="image" alt="Magenta" /></td>
</tr>
<tr>
<td>105</td>
<td>105</td>
<td>105</td>
<td><img src="image" alt="Gray" /></td>
</tr>
</tbody>
</table>
Color Data Type

**Color.** A sensation in the eye from electromagnetic radiation.

**Set of values.** [RGB representation] $256^3$ possible values, which quantify the amount of red, green, and blue, each on a scale of 0 to 255.

**API.** Application Programming Interface.

```java
public class java.awt.Color

Color(int r, int g, int b)
int getRed() red intensity
int getGreen() green intensity
int getBlue() blue intensity
Color brighter() brighter version of this color
Color darker() darker version of this color
String toString() string representation of this color
```

http://java.sun.com/j2se/1.5.0/docs/api/java/awt/Color.html
Albers Squares

Homage to the Square by Josef Albers (1949-1975)
Albers Squares

Josef Albers. Revolutionized the way people think about color.

% java AlbersSquares 9 90 166 100 100 100
Public class AlbersSquares {
    public static void main(String[] args) {
        int r1 = Integer.parseInt(args[0]);
        int g1 = Integer.parseInt(args[1]);
        int b1 = Integer.parseInt(args[2]);
        Color c1 = new Color(r1, g1, b1);

        int r2 = Integer.parseInt(args[3]);
        int g2 = Integer.parseInt(args[4]);
        int b2 = Integer.parseInt(args[5]);
        Color c2 = new Color(r2, g2, b2);

        StdDraw.setPenColor(c1);
        StdDraw.filledSquare(.25, .5, .2);
        StdDraw.setPenColor(c2);
        StdDraw.filledSquare(.25, .5, .1);

        StdDraw.setPenColor(c2);
        StdDraw.filledSquare(.75, .5, .2);
        StdDraw.setPenColor(c1);
        StdDraw.filledSquare(.75, .5, .1);
    }
}
Monochrome Luminance

**Monochrome luminance.** Effective brightness of a color.

**NTSC formula.** \( Y = 0.299r + 0.587g + 0.114b. \)

```java
import java.awt.Color;

public class Luminance {
    public static double lum(Color c) {
        int r = c.getRed();
        int g = c.getGreen();
        int b = c.getBlue();
        return .299*r + .587*g + .114*b;
    }
}
```
**Color Compatibility**

**Q.** Which font colors will be most readable with which background colors on computer monitors and cell phone screens?

**A.** Rule of thumb: difference in luminance should be $\geq 128$.

```
public static boolean compatible(Color a, Color b) {
    return Math.abs(lum(a) - lum(b)) >= 128.0;
}
```
Grayscale

Grayscale. When all three R, G, and B values are the same, resulting color is on grayscale from 0 (black) to 255 (white).

Convert to grayscale. Use luminance to determine value.

```java
public static Color toGray(Color c) {
    int y = (int) Math.round(lum(c));
    Color gray = new Color(y, y, y);
    return gray;
}
```

Bottom line. We are writing programs that manipulate color.
Possible memory representation.

<table>
<thead>
<tr>
<th>D0</th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
<th>D4</th>
<th>D5</th>
<th>D6</th>
<th>D7</th>
<th>D8</th>
</tr>
</thead>
<tbody>
<tr>
<td>255</td>
<td>0</td>
<td>255</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>105</td>
<td>105</td>
<td>105</td>
</tr>
</tbody>
</table>

Object reference is analogous to variable name.

- We can manipulate the value that it holds.
- We can pass it to (or return it from) a method.
References

René Magritte. "This is not a pipe."

Java. This is not a color.

```
Color sienna = new Color(160, 82, 45);
Color c = sienna.darker();
```

OOP. Natural vehicle for studying abstract models of the real world.
**Picture Data Type**

**Raster graphics.** Basis for image processing.

**Set of values.** 2D array of **Color** objects (pixels).

**API.**

```java
public class Picture {
    Picture(String filename)  // create a picture from a file
    Picture(int w, int h)     // create a blank w-by-h picture
    int width()               // return the width of the picture
    int height()              // return the height of the picture
    Color get(int i, int j)   // return the color of pixel (i, j)
    void set(int i, int j, Color c)  // set the color of pixel (i, j) to c
    void show()               // display the image in a window
    void save(String filename)  // save the image to a file
}
```
Image Processing: Grayscale Filter

**Goal.** Convert color image to grayscale according to luminance formula.

```java
import java.awt.Color;

public class Grayscale {
    public static void main(String[] args) {
        Picture pic = new Picture(args[0]);
        for (int i = 0; i < pic.width(); i++) {
            for (int j = 0; j < pic.height(); j++) {
                Color color = pic.get(i, j);
                Color gray = Luminance.toGray(color);
                pic.set(i, j, gray);
            }
        }
        pic.show();
    }
}
```
Image Processing: Grayscale Filter

Goal. Convert color image to grayscale according to luminance formula.

mandrill.jpg  % java Grayscale mandrill.jpg
**Image Processing: Scaling Filter**

**Goal.** Shrink or enlarge an image to desired size.

**Downscaling.** To shrink, delete half the rows and columns.

**Upscaling.** To enlarge, replace each pixel by 4 copies.
Image Processing: Scaling Filter

**Goal.** Shrink or enlarge an image to desired size.

**Uniform strategy.** To convert from $w_s$-by-$h_s$ to $w_t$-by-$h_t$:

- Scale row index by $w_s / w_t$.
- Scale column index by $h_s / h_t$.
- Set color of pixel $(i, j)$ in target image to color of pixel $(i \times w_s / w_t, j \times h_s / h_t)$ in source image.
import java.awt.Color;

public class Scale {
    public static void main(String args[]) {
        String filename = args[0];
        int w = Integer.parseInt(args[1]);
        int h = Integer.parseInt(args[2]);
        Picture source = new Picture(filename);
        Picture target = new Picture(w, h);
        for (int ti = 0; ti < w; ti++) {
            for (int tj = 0; tj < h; tj++) {
                int si = ti * source.width() / w;
                int sj = tj * source.height() / h;
                Color color = source.get(si, sj);
                target.set(ti, tj, color);
            }
        }
        source.show();
        target.show();
    }
}
Image Processing: Scaling Filter

**Scaling filter.** Creates two Picture objects and two windows.

```
% java Scale 400 200 mandrill.jpg
```
More Image Processing Effects

RGB color separation

swirl filter  wave filter  glass filter  Sobel edge detection
Text Processing
String Data Type

**String data type.** Basis for text processing.

**Set of values.** Sequence of Unicode characters.

### API

```java
public class String (Java string data type)

String(String s)  // create a string with the same value as s
int length()  // string length
char charAt(int i)  // i-th character
String substring(int i, int j)  // i-th through (j-1)st characters
boolean contains(String sub)  // does string contain sub as a substring?
boolean startsWith(String pre)  // does string start with pre?
boolean endsWith(String post)  // does string end with post?
int indexOf(String p)  // index of first occurrence of p
int indexOf(String p, int i)  // index of first occurrence of p after i
String concat(String t)  // this string with t appended
int compareTo(String t)  // string comparison
String replaceAll(String a, String b)  // result of changing as to bs
String[] split(String delim)  // strings between occurrences of delim
boolean equals(String t)  // is this string’s value the same as t’s?
```

[http://java.sun.com/j2se/1.5.0/docs/api/java/lang/String.html](http://java.sun.com/j2se/1.5.0/docs/api/java/lang/String.html)
**Typical String Processing Code**

<table>
<thead>
<tr>
<th>is the string a palindrome?</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static boolean isPalindrome(String s)</td>
</tr>
<tr>
<td>{</td>
</tr>
<tr>
<td>int N = s.length();</td>
</tr>
<tr>
<td>for (int i = 0; i &lt; N/2; i++)</td>
</tr>
<tr>
<td>if (s.charAt(i) != s.charAt(N-1-i))</td>
</tr>
<tr>
<td>return false;</td>
</tr>
<tr>
<td>return true;</td>
</tr>
<tr>
<td>}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>extract file name and extension from a command-line argument</th>
</tr>
</thead>
<tbody>
<tr>
<td>String s = args[0];</td>
</tr>
<tr>
<td>int dot = s.indexOf(&quot;.&quot;);</td>
</tr>
<tr>
<td>String base = s.substring(0, dot);</td>
</tr>
<tr>
<td>String extension = s.substring(dot + 1, s.length());</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>print all lines in standard input that contain a string specified on the command line</th>
</tr>
</thead>
<tbody>
<tr>
<td>String query = args[0];</td>
</tr>
<tr>
<td>while (!StdIn.isEmpty())</td>
</tr>
<tr>
<td>{</td>
</tr>
<tr>
<td>String s = StdIn.readLine();</td>
</tr>
<tr>
<td>if (s.contains(query)) StdOut.println(s);</td>
</tr>
<tr>
<td>}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>print all the hyperlinks (to educational institutions) in the text file on standard input</th>
</tr>
</thead>
<tbody>
<tr>
<td>while (!StdIn.isEmpty())</td>
</tr>
<tr>
<td>{</td>
</tr>
<tr>
<td>String s = StdIn.readString();</td>
</tr>
<tr>
<td>if (s.startsWith(&quot;http://&quot;) &amp;&amp; s.endsWith(&quot;.edu&quot;))</td>
</tr>
<tr>
<td>StdOut.println(s);</td>
</tr>
<tr>
<td>}</td>
</tr>
</tbody>
</table>
In and Out
Bird's Eye View (Revisited)

input streams

standard input

command-line arguments

output streams

standard output

pictures

drawings
Non-Standard Input

or use OS to redirect from one file

Standard input. Read from terminal window.
Goal. Read from several different input streams.

In data type. Read text from stdin, a file, a web site, or network.

Ex: Are two text files identical?

```java
public class Diff {
    public static void main(String[] args) {
        In in0 = new In(args[0]);
        In in1 = new In(args[1]);
        String s = in0.readLine();
        String t = in1.readLine();
        StdOut.println(s.equals(t));
    }
}
```
**Goal.** Find current stock price of Google.

The current stock price is displayed as $459.52 at 11:45 AM ET. The NYSE symbol for Google is goog.

Screen Scraping

**Goal.** Find current stock price of Google.
- `s.indexOf(t, i)`: index of first occurrence of pattern `t` in string `s`, starting at offset `i`.
- Find first string delimited by `<b>` and `</b>` after `Last Trade`.

```java
public class StockQuote {
    public static void main(String[] args) {
        String name = "http://finance.yahoo.com/q?s=";
        In in = new In(name + args[0]);
        String input = in.readAll();
        int start = input.indexOf("Last Trade:", 0);
        int from = input.indexOf("<b>", start);
        int to = input.indexOf("</b>", from);
        String price = input.substring(from + 3, to);
        StdOut.println(price);
    }
}
```

% java StockQuote goog
459.52
OOP Summary

Object. Holds a data type value; variable name refers to object.

In Java, programs manipulate references to objects.
- Exception: primitive types, e.g., boolean, int, double.
- Reference types: String, Picture, Color, arrays, everything else.
- OOP purist: language should not have separate primitive types.

Bottom line. We wrote programs that manipulate colors, pictures, and strings.

Next time. We'll write programs that manipulate our own abstractions.