CMSC 150
INTRODUCTION TO COMPUTING

ACKNOWLEDGEMENT: THESE SLIDES ARE ADAPTED FROM SLIDES PROVIDED WITH INTRODUCTION TO PROGRAMMING IN JAVA: AN INTERDISCIPLINARY APPROACH, SEDGEWICK AND WAYNE (PEARSON ADDISON-WESLEY 2007)

LECTURE 8

• USING DATA TYPES
A FOUNDATION FOR PROGRAMMING

Any program you might want to write

Objects

Functions and Modules

Graphic I/O

Arrays

Conditionals and Loops

Math/String

Text I/O

Primitive data types

Expressions

Create your own data types
DATA TYPES

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

• **Primitive types.** Values directly map to machine representation; operations directly map 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>$[-2^{31}, 2^{31})$</td>
<td>add, subtract, multiply</td>
</tr>
<tr>
<td>double</td>
<td>any of $2^{64}$ real numbers</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 (data and functions); variable name refers to object.

• **Object-oriented programming.** Paradigm of programming (design) where you
  • Create your own data types (set of values and ops on them – *Encapsulation*).
  • Use them in your programs (manipulate objects that hold values).

<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>getRed(), brighten()</td>
</tr>
<tr>
<td>Picture</td>
<td>2D array of Colors</td>
<td>getPixel(i, j), setPixel(i, j)</td>
</tr>
<tr>
<td>String</td>
<td>Sequence of characters</td>
<td>length(), substring(), compare()</td>
</tr>
</tbody>
</table>

• This lecture. Use existing data types.

• Next lecture. Create your own data types.
PRACTICE

• Describe objects (data and functions) for an Aquarium
  • Be descriptive
  • Objects can contain other objects!
  • Objects interact with other objects!
EXERCISE

• Describe objects (data and functions) for the world of Harry Potter
  • Be descriptive
  • Objects can contain other objects!
  • Objects interact with other objects!
CONSTRUCTORS AND METHODS

• To **construct** a new object:
  • Synonymous with “initialization” for variables
  • Use keyword `new` (to invoke constructor).
  • Use name of data type (to specify which type of object).

• To apply an operation:
  • Use name of object (to specify which object).
  • Use the dot operator (to invoke method).
  • Use the name of the method (to specify which operation).

• On Memory – All objects are “Java references”, aka, Pointers!
  • Just like arrays

```
String s;
s = new String("Hello, World");
System.out.println(s.substring(0, 5));
```
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>Red</td>
</tr>
<tr>
<td>0</td>
<td>255</td>
<td>0</td>
<td>Green</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>255</td>
<td>Blue</td>
</tr>
<tr>
<td>255</td>
<td>255</td>
<td>255</td>
<td>Black</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Black</td>
</tr>
<tr>
<td>255</td>
<td>0</td>
<td>255</td>
<td>Magenta</td>
</tr>
<tr>
<td>105</td>
<td>105</td>
<td>105</td>
<td>Grey</td>
</tr>
</tbody>
</table>
COLOR DATA TYPE

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

• 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
    boolean equals(Color c)  // is this color's value the same as c's?
}
```
ALBERS SQUARES

- Josef Albers. Revolutionized the way people think about color.

*Homage to the Square* by Josef Albers (1949-1975)
import java.awt.Color; //Access library

public class AlbersSquares {
    public static void main(String[] args) {
        //Construct first color
        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);

        //Construct second color
        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);

        //Draw first square
        StdDraw.setPenColor(c1);
        StdDraw.filledSquare(.25, .5, .2);
        StdDraw.setPenColor(c2);
        StdDraw.filledSquare(.75, .5, .1);

        //Draw second square
        StdDraw.setPenColor(c2);
        StdDraw.filledSquare(.75, .5, .2);
        StdDraw.setPenColor(c1);
        StdDraw.filledSquare(.25, .5, .1);
    }
}

% java AlbersSquares 9 90 166 100 100 100
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 and cell phone screens?

• A. Rule of thumb: difference in luminance should be ≥ 128.

```java
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.

<table>
<thead>
<tr>
<th>red</th>
<th>green</th>
<th>blue</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>90</td>
<td>166</td>
<td></td>
</tr>
</tbody>
</table>

This color

<table>
<thead>
<tr>
<th>red</th>
<th>green</th>
<th>blue</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>74</td>
<td>74</td>
<td>74</td>
<td></td>
</tr>
</tbody>
</table>

Grayscale version

<table>
<thead>
<tr>
<th>red</th>
<th>green</th>
<th>blue</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Black

0.299 * 9 + 0.587 * 90 + 0.114 * 166 = 74.445
Objects and Memory

• Possible memory representation. Data sequentially aligned in memory.

```
+---+---+---+---+---+---+---+---+
| 0x0| 0x1| 0x2| 0x3| 0x4| 0x5| 0x6| 0x7| 0x8 |
+---+---+---+---+---+---+---+---+
| 255|  0| 255|  0|  0|  0| 105| 105| 105 |
+---+---+---+---+---+---+---+---+
```

• 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.
# PICTURE DATA TYPE

- 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 x, int y)  // return the color of pixel (x, y)
    void set(int x, int y, Color c)  // set the color of pixel (x, y) to c
    void show()              // display the image in a window
    void save(String filename)  // save the image to a file
}
```
• Goal. Convert color image to grayscale according to luminance formula.

1. import java.awt.Color;
2. public class Grayscale {
3.   public static void main(String[] args) {
4.       Picture pic = new Picture(args[0]);
5.       for (int x = 0; x < pic.width(); x++) {
6.           for (int y = 0; y < pic.height(); y++) {
7.               Color color = pic.get(x, y);
8.               Color gray = Luminance.toGray(color);
9.               pic.set(x, y, gray);
10.          }
11.     }
12.     pic.show();
13.   }
14.}
IMAGE PROCESSING: GRAYSCALE FILTER

mandrill.Jpg

% java grayscale mandrill.Jpg
MORE IMAGE PROCESSING EFFECTS

RGB color separation

swirl filter  wave filter  glass filter  Sobel edge detection
STRING DATA TYPE

• String data type. Basis for text processing.

• Set of values. Sequence of Unicode characters.

• API.

```
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) // ith character

  String substring(int i, int j) // ith through (i-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://download.oracle.com/javase/6/docs/api/java/lang/String.html
# Typical String Processing Code

**TYPICAL STRING PROCESSING CODE**

<table>
<thead>
<tr>
<th>TYPICAL STRING PROCESSING CODE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>is the string a palindrome?</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>extract file name and extension from a command-line argument</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>print all lines in standard input that contain a string specified on the command line</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>print all the hyperlinks (to educational institutions) in the text file on standard input</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
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.