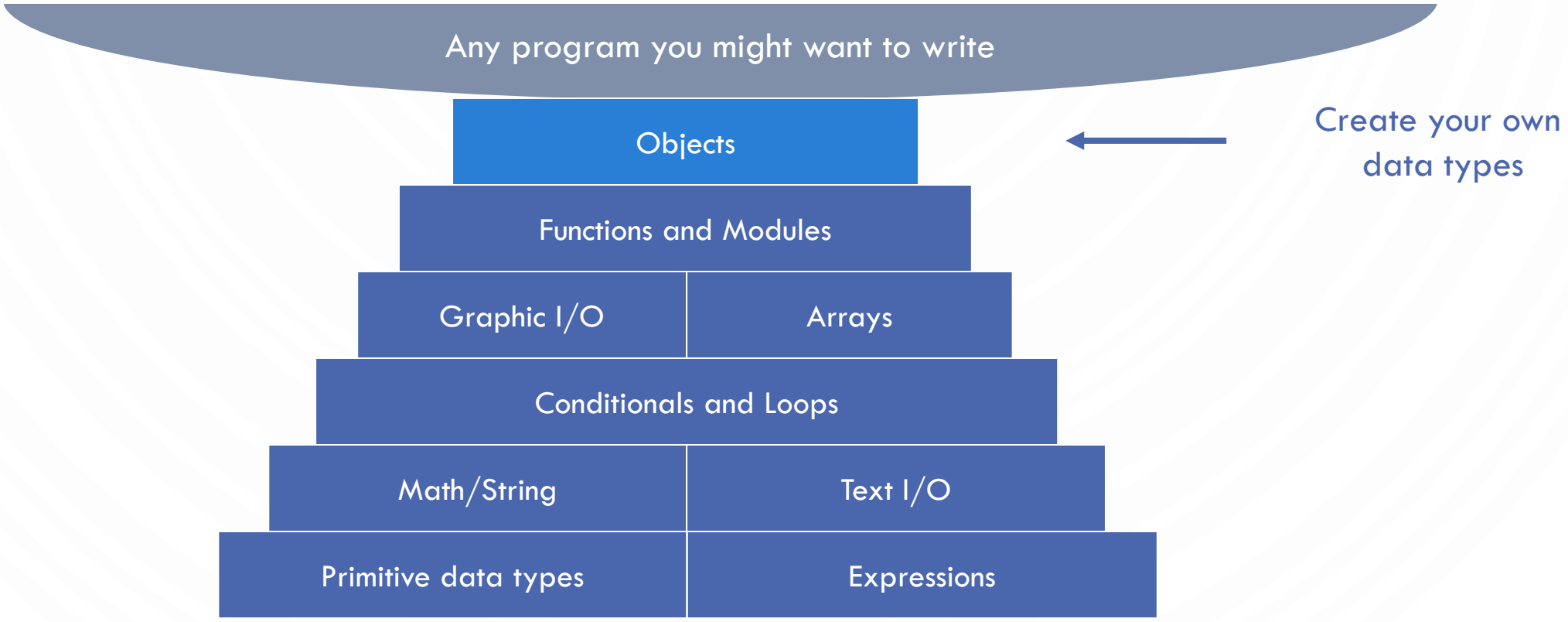
CMSC 150
INTRODUCTION TO COMPUTING

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



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.

Data Type	Set of Values	Operations
`boolean`	`true`, `false`	not, and, or, xor
`int`	$[-2^{31}, 2^{31})$	add, subtract, multiply
`double`	any of 2^64 real numbers	add, subtract, multiply

- 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

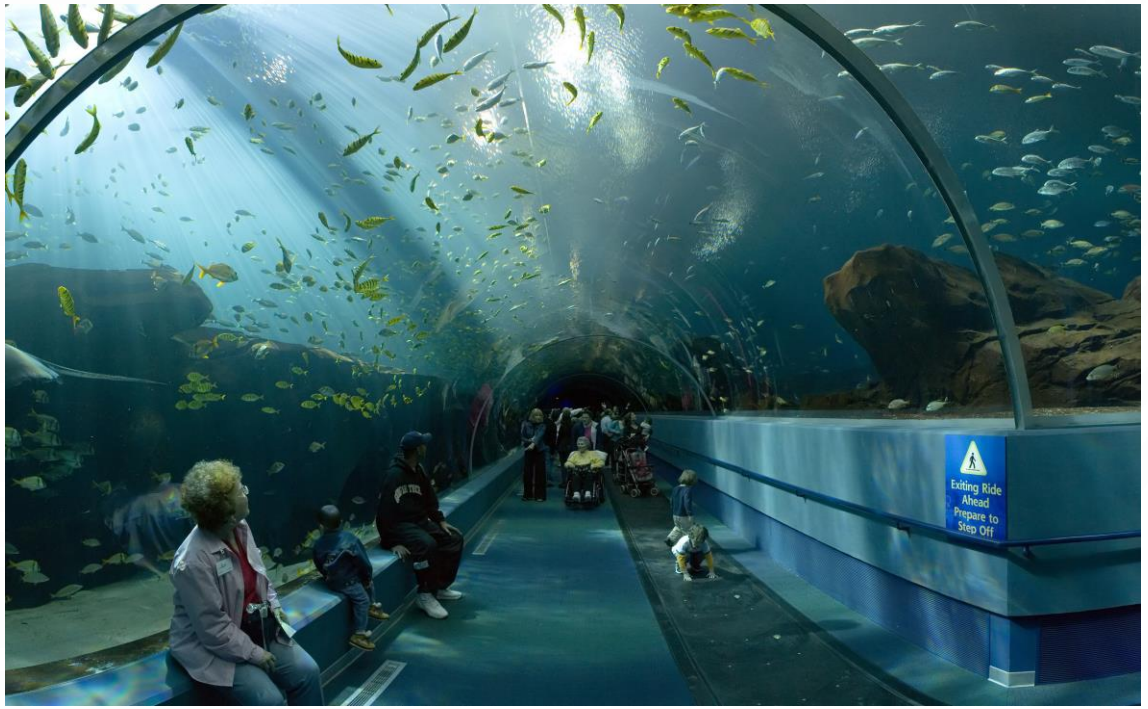
- **Object.** Holds a data type value (data and functions); variable name refers to object.
- **Object-oriented programming.** Paradigm of programming (design) where you
  - Create your own data types (set of values and ops on them – **Encapsulation**).
  - Use them in your programs (manipulate objects that hold values).

Data Type	Set of Values	Operations
`Color`	24 bits	`getRed(), brighten()`
`Picture`	2D array of `Color`s	`getPixel(i, j), setPixel(i, j)`
`String`	Sequence of characters	`length(), substring(), compare()`

- This lecture. Use existing data types.
- Next lecture. Create your own data types.

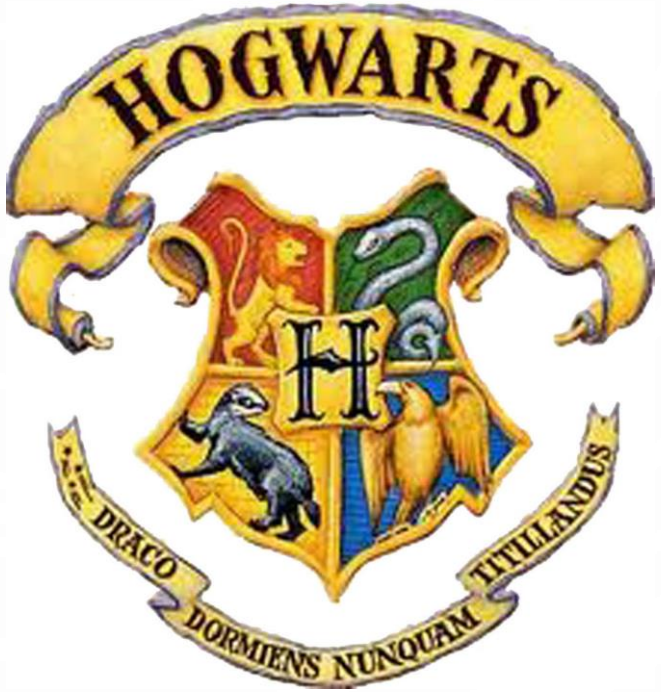
PRACTICE

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



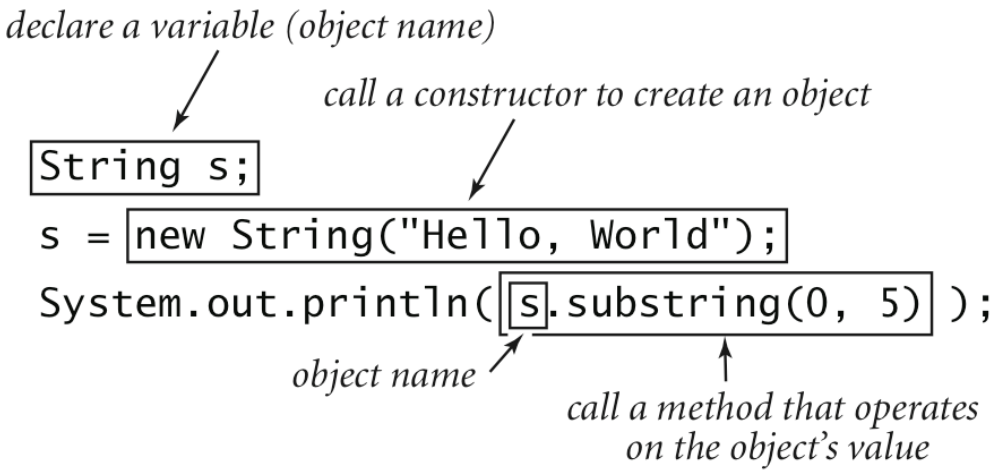
EXERCISE

- 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



CASE STUDY

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.

R	G	B	Color
255	0	0	
0	255	0	
0	0	255	
255	255	255	
0	0	0	
255	0	255	
105	105	105	

COLOR DATA TYPE

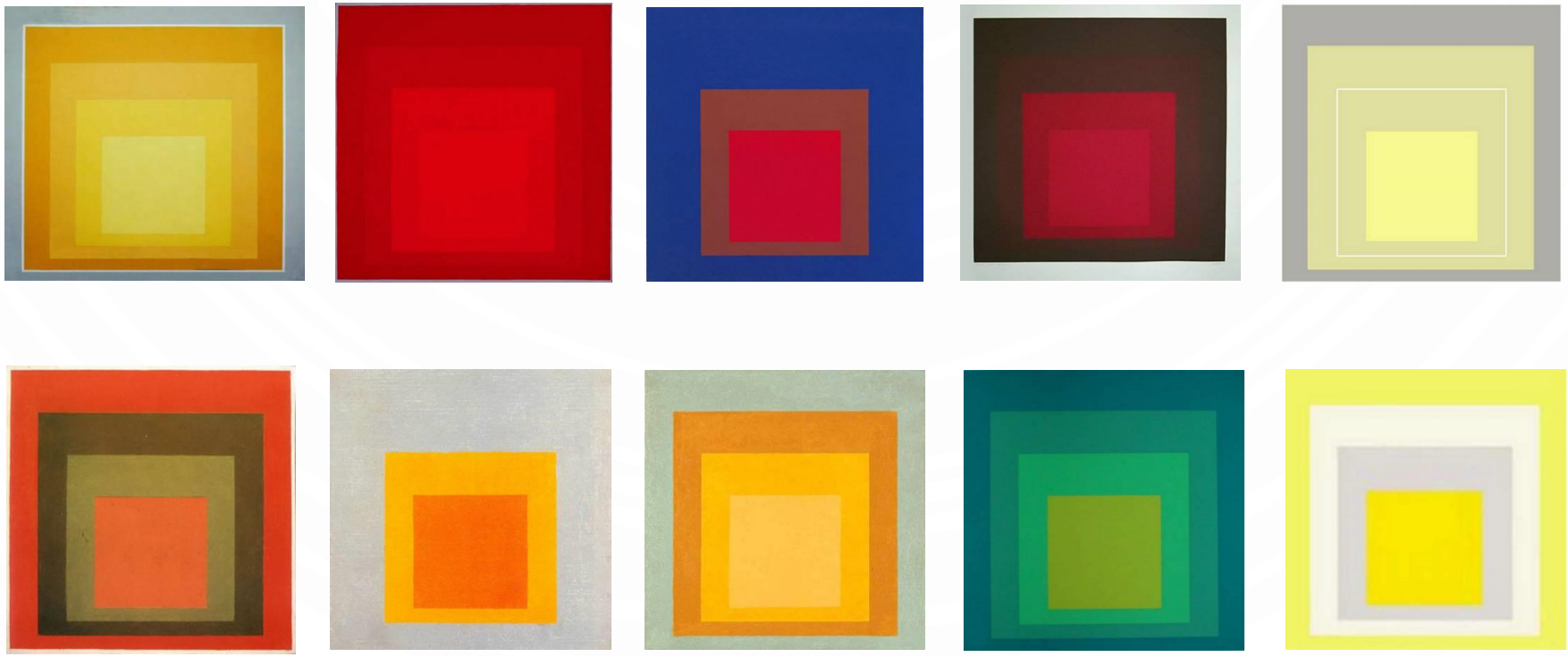
- Color. A sensation in the eye from electromagnetic radiation.
- API. Application Programming Interface.

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)

USING COLORS IN JAVA

```
1.  import java.awt.Color; //Access library
2.  public class AlbersSquares {
3.     public static void main(String[] args) {
4.        //Construct first color
5.        int r1 = Integer.parseInt(args[0]);
6.        int g1 = Integer.parseInt(args[1]);
7.        int b1 = Integer.parseInt(args[2]);
8.        Color c1 = new Color(r1, g1, b1);
9.        //Construct second color
10.       int r2 = Integer.parseInt(args[3]);
11.       int g2 = Integer.parseInt(args[4]);
12.       int b2 = Integer.parseInt(args[5]);
13.       Color c2 = new Color(r2, g2, b2);
14.       //Draw first square
15.       StdDraw.setPenColor(c1);
16.       StdDraw.filledSquare(.25, .5, .2);
17.       StdDraw.setPenColor(c2);
18.       StdDraw.filledSquare(.25, .5, .1);
19.       //Draw second square
20.       StdDraw.setPenColor(c2);
21.       StdDraw.filledSquare(.75, .5, .2);
22.       StdDraw.setPenColor(c1);
23.       StdDraw.filledSquare(.75, .5, .1);
24.    }
25. }
```

```
% java AlbersSquares 9 90 166  100 100 100
```

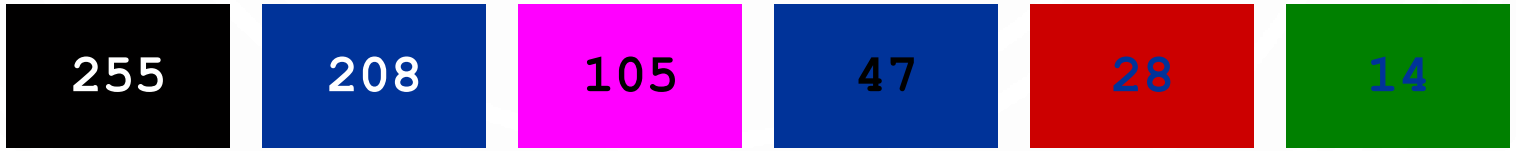
MONOCHROME LUMINANCE

- Monochrome luminance. Effective brightness of a color.
- NTSC formula. $Y = 0.299r + 0.587g + 0.114b$.

```
1. import java.awt.Color;
2. public class Luminance {
3.    public static double lum(Color c) {
4.       int r = c.getRed();
5.       int g = c.getGreen();
6.       int b = c.getBlue();
7.       return .299*r + .587*g + .114*b;
8.    }
9. }
```

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.

```
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.

```
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.

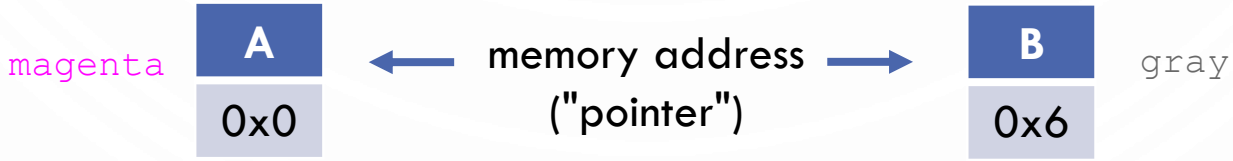
red	*green*	*blue*		
9	90	166	*this color*	
74	74	74	*grayscale version*	
0	0	0	*black*	

$$0.299 * 9 + 0.587 * 90 + 0.114 * 166 = 74.445$$

OBJECTS AND MEMORY

- Possible memory representation. Data sequentially aligned in memory.

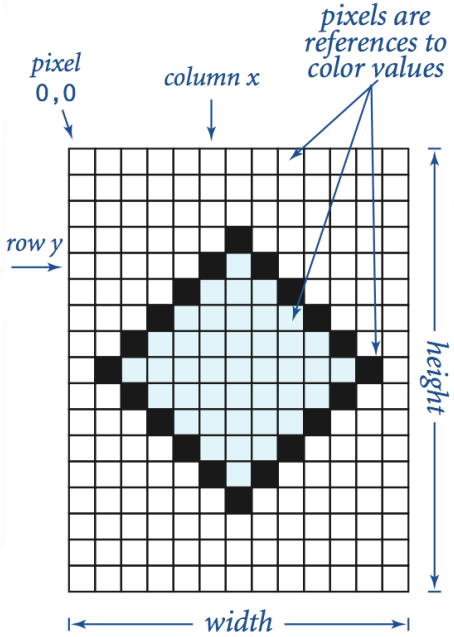
0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8
255	0	255	0	0	0	105	105	105



- 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.



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*

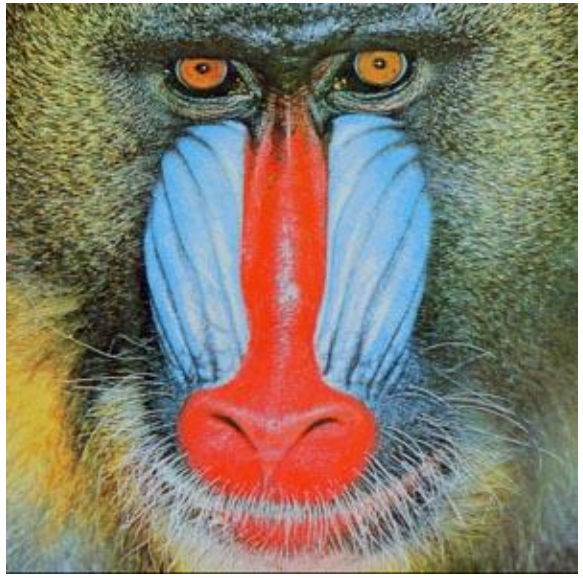
IMAGE PROCESSING: GRAYSCALE FILTER

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

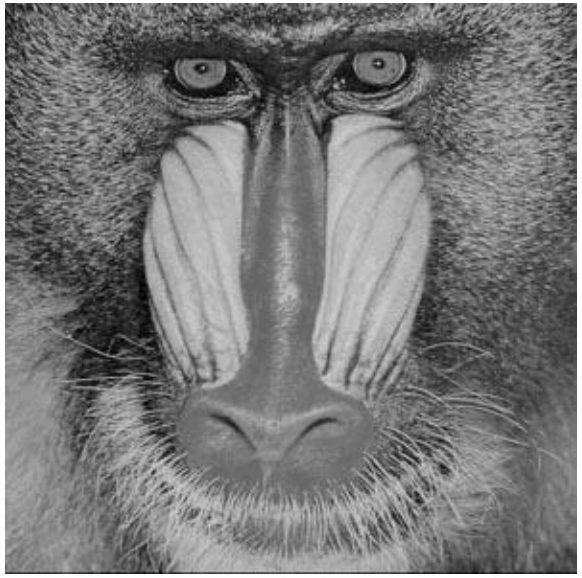
```
1. import java.awt.Color;
2. public class Grayscale {
3.    public static void main(String[] args) {
4.       Picture pic = new Picture(args[0]);
5.       for (int x = 0; x < pic.width(); x++) {
6.          for (int y = 0; y < pic.height(); y++) {
7.             Color color = pic.get(x, y);
8.  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

IMAGE PROCESSING: GRAYSCALE FILTER

mandrill.Jpg

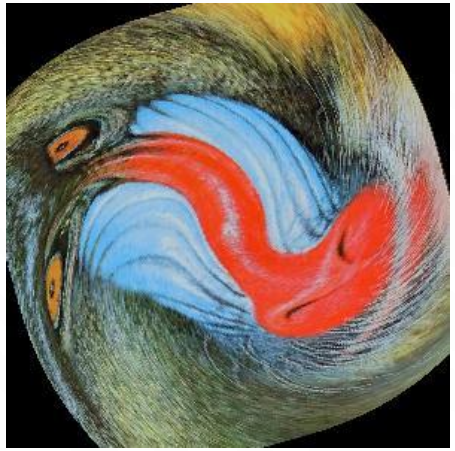


```
% java grayscale mandrill.Jpg
```



MORE IMAGE PROCESSING EFFECTS

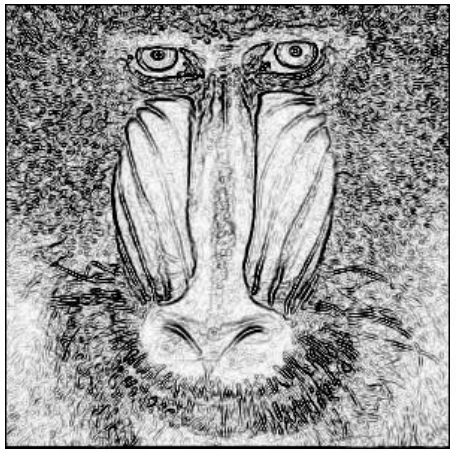
RGB color separation



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)	*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* a*s to* b*s*
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

is the string a palindrome?

```
public static boolean isPalindrome(String s)
{
   int N = s.length();
   for (int i = 0; i < N/2; i++)
      if (s.charAt(i) != s.charAt(N-1-i))
         return false;
   return true;
}
```

extract file name and extension from a command-line argument

```
String s = args[0];
int dot = s.indexOf(".");
String base      = s.substring(0, dot);
String extension = s.substring(dot + 1, s.length());
```

print all lines in standard input that contain a string specified on the command line

```
String query = args[0];
while (!StdIn.isEmpty())
{
   String s = StdIn.readLine();
   if (s.contains(query)) StdOut.println(s);
}
```

print all the hyperlinks (to educational institutions) in the text file on standard input

```
while (!StdIn.isEmpty())
{
   String s = StdIn.readString();
   if (s.startsWith("http://") && s.endsWith(".edu"))
      StdOut.println(s);
}
```

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.