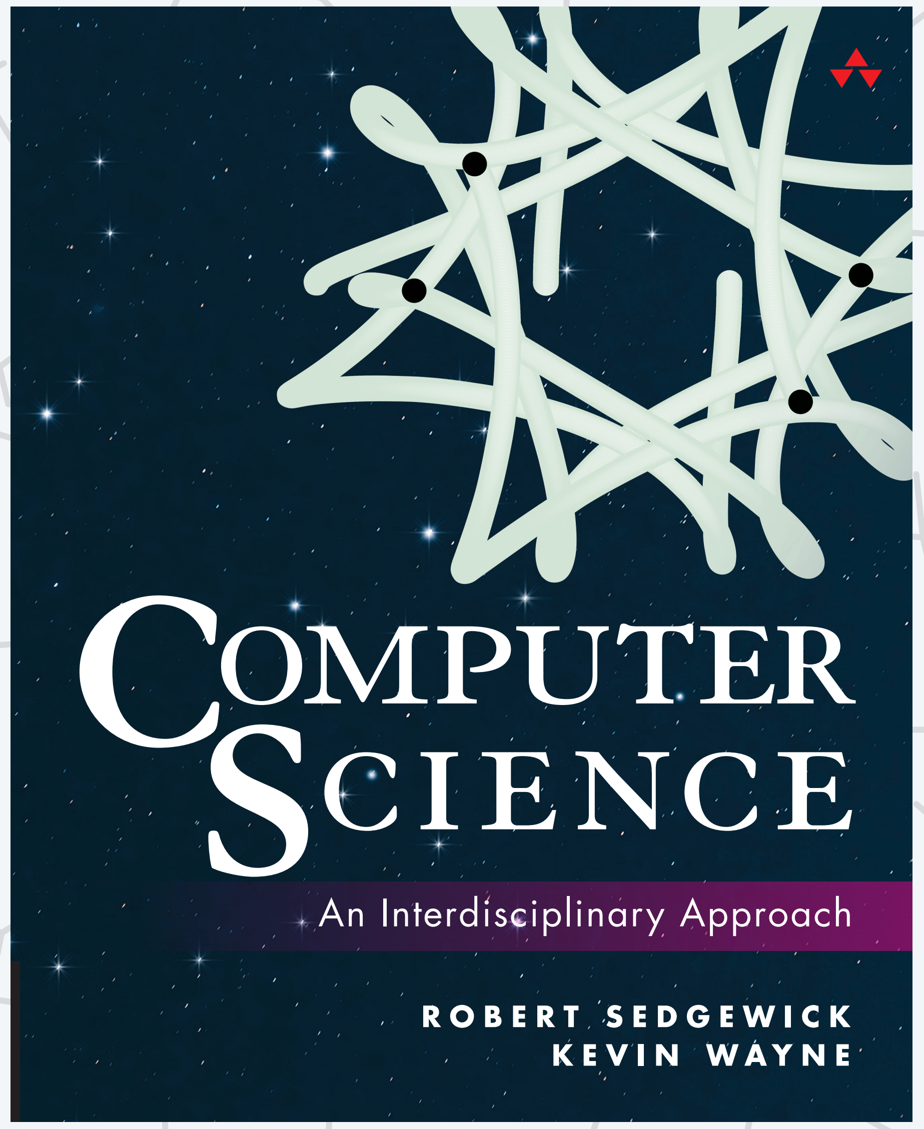
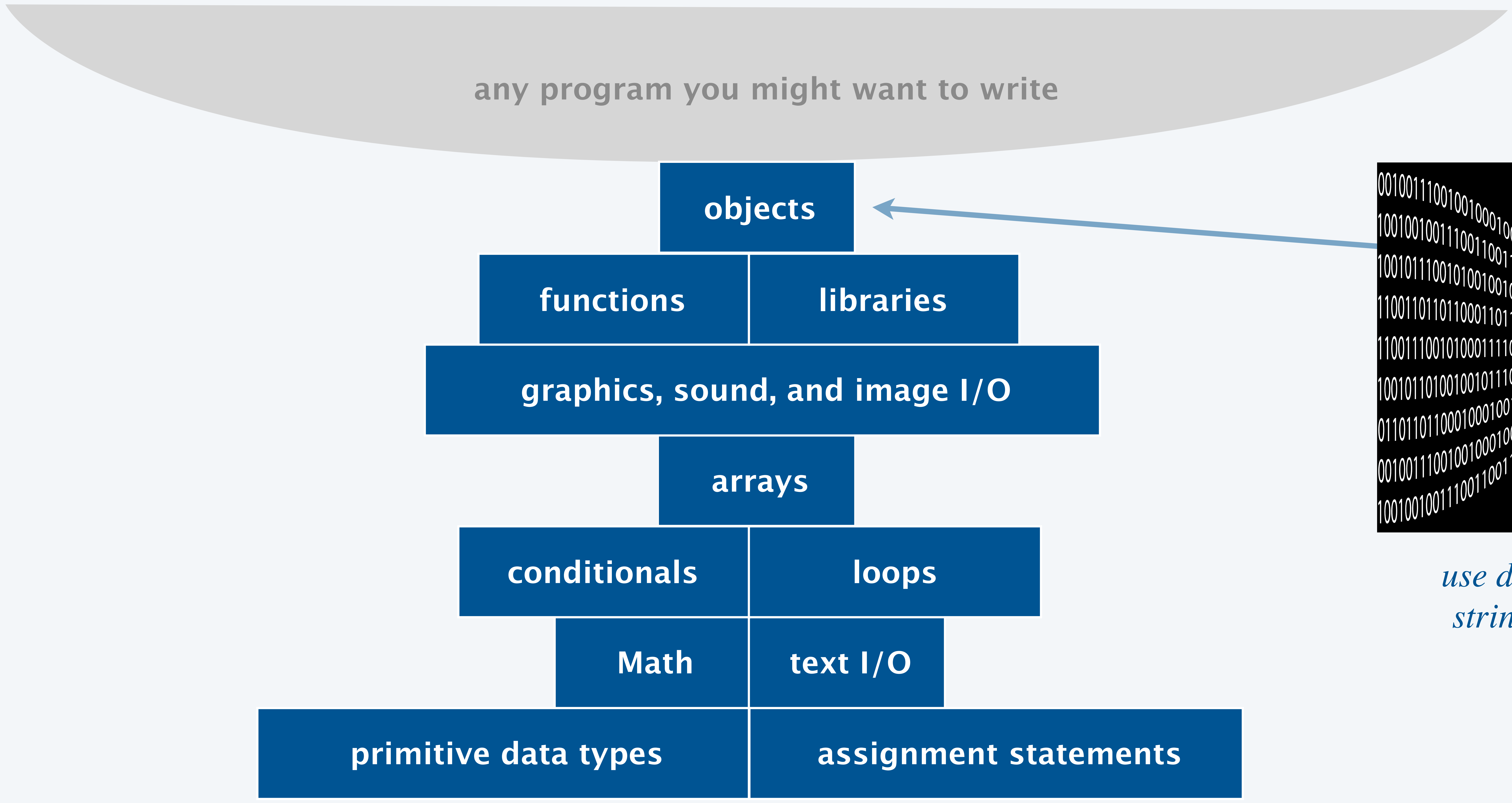


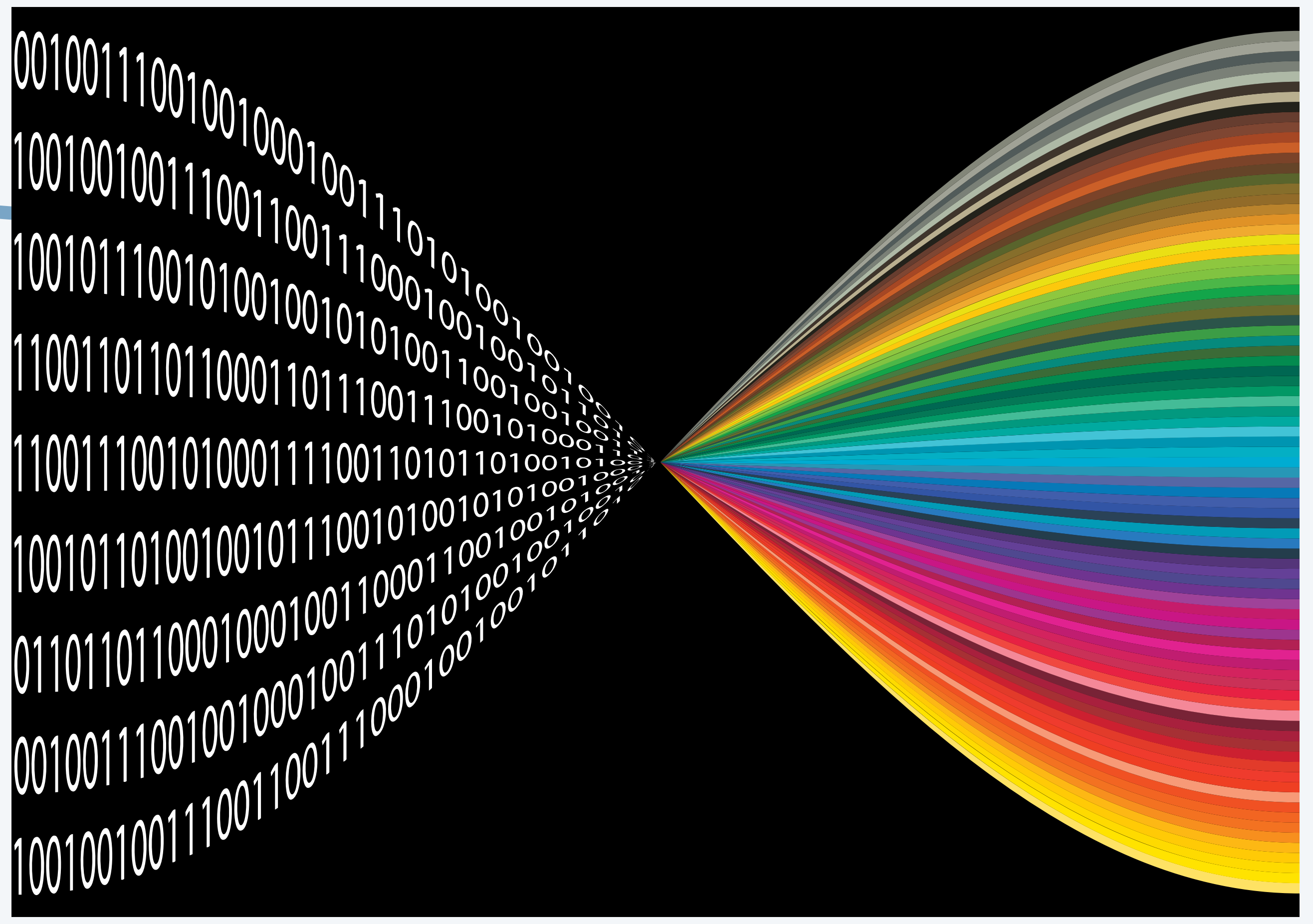
https://introcs.cs.princeton.edu

3.1 Using Data Types

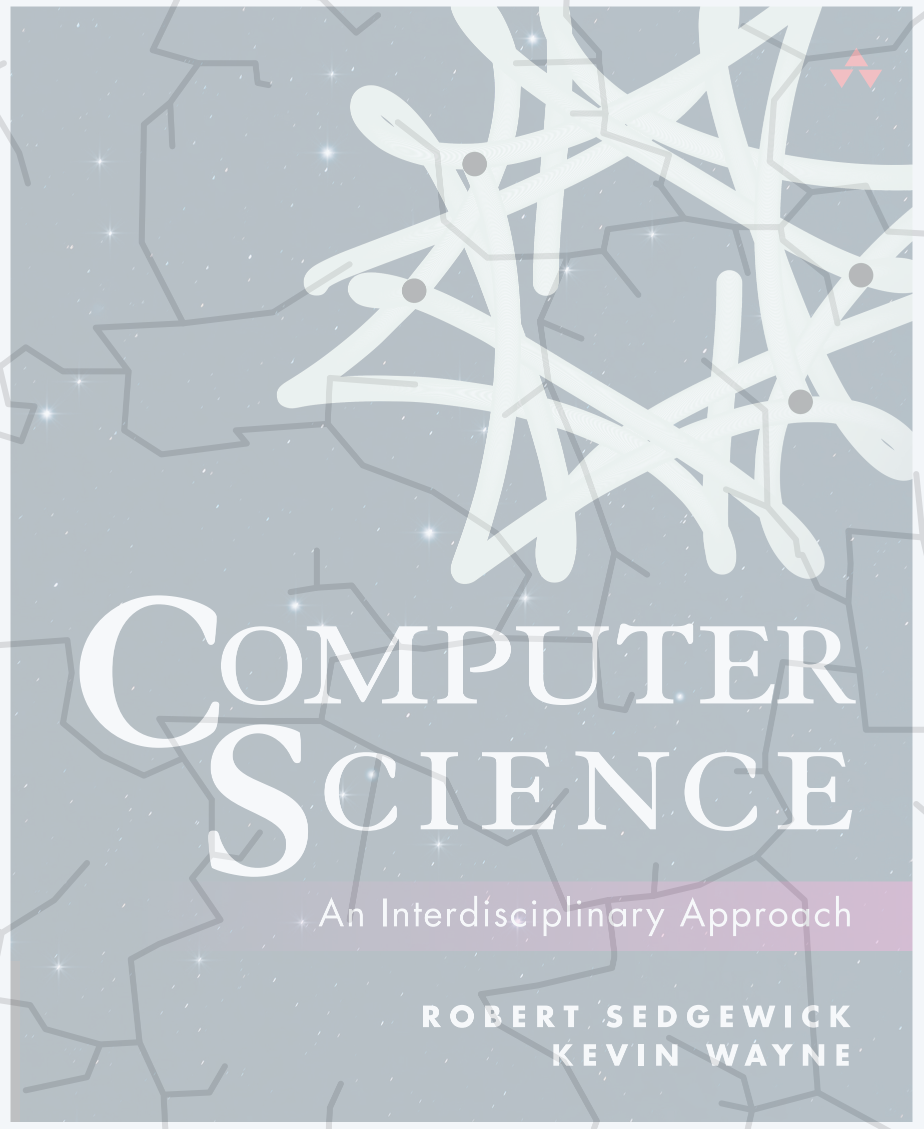
- *overview*
- *string processing*
- *color*
- *image processing*

Basic building blocks for programming





use data types that represent strings, colors, pictures, …



https://introcs.cs.princeton.edu

3.1 Using Data Types

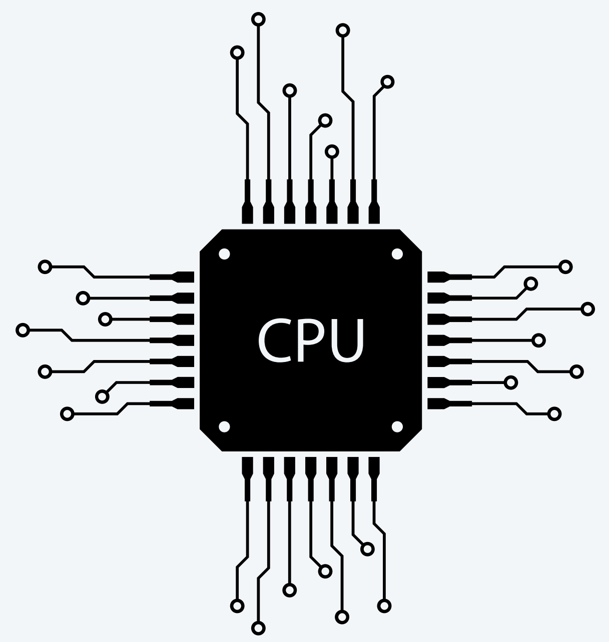
- ‣ *overview*
- ‣ *string processing*
- ‣ *color*
- ‣ *image processing*

Primitive data types

A data type is a set of values and a set of operations on those values.

Primitive types.

- Values map directly to machine representations.
- Operations map directly to machine instructions.



primitive type	set of values	example values	operations

`int`	*integers*	`17`
		`-12345`
			add, subtract, multiply, divide, …

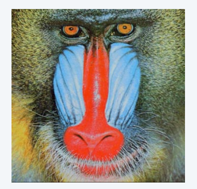
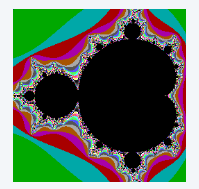
`double`	*floating-point numbers*	`2.5`
		`-0.125`
			add, subtract, multiply, divide, …

`boolean`	*truth values*	`true` `false`	and, or, not, …
⋮	⋮	⋮	⋮

Reference data types

Goal. Write programs that process other types of data.

- Strings, colors, pictures, …
- Points, circles, complex numbers, vectors, matrices, …
- GUIs, database connections, neural networks, plots, …

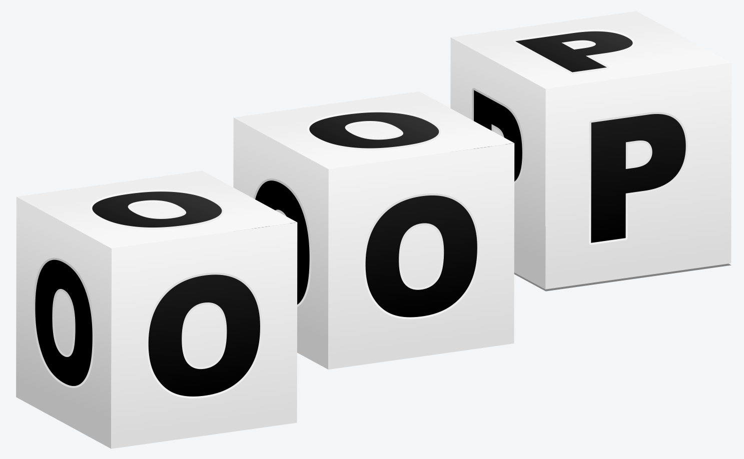
reference type	set of values	example values	operations	source	logo
`String`	*sequences of characters*	`"Hello, World"` `"I ❤️ COS 126"`	length, concatenate, compare, extract substring, search, …	Java language	
`Color`	*three 8-bit integers*	🟥 🟦 ◼	get RGB component, brighter, darker, …	Java library	
`Picture`	*2D array of colors*	 	get/set color of pixel, width, height, show, save, …	textbook library	
⋮	⋮	⋮	⋮		

Object-oriented programming (OOP)

Goal. Write programs that process other types of data.

- Strings, colors, pictures, …
- Points, circles, complex numbers, vectors, matrices, …
- GUIs, database connections, neural networks, plots, …

OOP empowers you to do this (and more!)



Object. An entity that combines a data-type value and associated operations.

- State: value from its data type.
- Behavior: the associated operations.
- Identity: unique identifier (e.g. memory address or “object reference”).

This lecture. Create and use objects from pre-existing data types.

Next lecture. Develop your own data types.

Using data types: quiz 1

Which reference data types have we encountered in this course so far?

A. Arrays.

B. Strings.

C. Both A and B.

D. Neither A nor B.

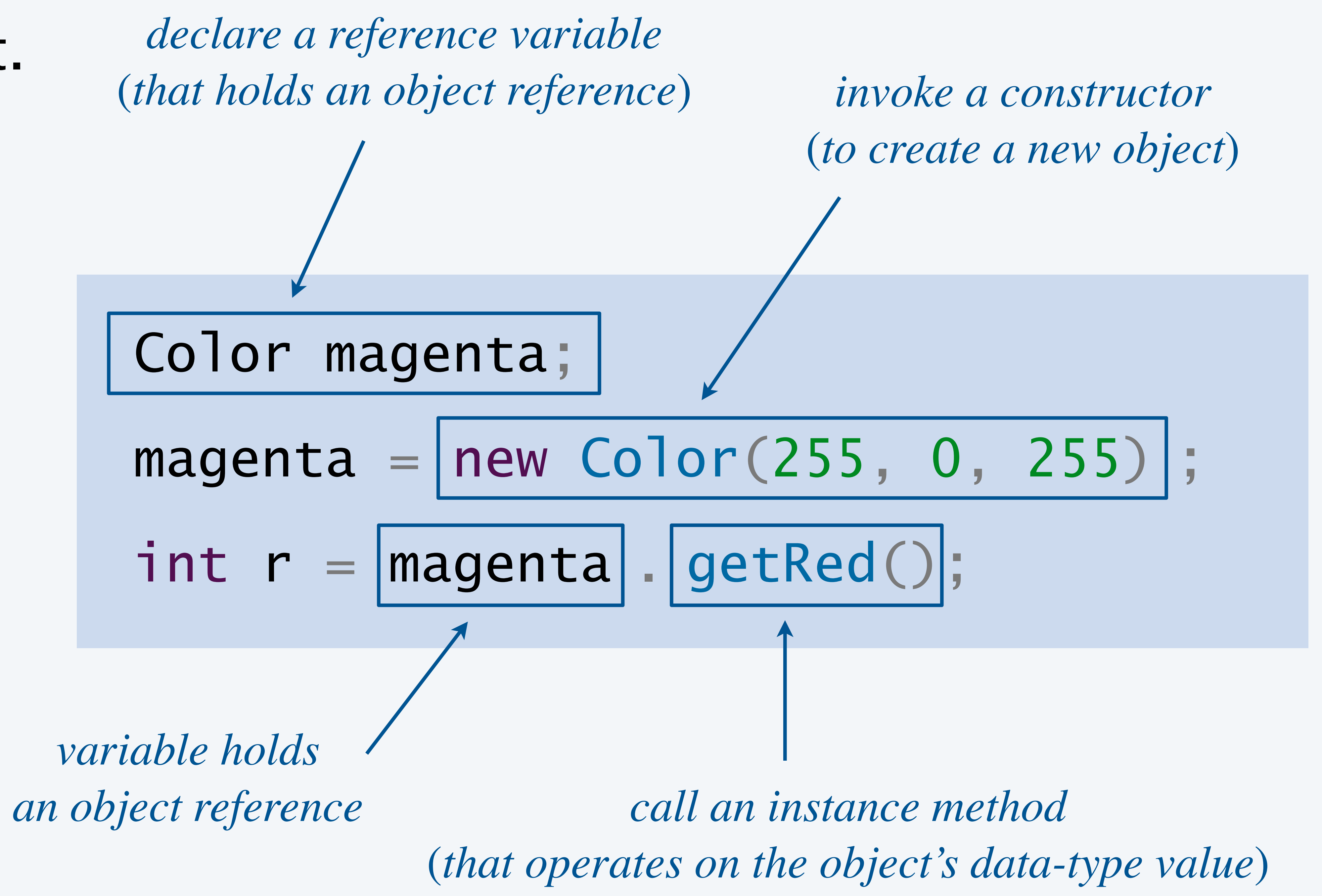
Using a reference data type: constructors and instance methods

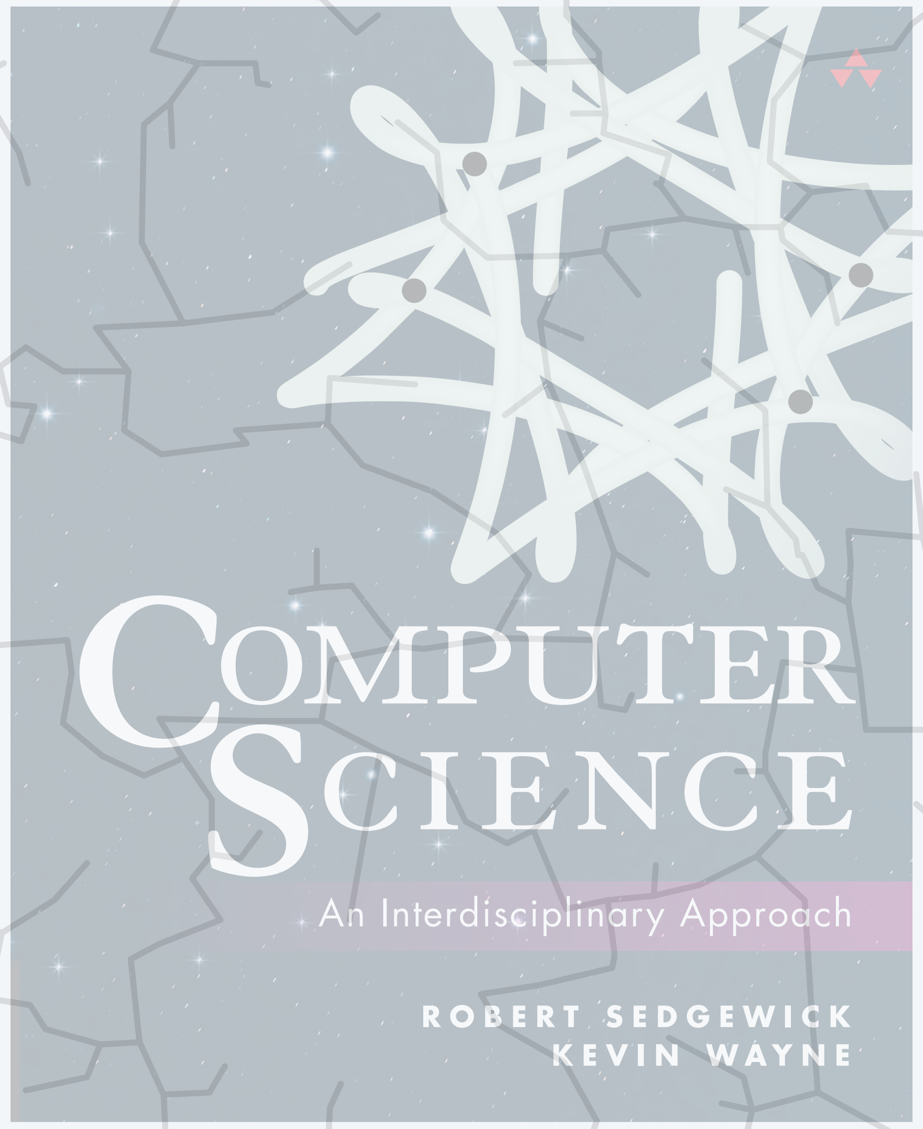
To construct a new object:

- Use the keyword *new* to invoke a constructor.
- Use data-type name to specify type of object to construct.
- Include any arguments.

To apply a data-type operation to a given object:

- Use an object reference to specify which object.
- Use the dot operator.
- Use a method name to specify which operation.
- Include any arguments.


```
Color magenta;
magenta = new Color(255, 0, 255);
int r = magenta.getRed();
```
declare a reference variable (that holds an object reference)
invoke a constructor (to create a new object)
variable holds an object reference
call an instance method (that operates on the object's data-type value)



https://introcs.cs.princeton.edu

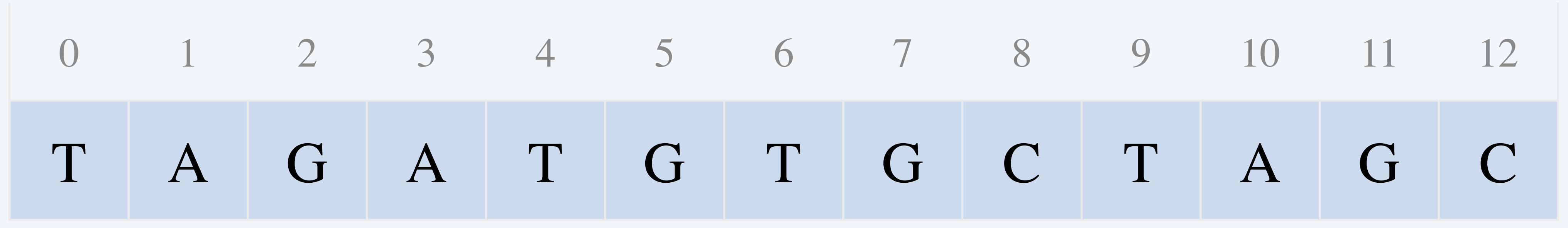
3.1 Using Data Types

- ‣ *overview*
- ‣ *string processing*
- ‣ *color*
- ‣ *image processing*

The *String* and *char* data types

A character is an individual letter, number, or symbol.

A string is a sequence of characters.



a DNA string

Important fundamental abstraction.

- Programming systems (e.g., Java code).
- Communication systems (e.g., text messages).
- Genomic sequences.
- …

type	set of values	example values	operations
`char`	*characters*	`'A'` `'B'` `'C'` `'6'` `'!'` `'ǎ'`	compare
`String`	*sequences of characters*	`"Hello, World"` `"Nǐ hǎo"`	length, concatenate, compare, extract substring, search, …

Note. Java provides special syntax for creating *String* objects. ⟵ *string literals and + operator (instead of* `new`*)*

String API

String data type. Java includes a *String* data type for manipulating strings.

`public class String`	**description**
`String(char[] values)`	*create new string from character array* ← seldom used
`int length()`	*length of string*
`char charAt(int i)`	*character at index* `i`
`boolean startsWith(String pre)`	*does string start with* `pre` *?*
`boolean endsWith(String post)`	*does string end with* `post` *?*
`boolean equals(Object obj)`	*do two strings correspond to same sequence of characters?*
`int indexOf(String t)`	*index of first occurrence of* `t`
`int lastIndexOf(String t)`	*index of last occurrence of* `t`
`String concat(String t)`	*concatenation of this string and* `t` ← typically use + operator instead
`String substring(int i, int j)`	*substring containing characters at indices* `i` *through* `j-1`
`String replace(char from, char to)`	*replace all occurrence of character* `from` *with* `to`
⋮	⋮

creates and returns a new `String` *(does not mutate existing string)*

Examples of *String* operations

Java	expression value	explanation

`String s = "PRINCETON";`
`String t = "TIGERS";`
	–	*string literals*

`s.length()`	`9`	*call an instance method*
`s.charAt(1)`	`'R'`	*0-based indexing*
`s.substring(0, 6)`	`"PRINCE"`	*left inclusive,* *right exclusive*
`s.length() <= t.length()`	`false`	*dot operator has higher precedence than* *arithmetic/logic operators*
`s.concat(t).length()`	`15`	*dot operator is* *left-to-right associative*

Examples of using the *String* data type

computation	Java code	examples

is the string a palindrome?
(string equal to its reverse)

```
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;
}
```

are two characters different?

yes	no
"noon"	"126"
"ACTATCA"	"ACTA"

convert DNA to mRNA
(replace base 'T' with 'U')

```
public static String transcribe(String dna) {
   String rna = dna.replace('T', 'U');
   return rna;
}
```

DNA	mRNA
"ACTG"	"ACUG"
"TTTAG"	"UUUAG"

extract base and extension
from filename

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

$$\underbrace{\texttt{arch}}_{\uparrow\ \textit{base}}\texttt{.}\underbrace{\texttt{jpg}}_{\uparrow\ \textit{extension}}$$

Using data types: quiz 2

Which is the the result of executing the following code fragment?

A. I*E

B. I*ER

C. TI*ERS

D. TIGERS

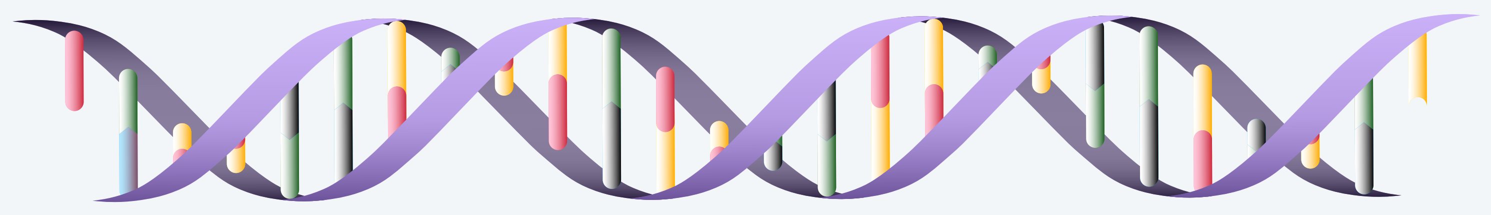
E. Run-time exception

```
String t = "TIGERS";
t.substring(1, 4);
t = t.replace('G', '*');
StdOut.println(t);
```

Identifying a potential gene

Pre-genomics era. Sequence a human genome.

Post-genomics era. Analyze the data and understand structure.



Genomics. Represent genome as a string over A C T G alphabet.

Gene. A substring of genome that represents a functional unit.

- Made up of codons (three A C T G nucleotides).
- Begins with start codon (A T G).
- Ends with a stop codon (T A G , T A A , or T G A).
- No intervening stop codons.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	T	G	C	A	T	A	G	C	G	C	A	T	A	G

start — *no intervening stop codons* — *stop*

DNA sequence	potential gene?
ATGCATAGCGCATAG	*yes*
ATGCGCTGCGTCTGTACTAG	*no*
ATGCCGTGACGTCTGTACTAG	*no*

intervening stop codon

20 nucleotides (not a multiple of 3)

Identifying a potential gene

Goal. Determine whether a given DNA string is a potential gene.

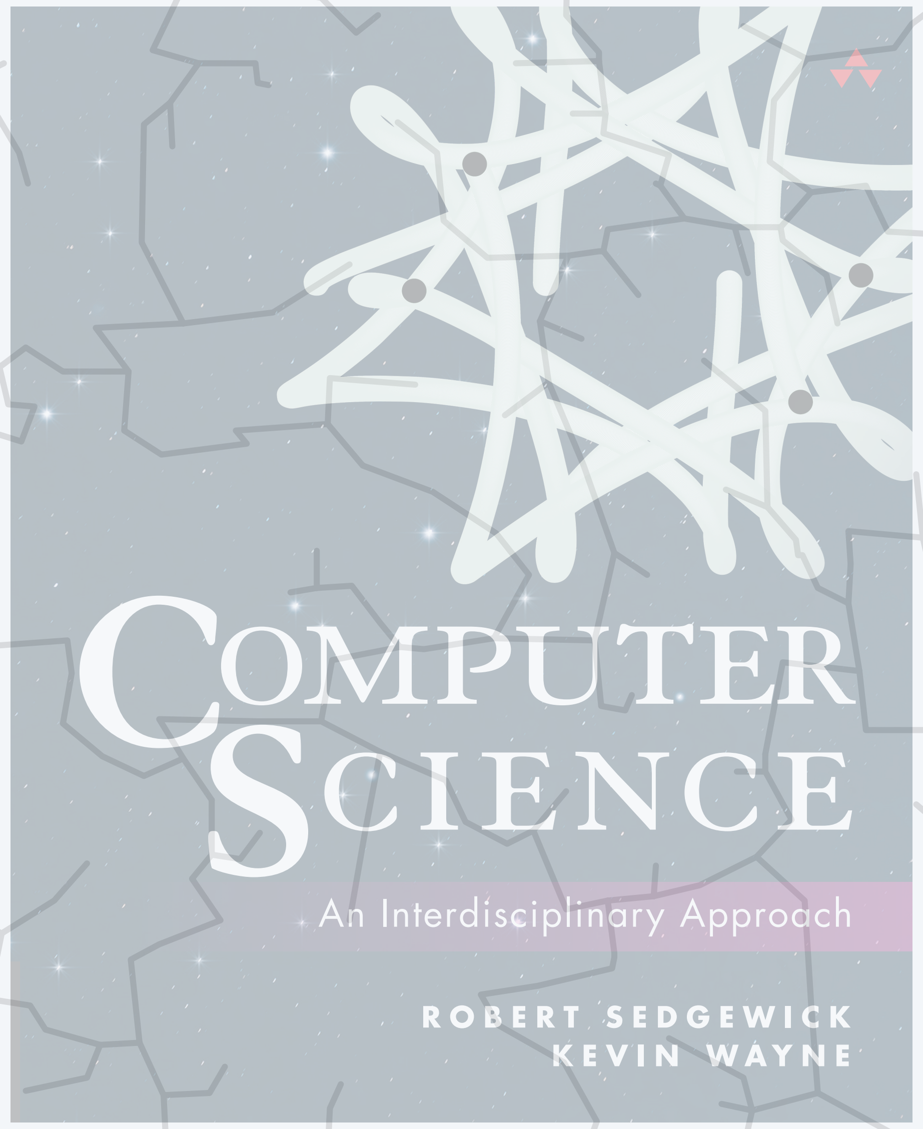
```
public static boolean isPotentialGene(String dna) {

    if (dna.length() % 3 != 0) return false;          ← length is a multiple of 3

    if (!dna.startsWith("ATG")) return false;          ← begins with a start codon

    for (int i = 3; i < dna.length() - 3; i += 3) {
       String codon = dna.substring(i, i+3);
       if (codon.equals("TAA")) return false;
       if (codon.equals("TAG")) return false;         ← no intervening stop codons
       if (codon.equals("TGA")) return false;
    }

    if (dna.endsWith("TAA")) return true;
    if (dna.endsWith("TAG")) return true;             ← ends with a stop codon
    if (dna.endsWith("TGA")) return true;
    return false;
}
```



https://introcs.cs.princeton.edu

3.1 Using Data Types

- overview
- string processing
- color
- image processing

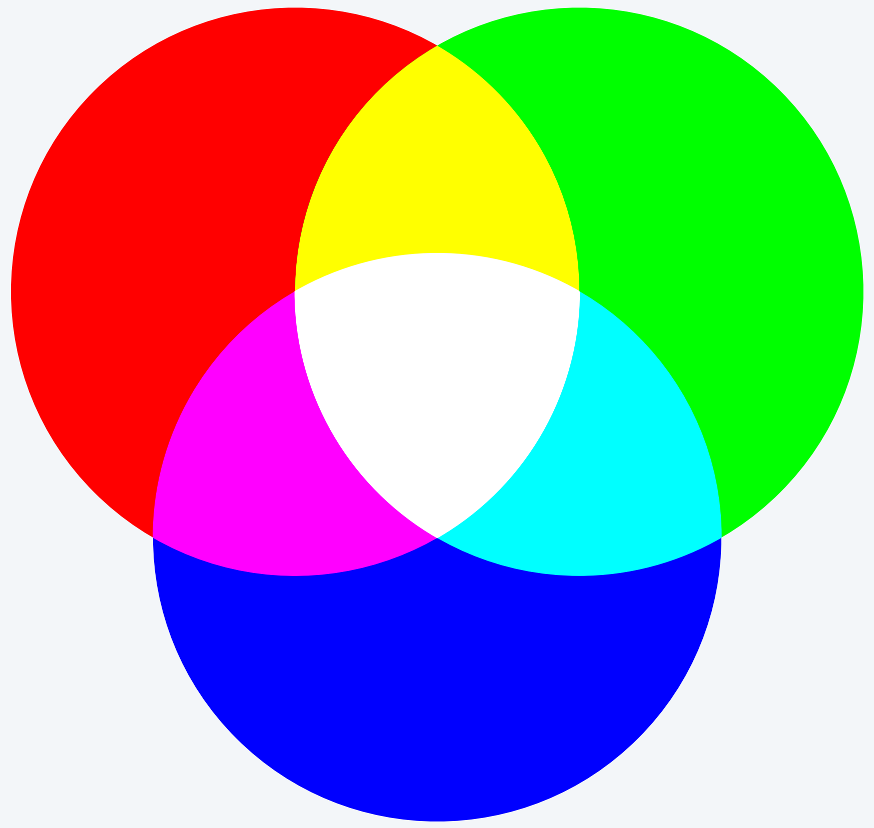
Review: RGB color model

Color is a sensation in the eye from electromagnetic radiation.

RGB color model. Popular format for representing color on digital displays.

- Color is composed of red, green, and blue components.
- Each color component is an integer between 0 to 255.

name	red	green	blue	color
red	255	0	0	🟥
green	0	255	0	🟩
blue	0	0	255	🟦
black	0	0	0	⬛
white	255	255	255	⬜
yellow	255	255	0	🟨
magenta	255	0	255	🟪
cyan	0	255	255	🟦
book blue	0	64	128	🟦



Color API

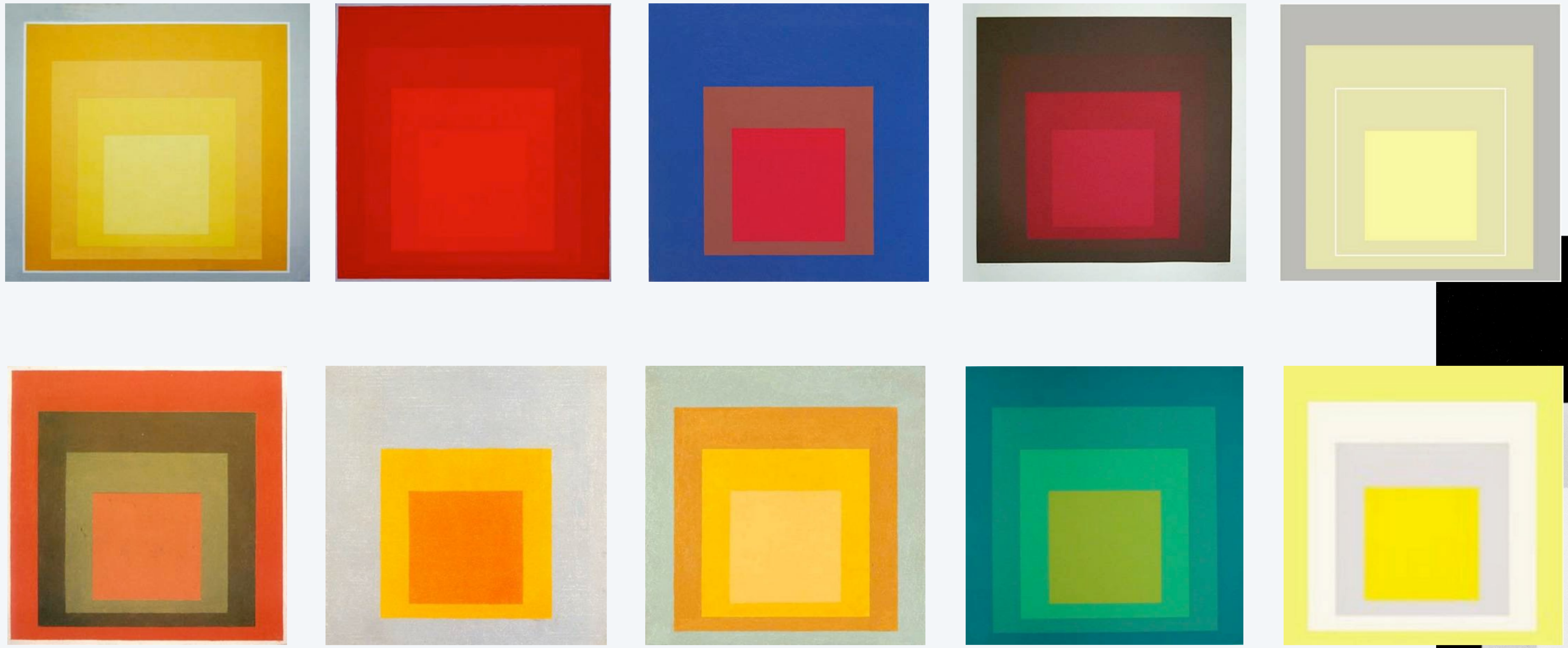
Color data type. Java includes a *`Color`* data type for manipulating colors.

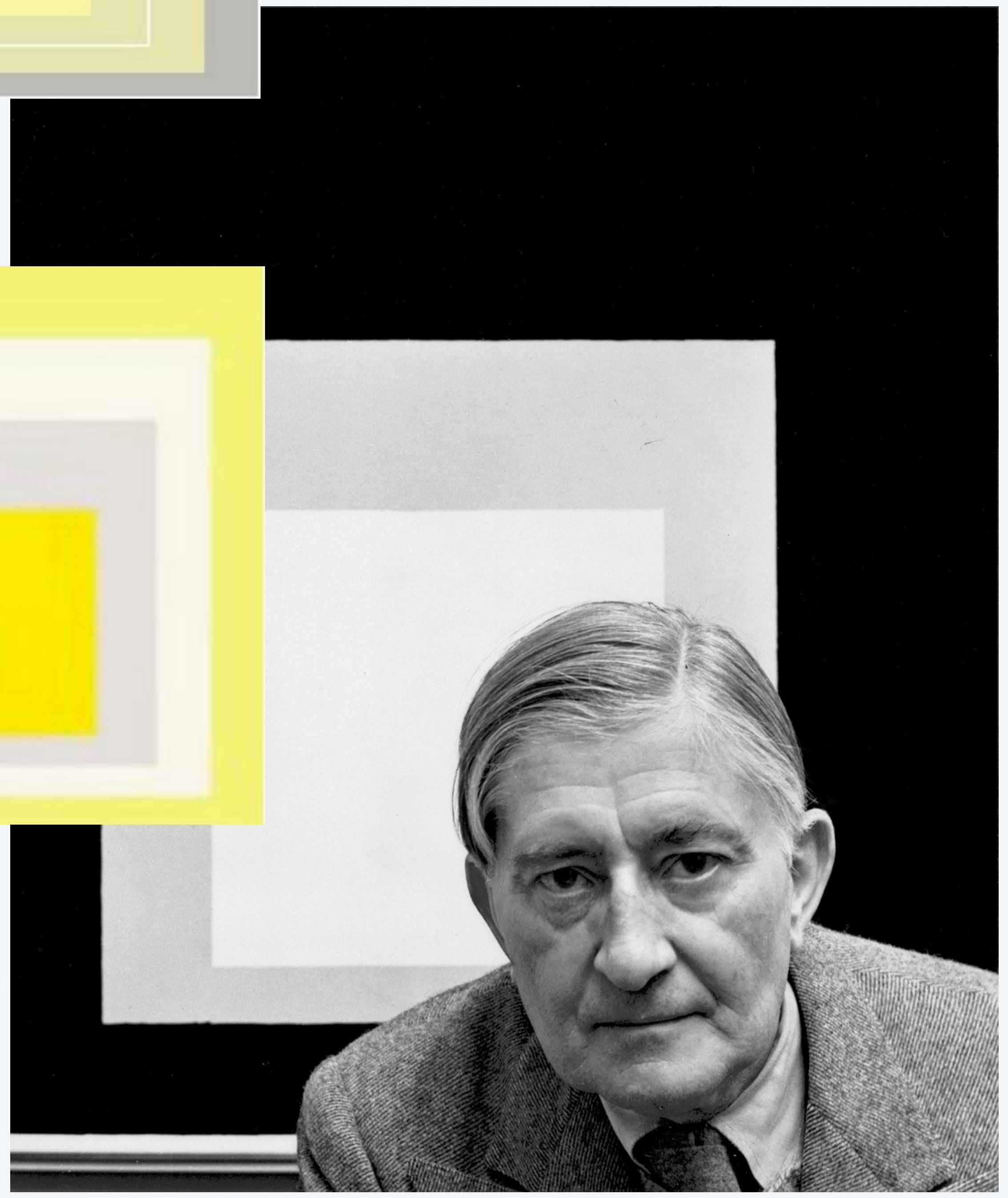
`public class Color`	**description**
`Color(int r, int g, int b)`	*create a new color with given RGB components*
`int getRed()`	*red intensity*
`int getGreen()`	*green intensity*
`int getBlue()`	*blue intensity*
`Color brighter()`	*brighter version of this color*
`Color darker()`	*darker version of this color*
`boolean equals(Object other)`	*do the two color objects correspond to same RGB values?*
`String toString()`	*string representation of this color*
⋮	⋮

Java library. It's located in *`java.awt.Color`*, so you need an *`import`* statement to use.

Albers squares

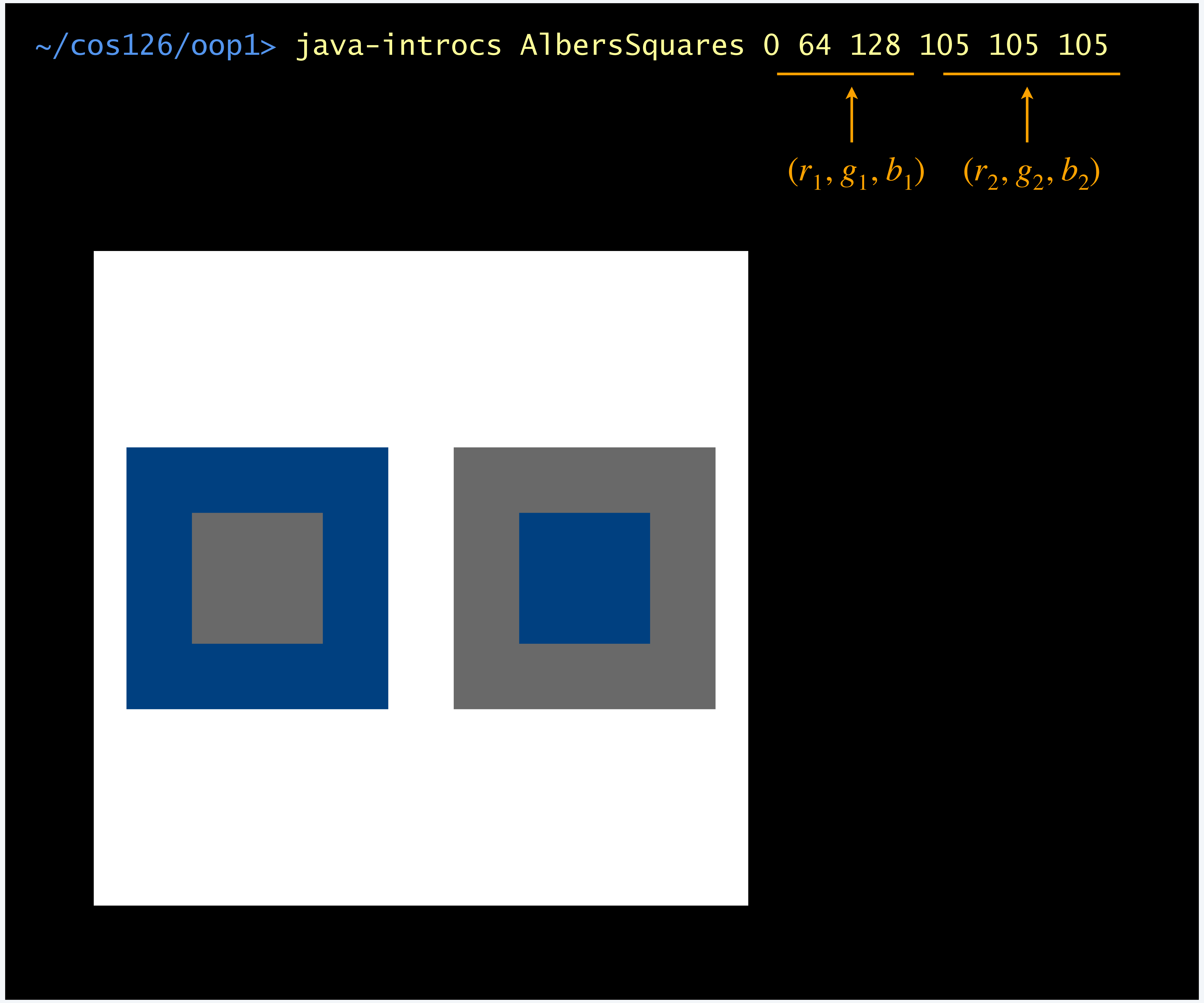
Josef Albers. A 20th century artist who revolutionized the way people think about color.


Homage to the Square (series)


Josef Albers

Albers squares

Goal. Write a Java program to generate Albers squares.



```
~/cos126/oop1> java-introcs AlbersSquares 0 64 128 105 105 105
```

$\underbrace{0\ 64\ 128}_{(r_1, g_1, b_1)}\ \underbrace{105\ 105\ 105}_{(r_2, g_2, b_2)}$

Albers squares implementation

```
import java.awt.Color;

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(0.25, 0.5, 0.2);
      StdDraw.setPenColor(c2);
      StdDraw.filledSquare(0.25, 0.5, 0.1);

      StdDraw.setPenColor(c2);
      StdDraw.filledSquare(0.75, 0.5, 0.2);
      StdDraw.setPenColor(c1);
      StdDraw.filledSquare(0.75, 0.5, 0.1);
   }
}
```

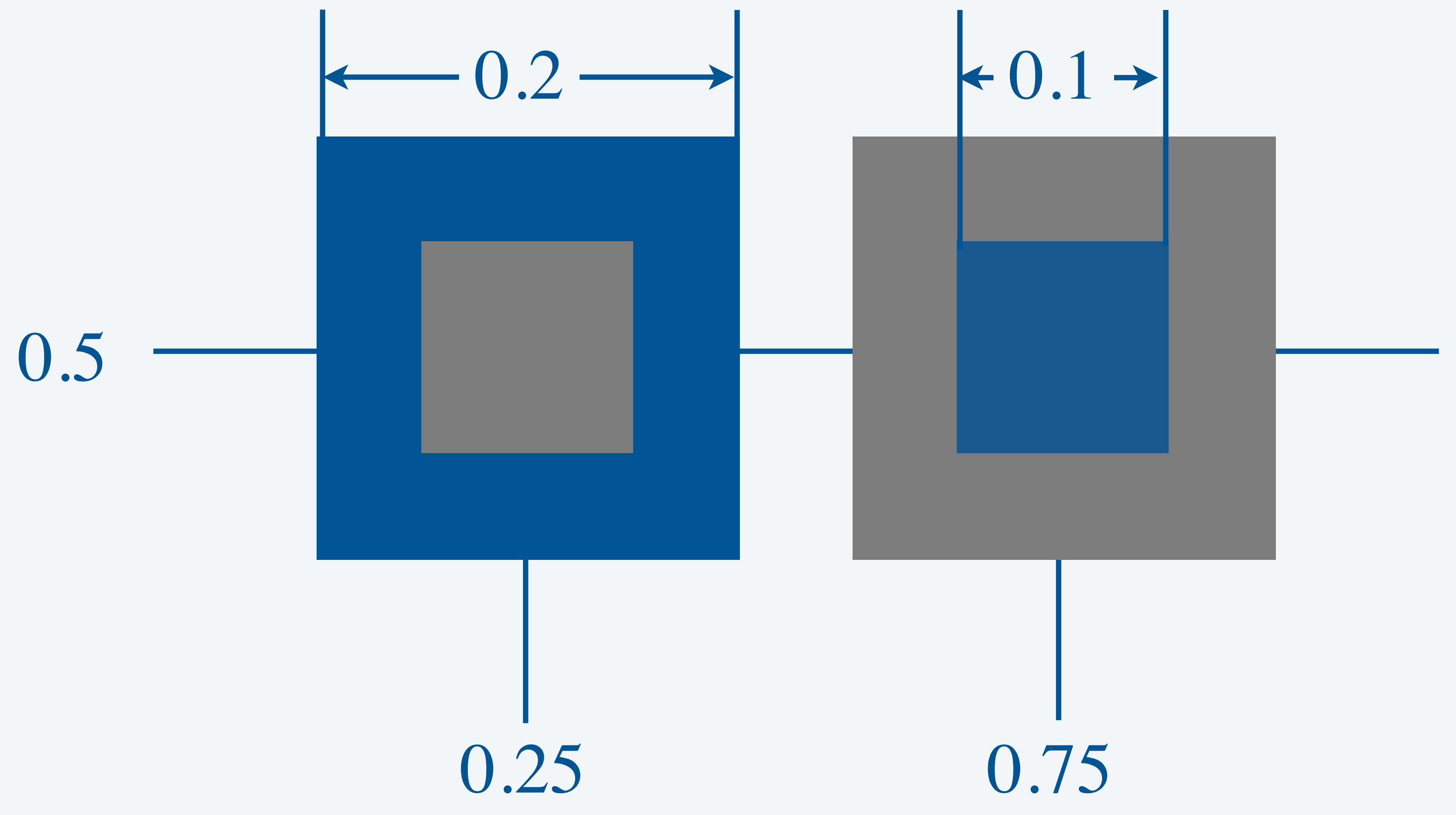
create first `Color` *object*

create second `Color` *object*

pass `Color` *object to* `StdDraw.setPenColor()`

draw first pair of nested squares

draw second pair of nested squares



Monochrome luminance

Def. The luminance of a color quantifies its effective brightness. ← *on a scale of 0 (black) to 255 (white)*

Standard formula. $Y = 0.299\,R + 0.587\,G + 0.114\,B.$ ← *pure green appears lighter than pure blue (so give higher weight)*

```
import java.awt.Color;

public class Luminance {

   public static double intensity(Color color) {
      int r = color.getRed();
      int g = color.getGreen();
      int b = color.getBlue();
      return 0.299*r + 0.587*g + 0.114*b;
   }

   public static void main(String[] args) {
      int r = Integer.parseInt(args[0]);
      int g = Integer.parseInt(args[1]);
      int b = Integer.parseInt(args[2]);
      Color color = new Color(r, g, b);
      StdOut.println(intensity(color));
   }
}
```

function takes a `Color` *object as an argument*

```
~/cos126/oop1> java-introcs Luminance 255 0 0
76.245

~/cos126/oop1> java-introcs Luminance 0 64 128
52.16
```

name	R	G	B	color	lum
red	255	0	0	🟥	76.245
green	0	255	0	🟩	149.685
blue	0	0	255	🟦	29.07
black	0	0	0	⬛	0.0
white	255	255	255	⬜	255.0
book blue	0	64	128	🟦	52.16

Foreground/background color accessibility

Goal. Determine whether text in one color will be readable if background is in another color.

Application. Make web content accessible.

Web standard. Readable if contrast ratio $\dfrac{lum_{max} + 0.05}{lum_{min} + 0.05} \geq 4.5.$

WCAG uses relative luminance, not monochrome luminance

WCAG 2.0 AA
COMPLIANT

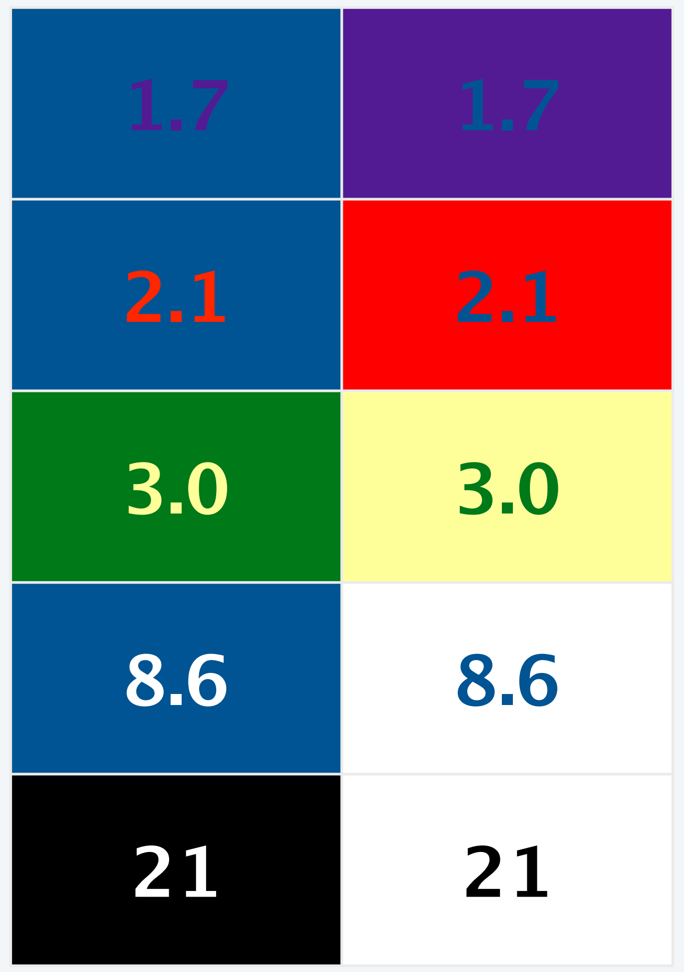
Web Content Accessibility Guidelines

Luminance.java

```java
public static double contrastRatio(Color a, Color b) {
   double min = Math.min(intensity(a), intensity(b)) / 255.0;
   double max = Math.max(intensity(a), intensity(b)) / 255.0;
   return (max + 0.05) / (min + 0.05);
}
```

normalized to be between 0 and 1

```java
public static boolean isAccessible(Color a, Color b) {
   return contrastRatio(a, b) >= 4.5;
}
```



1.7	1.7
2.1	2.1
3.0	3.0
8.6	8.6
21	21

contrast ratios (between 1 and 21)

Grayscale

Goal. Convert color image to grayscale.

- RGB color is gray when $R = G = B$.
- To convert RGB color to grayscale, use luminance for R, G, and B.

Luminance.java

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

round to nearest `int`

name	R	G	B	color	lum	gray
red	255	0	0	🟥	76.245	⬛
green	0	255	0	🟩	149.685	⬛
blue	0	0	255	🟦	29.07	⬛
black	0	0	0	⬛	0.0	⬛
white	255	255	255	⬜	255.0	⬜
book blue	0	64	128	🟦	52.16	⬛

Object references: memory representation

Object reference. Refers to a data-type value; it is not the data-type value.

object reference = unique identifier for object (e.g., memory address)

- Can manipulate the data-type value in the referenced object.
- Can use it to invoke instance methods (with the . operator).
- Can pass it to (or return it from) a method.

```
Color red   = new Color(255, 0, 0);
Color gray  = new Color(105, 105, 105);
Color color = red;
```

the reference variables red, gray, and color store object references

memory address
288
color
red
gray
330
105
105
105
330
330
288
255
0
0
color and red are now "aliases"
object reference (memory address of object)
memory contents (data-type value)

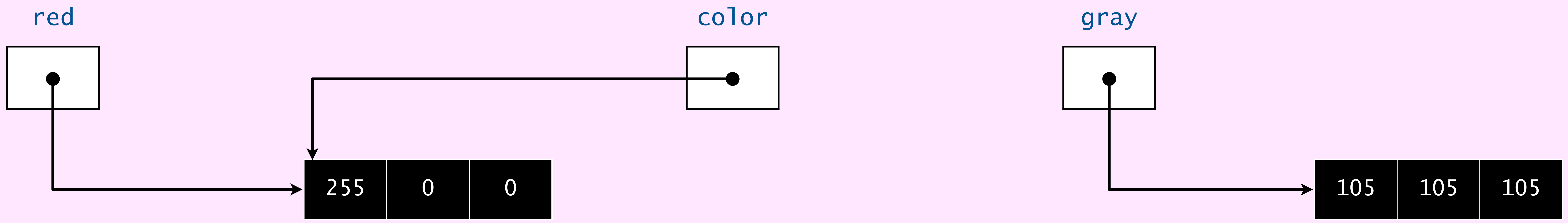
memory representation (using poetic license)

Object references: box-and-pointer diagrams

Box-and-pointer diagram.

- Put each object and reference variable in a box.
- Draw an arrow from each reference variable to the object it references.

```
Color red  = new Color(255, 0, 0);
Color gray = new Color(105, 105, 105);
Color color = red;
```



Using data types: quiz 3

Assume that the variables `red1`, `red2`, and `red3` are initialized as follows.

Which of the following expressions will evaluate to `false` ?

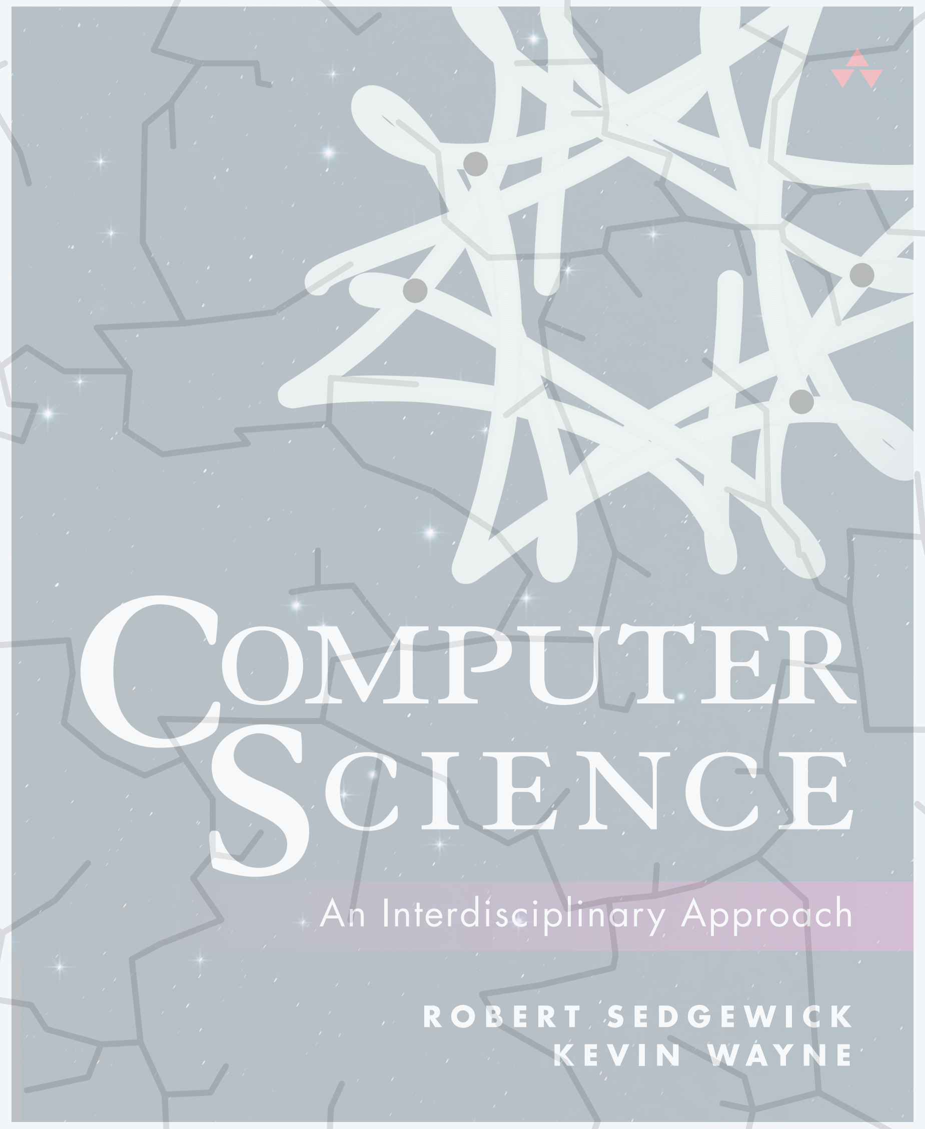
```
Color red1 = new Color(255, 0, 0);
Color red2 = new Color(255, 0, 0);
Color red3 = red1;
```

A. `red1 == red3`

B. `red2 == red3`

C. `red1.equals(red3)`

D. `red2.equals(red3)`



https://introcs.cs.princeton.edu

3.1 Using Data Types

- overview
- string processing
- color
- *image processing*

Input and output data types

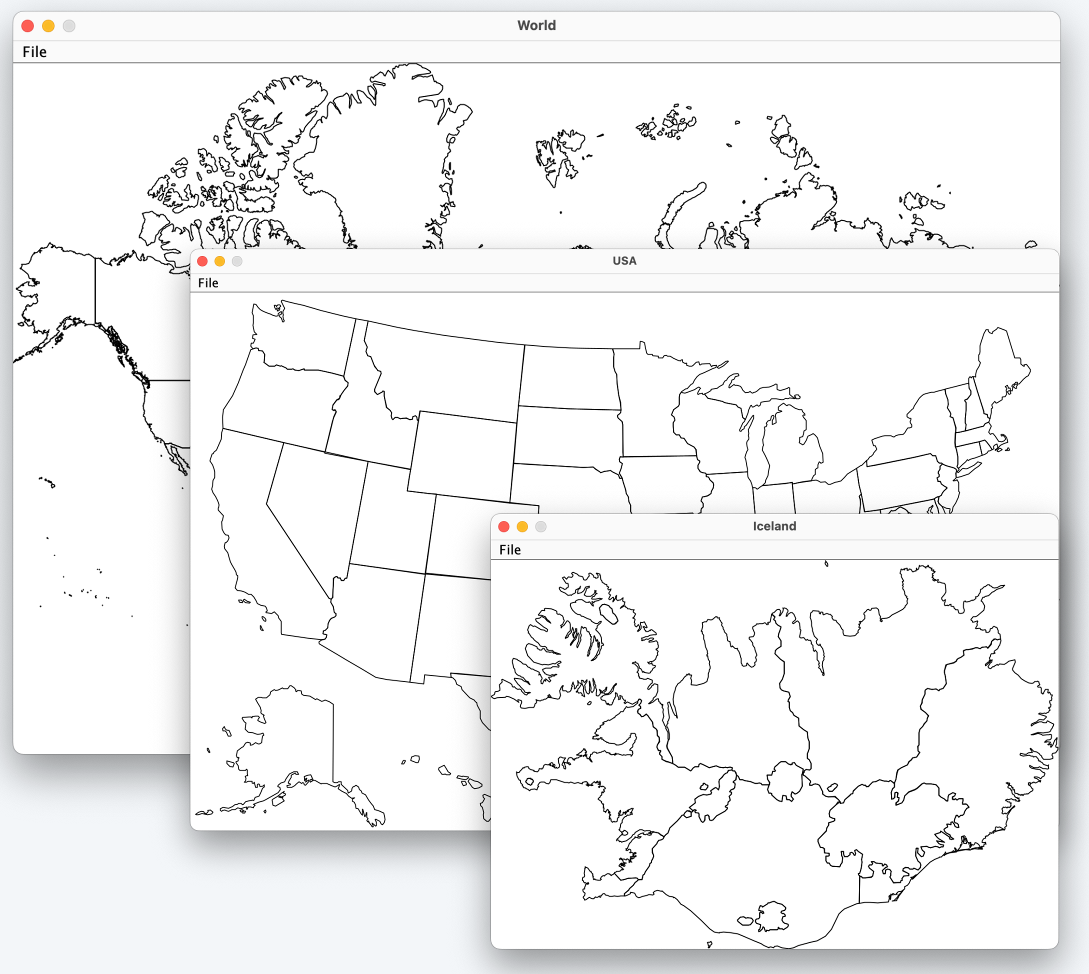
You have used. `StdIn`, `StdOut`, `StdDraw`, and `StdPicture`.

Key limitation. Only one entity per program.

one input stream, output stream, drawing, or picture per program execution

OOP versions. We also provide object-oriented versions. ← *available with* `javac-introcs` *and* `java-introcs` *commands*

data type	enables
`In`	*read from more than one input stream*
`Out`	*write to more than one output stream*
`Draw`	*create more than one drawing*
`Picture`	*process more than one image*



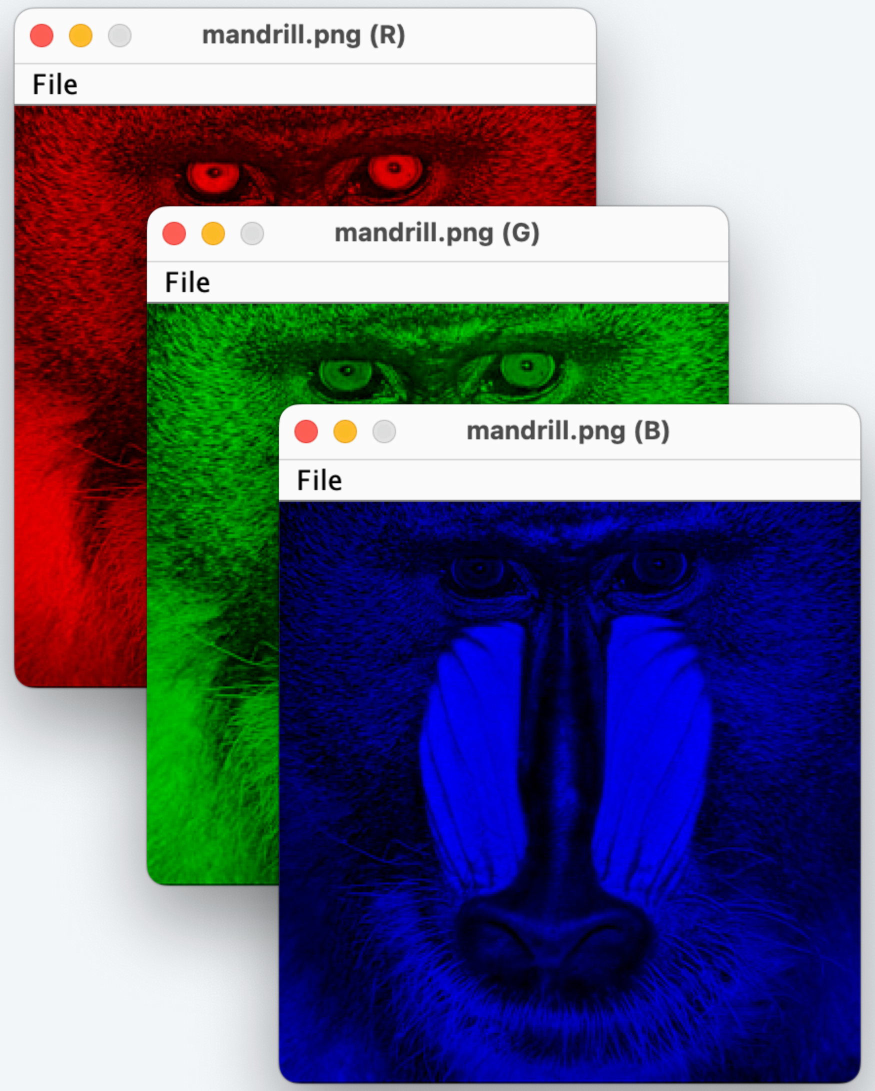


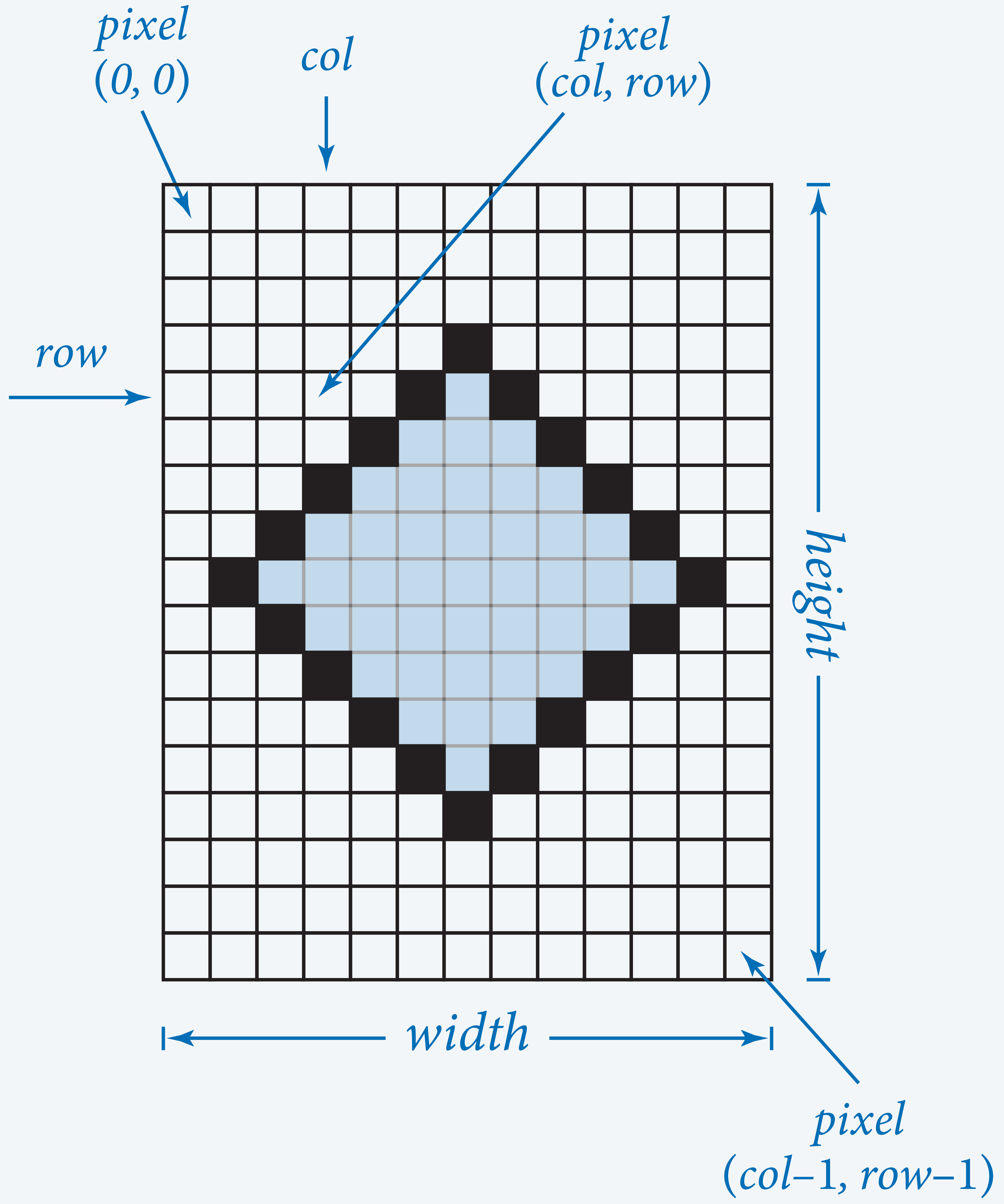
Image processing: review

A picture is a width-by-height grid of pixels; each pixel has an RGB color.

Ex.

mandrill.jpg

arch.jpg



Picture API

Picture data type. Our textbook data type for manipulating digital images.

- Can create many *Picture* objects in same program.
- Uses *Color* objects as arguments and return values.

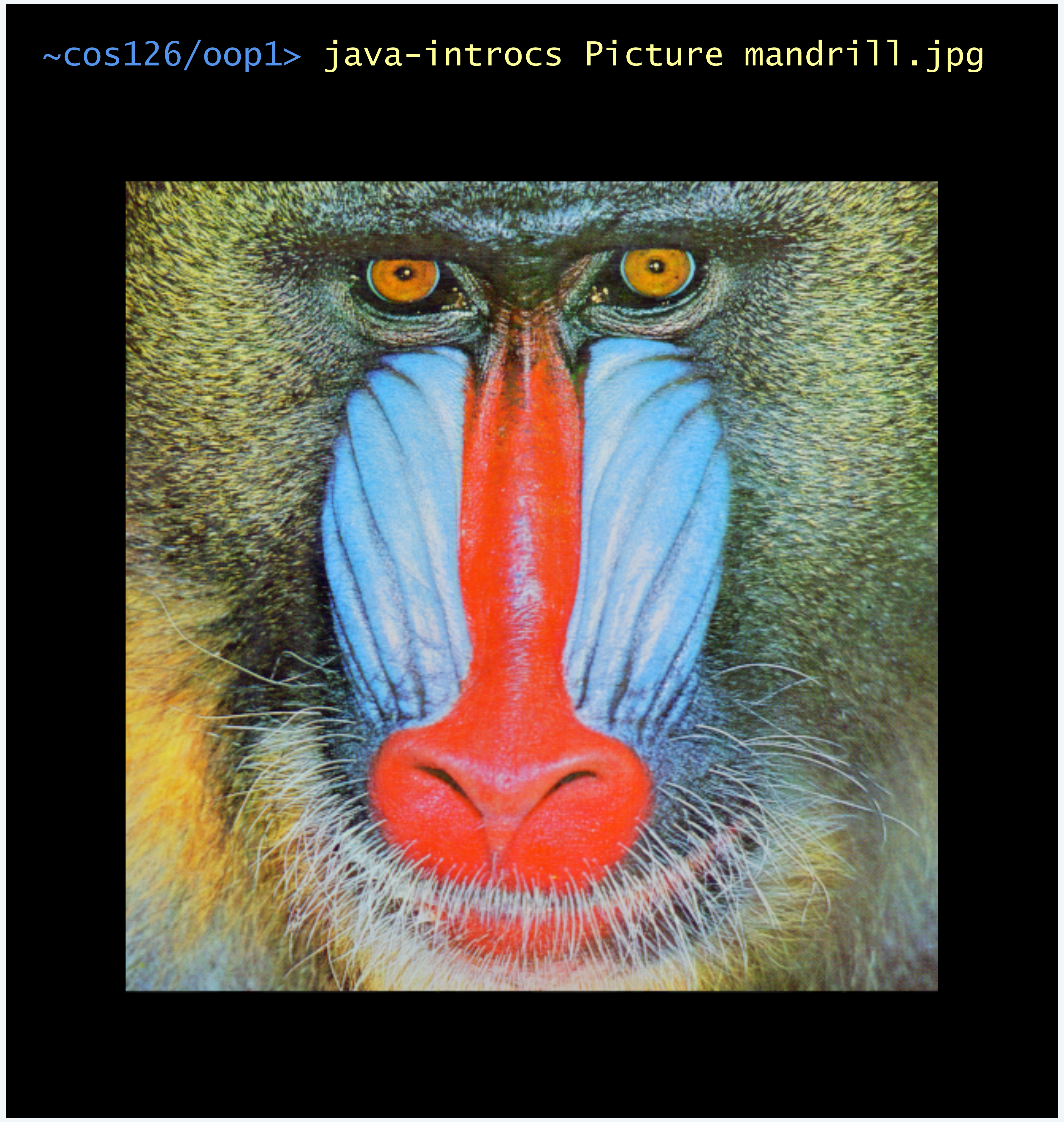
OOP version of `StdPicture` *(with a few important differences)*

`public class Picture`	**description**
`Picture(String filename)`	*create a picture from an image file*
`Picture(int width, int height)`	*create a blank* `width`*-by-*`height` *picture*
`int width()`	*width of the picture*
`int height()`	*height of the picture*
`Color get(int col, int row)`	*the color of pixel* `(col, row)`
`void set(int col, int row, Color color)`	*set the color of pixel* `(col, row)` *to* `color`
`void show()`	*display the image in its own window*
`void save(String filename)`	*save the picture to a file*

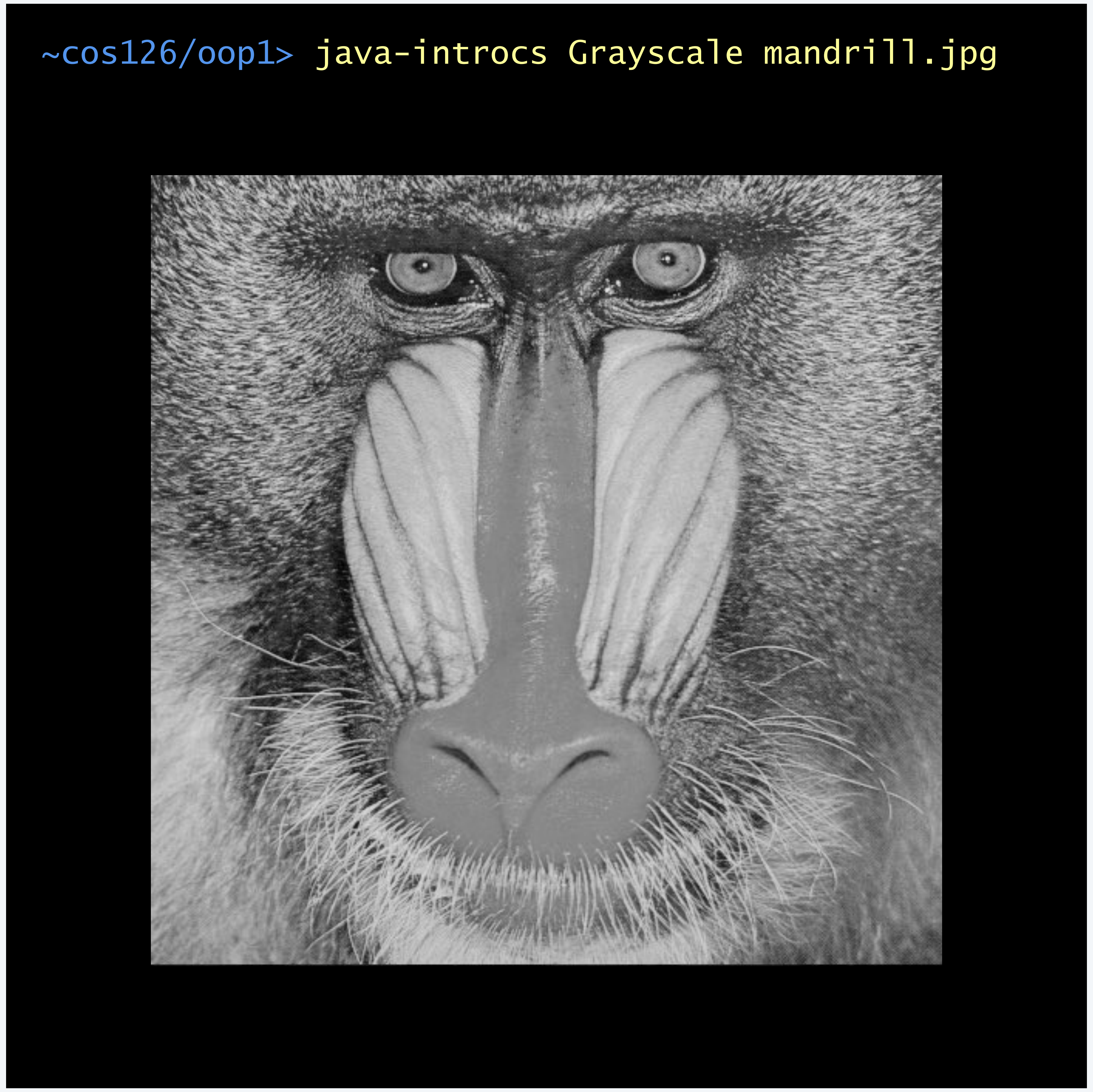
supported file formats: JPEG, PNG, GIF, TIFF, BMP

Grayscale filter

Goal. Write a Java program to convert an image to grayscale.



```
~cos126/oop1> java-introcs Picture mandrill.jpg
```



```
~cos126/oop1> java-introcs Grayscale mandrill.jpg
```

Grayscale filter implementation: object-oriented version

```
import java.awt.Color;

public class Grayscale {
   public static void main(String[] args) {
      Picture picture = new Picture(args[0]);
      for (int col = 0; col < picture.width(); col++) {
         for (int row = 0; row < picture.height(); row++) {
            Color color = picture.get(col, row);
            Color gray  = Luminance.toGray(color);
            picture.set(col, row, gray);
         }
      }
      picture.show();
   }
}
```

create a new picture from image file

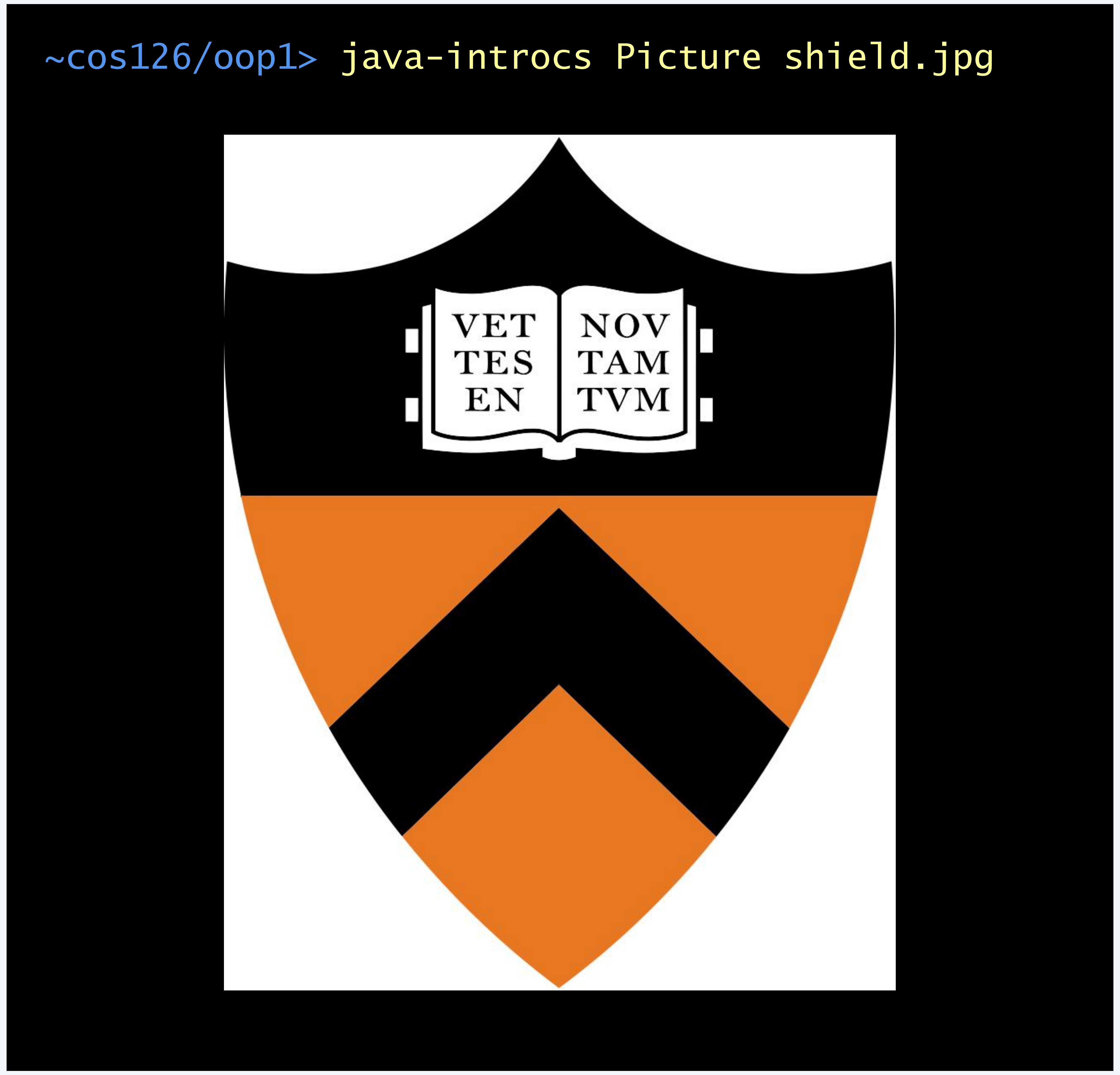
change each pixel to grayscale

display picture (in its own window)

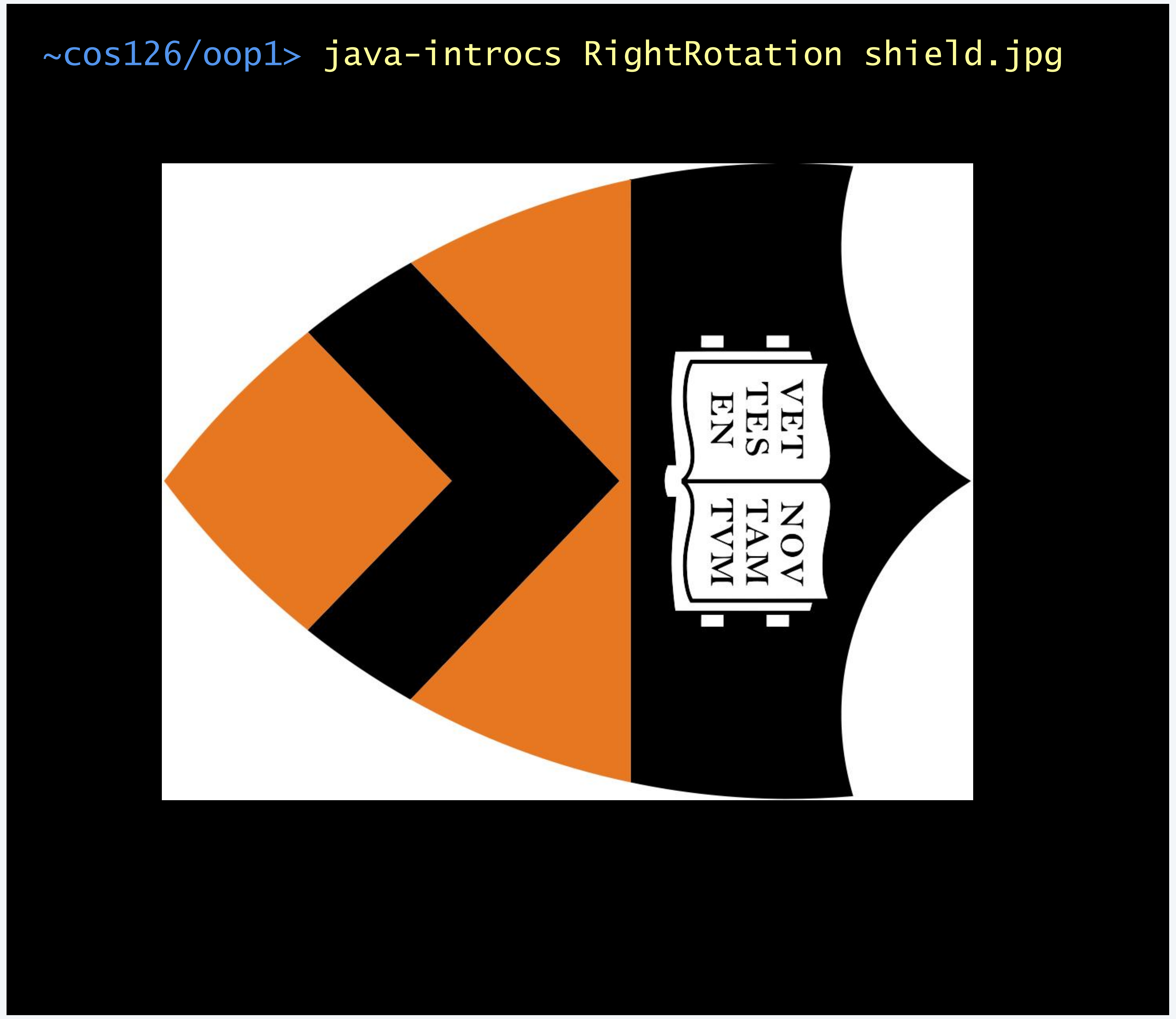
Rotate an image

Goal. Write a Java program to create a right-rotated (90° clockwise) version of an image.

Note. Need two *Picture* objects (since they are of different dimensions).



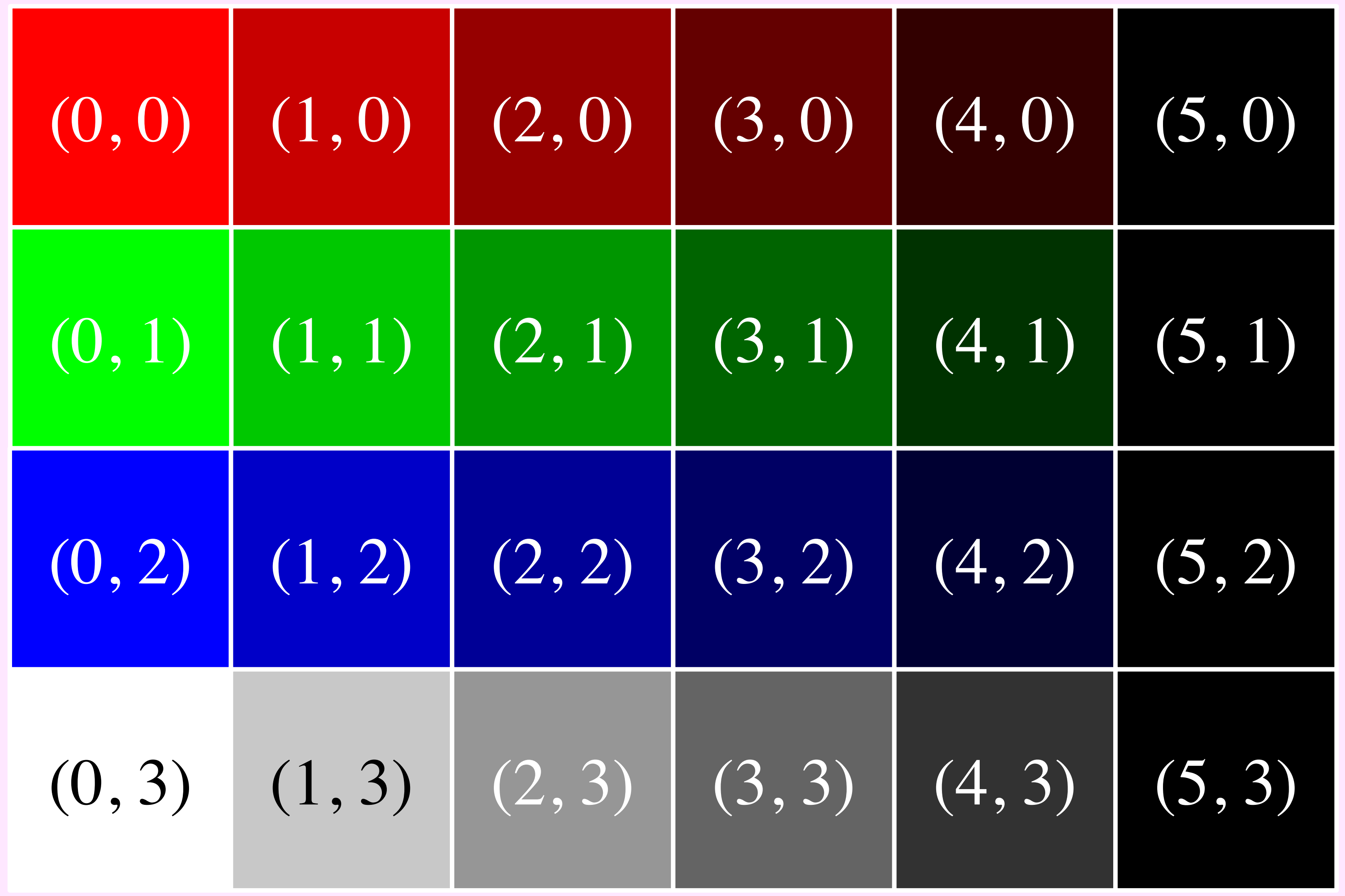
```
~cos126/oop1> java-introcs Picture shield.jpg
```



```
~cos126/oop1> java-introcs RightRotation shield.jpg
```

Rotate an image right: demo

Goal. Rotate an image right (90° clockwise).

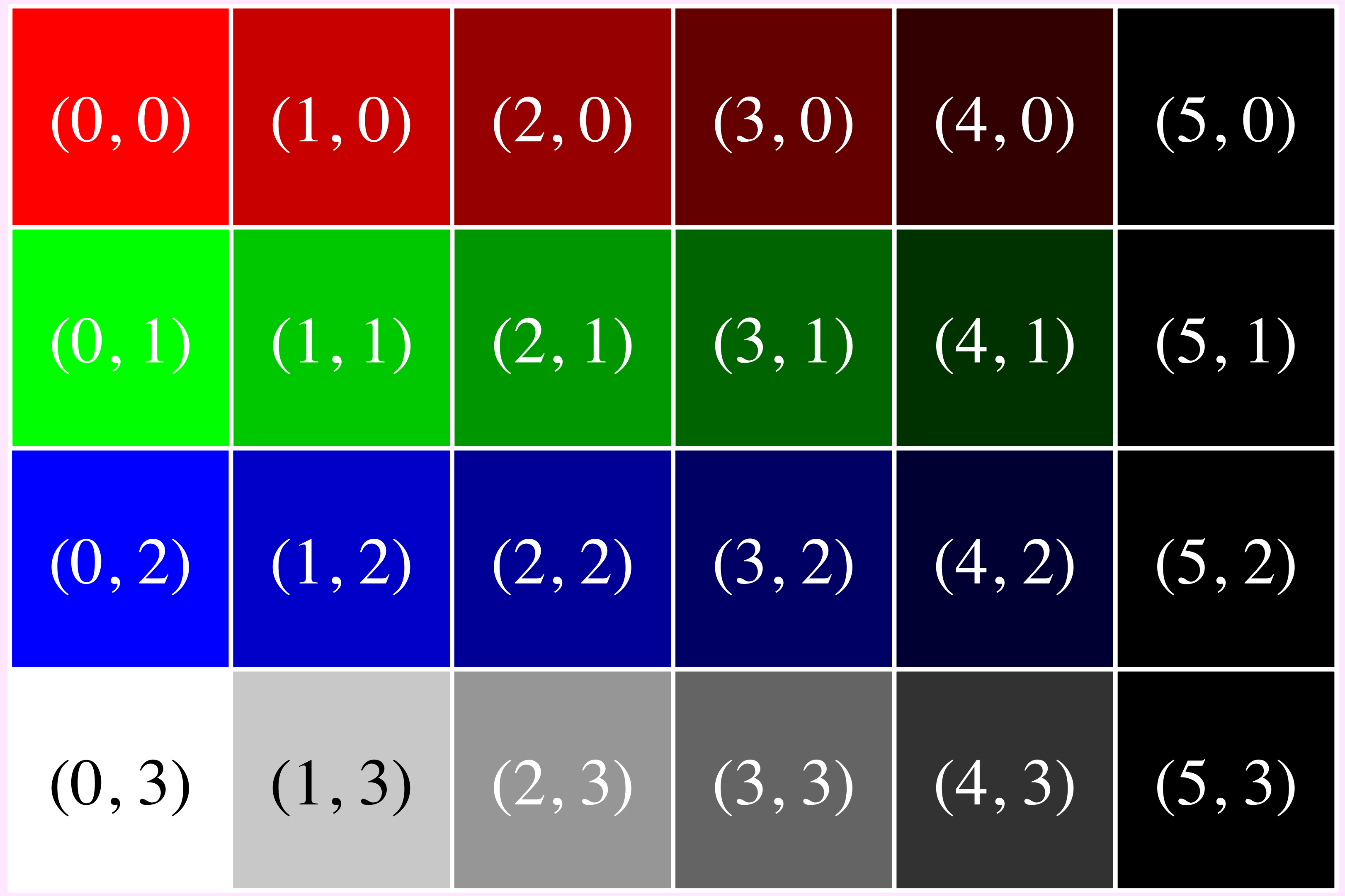


source image (6-by-4)

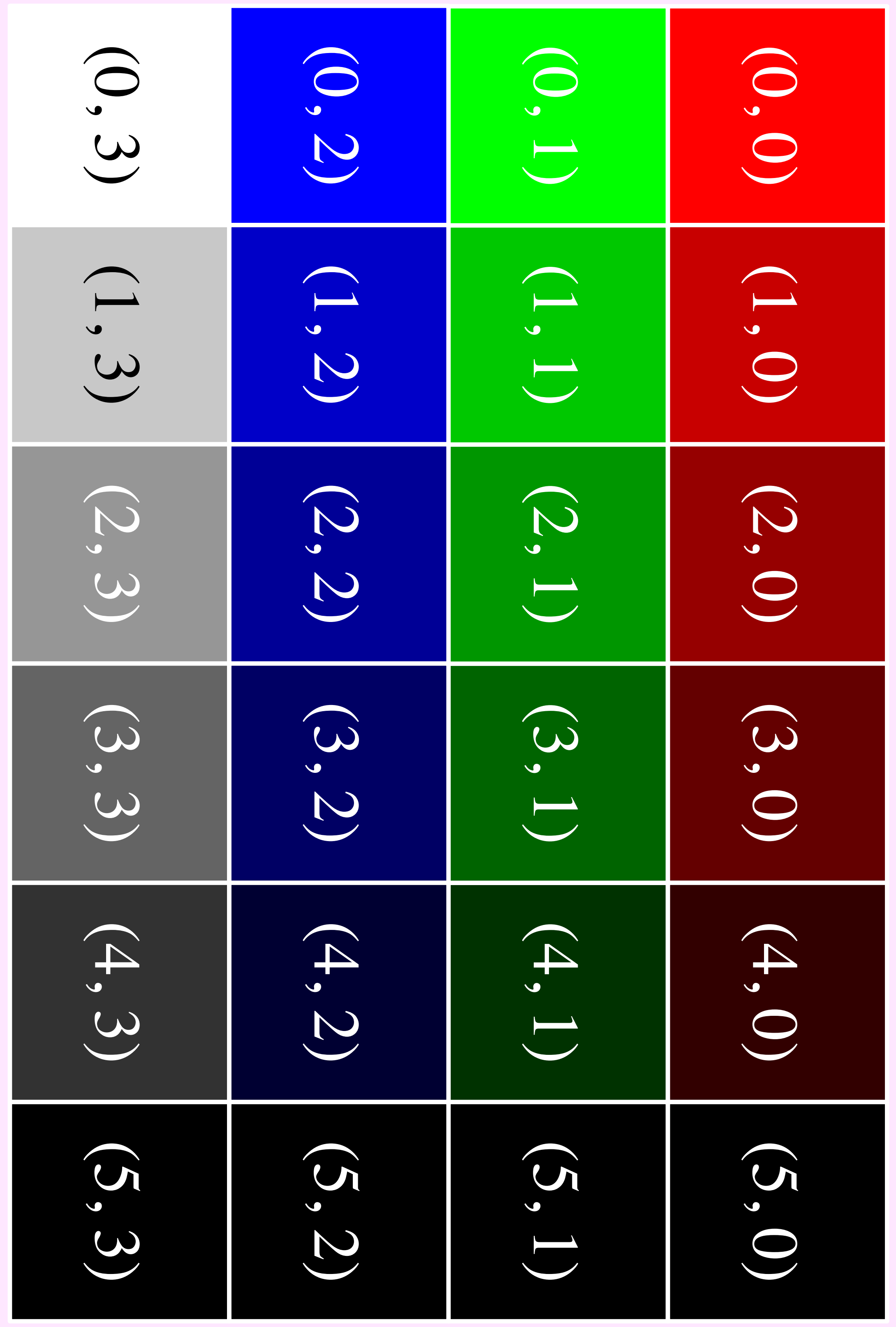
Rotate an image right: demo

Goal. Rotate an image right (90° clockwise).

Algorithm. Pixel (col, row) in source image becomes to pixel $(height - row - 1, col)$ in target image.



source image (6-by-4)



target image (4-by-6)

Right rotate an image implementation

```
import java.awt.Color;

public class RightRotation {
   public static void main(String[] args) {

      Picture source = new Picture(args[0]);
      int width  = source.width();
      int height = source.height();

      Picture target = new Picture(height, width);

      for (int col = 0; col < width; col++) {
         for (int row = 0; row < height; row++) {
            Color color = source.get(col, row);
            target.set(height - row - 1, col, color);
         }
      }

      source.show();
      target.show();
   }
}
```

create picture from file (and get dimensions)

create a new picture (of appropriate dimensions)

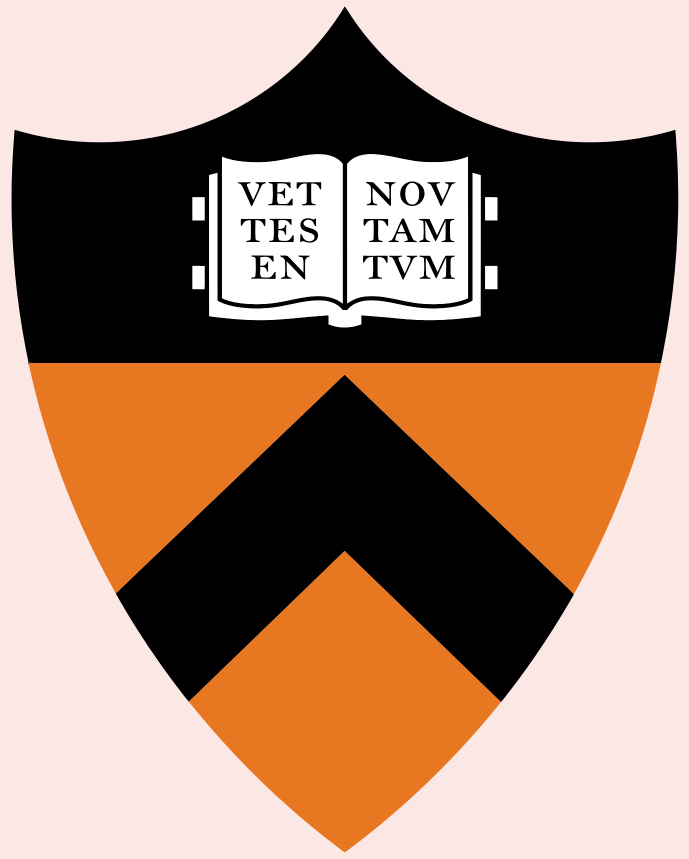
process each pixel

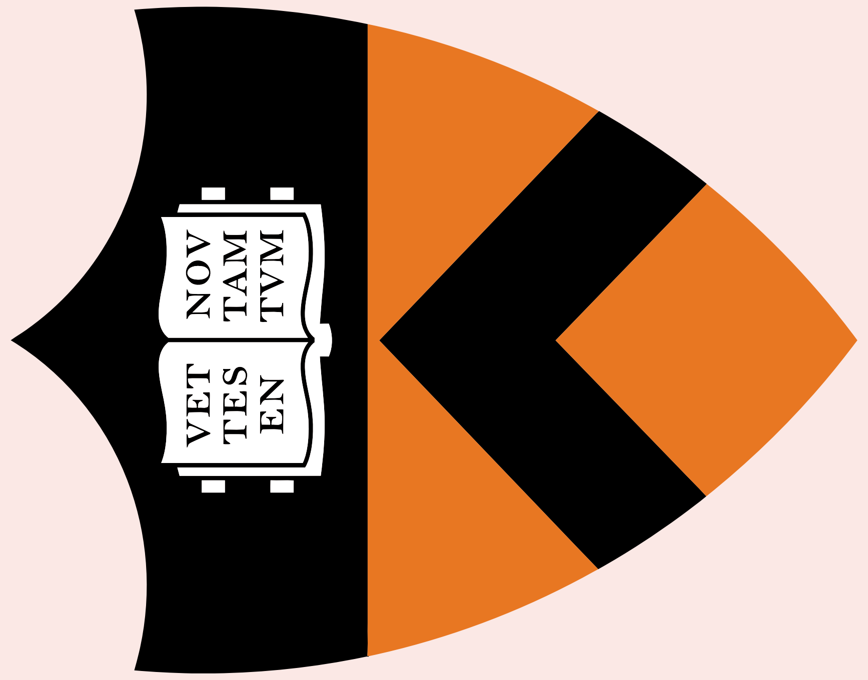
display each picture (in its own window)

Using data types: quiz 4

Fill in the missing code to left rotate (90° counterclockwise) an image?

```
for (int col = 0; col < width; col++) {
   for (int row = 0; row < height; row++) {
      Color color = source.get(col, row);
      target.set([redacted]);
   }
}
```





A. `target.set(col, row, color);`

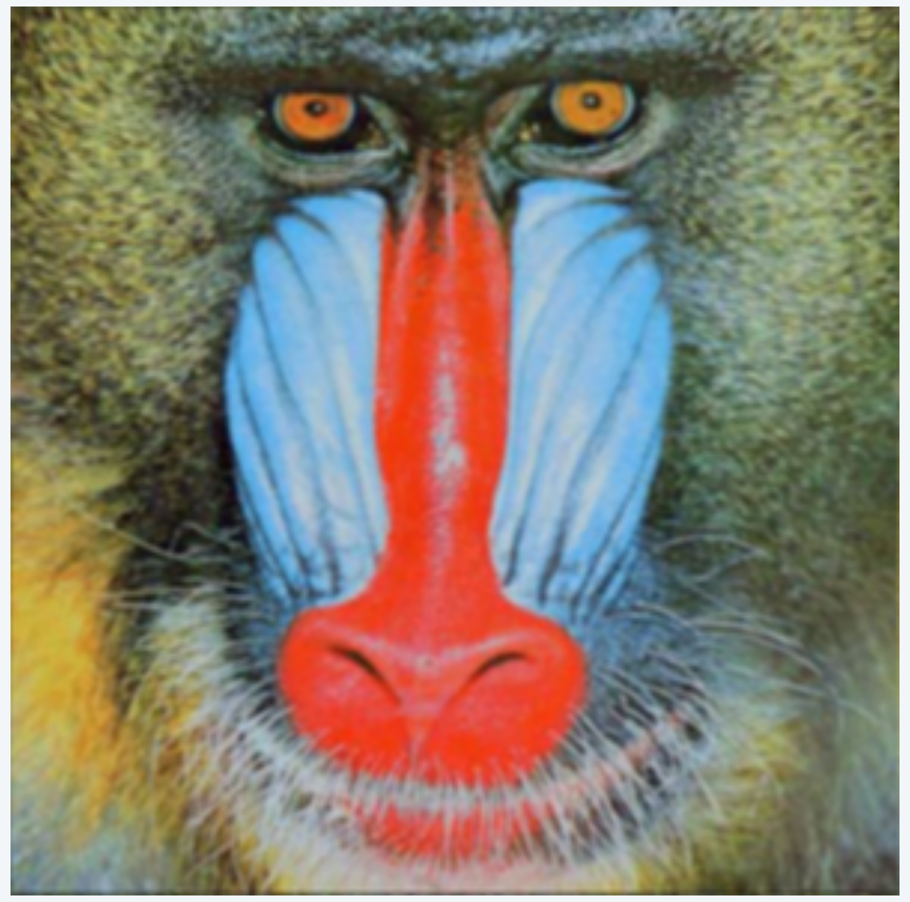
B. `target.set(row, col, color);`

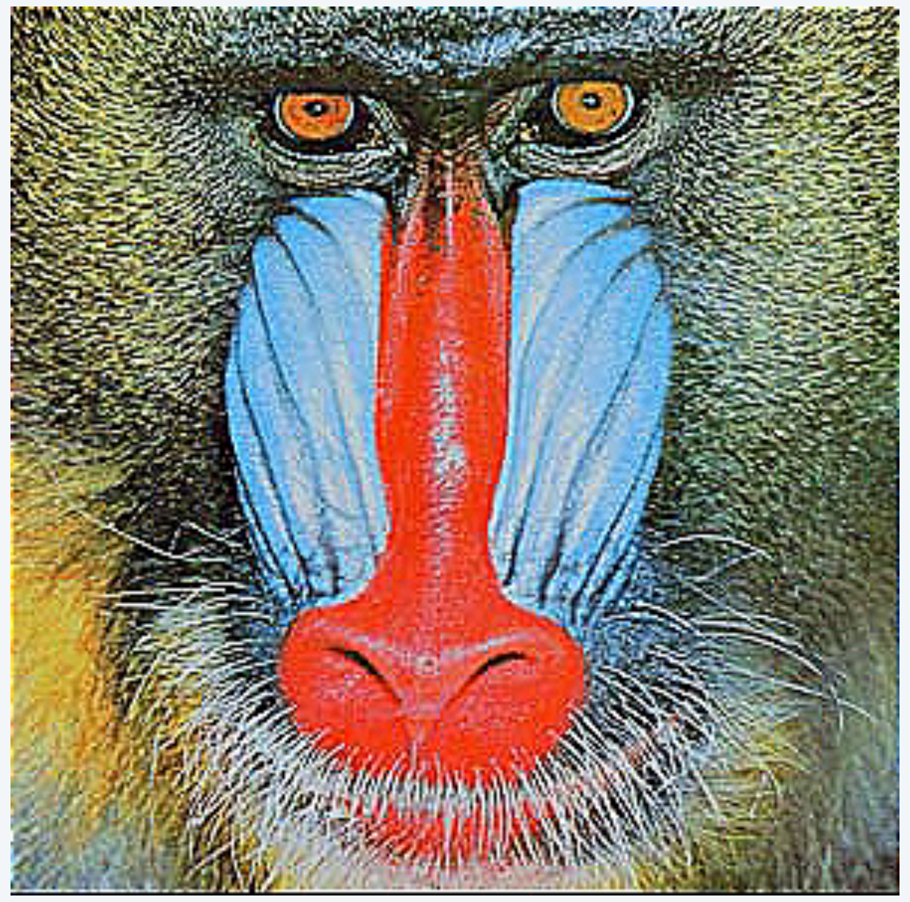
C. `target.set(height - row - 1, col, color);`

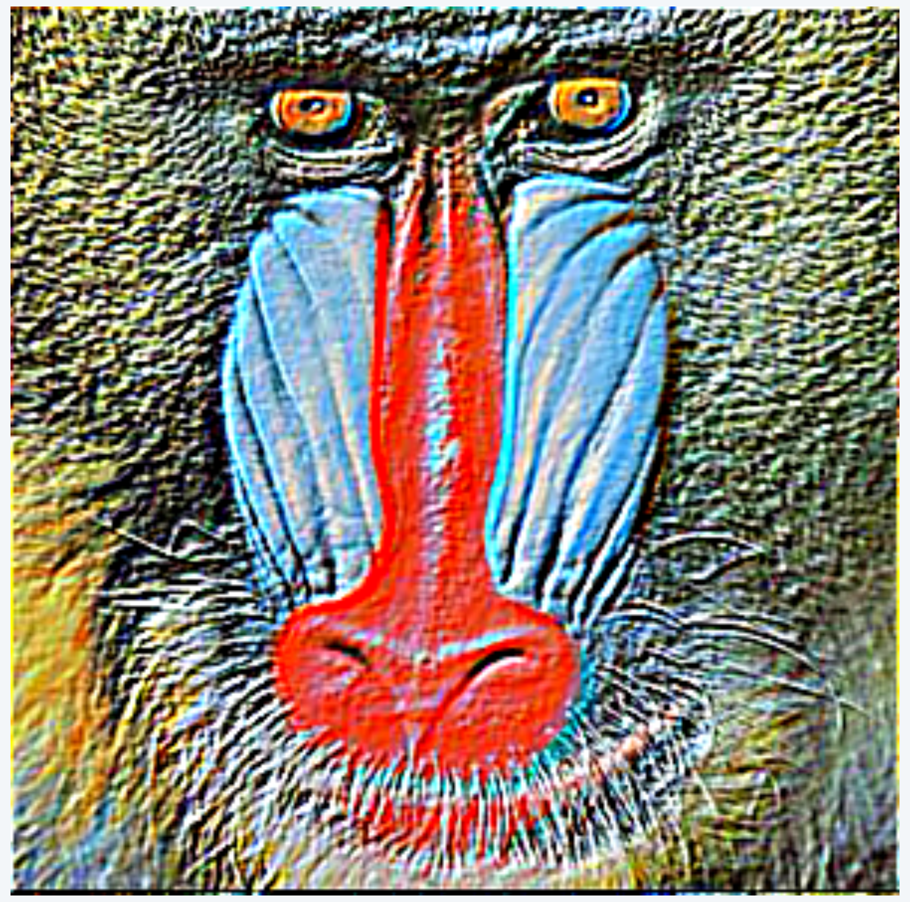
D. `target.set(row, width - col - 1, color);`

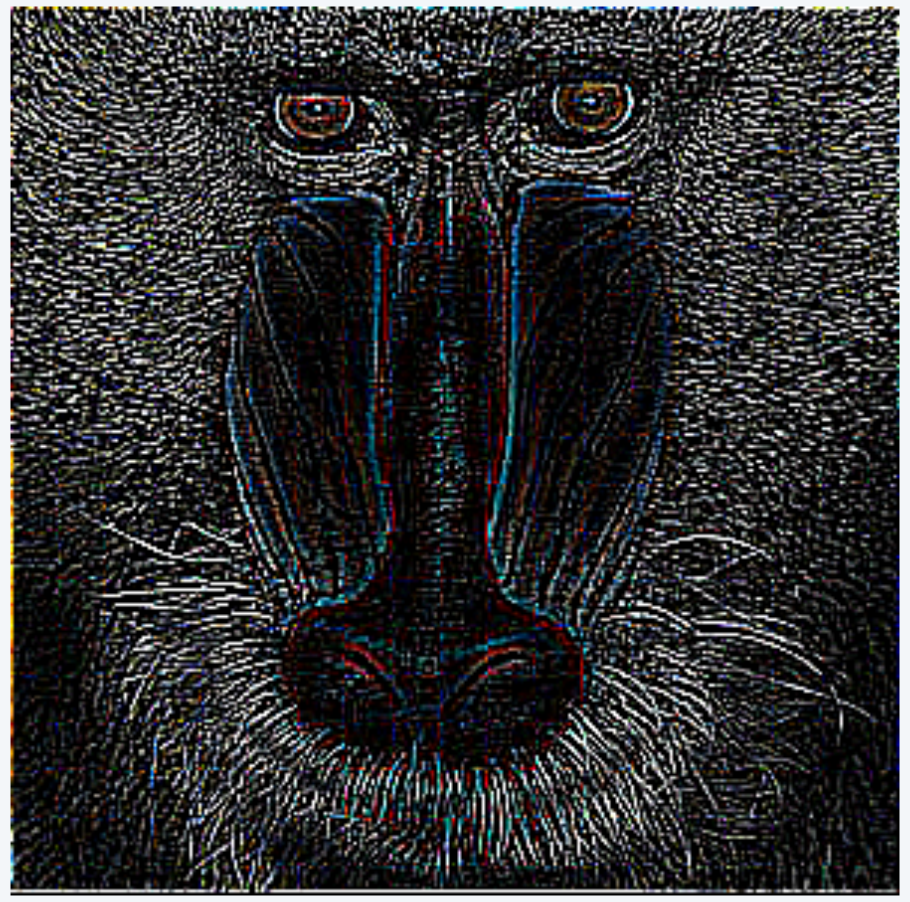
More image-processing effects

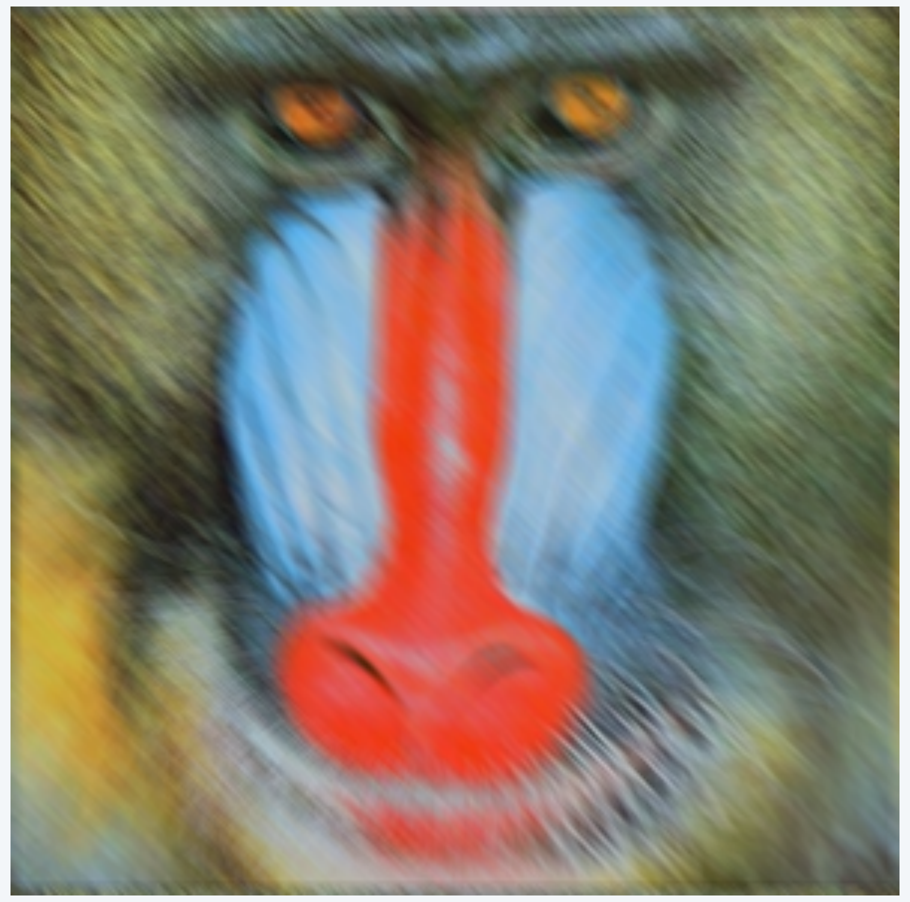
original


Gaussian blur

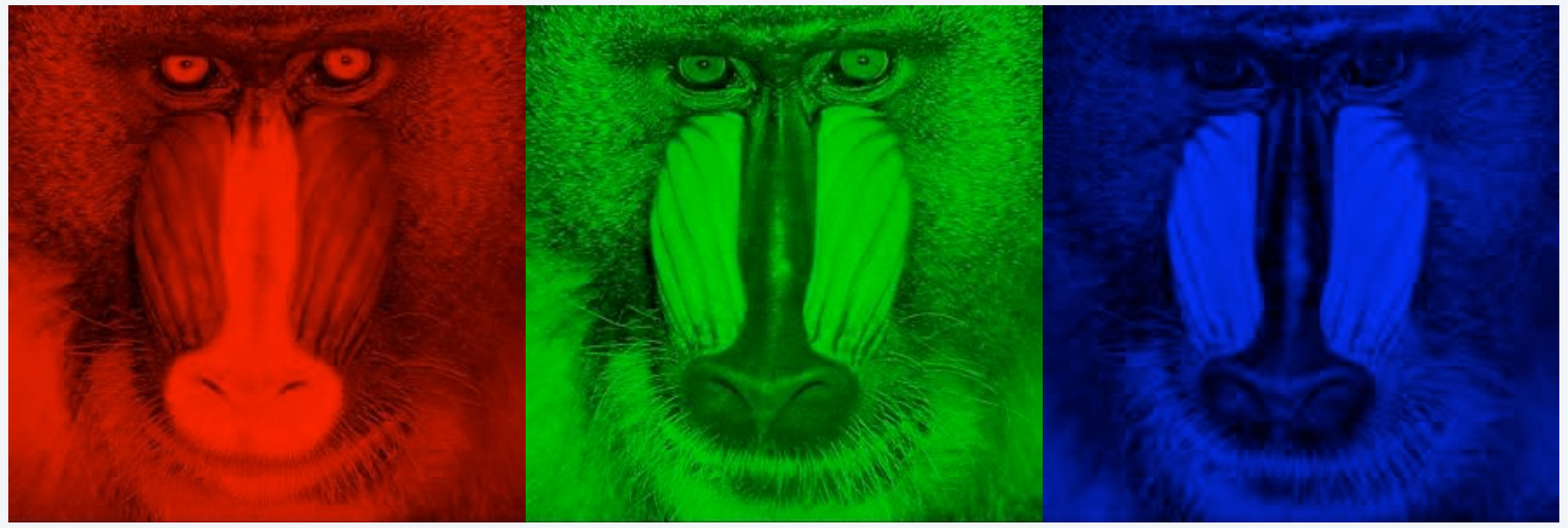

sharpen

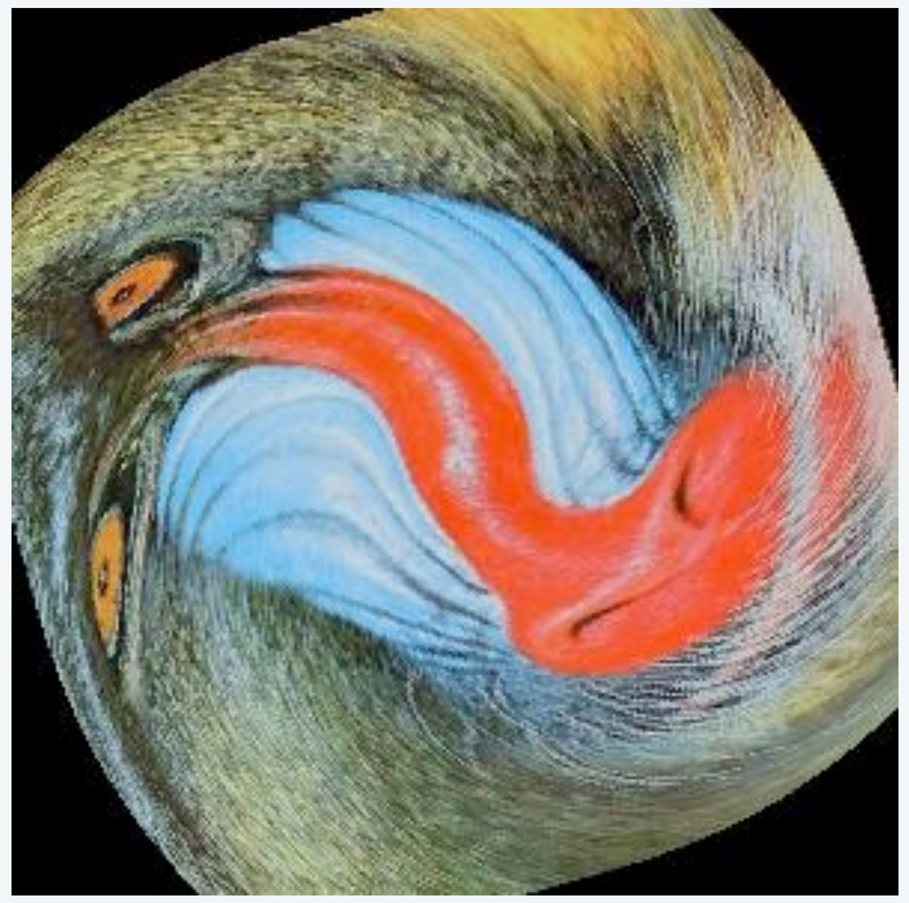

emboss

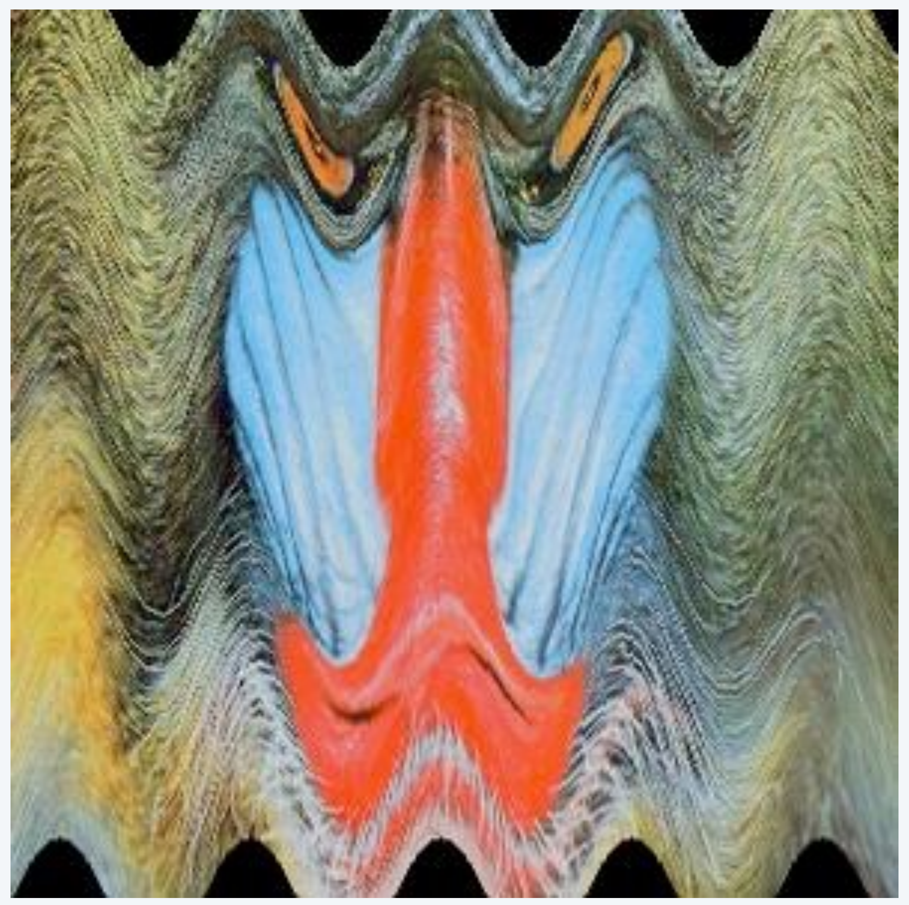

Laplacian

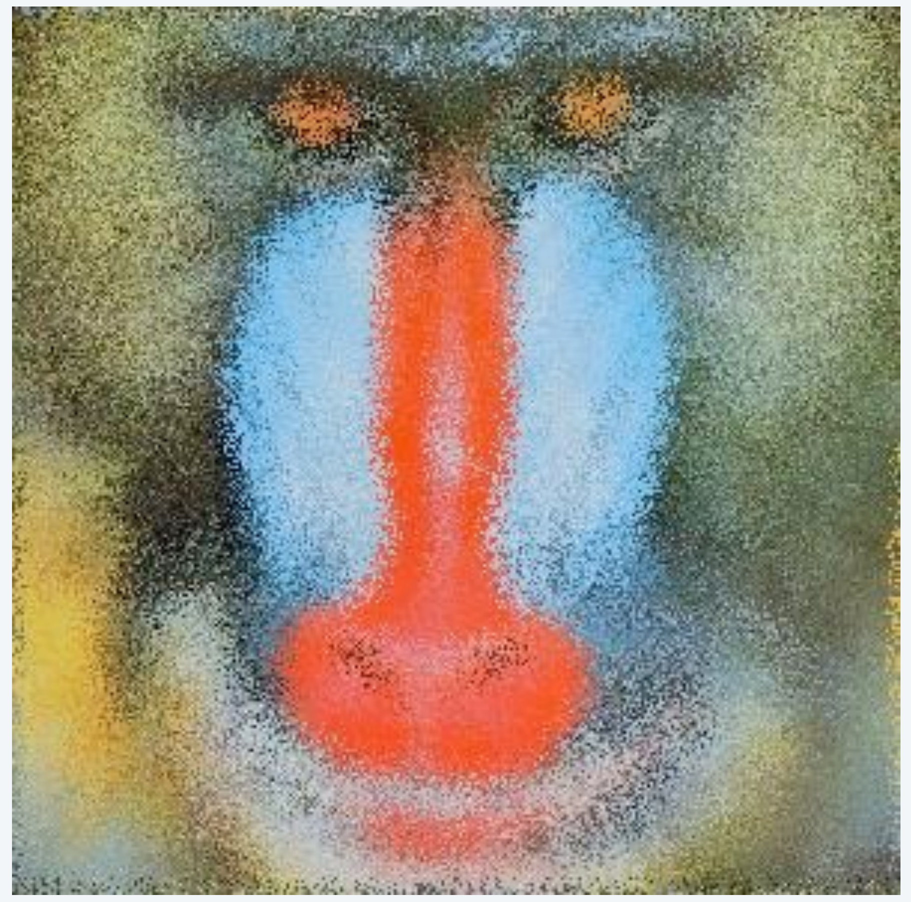

motion blur

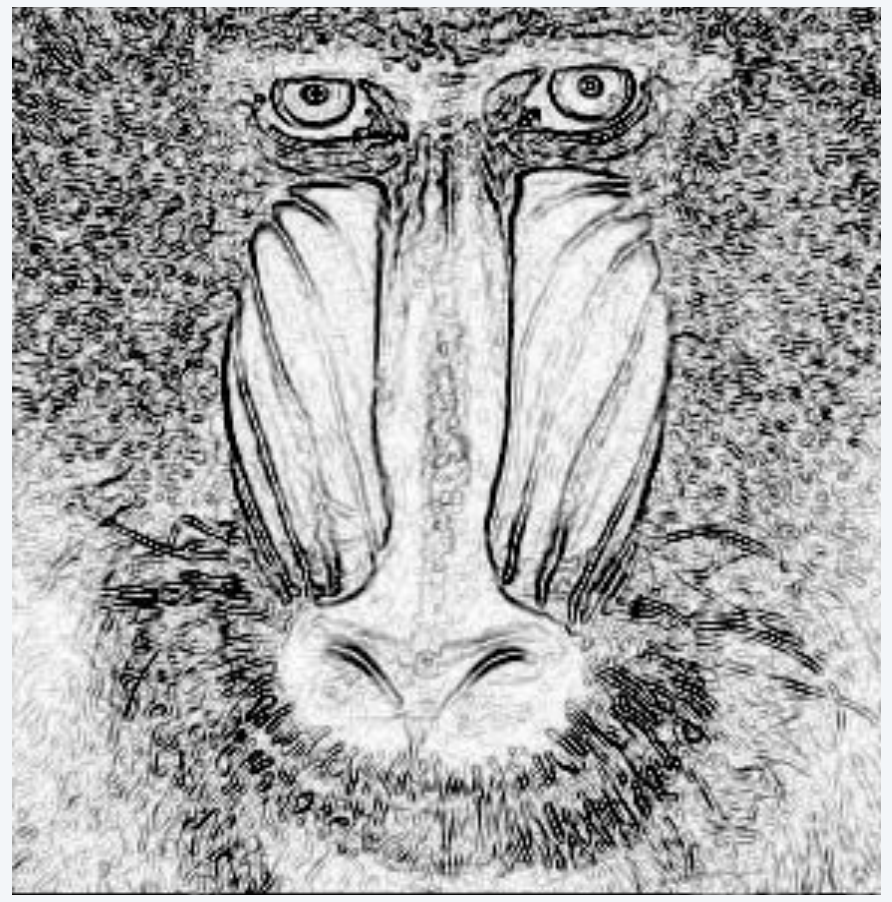
More image-processing effects


RGB color separation


swirl filter


wave filter


glass filter


Sobel edge detection

rescale

Data types

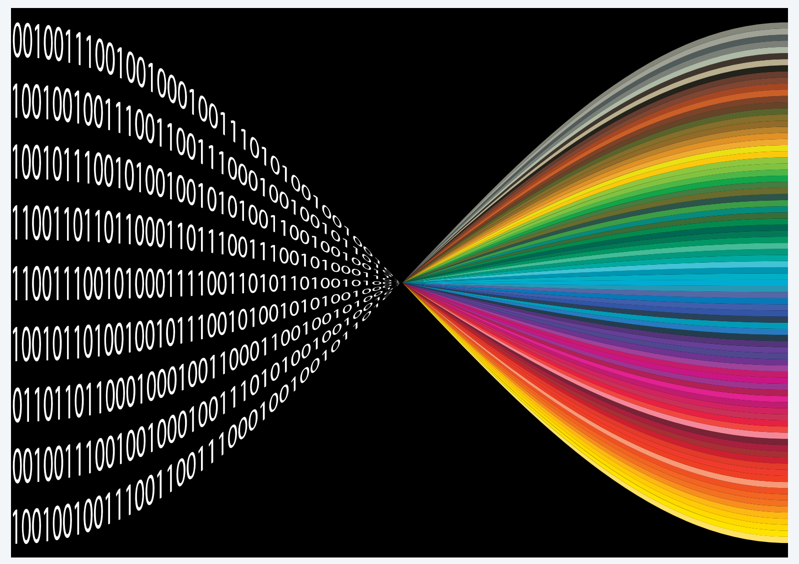
This lecture. Create and use objects from pre-existing data types. ← *strings, colors, pictures*

In Java, programs manipulate object references.

- Almost all data types in Java are reference types.
- Exceptions: primitive types.
- OOP purist: languages should have only reference types.

Next lecture. Develop your own data types.

T	A	G	A	T	G	T	G	C	T	A	G	C



Credits

image	source	license
Binary Code of Digital Images	Adobe Stock	education license
CPU Icon	Adobe Stock	education license
OOP Dice	Adobe Stock	education license
Molecular Structure of DNA	Adobe Stock	education license
RGB Color Model	Wikimedia	Kopimi
LGBTQ+ Eye	Christian Ibarra Santillan	CC BY 2.0
Josef Albers	Arnold Newman	
Homage to the Square	Josef Albers	

Lecture Slides © Copyright 2025 Robert Sedgewick and Kevin Wayne

Credits

image	source	license
WCAG 2.0 Compliant	REMOTEi	
The Treachery of Images	René Magritte	
Surrealist Painter and Plumber	Dan Piraro	Educational use
Select All Squares with Pipes	Noah Veltman	
Image Processing Icon	Adobe Stock	education license
Mandrill	SIPI Image Database	
Johnson Arch	Danielle Alio Capparella	by photographer
Princeton Shield	Wikimedia	public domain

Lecture Slides © Copyright 2025 Robert Sedgewick and Kevin Wayne