INTRODUCTION TO
Programming
in Java

An Interdisciplinary Approach

Robert Sedgewick • Kevin Wayne

Section 3.1

http://introcs.cs.princeton.edu

9. Abstract Data Types

9. Abstract Data Types

- **Overview**
- Color
- Image processing
- String processing

INTRODUCTION TO
Programming
in Java

An Interdisciplinary Approach

Robert Sedgewick · Kevin Wayne

http://introcs.cs.princeton.edu

Abstract data types

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

Primitive types
- *values* immediately map to machine representations
- *operations* immediately map to machine instructions.

Built-in data types

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

type	set of values	examples of values	examples of operations
char	characters	'A' '@'	compare
String	sequences of characters	"Hello World" "CS is fun"	concatenate
int	integers	17 12345	add, subtract, multiply, divide
double	floating-point numbers	3.1415 6.022e23	add, subtract, multiply, divide
boolean	truth values	true false	and, or, not

Java's built-in data types

We want to write programs that process other types of data.
- Colors, pictures, strings,
- Complex numbers, vectors, matrices,
- ...

An abstract data type is a data type whose representation is hidden from the user.

Object-oriented programming (OOP)

Object-oriented programming (OOP).
- Create your own data types (sets of values and ops on them).
- Use them in your programs (manipulate *objects*).

An object holds a data type value.
Variable names refer to objects.

data type	set of values	examples of operations
Color	three 8-bit integers	get red component, brighten
Picture	2D array of colors	get/set color of pixel (i, j)
String	sequence of characters	length, substring, compare

An abstract data type is a data type whose representation is hidden from the user.

Impact: We can use ADTs without knowing implementation details.
- This lecture: how to write client programs for several useful ADTs
- Next lecture: how to implement your own ADTs

Sound

We have *already* been using ADTs!

Crash course in sound
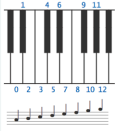
Sound is the perception of the vibration of molecules in our eardrums.

A musical tone is a steady periodic sound.

A pure tone is a sinusoidal waveform.

Western musical scale
• Concert A is 440 Hz.
• 12 notes, logarithmic scale.

pitch	i	frequency
A	0	440.
A# / B♭	1	466.
B	2	493.
C	3	523.
C# / D♭	4	554.
D	5	587.
D# / E♭	6	622.
E	7	659.
F	8	698.
F# / G♭	9	739.
G	10	783.
G# / A♭	11	830.
A	12	880.

Digital audio

To represent a wave, *sample* at regular intervals and save the values in an array. ← same as when plotting a function (previous lecture)

	samples/sec	samples	sampled waveform
1/40 second of concert A	5,512	137	
	11,025	275	
	22,050	551	
CD standard →	44,100	1102	

Bottom line. You can *write programs* to manipulate sound (arrays of double values).

Sound ADT

Values: Array of doubles.

Operations: specified in API.

public class StdAudio	
void play(double[] a)	*play the given sound wave*
void save(String file, double[] a)	*save to a* .wav *file*
double[] read(String file)	*read from a* .wav *file*

Representation: Hidden from user (.wav and other formats needed by devices).

Strings

We have *already* been using ADTs!

A String is a sequence of Unicode characters. ← defined in terms of its ADT values (typical)

Java's String ADT allows us to write Java programs that manipulate strings.

stay tuned for more complete API later in this lecture

Operations (API)

public class String	
String(String s)	*create a string with the same value*
int length()	*string length*
char charAt(int i)	*ith character*
String substring(int i, int j)	*ith through (j-1)st characters*
boolean contains(String sub)	*does string contain sub?*
boolean endsWith(String post)	*does string end with post?*
int indexOf(String p)	*index of 1st occurrence of p*
int indexOf(String p, int i)	*index of 1st occurrence of p after i*

Constructors and methods
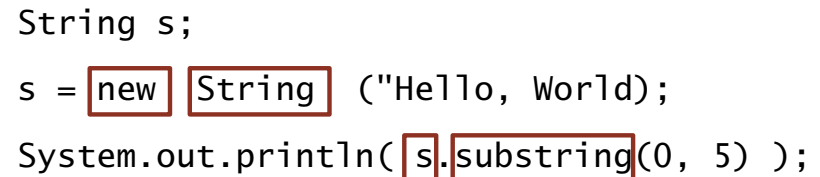
To use a data type, you need to know how to:
- Construct new objects.
- Apply operations to a given object.

To construct a new object
- Use the keyword new to invoke a "constructor."
- Use data type name to specify which type of object.

```
String s;
s = new  String  ("Hello, World);
System.out.println( s.substring(0, 5) );
```

To apply an operation
- Use object name to specify which object.
- Use the dot operator to indicate that an operation is to be applied.
- Use a method name to specify which operation.
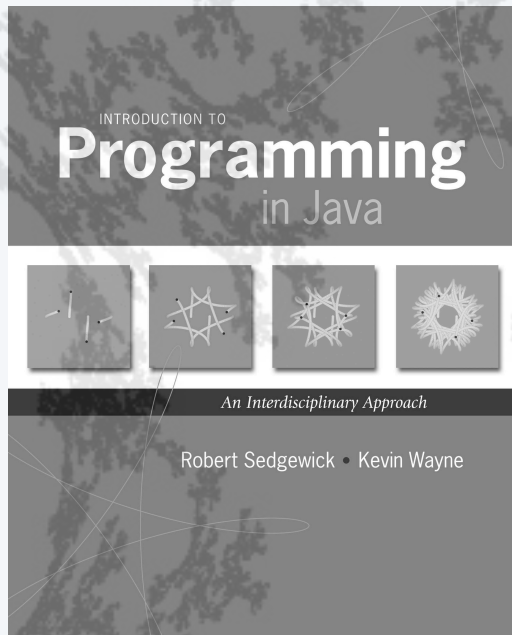
INTRODUCTION TO
Programming
in Java

An Interdisciplinary Approach

Robert Sedgewick · Kevin Wayne

http://introcs.cs.princeton.edu

9. Abstract Data Types

- **Overview**
- Color
- Image processing
- String processing

INTRODUCTION TO
Programming
in Java

An Interdisciplinary Approach

Robert Sedgewick · Kevin Wayne

http://introcs.cs.princeton.edu
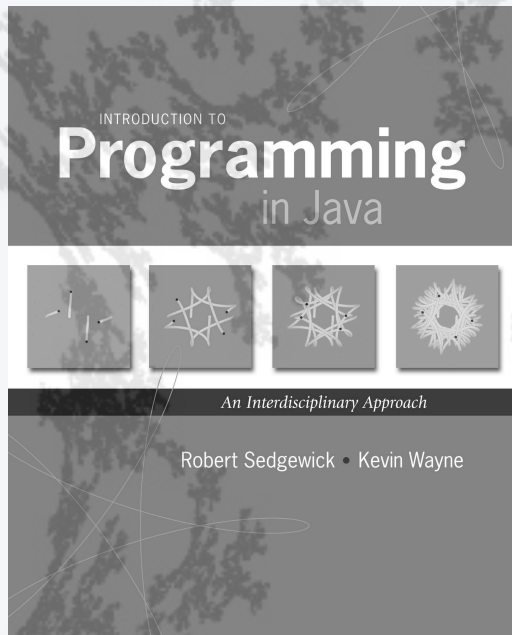
9. Abstract Data Types

- Overview
- **Color**
- Image processing
- String processing

Color ADT

Color is a sensation in the eye from electromagnetic radiation.

Values

An ADT allows us to write Java programs that manipulate color.

API (operations)

		examples							
R (8 bits)	red intensity	255	0	0	0	255	0	119	105
G (8 bits)	green intensity	0	255	0	0	255	64	33	105
B (8 bits)	blue intensity	0	0	255	0	255	128	27	105
color									

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 the same as c's ?*

Albers squares

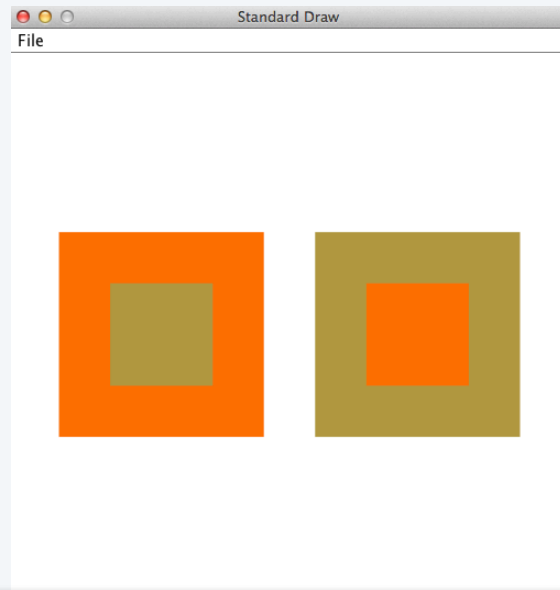Josef Albers. A 20th century artist who revolutionized the way people think about color.
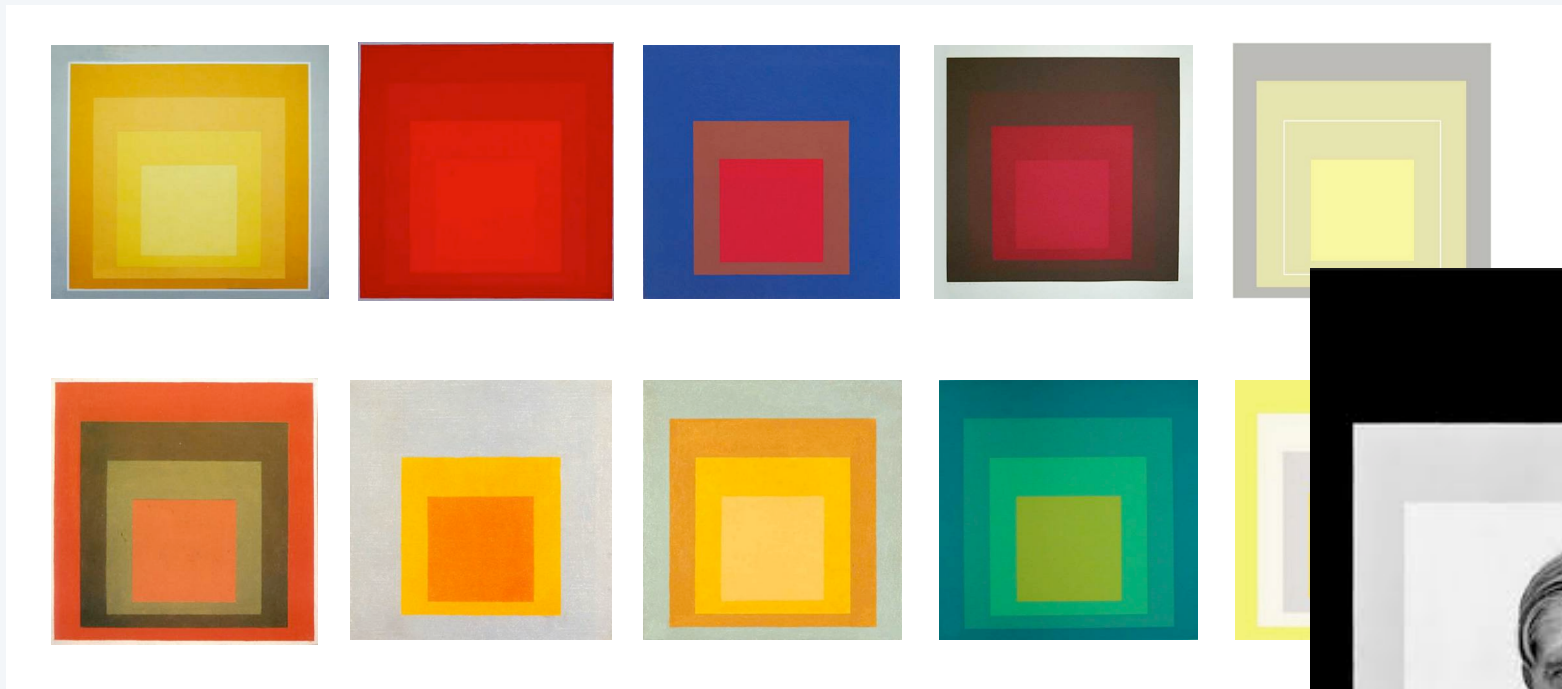
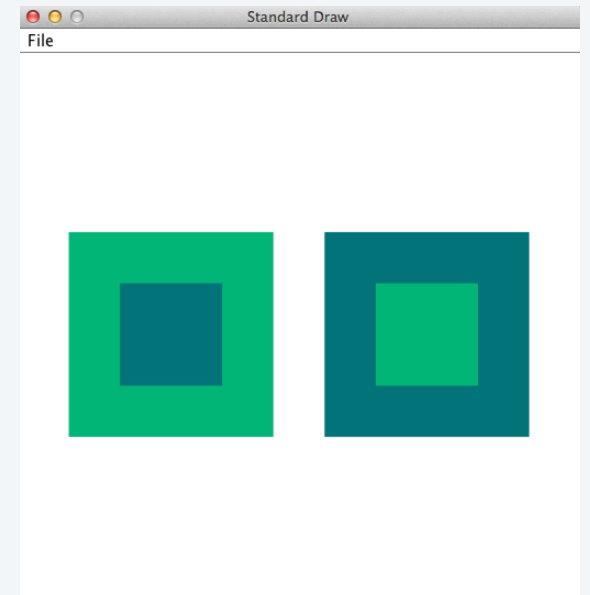Josef Albers 1888–1976

Color client example: Albers squares

Goal. Write a Java program to generate Albers squares.
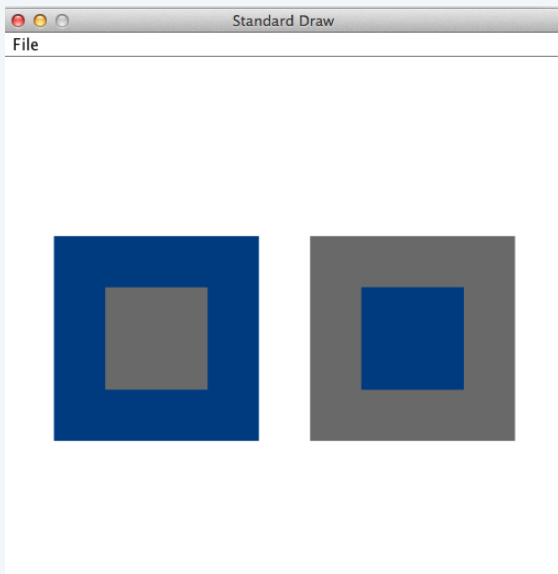
% java AlbersSquares 0 64 128 105 105 105

% java AlbersSquares 251 112 34 177 153 71

% java AlbersSquares 28 183 122 15 117 123

Color client example: Albers squares

```
public class AlbersSquares
{
    public static void main(String[] args)
    {
        int r1 = Integer.parseInt(args[0]);
        int g1 = Integer.parseInt(args[1]);        ← create first color
        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]);        ← create second color
        int b2 = Integer.parseInt(args[5]);
        Color c2 = new Color(r2, g2, b2);

        StdDraw.setPenColor(c1);
        StdDraw.filledSquare(.25, .5, .2);         ← draw first square
        StdDraw.setPenColor(c2);
        StdDraw.filledSquare(.25, .5, .1);

        StdDraw.setPenColor(c2);
        StdDraw.filledSquare(.75, .5, .2);         ← draw second square
        StdDraw.setPenColor(c1);
        StdDraw.filledSquare(.75, .5, .1);
    }
}
```

% java AlbersSquares 0 64 128 105 105 105

13
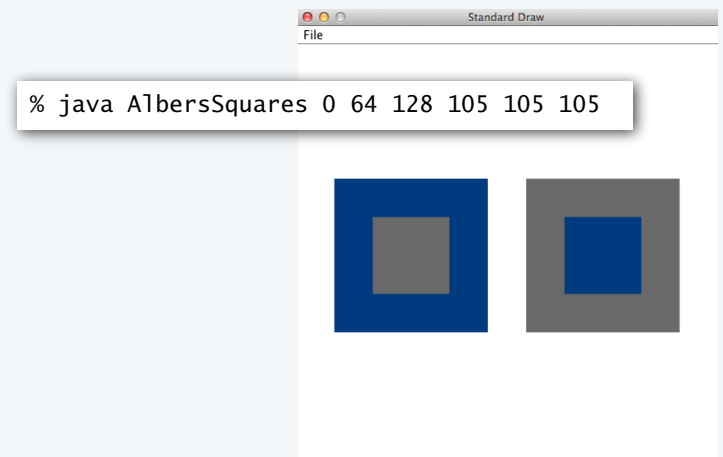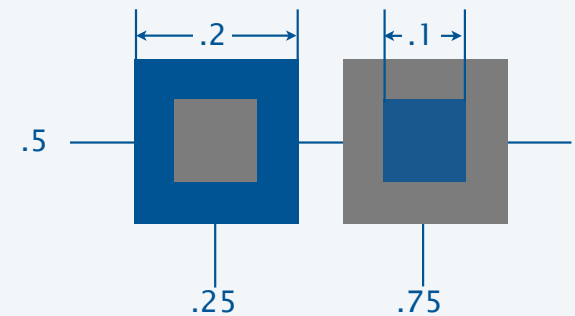
# Computing with color: monochrome luminance

Def. The *monochrome luminance* of a color quantifies its effective brightness.

NTSC standard formula for luminance: 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;
   }
   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 c = new Color(r, g, b);
      StdOut.println(Math.round(lum(c)));
   }
}
```

```
% java Luminance 0 64 128
52
```

	examples							
red intensity	255	0	0	0	255	0	119	105
green intensity	0	255	0	0	255	64	33	105
blue intensity	0	0	255	0	255	128	27	105
color								
luminance	76	150	29	0	255	52	58	105

Applications (next)
- Choose colors for displayed text.
- Convert colors to grayscale.

Computing with color: compatability

Q. Which font colors will be most readable with which background colors on a display?

Rule of thumb. Absolute value of difference in luminosity should be > 128.

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

		76	0	255	52
	76	255	76	179	24
	0	76		255	52
	255	179	255		203
	52	24	52	203	

Computing with color: grayscale

Goal. Convert a colors to grayscale values.

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


Q. What value for a given color?

A. Its luminance!

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

method for Luminance library

	examples							
red intensity	255	0	0	0	255	0	119	105
green intensity	0	255	0	0	255	64	33	105
blue intensity	0	0	255	0	255	128	27	105
color								
luminance	76	150	29	0	255	52	58	105
grayscale								

16

OOP context for color

Q. How does Java represent color? Three `int` values? Packed into one `int` value?

A. We don't know. The representation is hidden. It is an *abstract* data type.

Possible memory representation of red = new Color(255, 0, 0)
and gray = new Color(105, 105, 105);

memory address
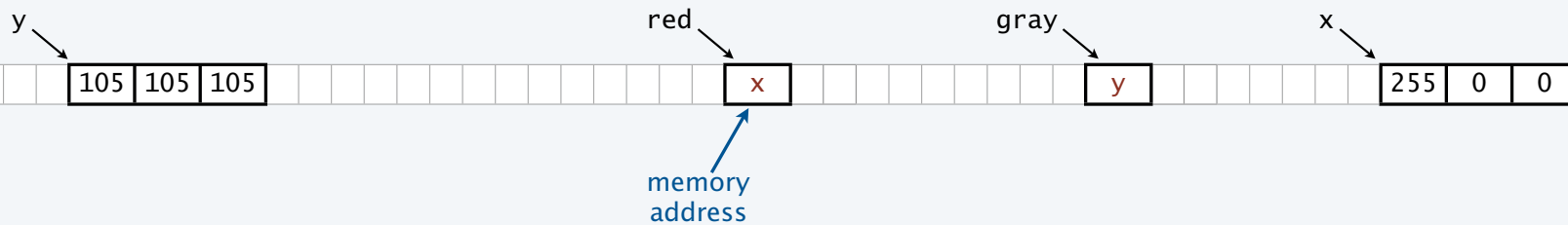
An object reference is analogous to a variable name.
- It is not the value but it refers to the value.
- We can manipulate the value in the object it refers to.
- We can pass it to (or return it from) a method.

We also use object references to *invoke* methods (with the . operator)

References and abstraction

René Magritte. This is not a pipe.   ←— It is a picture of a painting of a pipe.

Java. These are not colors.

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

Object-oriented programming. A natural vehicle for studying abstract models of the real world.

"This is not a pipe."

Dan Piraro, http://www.uexpress.com
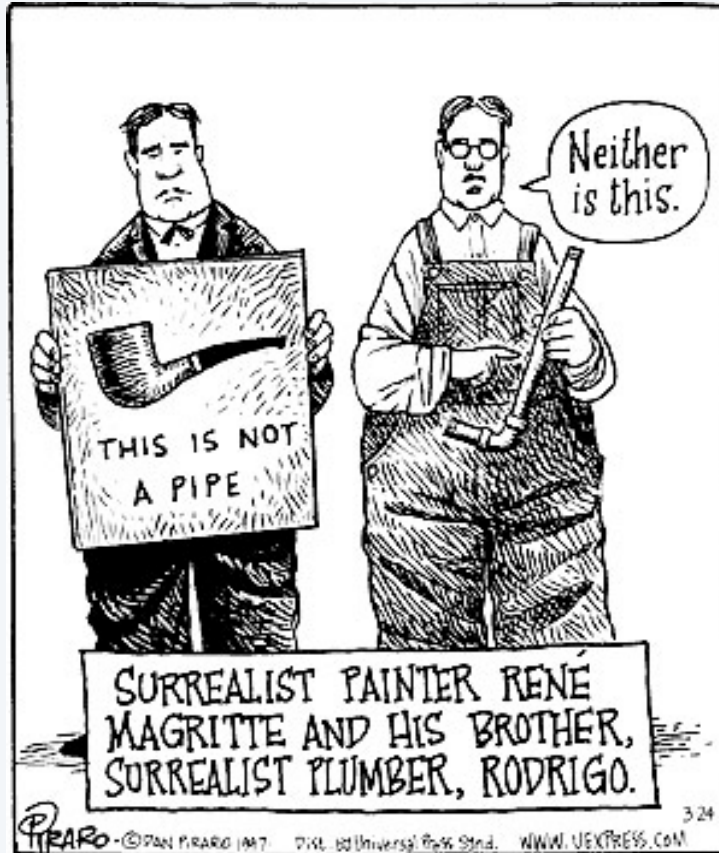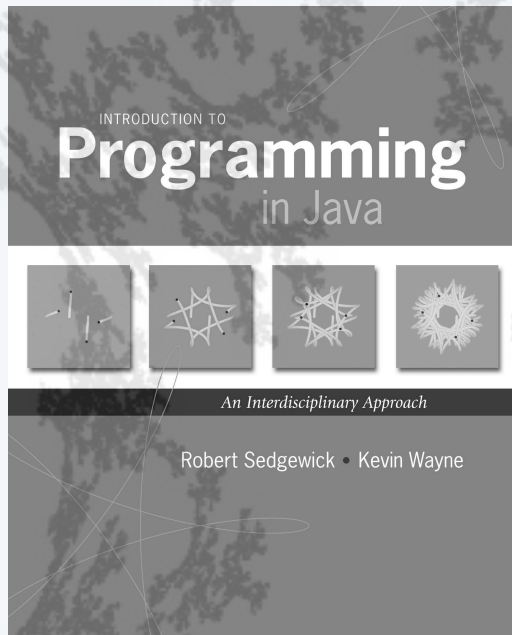
Yes it is! He's referring to the physical object he's holding.
He needs to be holding a *picture* of a pipe.

This is not a pipe.
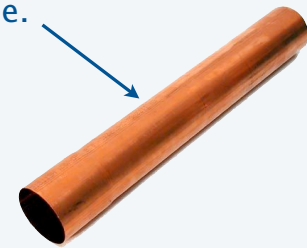
Surrealist computer scientist:
Neither is this.
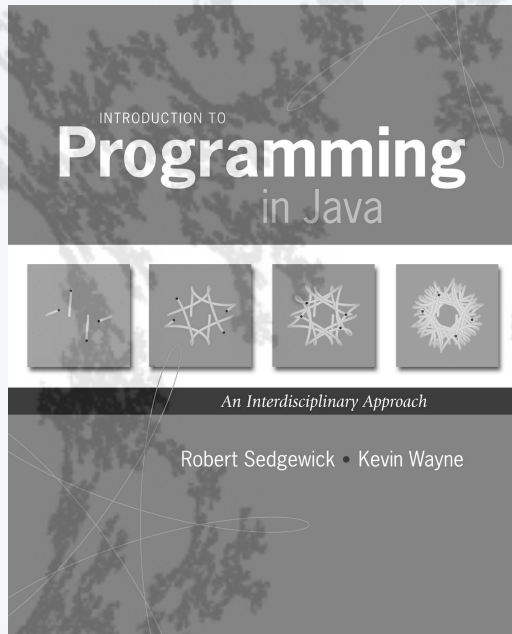
```
% java RandomSeq 10000 | java Average
```

INTRODUCTION TO
Programming
in Java

An Interdisciplinary Approach

Robert Sedgewick · Kevin Wayne

9. Abstract Data Types

- Overview
- **Color**
- Image processing
- String processing

INTRODUCTION TO
Programming
in Java

An Interdisciplinary Approach

Robert Sedgewick · Kevin Wayne

http://introcs.cs.princeton.edu

9. Abstract Data Types

- Overview
- Color
- **Image processing**
- String processing

Picture ADT

A Picture is a 2D array of pixels.

defined in terms of its ADT values (typical)

An ADT allows us to write Java programs that manipulate pictures.

Values (arrays of Colors)

column j

pixel (0, 0)

row i →

pixel (i, j)

height

width

API (operations)
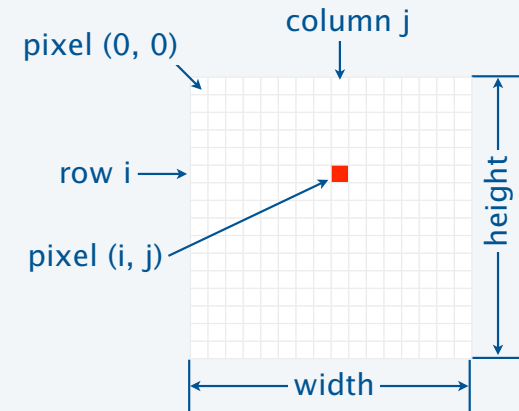
public class java.awt.color	
Picture(String filename)	*create a picture from a file*
Picture(int w, int h)	*create a blank w-by-h picture*
int width()	*width of the picture*
int height()	*height of the picture*
Color get(int i, int j)	*the color of pixel (i, j)*
void set(int i, int j, Color c)	*set the color of pixel (i, j) to c*
void show()	*display the image in a window*
void save(String filename)	*is this color the same as c's?*

Picture client example: Grayscale filter

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

Source: `mandrill.jpg`
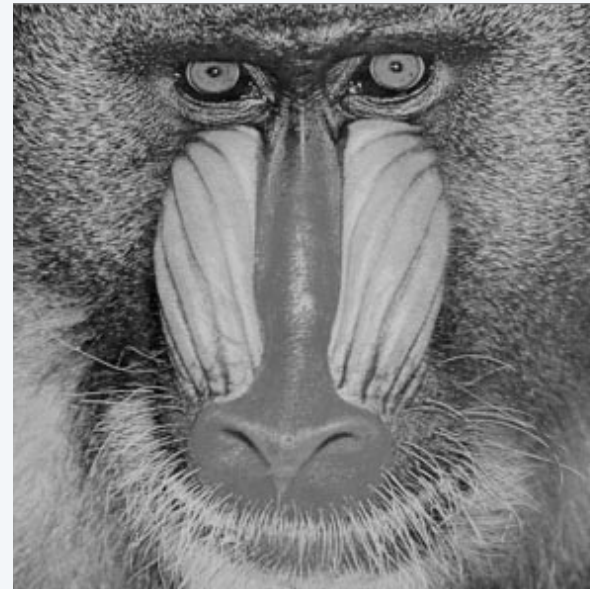
`% java Grayscale mandrill.jpg`

Picture client example: Grayscale filter

```java
import java.awt.Color;
public class Grayscale
{
    public static void main(String[] args)
    {
        Picture pic = new Picture(args[0]);     ←——— create a new picture
        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);   ←——— fill in each pixel
                pic.set(i, j, gray);
            }
        pic.show();
    }
}
```

% java Grayscale mandrill.jpg
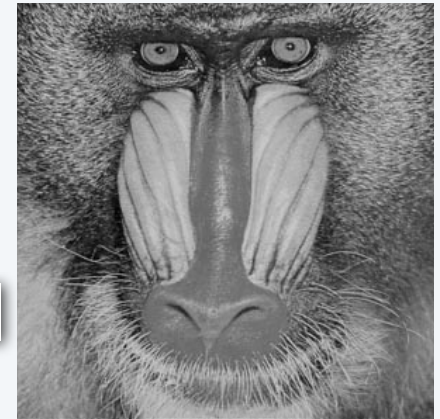
TEQ 1a on image processing

Q. What is the effect of the following code (easy question)?

```
Picture pic = new Picture(args[0]);
for (int i = 0; i < pic.width(); i++)
   for (int j = 0; j < pic.height(); j++)
      pic.set(i, j, pic.get(i, j));
pic.show();
```

Q. What is the effect of the following code (easy question)?

```
Picture pic = new Picture(args[0]);
for (int i = 0; i < pic.width(); i++)
    for (int j = 0; j < pic.height(); j++)
        pic.set(i, j, pic.get(i, j));
pic.show();
```

A. None. Just shows the picture.

Q. What is the effect of the following code (not-so-easy question)?

```
Picture pic = new Picture(args[0]);
for (int i = 0; i < pic.width(); i++)
   for (int j = 0; j < pic.height(); j++)
      pic.set(i, pic.height()-j-1, pic.get(i, j));
pic.show();
```

Q. What is the effect of the following code (not-so-easy question)?

```
Picture pic = new Picture(args[0]);
for (int i = 0; i < pic.width(); i++)
   for (int j = 0; j < pic.height(); j++)
      pic.set(i, pic.height()-j-1, pic.get(i, j));
pic.show();
```
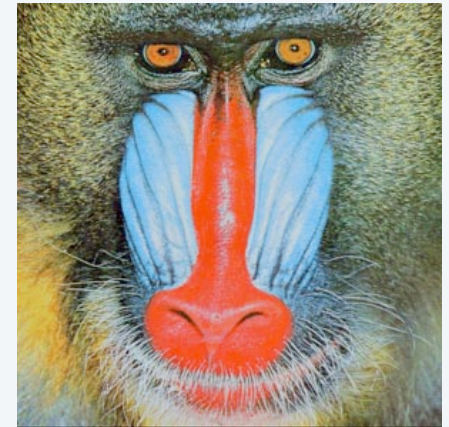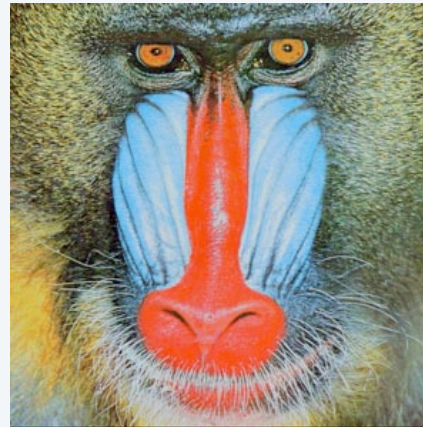
A. Tries to turn image upside down, but fails.
   An instructive bug!.

Q. What is the effect of the following code?

```
Picture source = new Picture(args[0]);
int width  = source.width();
int height = source.height();
Picture target = new Picture(width, height);
for (int i = 0; i < width; i++)
   for (int j = 0; j < height; j++)
      target.set(i, height-j-1, source.get(i, j));
target.show();
```

Q. What is the effect of the following code?

```
Picture source = new Picture(args[0]);
int width  = source.width();
int height = source.height();
Picture target = new Picture(width, height);
for (int i = 0; i < width; i++)
    for (int j = 0; j < height; j++)
        target.set(i, height-j-1, source.get(i, j));
target.show();
```

A. Makes an upside down copy of the image.
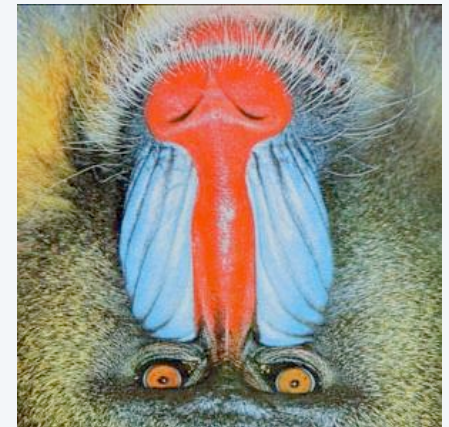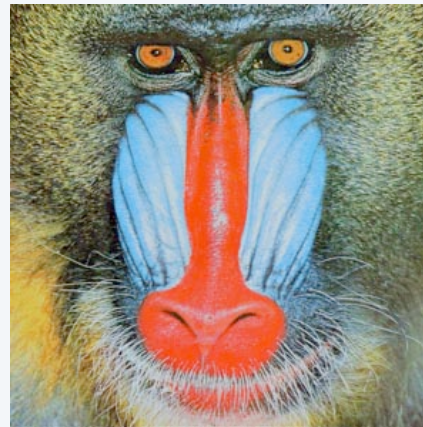
Picture client example: Scaling filter

Goal. Write a Java program to scale an image (arbitrarily and independently on *x* and *y*).

600x200

% java Scale mandrill.jpg 600 200

500x500
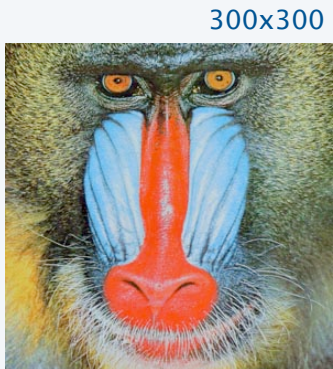
% java Scale mandrill.jpg 500 500

300x300

Source: mandrill.jpg

99x99
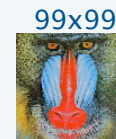
% java Scale mandrill.jpg 99 99

200x400
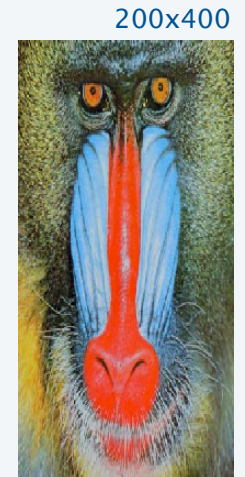
% java Scale mandrill.jpg 200 400

Picture client example: Scaling filter

Goal. Write a Java program to scale an image (arbitrarily and independently on *x* and *y*).

Ex. Downscaling by halving.
Shrink in half by deleting
alternate rows and columns.
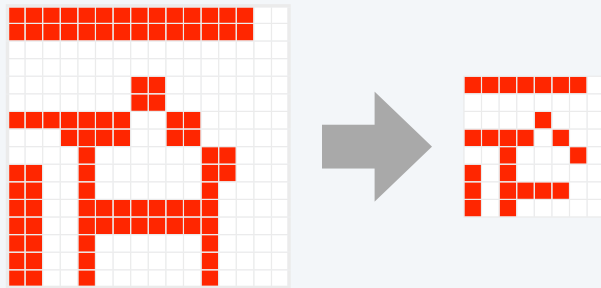
Ex. Upscaling by doubling.
Double in size by replacing
each pixel with four copies.
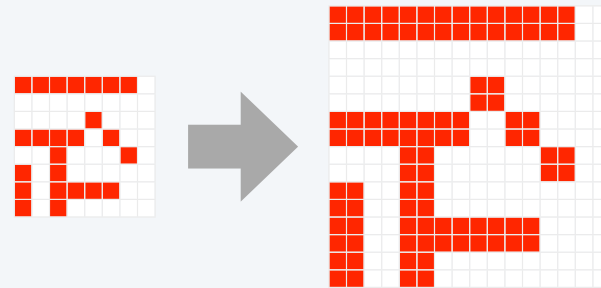
Picture client example: Scaling filter

Goal. Write a Java program to scale an image (arbitrarily and independently on *x* and *y*).

A uniform strategy to scale from ws-by-hs to wt-by-ht.
- Scale column index by ws/wt .
- Scale row index by hs/ht .

Approach. Arrange computation to compute exactly one value for each *target* pixel.
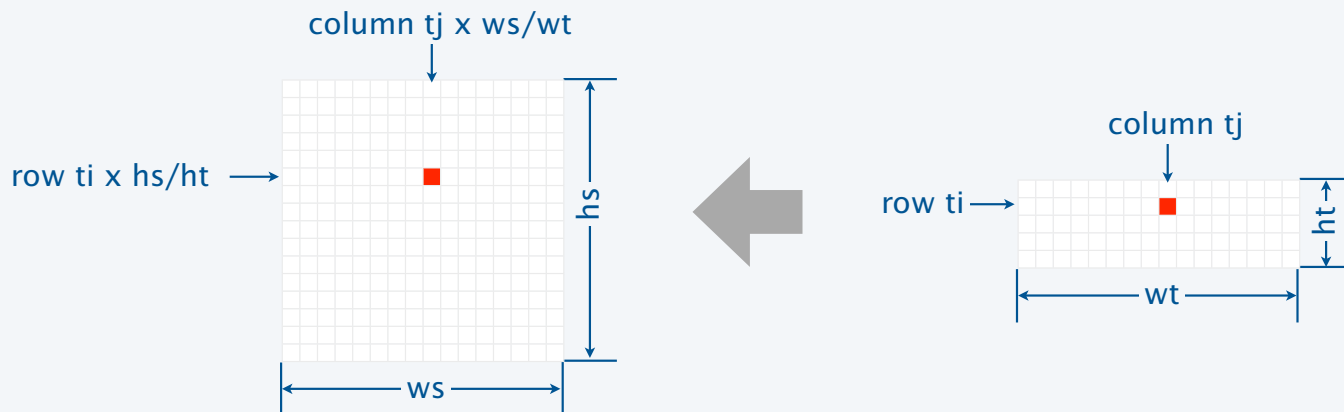
Picture client example: Scaling filter

```java
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);
         }
      target.show();
   }
}
```
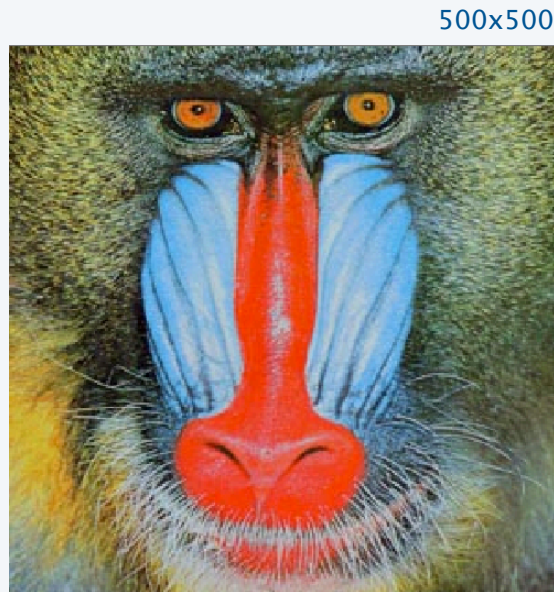
% java Scale mandrill.jpg 300 900
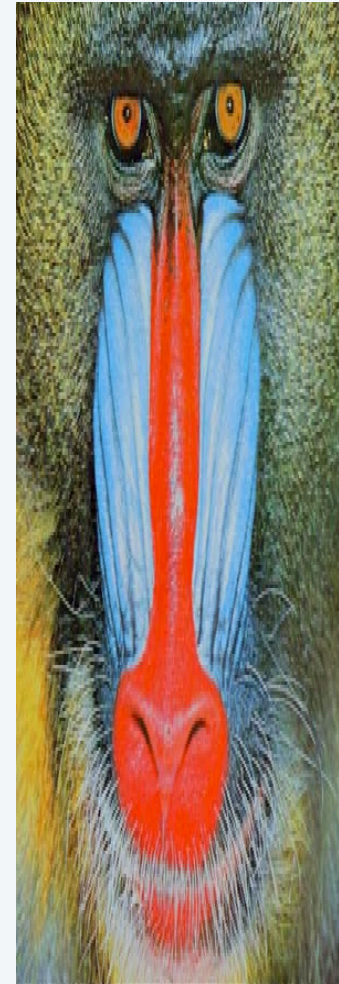
More image-processing effects
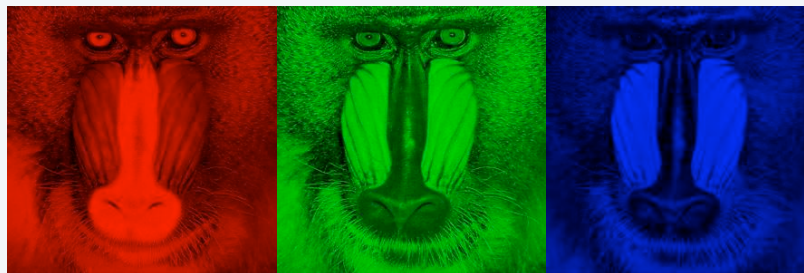
RGB color separation
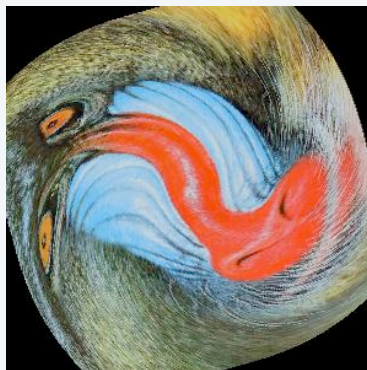
swirl filter   wave filter   glass filter   Sobel edge detection
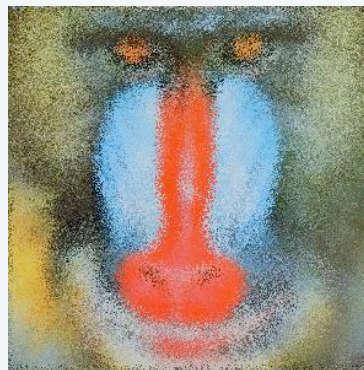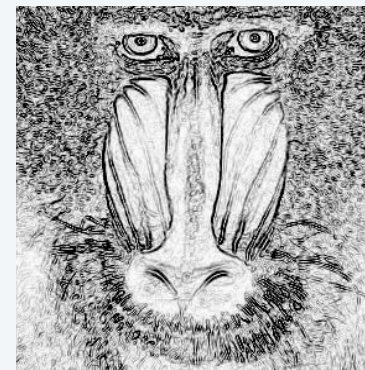
INTRODUCTION TO
Programming
in Java

An Interdisciplinary Approach

Robert Sedgewick · Kevin Wayne

http://introcs.cs.princeton.edu

9. Abstract Data Types

- Overview
- Color
- **Image processing**
- String processing

INTRODUCTION TO
Programming
in Java

An Interdisciplinary Approach

Robert Sedgewick · Kevin Wayne

http://introcs.cs.princeton.edu

9. Abstract Data Types

- Overview
- Color
- Image processing
- **String processing**

String ADT

A `String` is a sequence of Unicode characters. ←——— defined in terms of its ADT values (typical)

Java's ADT allows us to write Java programs that manipulate strings.

Operations (API)

public class String	
String(String s)	*create a string with the same value*
int length()	*string length*
char charAt(int i)	*ith character*
String substring(int i, int j)	*ith through (j-1)st characters*
boolean contains(String sub)	*does string contain sub?*
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?*

Programming with strings: typical examples

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;
}
```

Find occurrences of a specified string in StdIn

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

Search for *.edu hyperlinks in the text file on StdIn

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

String client example: Gene finding

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 of *codons* (three A C T G *nucleotides*).
- Preceded by ATG (*start* codon).
- Succeeded by TAG (*stop* codon). ⟵ simplified for lecture (TAA or TGA are also stop codons)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
A	T	A	G	A	T	G	C	A	T	A	G	C	G	C	A	T	A	G	C	T	A	G	A	T	G	T	G	C	T	A	G	C

⊢ *start* ⊣ ⊢ *gene* ⊣ ⊢ *stop* ⊣   ⊢ *start* ⊣ ⊢ *gene* ⊣ ⊢ *stop* ⊣
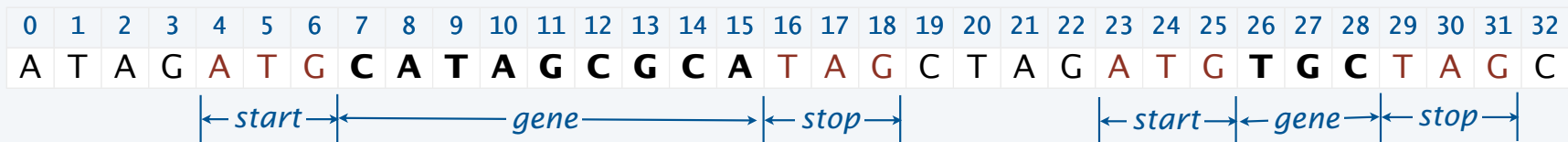
Goal. Write a Java program to find genes in a given genome.

String client example: Gene finding

Algorithm. Scan left-to-right through genome.
- If start codon ATG found, set beg to index i.
- If stop codon TAG found and substring length is a multiple of 3, print gene and reset beg to –1.

i	codon		beg	output	remainder of input string
	start	*stop*			
0			–1		ATAGATGCATAGCGCATAGCTAGATGTGCTAGC
1		TAG	–1		TAGATGCATAGCGCATAGCTAGATGTGCTAGC
4	ATG		4		ATGCATAGCGCATAGCTAGATGTGCTAGC
9		TAG	4		TAGCGCATAGCTAGATGTGCTAGC
16		TAG	4	CATAGCGCA	TAGCTAGATGTGCTAGC
20		TAG	–1		TAGATGTGCTAGC
23	ATG		23		ATGTGCTAGC
29		TAG	23	TGC	TAGC

String client example: Gene finding

```
public class GeneFind
{
   public static void main(String[] args)
   {
      String start  = args[0];
      String stop   = args[1];
      String genome = StdIn.readAll();

      int beg = -1;
      for (int i = 0; i < genome.length() - 2; i++)
      {
         String codon = genome.substring(i, i+3);
         if (codon.equals(start)) beg = i;
         if (codon.equals(stop) && beg != -1 && beg+3 < i)
         {
            String gene = genome.substring(beg+3, i);
            if (gene.length() % 3 == 0)
            {
               StdOut.println(gene);
               beg = -1;
            }
         }
      }
   }
}
```

Fixes bug in Program 3.1.8
TEQ 1: What's the bug?
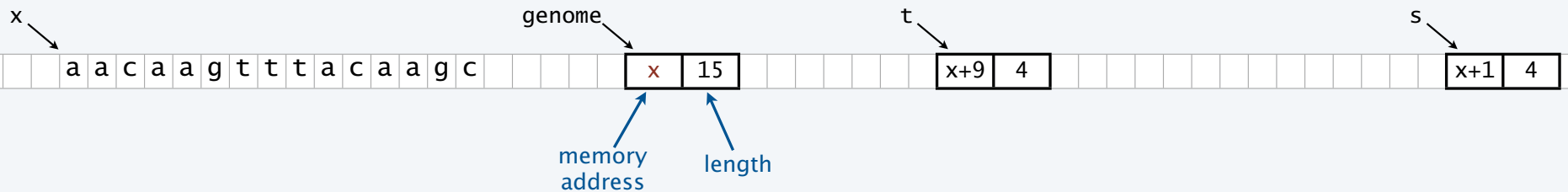TEQ 2: Give input that causes
       Program 3.1.8 to crash

```
% more genomeTiny.txt
ATAGATGCATAGCGCATAGCTAGATGTGCTAGC

% java GeneFind ATG TAG < genomeTiny.txt
CATAGCGCA
TGC
```
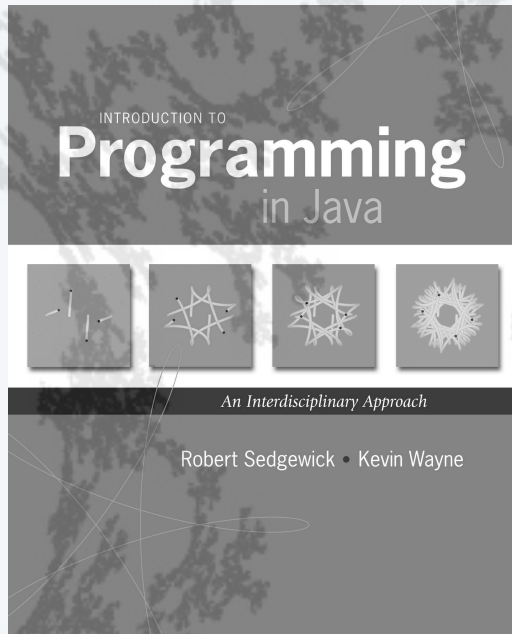
OOP context for strings

Possible memory representation of

```
String genome = "aacaagtttacaagc";

String s = genome.substring(1, 5);

String t = genome.substring(9, 13);
```


Implications for clients
- Substrings are different strings that share the same value "acaa".
- (s == t) is false (because it compares addresses).
- (s.equals(t)) is true (because it compares character sequences).
- Java String interface is more complicated than the API (and not really an ADT).

INTRODUCTION TO
Programming
in Java

An Interdisciplinary Approach

Robert Sedgewick · Kevin Wayne

9. Abstract Data Types

- Overview
- Color
- Image processing
- **String processing**

Object-oriented programming: summary

Object-oriented programming.
- Create your own data types (sets of values and ops on them).
- Use them in your programs (manipulate *objects*). ⟵ An object holds a data type value.
Variable names refer to objects.

In Java, programs manipulate references to objects.
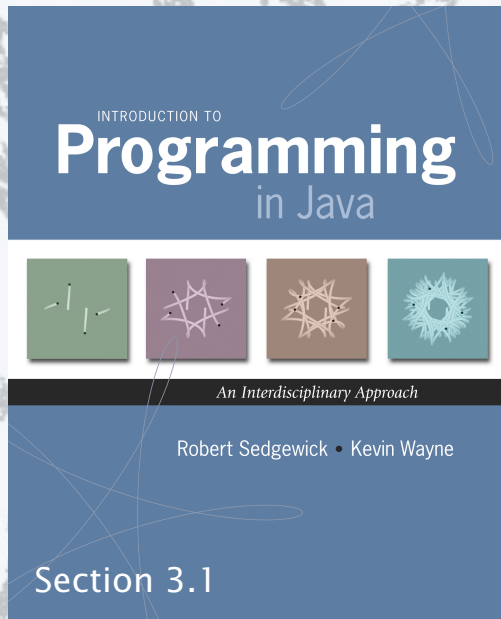- `String`, `Picture`, `Color`, arrays, (and everything else) are *reference types*.
- Exceptions: boolean, `int`, `double` and other *primitive types*.
- OOP purist: Languages should not have separate primitive types.
- Practical programmer: Primitive types provide needed efficiency.

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

This lecture: You can write programs to manipulate sounds, colors, pictures, and strings.
Next lecture: You can *define your own abstractions* and write programs that manipulate them.

INTRODUCTION TO
Programming
in Java

An Interdisciplinary Approach

Robert Sedgewick • Kevin Wayne

Section 3.1

9. Abstract Data Types