3. Object Oriented Programming

Data Types

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

Java has 8 primitive types.

- Language support for common operations: `+ - * / % && || !`
- `boolean`, `char`, `int`, `byte`, `short`, `long`, `float`, `double`

Data Type	Set of Values	Built-In Operations
`boolean`	`true`, `false`	not, and, or, xor
`int`	any of 2^{32} possible integers	add, subtract, multiply
`double`	any of 2^{64} possible reals	add, subtract, multiply

We want to write programs that process other types of data.

- Colors, pictures, strings, input streams.
- Complex numbers, matrices, rational numbers, polynomials, . . .
- Points, lines, rectangles, circles, spheres, . . .

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.

Data Type	*Set of Values*	*Operations*
`Color`	24 bits	get red component, brighten
`Picture`	2D array of colors	get/set color of pixel (i, j)
`String`	sequence of characters	length, substring, get ith char

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.

```
String s = new String("Hello");
int N    = s.length();
String t = s.substring(2, 4);
```

3.1. Using Data Types

Colors

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	

Colors

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.

Operations (partial): for `Color` variable `c`.

Operation	Description	Return type
`c.getRed()`	return red component	`int`
`c.getGreen()`	return green component	`int`
`c.getBlue()`	return blue component	`int`
`c.brighter()`	return brighter version of `c`	`Color`
`c.darker()`	return darker version of `c`	`Color`
`c.toString()`	return string representation of `c`	`String`

Reference: `http://java.sun.com/j2se/1.5.0/docs/api/java/awt/Color.html`

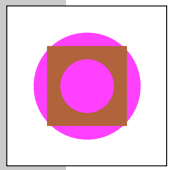
Using Colors in Java

One use of objects. Store aggregate data in one variable.

use import to access Java library

```
import java.awt.Color;

public class ColorTest {
   public static void main(String[] args) {
      StdDraw.create(600, 600);
      Color magenta = new Color(255,  0, 255);
      Color sienna  = new Color(160, 82,  45);
      StdDraw.moveTo(0.5, 0.5);
      StdDraw.setColor(magenta);
      StdDraw.spot(0.667);
      StdDraw.setColor(sienna);
      StdDraw.spot(0.5, 0.5);
      StdDraw.setColor(magenta);
      StdDraw.spot(0.333);
      StdDraw.show();
   }
}
```



Monochrome Luminance

Monochrome luminance (Y). Effective brightness of a color.

Application: Which font colors will be most readable with what background colors on TV and computer screens?

NTSC formula: grayscale = 0.2989r + 0.5870g + 0.1140b.

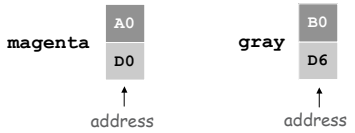
```
public static Color toGray(Color c) {
   int r = c.getRed();
   int g = c.getGreen();
   int b = c.getBlue();
   int luminance = (int) (0.2989*r + 0.5870*g + 0.1140*b);
   Color gray = new Color(luminance, luminance, luminance);
   return gray;
}
```

More uses. Pass colors to functions; return colors from functions.

OOP Context for Color

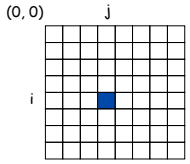
Possible memory representation.

D0	D1	D2	D3	D4	D5	D6	D7	D8
255	0	255	0	0	0	105	105	105



Bottom line: we are writing programs that manipulate color.

Raster Graphics



Raster graphics: basis for image processing.

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

Operations: for `Picture` variable `pic`.

Operation	Description	Return type
`pic.getHeight()`	return height of image	`int`
`pic.getWidth()`	return width of image	`int`
`pic.getColor(i, j)`	return color of pixel (i, j)	`Color`
`pic.setColor(i, j, c)`	set color of pixel (i, j) to c	`void`
`pic.show()`	display the picture in a window	`void`
`pic.save(s)`	save the picture to file s	`void`

Reference: `http://www.cs.princeton.edu/IntroCS/31datatype`

Image Processing: Monochrome Luminance

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

```
import java.awt.Color;

public class Grayscale {
   public static void main(String args[]) {
      Picture pic1 = new Picture(args[0]);
      int width  = pic1.width();
      int height = pic1.height();
      Picture pic2 = new Picture(width, height);
      for (int i = 0; i < width; i++) {
         for (int j = 0; j < height; j++) {
            Color c    = pic.getColor(i, j);
            Color gray = toGray(c);
            pic2.setColor(i, j, gray);
         }
      }
      pic1.show();
      pic2.show();
   }
}
```

Image Processing: Monochrome Luminance

Grayscale.java. Creates two `Picture` objects and windows.

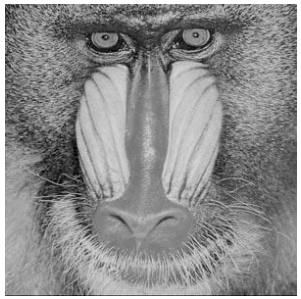


Image Processing: Swirl Filter

Swirl.java. Creates two `Picture` objects and windows.

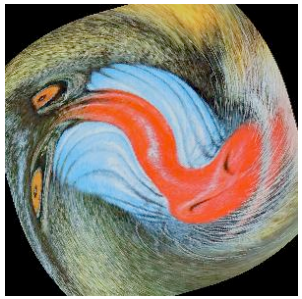


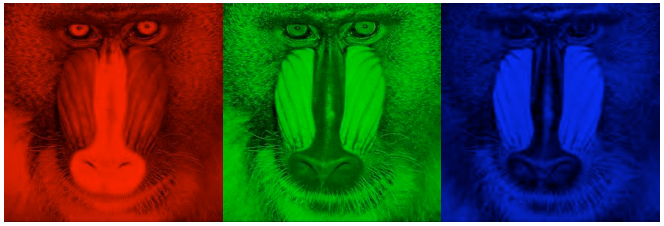
Image Processing: Swirl Filter

Goal. Create swirl special effect by setting color of output pixel (i, j) to color of some other input pixel (ii, jj).

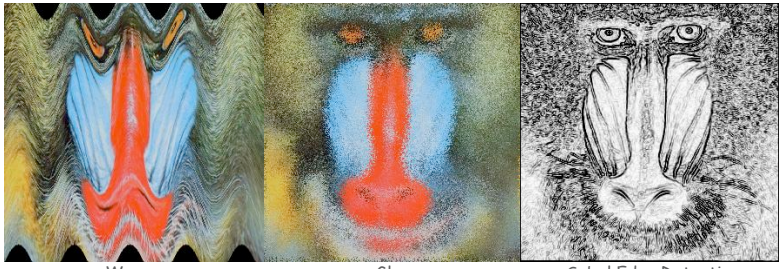
```
double i0 = 0.5 * width;
double j0 = 0.5 * height;

for (int i = 0; i < width; i++) {
   for (int j = 0; j < height; j++) {
      double di = i - i0;
      double dj = j - j0;
      double r = Math.sqrt(di*di +dj*dj);
      double a = Math.PI / 256 * r;
      int ii = (int)(-di*Math.cos(a) + dj*Math.sin(a) + i0);
      int jj = (int)( di*Math.sin(a) + dj*Math.cos(a) + j0);
      if (ii >= 0 && ii < width && jj >= 0 && jj < height)
         pic2.setColor(i, j, pic1.getColor(ii, jj));
   }
}
```


More Image Processing Effects



RGB color separation



Wave Glass Sobel Edge Detection

OOP Context for Picture

Immutability. Can't change a `Color` object's value once created.
Mutability. Can change a `Picture` object's value.

```
Color c = pic.getColor(0, 0);
pic.setColor(3, 0, c);
```

Object reference is analogous to variable name.

- We can manipulate the value that it holds.
- We can pass it to a function / method.

Strings

String data type: basis for text processing.
Set of values: sequence of Unicode characters.
Operations (partial): for `String` variable `s`.

Operation	Description	s	Return
`s.length()`	return length of `s`	`Hello`	`5`
`s.charAt(1)`	return char of `s` at index 1	`Hello`	`e`
`s.substring(1, 4)`	return substring of `s` from 1 (inclusive) to 4 (exclusive)	`Hello`	`ell`
`s.toUpperCase()`	return uppercase version of `s`	`Hello`	`HELLO`
`s.endsWith(".com")`	does `s` end with `.com` ?	`cnn.com`	`true`
`s.compareTo("Bye")`	return a positive integer, negative integer or zero based on string comparison	`Hello`	`-1`

Reference: `http://java.sun.com/j2se/1.5.0/docs/api/java/lang/String.html`

Sorting Strings

Analysis of algorithms lecture: quicksort array of real numbers.

To sort strings:

- Replace all occurrences of `double` with `String`.

```
static void exch(String[] a, int i, int j) {
   String swap = a[i];
   a[i] = a[j];
   a[j] = swap;
}
```

- Rewrite `less` to compare elements lexicographically.

```
static boolean less(String s, String t) {
   return (s.compareTo(t) < 0);
}
```


Redundancy Detector

Longest repeated substring. Given a string, find the longest substring that appears at least twice.

Brute force.

- Try all indices i and j for start of possible match.
- Compute longest common prefix for each pair (quadratic).

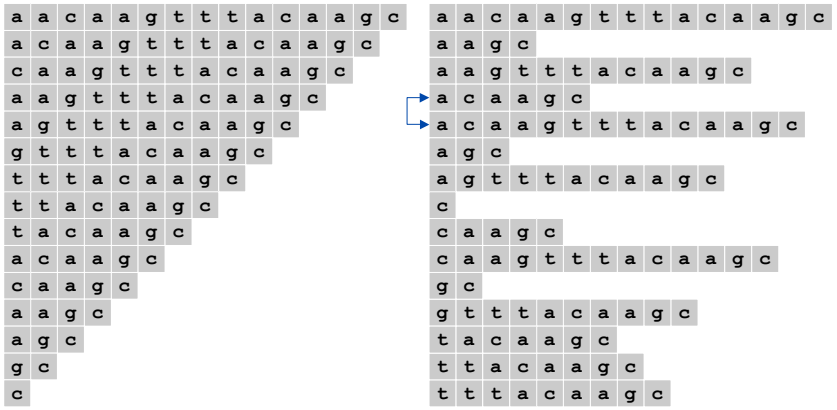
Applications.

- Computational molecular biology.
- Burrows-Wheeler transform for data compression.

Longest Repeated Substring: A Sorting Solution

A sorting solution.

- Form suffixes, and sort them to bring repeated substrings together.
- Compute longest prefix between N-1 adjacent suffixes.



Longest Repeated Substring: Java Implementation

Suffix sorting implementation.

```
int N = s.length();
String[] suffixes = new String[N];
for (int i = 0; i < N; i++)
   suffixes[i] = s.substring(i, N);
Arrays.sort(suffixes);
```

Longest repeated substring. Search only adjacent suffixes.

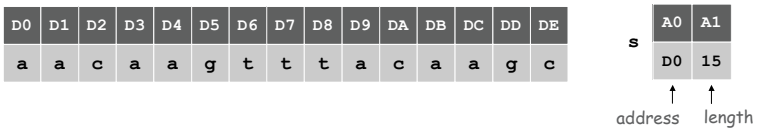
```
String lrs = "";
for (int i = 0; i < N-1; i++) {
   String x = lcp(suffixes[i], suffixes[i+1]);
   if (x.length() > lrs.length()) lrs = x;
}
```

LCP. Find the longest string that is a prefix of both s and t.
Ex: lcp("acaagtttac", "acaagc") = "acaag".

OOP Context for Strings

Possible memory representation of a string.

- `s = "aacaagtttacaagc";`



- `t = s.substring(5, 15);`

- No characters are copied when you invoke `substring` method!

Consequences.

- Calling `substring` takes constant time (instead of linear).
- Creating suffixes takes linear space (instead of quadratic).

Longest Repeated Substring: Analysis

Computational experiments.

Input File	Characters	Brute	Suffix Sort	Length
LRS.java	2,162	0.6 sec	0.14 sec	73
Amendments	18,369	37 sec	0.25 sec	216
Aesop's Fables	191,945	3958 sec	1.0 sec	58
Moby Dick	1.2 million	43 hours †	7.6 sec	79
Bible	4.0 million	20 days †	34 sec	11
Chromosome 11	7.1 million	2 months †	61 sec	12,567
Pi	10 million	4 months †	84 sec	14

† estimated

Lesson. Sorting to the rescue; enables new research.

Non-Standard Input

Standard input: from keyboard, or use OS to redirect from one file.
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?

```
public class Diff {
   public static void main(String[] args) {
      In in0 = new In(args[0]);
      In in1 = new In(args[1]);
      String s = in0.readAll();
      String t = in1.readAll();
      System.out.println(s.equals(t));
   }
}
```


Screen Scraping

Find current stock price of Google.

- `s.indexOf(t, i)`: index of 1st occurrence of pattern `t` in string `s`, starting at offset `i`.
- Read raw html from `http://finance.yahoo.com/q?s=goog` (NYSE symbol)
- Find 1st string delimited by `<b>` and `</b>` appearing after `Last Trade`

```
public class StockQuote {
   public static void main(String[] args) {
      String name = "http://finance.yahoo.com/q?s=" + args[0];
      In in = new In(name);
      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);
      System.out.println(price);
   }
}
```

```
% java StockQuote goog
131.00
```


Day Trader

Add bells and whistles.

- Plot price in real-time.
- Notify user if price dips below a certain price.
- Embed logic to determine when to buy and sell.
- Automatically send buy and sell orders to trading firm.

Warning. Use at your own financial risk.

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, strings, and input streams.

Next time. We'll learn how to create new data types so that we can write programs to manipulate our own abstractions.