3.1 Data Types

any program you might want to write

objects

functions and modules

graphics, sound, and image I/O

arrays

conditionals and loops

Math  text I/O

primitive data types  assignment statements

create your own data types
Abstract Data Types

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

**Abstract data type.** Data type whose representation is hidden from the user.

**Primitive types.**
- values directly map to machine representations
- operations directly translate 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}) to (2^{31} - 1)</td>
<td>add, subtract, multiply</td>
</tr>
<tr>
<td>double</td>
<td>any of (2^{64}) possible reals</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; variable name refers to object.

Object-oriented programming.
• Create your own data types (sets of values and ops on them)
• 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>get red component, brighten</td>
</tr>
<tr>
<td>Picture</td>
<td>2D array of colors</td>
<td>get/set color of pixel (i, j)</td>
</tr>
<tr>
<td>String</td>
<td>sequence of characters</td>
<td>length, substring, compare</td>
</tr>
</tbody>
</table>

Abstract data type (ADT). Object representation is hidden.

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
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 keyword `new` to invoke a “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 indicate an operation is to be applied
• Use a method name to specify which operation

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

<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>white</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>gray</td>
</tr>
</tbody>
</table>
**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.

**API** (Application Programming Interface) specifies *set of operations*.

```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?

[http://java.sun.com/j2se/1.5.0/docs/api/java/awt/Color.html](http://java.sun.com/j2se/1.5.0/docs/api/java/awt/Color.html)
Josef Albers. Revolutionized the way people think about color.

Homage to the Square by Josef Albers (1949-1975)
Josef Albers. Revolutionized the way people think about color.

% java AlbersSquares 9 90 166 100 100 100
import java.awt.Color;

to access Color library

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(.25, .5, .2);
        StdDraw.setPenColor(c2);
        StdDraw.filledSquare(.25, .5, .1);

        StdDraw.setPenColor(c2);
        StdDraw.filledSquare(.75, .5, .2);
        StdDraw.setPenColor(c1);
        StdDraw.filledSquare(.75, .5, .1);
    }
}
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;
    }
}
```
Q. Which font colors will be most readable with which background colors on computer monitors 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. 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;
}
```

<table>
<thead>
<tr>
<th>red</th>
<th>green</th>
<th>blue</th>
<th>this color</th>
<th>grayscale version</th>
<th>black</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>90</td>
<td>166</td>
<td></td>
<td>74 74 74</td>
<td>0 0 0</td>
</tr>
</tbody>
</table>

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

Bottom line. We are writing programs that manipulate color.
OOP Context for Color

Possible memory representation (in TOY).

<table>
<thead>
<tr>
<th>D0</th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
<th>D4</th>
<th>D5</th>
<th>D6</th>
<th>D7</th>
<th>D8</th>
</tr>
</thead>
<tbody>
<tr>
<td>255</td>
<td>0</td>
<td>255</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>105</td>
<td>105</td>
<td>105</td>
</tr>
</tbody>
</table>

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.
References

**René Magritte.** "This is not a pipe."

![Image of Rene Magritte's painting](image)

**Java.** This is not a color.

```java
Color sienna = new Color(160, 82, 45);
Color c = sienna.darker();
```

**OOP.** Natural vehicle for studying abstract models of the real world.
Picture Data Type

**Raster graphics.** Basis for image processing.

**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.

```java
import java.awt.Color;

public class Grayscale {
    public static void main(String[] args) {
        Picture pic = new Picture(args[0]);
        for (int x = 0; x < pic.width(); x++)
            for (int y = 0; y < pic.height(); y++)
                {
                    Color color = pic.get(x, y);
                    Color gray = Luminance.toGray(color);
                    pic.set(x, y, gray);
                }
        pic.show();
    }
}
```
Image Processing: Grayscale Filter

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

*mandrill.jpg*

```java
% java Grayscale mandrill.jpg
```
Image Processing Challenge 1

What does the following code do? (Easy question!)

```java
Picture pic = new Picture(args[0]);
for (int x = 0; x < pic.width(); x++)
    for (int y = 0; y < pic.height(); y++)
        pic.set(x, y, pic.get(x, y));
pic.show();
```
What does the following code do? (Hard question.)

```java
Picture pic = new Picture(args[0]);
for (int x = 0; x < pic.width(); x++)
    for (int y = 0; y < pic.height(); y++)
        pic.set(x, pic.height() - y - 1, pic.get(x, y));
pic.show();
```
What does the following code do? (Hard question.)

```java
Picture source = new Picture(args[0]);
int width  = source.width();
int height = source.height();
Picture target = new Picture(width, height);
for (int x = 0; x < width; x++)
    for (int y = 0; y < height; y++)
        target.set(x, height - y - 1, source.get(x, y));
target.show();
```
Image Processing: Scaling Filter

**Goal.** Shrink or enlarge an image to desired size.

**Downscaling.** To shrink in half, delete half the rows and columns.
**Upscaling.** To enlarge to double, replace each pixel by 4 copies.
**Goal.** Shrink or enlarge an image to desired size.

**Uniform strategy.** To convert from $w_s$-by-$h_s$ to $w_t$-by-$h_t$:

- Scale column index by $w_s / w_t$.
- Scale row index by $h_s / h_t$.
- Set color of pixel $(x, y)$ in target image to color of pixel $(x \times w_s / w_t, y \times h_s / h_t)$ in source image.
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 tx = 0; tx < w; tx++)
            for (int ty = 0; ty < h; ty++)
                {
                    int sx = tx * source.width() / w;
                    int sy = ty * source.height() / h;
                    Color color = source.get(sx, sy);
                    target.set(tx, ty, color);
            }
        source.show();
        target.show();
    }
}
Scaling filter. Creates two Picture objects and two windows.

mandrill.jpg

% java Scale mandrill.jpg 400 200
More Image Processing Effects

- wave filter
- glass filter
- Sobel edge detection

RGB color separation

- swirl filter
- wave filter
- glass filter
- Sobel edge detection
String Processing
String data type. Basis for text processing.
Set of values. Sequence of Unicode characters.

API:

public class String (Java string data type)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String(String s)</td>
<td>create a string with the same value as s</td>
</tr>
<tr>
<td>int length()</td>
<td>string length</td>
</tr>
<tr>
<td>char charAt(int i)</td>
<td>i\textsuperscript{th} character</td>
</tr>
<tr>
<td>String substring(int i, int j)</td>
<td>i\textsuperscript{th} through (j-1)\textsuperscript{st} characters</td>
</tr>
<tr>
<td>boolean contains(String sub)</td>
<td>does string contain \texttt{sub} as a substring?</td>
</tr>
<tr>
<td>boolean startsWith(String pre)</td>
<td>does string start with \texttt{pre}?</td>
</tr>
<tr>
<td>boolean endsWith(String post)</td>
<td>does string end with \texttt{post}?</td>
</tr>
<tr>
<td>int indexOf(String p)</td>
<td>index of first occurrence of \texttt{p}</td>
</tr>
<tr>
<td>int indexOf(String p, int i)</td>
<td>index of first occurrence of \texttt{p} after \texttt{i}</td>
</tr>
<tr>
<td>String concat(String t)</td>
<td>this string with \texttt{t} appended</td>
</tr>
<tr>
<td>int compareTo(String t)</td>
<td>string comparison</td>
</tr>
<tr>
<td>String replaceAll(String a, String b)</td>
<td>result of changing as to \texttt{bs}</td>
</tr>
<tr>
<td>String[] split(String delim)</td>
<td>strings between occurrences of \texttt{delim}</td>
</tr>
<tr>
<td>boolean equals(String t)</td>
<td>is this string's value the same as \texttt{t}'s?</td>
</tr>
</tbody>
</table>

http://java.sun.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); |
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.

- Preceded by ATG. [start codon]
- Multiple of 3 nucleotides. [codons other than start/stop]
- Succeeded by TAG, TAA, or TGA. [stop codons]
Gene Finding: Algorithm

Algorithm. Scan left-to-right through genome.

- If start codon found, then set \( \text{beg} \) to index \( i \).

- If stop codon found \textbf{and} \( \text{beg} \neq -1 \) \textbf{and} substring is a multiple of 3
  - output gene
  - reset \( \text{beg} \) to -1

<table>
<thead>
<tr>
<th>( i )</th>
<th>codon</th>
<th>( \text{beg} )</th>
<th>gene</th>
<th>remaining portion of input string</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>-1</td>
<td>ATGATGCATAGCGC CATAGCTAGATGTGCTAGC</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>TAG</td>
<td>-1</td>
<td>ATAGATGCATAGCGC CATAGCTAGATGTGCTAGC</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>ATG</td>
<td>4</td>
<td>ATAGATGCATAGCGC CATAGCTAGATGTGCTAGC</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>TAG</td>
<td>4</td>
<td>ATAGATGCATAGCGC CATAGCTAGATGTGCTAGC</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>TAG</td>
<td>4</td>
<td>CATAGCGCA CATAGCTAGATGTGCTAGC</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>TAG</td>
<td>-1</td>
<td>ATAGATGCATAGCGC CATAGCTAGATGTGCTAGC</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>ATG</td>
<td>23</td>
<td>ATAGATGCATAGCGC CATAGCTAGATGTGCTAGC</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>TAG</td>
<td>23</td>
<td>TGC</td>
<td>ATAGATGCATAGCGC CATAGCTAGATGTGCTAGC</td>
</tr>
</tbody>
</table>
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;
                }
            }
        }
    }
}
Possible memory representation of a string (using TOY addresses).

- genome = "aacaagtttacaagc"

- \( s = \text{genome.substring}(1, 5); \)
- \( t = \text{genome.substring}(9, 13); \)

\( s \) and \( t \) are different strings that have the same value: "acaa"

\( (s == t) \) is false, but \( (s.equals(t)) \) is true!
Input and Output
Bird's Eye View (Re-Revisited)
Non-Standard Input

Standard input. Read from terminal window.

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?

```java
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();
        StdOut.println(s.equals(t));
    }
}
```
Goal. Find current stock price of Google.

Step 1. Find web source.


NYSE symbol
Screen Scraping

**Goal.** Find current stock price of Google.

**Step 2.** Find string representation (HTML code) of web source.

```html
<h3 class=""><span itemprop="tickersymbol">GOOG</span>:US </h3>
<span class="price">
<meta itemprop="price" content="559.500" />
559.500
<span itemprop="priceCurrency">USD</span>
</span>
```

price is string between `content=559.500` and next `>`, after `tickersymbol`
Screen Scraping

**Goal.** Find current stock price of Google.

**Step 3.** Write code to extract stock price from HTML code.

```java
public class StockQuote {
    public static void main(String[] args) {
        String name = "http://www.bloomberg.com/quote/";
        In in = new In(name + args[0] + ":US");
        String input = in.readAll();
        int start = input.indexOf("tickersymbol", 0);
        int from = input.indexOf("content=", start);
        int to = input.indexOf("/>", from);
        String price = input.substring(from + 9, to - 2); //small fiddles
        StdOut.println(price);
    }
}
```

```
% java-introcs StockQuote GOOG
559.500
```

- `s.indexOf(t, i)`: index of first occurrence of `t` in `s`, starting at offset `i`.
- Find string delimited (with small fiddles) by `content=` and `/>`, after `tickersymbol`.
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.
Today, you saw how to to write programs that manipulate colors, pictures, strings, and I/O streams.

Next time.
You will learn to define your own abstractions and to write programs that manipulate them.