3.1 Using Data Types
A Foundation for Programming

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
Data Types

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

Primitive types. Values directly map to machine representation; ops directly map 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 (set 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>

This lecture. Use existing data types.
Next lecture. Create your own data types.
Constructors and Methods

To construct a new object:
- 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).

```
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.

```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://download.oracle.com/javase/6/docs/api/java/awt/Color.html
Monochrome Luminance

Monochrome luminance. Effective brightness of a color.

NTSC formula. \( Y = 0.299r + 0.587g + 0.114b. \)

```java
import java.awt.Color;

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;
    }
}
```
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 $\geq 128$.

```java
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.

```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>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.299 * 9 + 0.587 * 90 + 0.114 * 166 = 74.445</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Bottom line.** We are writing programs that manipulate color.
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."

Java. This is not a color.

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

OOP. Natural vehicle for studying abstract models of the real world.
This is Not a Pipe

Neither is this.

% java RandomSeq 10000 | java Average

Dan Piraro, http://www.uexpress.com
Picture Data Type

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

**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.

```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.

```
% java Grayscale mandrill.jpg
```

mandrill.jpg

% java Grayscale mandrill.jpg
**Image Processing: Scaling Filter**

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

**Downscaling.** To shrink, delete half the rows and columns.

**Upscaling.** To enlarge, replace each pixel by 4 copies.
**Image Processing: Scaling Filter**

**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 $\frac{w_s}{w_t}$.
- Scale row index by $\frac{h_s}{h_t}$.
- Set color of pixel $(x, y)$ in target image to color of pixel $(x \times \frac{w_s}{w_t}, y \times \frac{h_s}{h_t})$ 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 < target.width(); tx++) {
            for (int ty = 0; ty < target.height(); ty++) {
                int sx = tx * source.width() / target.width();
                int sy = ty * source.height() / target.height();
                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
(298-by-298)

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

RGB color separation

swirl filter
wave filter
glass filter
Sobel edge detection
Text Processing
## String Data Type

**String data type.** Basis for text processing.  
**Set of values.** Sequence of Unicode characters.

### API.

```java
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>ith character</td>
</tr>
<tr>
<td>String substring(int i, int j)</td>
<td>ith through (j-1)st characters</td>
</tr>
<tr>
<td>boolean contains(String sub)</td>
<td>does string contain sub as a substring?</td>
</tr>
<tr>
<td>boolean startsWith(String pre)</td>
<td>does string start with pre?</td>
</tr>
<tr>
<td>boolean endsWith(String post)</td>
<td>does string end with post?</td>
</tr>
<tr>
<td>int indexOf(String p)</td>
<td>index of first occurrence of p</td>
</tr>
<tr>
<td>int indexOf(String p, int i)</td>
<td>index of first occurrence of p after i</td>
</tr>
<tr>
<td>String concat(String t)</td>
<td>this string with 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 bs</td>
</tr>
<tr>
<td>String[] split(String delim)</td>
<td>strings between occurrences of delim</td>
</tr>
<tr>
<td>boolean equals(String t)</td>
<td>is this string’s value the same as t’s?</td>
</tr>
</tbody>
</table>

[http://download.oracle.com/javase/6/docs/api/java/lang/String.html](http://download.oracle.com/javase/6/docs/api/java/lang/String.html)
## Typical String Processing Code

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Code</th>
</tr>
</thead>
</table>
| **is the string a palindrome?** | ```java
def isPalindrome(s):
    N = len(s)
    for i in range(N//2):
        if s[i] != s[N-1-i]:
            return False
    return True
``` |
| **extract file name and extension from a command-line argument** | ```java
s = args[0]
dot = s.indexof(".")
base = s.substr(0, dot)
extension = s.substr(dot + 1, s.length())
``` |
| **print all lines in standard input that contain a string specified on the command line** | ```java
query = args[0]
while not StdIn.isEmpty():
    s = StdIn.readLine()
    if s.contains(query):
        StdOut.println(s)
``` |
| **print all the hyperlinks (to educational institutions) in the text file on standard input** | ```java
while not StdIn.isEmpty():
    s = StdIn.readString()
    if s.startsWith("http://") and 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]

Goal. Find all genes.
Gene Finding: Algorithm

**Algorithm.** Scan left-to-right through genome.

- If start codon, then set $\text{beg}$ to index $i$.
- If stop codon 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>ATAGATGCATAGCGCATAGCTAGATGTGCTAGC</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>TAG</td>
<td>-1</td>
<td>ATAGATGCATAGCGCATAGCTAGATGTGCTAGC</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>ATG</td>
<td>4</td>
<td>ATAGATGCATAGCGCATAGCTAGATGTGCTAGC</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>TAG</td>
<td>4</td>
<td>ATAGATGCATAGCGCATAGCTAGATGTGCTAGC</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>TAG</td>
<td>4</td>
<td>CATAGCGCA</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>TAG</td>
<td>-1</td>
<td>ATAGATGCATAGCGCATAGCTAGATGTGCTAGC</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>ATG</td>
<td>23</td>
<td>ATAGATGCATAGCGCATAGCTAGATGTGCTAGC</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>TAG</td>
<td>23</td>
<td>TGC</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ATAGATGCATAGCGCATAGCTAGATGTGCTAGC</td>
<td></td>
</tr>
</tbody>
</table>
Gene Finding: Implementation

```java
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;
                }
            }
        }
    }
}
```

```
% more genomeTiny.txt
ATAGATGCATAGCGCATAGCTAGATGTGCTAGC

% java GeneFind ATG TAG < genomeTiny.txt
CATAGCGCA
TGC
```
Possible memory representation of a string.

```
genome = "aacaagtttacaagc";
```

| D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | DA | DB | DC | DD | DE |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| a  | a  | c  | a  | a  | g  | t  | t  | t  | a  | c  | a  | a  | g  | c  |

genome

<table>
<thead>
<tr>
<th>A0</th>
<th>A1</th>
</tr>
</thead>
<tbody>
<tr>
<td>D0</td>
<td>15</td>
</tr>
</tbody>
</table>

memory address

length
In and Out
Bird's Eye View (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]);  // read from one file
        In in1 = new In(args[1]);  // read from another file
        String s = in0.readAll();
        String t = in1.readAll();
        StdOut.println(s.equals(t));
    }
}
```
**Goal.** Find current stock price of Google.

```html
... <tr> <td class="yfnc_tablehead1" width="48%"> Last Trade: </td> <td class="yfnc_tabledata1"> <big> <b>576.50</b> </big> </td> </tr> ... <tr> <td class="yfnc_tablehead1" width="48%"> Trade Time: </td> <td class="yfnc_tabledata1"> 11:45AM ET </td> </tr> ...
```

http://finance.yahoo.com/q?s=goog
Screen Scraping

Goal. Find current stock price of Google.

- `s.indexOf(t, i)`: index of first occurrence of pattern `t` in string `s`, starting at offset `i`.
- Find first string delimited by `<b>` and `</b>` after `Last Trade`.

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

% java StockQuote goog
576.50
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. Please, please use at your own financial risk.

The New Yorker, September 6, 1999
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.
Extra Slides
import java.awt.Color;

public class ColorSeparation {
    public static void main(String args[]) {
        Picture pic = new Picture(args[0]);
        int width = pic.width();
        int height = pic.height();

        Picture R = new Picture(width, height);
        Picture G = new Picture(width, height);
        Picture B = new Picture(width, height);

        for (int x = 0; x < width; x++) {
            for (int y = 0; y < height; y++) {
                Color c = pic.get(x, y);
                int r = c.getRed();
                int g = c.getGreen();
                int b = c.getBlue();
                R.set(x, y, new Color(r, 0, 0));
                G.set(x, y, new Color(0, g, 0));
                B.set(x, y, new Color(0, 0, b));
            }
        }
        R.show();
        G.show();
        B.show();
    }
}
Color Separation

*ColorSeparation.java.* Creates three *picture* objects and windows.
Memory Management

Value types.
- Allocate memory when variable is declared.
- Can reclaim memory when variable goes out of scope.

Reference types.
- Allocate memory when object is created with `new`.
- Can reclaim memory when last reference goes out of scope.
- Significantly more challenging if several references to same object.

Garbage collector. System automatically reclaims memory; programmer relieved of tedious and error-prone activity.