CS 112 Introduction to Programming
(Spring 2012)

Lecture #18: Using Data Types
Zhong Shao

Department of Computer Science
Yale University
Office: 314 Watson

http://flint.cs.yale.edu/cs112

Acknowledgements: some slides used in this class are taken directly or adapted from those accompanying the textbook: Introduction to Programming in Java: An Interdisciplinary Approach by Robert Sedgewick and Kevin Wayne (Copyright 2002-2010)
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. Ops 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.

Impact. Enables us to create our own data types; define operations on them; and integrate into our programs.

<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>
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.

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

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

### API.

```java
class String {
    // (Java string data type)

    String(String s)
    int length()
    char charAt(int i)
    String substring(int i, int j)
    boolean contains(String sub)
    boolean startsWith(String pre)
    boolean endsWith(String post)
    int indexOf(String p)
    int indexOf(String p, int i)
    String concat(String t)
    int compareTo(String t)
    String replaceAll(String a, String b)
    String[] split(String delim)
    boolean equals(String t)
}
```

- `create a string with the same value as s`
- `string length`
- `i\text{th} \text{ character}`
- `i\text{th} \text{ through} \ (j-1)\text{st} \text{ characters}`
- `does \text{ string contain } \text{ sub as a substring?}`
- `does \text{ string start with } \text{ pre?}`
- `does \text{ string end with } \text{ post?}`
- `index of first occurrence of p`
- `index of first occurrence of p after i`
- `this \text{ string with } t \text{ appended}`
- `string \text{ comparison}`
- `result of changing as to bs`
- `strings between occurrences of delim`
- `is \text{ this string's value the same as } t's?`

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

### Extract file name and extension from a command-line argument

```java
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;
}
```

```java
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

```java
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

```java
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 \textbf{ATG}. [start codon]
- Multiple of 3 nucleotides. [codons other than start/stop]
- Succeeded by \textbf{TAG}, \textbf{TAA}, or \textbf{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>start stop</td>
<td>-1</td>
<td>ATAGATGCATAAGCAGCATAGCTAGATGTGCTAGC</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>TAG</td>
<td>-1</td>
<td>ATAGATGCATAAGCAGCATAGCTAGATGTGCTAGC</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>ATG</td>
<td>4</td>
<td>ATAGATGCATAAGCAGCATAGCTAGATGTGCTAGC</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>TAG</td>
<td>4</td>
<td>ATAGATGCATAAGCAGCATAGCTAGATGTGCTAGC</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>TAG</td>
<td>4</td>
<td>CATAGCGCA</td>
<td>ATAGATGCATAAGCAGCATAGCTAGATGTGCTAGC</td>
</tr>
<tr>
<td>20</td>
<td>TAG</td>
<td>-1</td>
<td>ATAGATGCATAAGCAGCATAGCTAGATGTGCTAGC</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>ATG</td>
<td>23</td>
<td>ATAGATGCATAAGCAGCATAGCTAGATGTGCTAGC</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>TAG</td>
<td>23</td>
<td>TGC</td>
<td>ATAGATGCATAAGCAGCATAGCTAGATGTGCTAGC</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) {
                String gene = genome.substring(beg+3, i);
                if (gene.length() % 3 == 0) {
                    StdOut.println(gene);
                    beg = -1;
                }
            }
        }
    }
}

% more genomeTiny.txt
ATAGATGCATAGCGCATAGCTAGCTAGCTAGC

% java GeneFind ATG TAG < genomeTiny.txt
CATAGCGCA
TGC
### OOP Context for Strings

Possible memory representation of a string.

- `genome = "aacaagtttacaagc";`

<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>
<th>D9</th>
<th>DA</th>
<th>DB</th>
<th>DC</th>
<th>DD</th>
<th>DE</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
<td>c</td>
<td>a</td>
<td>a</td>
<td>g</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>a</td>
<td>c</td>
<td>a</td>
<td>a</td>
<td>g</td>
<td>c</td>
</tr>
</tbody>
</table>

- `genome` length memory address

- D0 15
OOP Context for Strings

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.length = 15

- s = genome.substring(1, 5);
- t = genome.substring(9, 13);

- (s == t) is false, but (s.equals(t)) is true.

s and t refer to different strings that have the same value "acaa"
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>![RGB color]</td>
</tr>
<tr>
<td>0</td>
<td>255</td>
<td>0</td>
<td>![RGB color]</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>255</td>
<td>![RGB color]</td>
</tr>
<tr>
<td>255</td>
<td>255</td>
<td>255</td>
<td>![RGB color]</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>![RGB color]</td>
</tr>
<tr>
<td>255</td>
<td>0</td>
<td>255</td>
<td>![RGB color]</td>
</tr>
<tr>
<td>105</td>
<td>105</td>
<td>105</td>
<td>![RGB color]</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
Albers Squares

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
Using Colors in Java

```java
import java.awt.Color;

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;
    }
}
```
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>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>90</td>
<td>166</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>grayscale version</th>
</tr>
</thead>
<tbody>
<tr>
<td>74</td>
<td>74</td>
<td>74</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>black</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

\[0.299 \times 9 + 0.587 \times 90 + 0.114 \times 166 = 74.445\]

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

Possible memory representation.

<table>
<thead>
<tr>
<th></th>
<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></td>
<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."

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
}
```
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
% 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.

```
downscaling
   source
   target

upscaling
   source
   target
```
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 \(w_s / w_t\).
- 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.

![Diagram of image scaling process](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();
    }
}
Image Processing: Scaling Filter

Scaling filter. Creates two Picture objects and two windows.

```
mandrill.jpg
(298-by-298)

% java Scale mandrill.jpg 400 200
```
In and Out
Bird's Eye View (Revisited)

input streams
standard input

pictures

drawings

command-line arguments

output streams
standard output
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));
    }
}
```
**Screen Scraping**

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

```html
...<tr><td class="yfnc_tablehead1" width="48%"><b>Last Trade: </b></td><td class="yfnc_tabledata1"><b>576.50</b></td></tr><tr><td class="yfnc_tablehead1" width="48%"><b>Trade Time: </b></td><td class="yfnc_tabledata1">11:45AM ET</td></tr>
...  
http://finance.yahoo.com/q?s=goog
```

NYSE symbol
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
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. 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.