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 were 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

functions and modules
arrays
graphics, sound, and image I/O
notations and loops
Math
text I/O
primitive data types
assignment statements

newtype

objects

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^n ) to (2^n - 1)</td>
<td>add, subtract, multiply</td>
</tr>
<tr>
<td>double</td>
<td>any of (2^{int}) 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.

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

String Data Type

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

API

```
public class String {
    // Java string data type
    // create a string with the same value as $ s $.
    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 s);
    int indexOf(String s, int i);
    int compareTo(String s);
    String replaceAll(String a, String b);
    String[] split(String delimiters);
    boolean equals(String s);
    // end of string methods
}
```

Text Processing

String Processing Code

```
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;
}
```

Typical String Processing Code

```
String query = args[0];
String base = s.substring(0, del1);
String extension = s.substring(del1 + 1, s.length());
```

```
String s = StdIn.readString();
if (s.contains(query)) StdOut.println(s);
```

```
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.
- Multiple of 3 nucleotides.
- Succeeded by TAG, TAA, or TGA.

Goal. Find all genes.

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

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

Gene Finding: Implementation

Possible memory representation of a string.
- genome = "aacaagttaaacagc";

```
 genome
0 1 2 3 4 5 6 7 8 9 10 11 12 13
a a c a g t t a c a a c g
```

OOP Context for Strings

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>start</th>
<th>stop</th>
<th>beg</th>
<th>gene</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>TAG</td>
<td></td>
<td></td>
<td>-3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>ATG</td>
<td></td>
<td></td>
<td>4</td>
<td>multiple of 3</td>
</tr>
<tr>
<td>9</td>
<td>TAG</td>
<td></td>
<td></td>
<td>4</td>
<td>CATAGCCA</td>
</tr>
<tr>
<td>16</td>
<td>TAG</td>
<td></td>
<td></td>
<td>4</td>
<td>CATAGCCA</td>
</tr>
<tr>
<td>20</td>
<td>TAG</td>
<td></td>
<td></td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>ATG</td>
<td></td>
<td></td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>TAG</td>
<td></td>
<td></td>
<td>23</td>
<td>TOC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
OOP Context for Strings

Possible memory representation of a string.

- genome = "aacaagtttacaagc";
- 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".

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>0</td>
<td>Black</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Transparent</td>
</tr>
<tr>
<td>255</td>
<td>0</td>
<td>255</td>
<td>Pink</td>
</tr>
<tr>
<td>105</td>
<td>105</td>
<td>105</td>
<td>Purple</td>
</tr>
</tbody>
</table>

Image Processing

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 {
    // Constructor and methods...
    Color(int r, int g, int b) {
        // Initialize color
    }
    int getRed() {
        // Get red intensity
    }
    int getGreen() {
        // Get green intensity
    }
    int getBlue() {
        // Get 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?
    }
}
```

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)

```java
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(.75, .5, .2);

        StdDraw.setPenColor(c1);
        StdDraw.filledSquare(.25, .5, .1);
        StdDraw.setPenColor(c2);
        StdDraw.filledSquare(.75, .5, .1);
    }
}
```

Monochrome Luminance

Monochrome luminance. Effective brightness of a color.

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

```java
public class Luminance {
    public static double lum(Color c) {
        int r = c.getRed();
        int g = c.getGreen();
        int b = c.getBlue();
        return 0.299*r + 0.587*g + 0.114*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 \( \geq 128 \).

```
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.

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

Bottom line. We are writing programs that manipulate color.

**OOP Context for Color**

Possible memory representation.

```
<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."

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.
**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: 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.
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.

```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 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.

In and Out
Non-Standard Input

Goal. Read from terminal window.

Standard input. 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.readLine();
        String t = in1.readLine();
        StdOut.println(s.equals(t));
    }
}
```

or use OS to redirect from one file

Screen Scraping

Goal. Find current stock price of Google.

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

```
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.readLine();
        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.

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, and strings.

Next time. We’ll write programs that manipulate our own abstractions.