3.1 Using 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).

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).

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.

```
R   G   B   Color
255 0  0  red
0   255 0  green
255 255 0  blue
0   0   255 cyan
255 0  255 magenta
0   255 255 yellow
```

Monochrome Luminance

Monochrome luminance. Effective brightness of a color.

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

```
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 0.299 * r + 0.587 * g + 0.114 * b;
    }
}
```

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;
}
```

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 \( \geq 128 \).

```
255 208 105 47 23 17
```

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.

OOP Context for Color
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

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

public class Picture

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.

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.

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.

**Uniform strategy.**
1. Scale column index by $w_s / w_t$.
2. Scale row index by $h_s / h_t$.
3. 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();
    }
}
```

**More Image Processing Effects**

- Wave filter
- Glass filter
- Sobel edge detection
- RGB color separation

**Text Processing**

**String data type.** Basis for text processing.

Set of values: Sequence of Unicode characters.

```java
public class String {
    String(String s) { ... }
    int length() { ... }
    char charAt(int i) { ... }
    String substring(int f, int t) { ... }
    boolean contains(String sub) { ... }
    boolean startsWith(String pre) { ... }
    boolean endsWith(String post) { ... }
    int indexOf(String g) { ... }
    int lastIndexOf(String g) { ... }
    int compareTo(String o) { ... }
    String replaceAll(String s, String b) { ... }
    String[] split(String del) { ... }
    boolean equals(String s) { ... }
}

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

```
public class Genomics {
  public static void main(String[] args) {
    String genome = readAll();
    System.out.println("Gene Find: ");
    for (int i = 0; i < genome.length() - 2; i++) {
      if (genome.substring(i, i + 3).equals("ATG") && genome.substring(i + 3, i + 6).equals("TAG")) {
        System.out.println("Gene: ");
      }
    }
  }
}
```

Gene Finding: Algorithm

Algorithm. Scan left-to-right through genome.
- If start codon, then set beg to index i.
- If stop codon and substring is a multiple of 3
  - output gene.
  - reset beg to -1.

```
<table>
<thead>
<tr>
<th>codon</th>
<th>start</th>
<th>beg</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATG</td>
<td>0</td>
<td>-3</td>
</tr>
<tr>
<td>TAG</td>
<td>1</td>
<td>-3</td>
</tr>
<tr>
<td>TAC</td>
<td>4</td>
<td>-4</td>
</tr>
<tr>
<td>TAG</td>
<td>4</td>
<td>-3</td>
</tr>
<tr>
<td>TAC</td>
<td>7</td>
<td>-4</td>
</tr>
<tr>
<td>TGG</td>
<td>10</td>
<td>-3</td>
</tr>
<tr>
<td>TCA</td>
<td>13</td>
<td>-3</td>
</tr>
<tr>
<td>TAA</td>
<td>16</td>
<td>-3</td>
</tr>
<tr>
<td>TGA</td>
<td>19</td>
<td>-3</td>
</tr>
<tr>
<td>TGC</td>
<td>22</td>
<td>-3</td>
</tr>
<tr>
<td>CTC</td>
<td>25</td>
<td>-3</td>
</tr>
<tr>
<td>CTG</td>
<td>28</td>
<td>-3</td>
</tr>
</tbody>
</table>
```

OOP Context for Strings

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

```
<table>
<thead>
<tr>
<th>genome</th>
</tr>
</thead>
<tbody>
<tr>
<td>a c a a g t t a c a a g c</td>
</tr>
</tbody>
</table>
```

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

```
<table>
<thead>
<tr>
<th>genome</th>
</tr>
</thead>
<tbody>
<tr>
<td>a c a a g t t a c a a g c</td>
</tr>
</tbody>
</table>
```

Gene Finding: Implementation

```
public class Genomics {
  public static void main(String[] args) {
    String genome = readAll();
    System.out.println("Gene Find: ");
    for (int i = 0; i < genome.length() - 2; i++) {
      if (genome.substring(i, i + 3).equals("ATG") && genome.substring(i + 3, i + 6).equals("TAG")) {
        System.out.println("Gene: ");
      }
    }
  }
}
```
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]);
        In in1 = new In(args[1]);
        String s = in0.readLine();
        String t = in1.readLine();
        StdOut.println(s.equals(t));
    }
}
```

Screen Scraping

Goal. Find current stock price of Google.

```java
public class StockQuote {
   public static void main(String[] args) {
       String name = "http://finance.yahoo.com/q?t=";
       In in = new In(name + args[0]);
       String input = in.readLine();
       int start = input.indexOf("Last Trade:");
       int from = input.indexOf("<b>" , start);
       int to = input.indexOf("</b>", from);
       String price = input.substring(from + 3, to);
       StdOut.println(price);
   }
}
```

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.

Extra Slides

Color Separation

```java
import java.awt.Color;
public class ColorSeparation {
    public static void main(String args[]) {
        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. 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.