3.1 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 $2^n$ 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).

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

Objects

<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 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

- declare a variable (object name)
- call a constructor to create an object
- statement
- call a method that operates on the object's value
Color Data Type

Color. A sensation in the eye from electromagnetic radiation.

Set of values. [RGB representation] 256³ 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?
}
```

Albers Squares

Josef Albers. Revolutionized the way people think about color.

Homage to the Square by Josef Albers (1949-1975)
Albers Squares

Josef Albers. Revolutionized the way people think about color.

Example Client Program for Color ADT

```java
import java.awt.Color;

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, .1);
    }
}
```

Monochrome Luminance

Monochrome luminance. Effective brightness of a color.

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

Color Compatibility

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 \( \geq 128 \).

```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;
    }
}
```

```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.

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.

References

René Magritte. "This is not a pipe."

Java. This is not a color.

OOP. Natural vehicle for studying abstract models of the real world.

OOP Context for Color

Possible memory representation (in TOY).

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.

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)
    Picture(int w, int h)
    int width()
    int height()
    Color get(int x, int y)
    void set(int x, int y, Color c)
    void show()
    void save(String filename)
    create a picture from a file
    create a blank w-by-h picture
    return the width of the picture
    return the height of the picture
    return the color of pixel (x, y)
    set the color of pixel (x, y) to c
    display the image in a window
    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 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();
```

**Image Processing Challenge 2**

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?

```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();
```

**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.

```
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();
    }
}
```
Image Processing: Scaling Filter

**Scaling filter.** Creates two Picture objects and two windows.

```
% java Scale mandrill.jpg 400 200
```

More Image Processing Effects

- WAV filter
- Glass filter
- Sobel edge detection

String Processing

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

**Set of values.** Sequence of Unicode characters.

**API:**

```java
public class String (Java string data type)

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

[Link to Java API documentation](https://java.sun.com/j2se/1.5/docs/api/java/lang/String.html)
Typical String Processing Code

```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
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++)
            if (genome.substring(i, i+3).equals(start))
                beg = i;
        String gene = genome.substring(beg+3, i) +
            if (gene.length() % 3 == 0)
                StdOut.println(gene);
        beg = -1;
    }
}
```

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

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++)
            if (genome.substring(i, i+3).equals(start))
                beg = i;
        String gene = genome.substring(beg+3, i) +
            if (gene.length() % 3 == 0)
                StdOut.println(gene);
        beg = -1;
    }
}
```

Pre-genomics era. Sequence a human genome.

Post-genomics era. Analyze the data and understand structure.

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]
OOP Context for Strings

Possible memory representation of a string (using TOY addresses).

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

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

In and Out

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.

http://www.thestreet.com/quote/goog.html

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

```html
<div id="topTradeInfo">
  <div id="tradeInfo">
    <span id="price-tabs">$642.59</span>
    <span class="valueRed-tabs" alt="Down"></span><span class="chg">. . .</span>
  </div>
</div>
```

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

```java
import java.io.*;
import java.net.*;
import java.util.*;
import java.math.*;

public class StockQuote {
  public static void main(String[] args) {
    String name = "http://www.thestreet.com/quote/";
    In in = new In(name + args[0] + ".html");
    String input = in.readAll();
    int start = input.indexOf("topTradeInfo", 0);
    int from = input.indexOf("price-tabs", start);
    int to = input.indexOf("</span>", from);
    String price = input.substring(from + 12, to);
    StdOut.println(price);
  }
}
```

%!java StockQuote goog
$642.59

• s.indexOf(t, i): index of first occurrence of t in s, starting at offset i.
• Read raw html from http://www.thestreet.com/quote/goog.html
• Find string delimited by "price-tabs" and </span>, after topTradeInfo

Warning. Use at your own financial risk.
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 learned 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.