8. Abstract Data Types

Abstract data types

A data type is a set of values and a set of operations on those values.

Primitive types
- values immediately map to machine representations
- operations immediately map to machine instructions.

We want to write programs that process other types of data.
- Colors, pictures, strings,
- Complex numbers, vectors, matrices,
- ... 

An abstract data type is a data type whose representation is hidden from the client.

Object-oriented programming (OOP)

Object-oriented programming (OOP).
- Create your own data types.
- Use them in your programs (manipulate objects).

Examples (stay tuned for details)

<table>
<thead>
<tr>
<th>data type</th>
<th>set of values</th>
<th>examples of operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color</td>
<td>three 8-bit integers</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>

Best practice: Use abstract data types (representation is hidden from the client).

Impact: Clients 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
Strings

We have already been using ADTs!

A String is a sequence of Unicode characters. defined in terms of its ADT values (typical)

Java’s String ADT allows us to write Java programs that manipulate strings. The exact representation is hidden (it could change and our programs would still work).

Operations (API)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public class String</td>
<td></td>
</tr>
<tr>
<td>String(String s)</td>
<td>create a string with the same value</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-1st characters</td>
</tr>
<tr>
<td>boolean contains(String sub)</td>
<td>does string contain sub?</td>
</tr>
</tbody>
</table>

Using a data type: constructors and methods

To use a data type, you need to know:
- Its name (capitalized, in Java).
- How to construct new objects.
- How to apply operations to a given object.

To construct a new object
- Use the keyword new to invoke a constructor.
- Use data type name to specify type of object.

To apply an operation (invoke a method)
- Use object name to specify which object.
- Use the dot operator to indicate that an operation is to be applied.
- Use a method name to specify which operation.

Pop quiz on ADTs

Q. What is a data type?
A. A set of values and a set of operations on those values.

Q. What is an abstract data type?
A. A data type whose representation is hidden from the client.
9. Abstract Data Types

- Overview
- Color
- Image processing
- String processing

Color ADT

Color is a sensation in the eye from electromagnetic radiation.

An ADT allows us to write Java programs that manipulate color.

API (operations)

<table>
<thead>
<tr>
<th></th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>R (8 bits)</td>
<td>red intensity</td>
</tr>
<tr>
<td>G (8 bits)</td>
<td>green intensity</td>
</tr>
<tr>
<td>B (8 bits)</td>
<td>blue intensity</td>
</tr>
<tr>
<td>color</td>
<td></td>
</tr>
</tbody>
</table>

R: 0 0 255 0 0 119 105
G: 0 255 0 0 64 33 105
B: 0 0 255 0 128 27 105

public class Color

Color(int r, int g, int b)
int getRed()
int getGreen()
int getBlue()
Color brighter()
Color darker()
String toString()
boolean equals(Color c)

Albers squares

Josef Albers. A 20th century artist who revolutionized the way people think about color.
Color client example: Albers squares

Goal. Write a Java program to generate Albers squares.

```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.setPenColor(c2);
        StdDraw.setPenColor(c1);
        StdDraw.setPenColor(c2);
        StdDraw.setPenColor(c1);
        StdDraw.setPenColor(c2);
        StdDraw.setPenColor(c1);
        StdDraw.setPenColor(c2);
    }
}
```

Computing with color: monochrome luminance

**Definition.** The monochrome luminance of a color quantifies its effective brightness.

NTSC standard formula for luminance: \(0.299r + 0.587g + 0.114b\).

```
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 .299*r + .587*g + .114*b;
    }
    public static void main(String[] args) {
        int r = Integer.parseInt(args[0]);
        int g = Integer.parseInt(args[1]);
        int b = Integer.parseInt(args[2]);
        Color c = new Color(r, g, b);
        StdOut.println(Math.round(lum(c)));
    }
}
```

### Examples

<table>
<thead>
<tr>
<th>intensity</th>
<th>red</th>
<th>green</th>
<th>blue</th>
</tr>
</thead>
<tbody>
<tr>
<td>119</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>105</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### Applications (next)

- Choose colors for displayed text.
- Convert colors to grayscale.

Computing with color: compatibility

**Question.** Which font colors will be most readable with which background colors on a display?

**Rule of thumb.** Absolute value of difference in luminosity should be > 128.

```
public static boolean compatible(Color a, Color b) {
    return Math.abs(lum(a) - lum(b)) > 128.0;
}
```
Computing with color: grayscale

**Goal.** Convert colors to grayscale values.

**Fact.** When all three R, G, and B values are the same, resulting color is on grayscale from 0 (black) to 255 (white).

**Q.** What value for a given color?

**A.** Its luminance!

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

**Examples**

<table>
<thead>
<tr>
<th>red intensity</th>
<th>255</th>
<th>0</th>
<th>0</th>
<th>255</th>
<th>0</th>
<th>119</th>
<th>105</th>
</tr>
</thead>
<tbody>
<tr>
<td>green intensity</td>
<td>0</td>
<td>255</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>64</td>
<td>105</td>
</tr>
<tr>
<td>blue intensity</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>0</td>
<td>255</td>
<td>128</td>
<td>27</td>
</tr>
<tr>
<td>color</td>
<td>red</td>
<td>green</td>
<td>blue</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>luminance</td>
<td>76</td>
<td>150</td>
<td>29</td>
<td>0</td>
<td>255</td>
<td>52</td>
<td>58</td>
</tr>
<tr>
<td>grayscale</td>
<td>light gray</td>
<td>gray</td>
<td>dark gray</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

OOP context for color

**Q.** How does Java represent color? Three int values? Packed into one int value?

**A.** We don't know. The representation is hidden. It is an abstract data type.

**Possible memory representation of**

```
red = new Color(255, 0, 0)
and gray = new Color(105, 105, 105);
```

**An object reference is analogous to a variable name.**
- It is not the value but it refers to the value.
- We can manipulate the value in the object it refers to.
- We can pass it to (or return it from) a method.

References and abstraction

**René Magritte.** This is not a pipe.

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

---

**Java.** These are not colors.

---

"This is not a pipe."

Yes it is! He's referring to the physical object he's holding. Joke would be better if he were holding a picture of a pipe.

---

**Object-oriented programming.** A natural vehicle for studying abstract models of the real world.
9. Abstract Data Types

- Overview
- Color
- Image processing
- String processing

Picture ADT

A Picture is a 2D array of pixels.

An ADT allows us to write Java programs that manipulate pictures.

API (operations)

public class Picture

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Picture(String filename)</td>
<td>create a picture from a file</td>
</tr>
<tr>
<td>Picture(int w, int h)</td>
<td>create a blank w-by-h picture</td>
</tr>
<tr>
<td>int width()</td>
<td>width of the picture</td>
</tr>
<tr>
<td>int height()</td>
<td>height of the picture</td>
</tr>
<tr>
<td>Color get(int col, int row)</td>
<td>the color of pixel (col, row)</td>
</tr>
<tr>
<td>void set(int col, int row, Color c)</td>
<td>set the color of pixel (col, row) to c</td>
</tr>
<tr>
<td>void show()</td>
<td>display the image in a window</td>
</tr>
<tr>
<td>void save(String filename)</td>
<td>save the picture to a file</td>
</tr>
</tbody>
</table>

Picture client example: Grayscale filter

Goal. Write a Java program to convert an image to grayscale.
**Picture client example: Grayscale filter**

```java
import java.awt.Color;
public class Grayscale
{
    public static void main(String[] args)
    {
        Picture pic = new Picture(args[0]);
        for (int col = 0; col < pic.width(); col++)
            for (int row = 0; row < pic.height(); row++)
            {
                Color color = pic.get(col, row);
                Color gray = Luminance.toGray(color);
                pic.set(col, row, gray);
            }
        pic.show();
    }
}
```

**Pop quiz 1a on image processing**

**Q.** What is the effect of the following code (easy question)?

```java
Picture pic = new Picture(args[0]);
for (int col = 0; col < pic.width(); col++)
    for (int row = 0; row < pic.height(); row++)
        pic.set(col, row, pic.get(col, row));
pic.show();
```

**A.** None, just shows the picture.

**Pop quiz 1b on image processing**

**Q.** What is the effect of the following code (not-so-easy question)?

```java
Picture pic = new Picture(args[0]);
for (int col = 0; col < pic.width(); col++)
    for (int row = 0; row < pic.height(); row++)
        pic.set(col, pic.height()-row-1, pic.get(col, row));
pic.show();
```
Pop quiz 1b on image processing

Q. What is the effect of the following code (not-so-easy question)?

```java
Picture pic = new Picture(args[0]);
for (int col = 0; col < pic.width(); col++)
    for (int row = 0; row < pic.height(); row++)
        pic.set(col, pic.height()-row-1, pic.get(col, row));
pic.show();
```

A. Tries to turn image upside down, but fails.
An instructive bug.

Pop quiz 1c on image processing

Q. What is the effect of the following code?

```java
Picture source = new Picture(args[0]);
int width = source.width();
int height = source.height();
Picture target = new Picture(width, height);
for (int col = 0; col < width; col++)
    for (int row = 0; row < height; row++)
        target.set(col, height-row-1, source.get(col, row));
target.show();
```

A. Makes an upside down copy of the image.

Picture client example: Scaling filter

Goal. Write a Java program to scale an image (arbitrarily and independently on x and y).
**Picture client example: Scaling filter**

**Goal.** Write a Java program to scale an image (arbitrarily and independently on x and y).

**Ex. Downscaling by halving.**
Shrink in half by deleting alternate rows and columns.

**Ex. Upscaling by doubling.**
Double in size by replacing each pixel with four copies.

```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 tcol = 0; tcol < w; tcol++)
            for (int trow = 0; trow < h; trow++)
                target.set(tcol, trow, source.get(tcol * w / ws, trow * hs / ht));
        target.show();
    }
}
```

**Picture client example: Scaling filter**

**Goal.** Write a Java program to scale an image (arbitrarily and independently on x and y).

**A uniform strategy to scale from ws-by-hs to wt-by-ht.**
- Scale column index by ws/wt.
- Scale row index by hs/ht.

**Approach.** Arrange computation to compute exactly one value for each target pixel.

**More image-processing effects**

- Glass filter
- Sobel edge detection
- RGB color separation
- Swirl filter
9. Abstract Data Types

- Overview
- Color
- Image processing
- String processing

String ADT

A String is a sequence of Unicode characters. Defined in terms of its ADT values (typical)

Java's ADT allows us to write Java programs that manipulate strings.

Operations (API)

<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</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 jth characters</td>
</tr>
<tr>
<td>boolean contains(String sub)</td>
<td>does string contain sub?</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>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 a to b</td>
</tr>
<tr>
<td>String[] split(String delim)</td>
<td>strings between occurrences of delim</td>
</tr>
<tr>
<td>boolean equals(Object t)</td>
<td>is this string's value the same as t's?</td>
</tr>
</tbody>
</table>

Programming with strings: typical examples

Is the string a palindrome?

```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;
}
```

Find lines containing a specified string in StdIn

```java
while (!Stdin.isEmpty()) {
    String query = args[0];
    if (!Stdin.isEmpty()) {
        String s = Stdin.readLine();
        if (s.startsWith(http) && s.endsWith(edu))
            StdOut.println(s);
    }
}
```

Search for *.edu hyperlinks in the text file on StdIn

```java
while (!Stdin.isEmpty()) {
    String s = Stdin.readString();
    if (s.startsWith("http://") & & s.endsWith(".edu"))
        StdOut.println(s);
}
```
String client exercise: Gene finding

Goal. Write a Java program to find genes in a given genome.

Algorithm. Scan left-to-right through dna.
- If start codon ATG found, set beg to index i.
- If stop codon found and substring length is a multiple of 3, print gene and reset beg to -1.

<table>
<thead>
<tr>
<th>i</th>
<th>codon</th>
<th>beg</th>
<th>output</th>
<th>remainder of input string</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>-1</td>
<td></td>
<td>ATAGATGCATAGCCTAGCTAGCC</td>
</tr>
<tr>
<td>1</td>
<td>TAG</td>
<td>-1</td>
<td>ATAGATGCATAGCCTAGCTAGCC</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>ATG</td>
<td>4</td>
<td>ATAGATGCATAGCCTAGCTAGCC</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>TAG</td>
<td>4</td>
<td>ATAGATGCATAGCCTAGCTAGCC</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>TAG</td>
<td>-1</td>
<td>ATAGATGCATAGCCTAGCTAGCC</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>TAG</td>
<td>-1</td>
<td>ATAGATGCATAGCCTAGCTAGCC</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>ATG</td>
<td>23</td>
<td>ATAGATGCATAGCCTAGCTAGCC</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>TAG</td>
<td>23</td>
<td>ATAGATGCATAGCCTAGCTAGCC</td>
<td></td>
</tr>
</tbody>
</table>

Implementation. Entertaining programming exercise!

String client warmup: Identifying a potential gene

Goal. Write a Java program to determine whether a given string is a potential gene.

```
public class Gene
{
    public static boolean isPotentialGene(String dna)
    {
        if (dna.length() % 3 != 0) return false;
        if (!dna.startsWith("ATG")) return false;
        for (int i = 0; i < dna.length() - 3; i++)
        {
            String codon = dna.substring(i, i + 3);
            if (!codon.equals("ATG")) return false;
            if (codon.equals("TAG")) return false;
            if (codon.equals("TGA")) return false;
        }
        if (dna.endsWith("TAA")) return true;
        if (dna.endsWith("TGA")) return true;
        if (dna.endsWith("TGG")) return true;
        return false;
    }
    public static void main(String[] args)
    {
        System.out.println(isPotentialGene(args[0]));
    }
}
```

OOP context for strings

Possible memory representation of

```
public class Gene
{
    public static boolean isPotentialGene(String dna)
    {
        if (dna.length() % 3 != 0) return false;
        if (!dna.startsWith("ATG")) return false;
        for (int i = 0; i < dna.length() - 3; i++)
        {
            String codon = dna.substring(i, i + 3);
            if (!codon.equals("ATG")) return false;
            if (codon.equals("TAG")) return false;
            if (codon.equals("TGA")) return false;
        }
        if (dna.endsWith("TAA")) return true;
        if (dna.endsWith("TGA")) return true;
        if (dna.endsWith("TGG")) return true;
        return false;
    }
    public static void main(String[] args)
    {
        System.out.println(isPotentialGene(args[0]));
    }
}
```

Implications
- s and t are different strings that share the same value “aaca”.
- (s.equals(t)) is false (because it compares addresses).
- (s.equals(t)) is true (because it compares character sequences).
- Java String interface is more complicated than the API.
Object-oriented programming: summary

Object-oriented programming.
- Create your own data types (sets of values and ops on them).
- Use them in your programs (manipulate objects).

In Java, programs manipulate references to objects.
- String, Picture, Color, arrays, (and everything else) are reference types.
- Exceptions: boolean, int, double and other primitive types.
- OOP purist: Languages should not have separate primitive types.
- Practical programmer: Primitive types provide needed efficiency.

This lecture: You can write programs to manipulate sounds, colors, pictures, and strings.
Next lecture: You can define your own abstractions and write programs that manipulate them.