9. 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>

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

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
Sound

We have already been using ADTs!

Sound ADT

Values: Array of doubles.

Operations: specified in API.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>play(double[] a)</td>
<td>play the given sound wave</td>
</tr>
<tr>
<td>save(String file, double[] a)</td>
<td>save to a .wav file</td>
</tr>
<tr>
<td>read(String file)</td>
<td>read from a .wav file</td>
</tr>
</tbody>
</table>

Representation: Hidden from user (.wav and other formats needed by devices).

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.

<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 j-1st characters</td>
</tr>
<tr>
<td>boolean contains(String sub)</td>
<td>does string contain sub?</td>
</tr>
</tbody>
</table>

Operations (API)

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.

Strings

We have already been using ADTs!

A String is a sequence of Unicode characters.

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

<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 j-1st characters</td>
</tr>
<tr>
<td>boolean contains(String sub)</td>
<td>does string contain sub?</td>
</tr>
</tbody>
</table>

Operations (API)

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.

Strings

We have already been using ADTs!

A String is a sequence of Unicode characters.

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

<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 j-1st characters</td>
</tr>
<tr>
<td>boolean contains(String sub)</td>
<td>does string contain sub?</td>
</tr>
</tbody>
</table>

Operations (API)

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.

Strings

We have already been using ADTs!

A String is a sequence of Unicode characters.

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

<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 j-1st characters</td>
</tr>
<tr>
<td>boolean contains(String sub)</td>
<td>does string contain sub?</td>
</tr>
</tbody>
</table>

Operations (API)
9. Abstract Data Types

- Overview
- Color
- Image processing
- String processing

Color ADT

**Color** is a sensation in the eye from electromagnetic radiation.

### Values

<table>
<thead>
<tr>
<th>Examples</th>
<th>R (8 bits)</th>
<th>red intensity</th>
<th>255</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>255</th>
<th>0</th>
<th>119</th>
<th>105</th>
</tr>
</thead>
<tbody>
<tr>
<td>G (8 bits)</td>
<td>green intensity</td>
<td>255</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>64</td>
<td>33</td>
<td>105</td>
<td></td>
</tr>
<tr>
<td>B (8 bits)</td>
<td>blue intensity</td>
<td>0</td>
<td>255</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>128</td>
<td>27</td>
<td>105</td>
<td></td>
</tr>
</tbody>
</table>

### API (operations)

- `public class java.awt.Color`

  - `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 the same as c’s?

Color client example: Albers squares

**Goal**: Write a Java program to generate Albers squares.

Example:

```java
public class AlbersSquares {
    public static void main(String[] args) {
        // Generate Albers squares...
    }
}
```
Color client example: 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.filledSquare(.25, .5, .75);
        StdDraw.setPenColor(c2);
        StdDraw.filledSquare(.25, .5, .75);
    }
}
```

Computing with color: monochrome luminance

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

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

```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;
    }
    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)));
    }
}
```

Computing with color: grayscale

**Goal.** Convert a color 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!

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

Computing with color: compatibility

**Q.** 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.

```java
public static boolean compatible(Color a, Color b) {
    return Math.abs(lum(a) - lum(b)) > 128.0;
}
```
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.

Object-oriented programming. A natural vehicle for studying abstract models of the real world.

"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.

Surrealist painter Rene Magritte and his brother Surrealist Plumber, Rodney.\[4\]
### Picture client example: Grayscale filter

**Goal.** Write a Java program to convert an image to grayscale.

#### API (operations)

- `Picture(String filename)`
  - create a picture from a file

- `int width()`
  - width of the picture

- `int height()`
  - height of the picture

- `Color get(int i, int j)`
  - the color of pixel (i, j)

- `void set(int i, int j, Color c)`
  - set the color of pixel (i, j) to c

- `void show()`
  - display the image in a window

- `void save(String filename)`
  - save the picture to a file

#### Values (arrays of Colors)

- `Pixel(i, j)`
- `Width` and `Height`

#### Picture client example: Grayscale filter

```java
import java.awt.Color;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.util.Scanner;

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

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

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

**A.** The code copies the original image to a new grayscale image.
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 i = 0; i < pic.width(); i++)
        for (int j = 0; j < pic.height(); j++)
            pic.set(i, pic.height() - j - 2, pic.get(i, j));
    pic.show();
```

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 i = 0; i < width; i++)
        for (int j = 0; j < height; j++)
            target.set(i, height - j - 1, source.get(i, j));
    target.show();
```

Picture client example: Scaling filter

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

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

Ex. Upsampling by doubling.
Double in size by replacing each pixel with four copies.
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 \( ws/\text{wt} \).
- Scale row index by \( hs/\text{ht} \).

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

```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 ti = 0; ti < w; ti++)
            for (int tj = 0; tj < h; tj++)
                {
                int si = ti * source.width() / w;
                int sj = tj * source.height() / h;
                Color color = source.get(si, sj);
                target.set(ti, tj, color);
            }
        target.show();
    }
}
```

More image-processing effects

- **RGB color separation**
- **swirl filter**
- **wave filter**
- **glass filter**
- **Sobel edge detection**
A **String** is a sequence of Unicode characters.

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><code>String(String s)</code></td>
<td>creates a string with the same value</td>
</tr>
<tr>
<td><code>int length()</code></td>
<td>returns the length of the string</td>
</tr>
<tr>
<td><code>char charAt(int i)</code></td>
<td>returns the character at index <code>i</code></td>
</tr>
<tr>
<td><code>String substring(int i, int j)</code></td>
<td>returns a substring from index <code>i</code> to <code>j</code></td>
</tr>
<tr>
<td><code>boolean contains(String sub)</code></td>
<td>checks if the string contains the substring</td>
</tr>
<tr>
<td><code>boolean startsWith(String pre)</code></td>
<td>checks if the string starts with the prefix</td>
</tr>
<tr>
<td><code>boolean endsWith(String post)</code></td>
<td>checks if the string ends with the suffix</td>
</tr>
<tr>
<td><code>int indexOf(String p, int i)</code></td>
<td>returns the index of the first occurrence of <code>p</code> after <code>i</code></td>
</tr>
<tr>
<td><code>int compareTo(String t)</code></td>
<td>compares the string with another string</td>
</tr>
<tr>
<td><code>String replaceAll(String a, String b)</code></td>
<td>returns a new string with all occurrences of <code>a</code> replaced by <code>b</code></td>
</tr>
<tr>
<td><code>String[] split(String delim)</code></td>
<td>splits the string by the delimiter and returns an array of substrings</td>
</tr>
<tr>
<td><code>boolean equals(String t)</code></td>
<td>checks if the string’s value is the same as <code>t</code></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;
}
```

**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 example: 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.

- Made of **codons** (three A C T G nucleotides).
- Preceded by ATG (start codon).
- Succeeded by TAG (stop codon).

[Diagram: ATAG CATG CATAG CGC ATAG CATG CATG TGC ATACC]

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

---

**Algorithm.** Scan left-to-right through genome.

- If start codon ATG found, set `beg` to index `i`.
- If stop codon TAG 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>start</th>
<th>stop</th>
<th>beg</th>
<th>output</th>
<th>remainder of input string</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>-1</td>
<td>ATAGCATGATGCATAGCTAGCTAGTCATACC</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>TAG</td>
<td>-1</td>
<td></td>
<td></td>
<td>TAGATGCATAGCGCATAGCTAGCTAGTCATACC</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>ATG</td>
<td>4</td>
<td></td>
<td></td>
<td>ATGCATAGCGCATAGCTAGCTAGTCATACC</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>TAG</td>
<td>4</td>
<td></td>
<td></td>
<td>TAGCATAGCGCATAGCTAGCTAGTCATACC</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>TAG</td>
<td>4</td>
<td></td>
<td></td>
<td>TAGCATAGCGCATAGCTAGCTAGTCATACC</td>
<td>CATAGCGCA</td>
</tr>
<tr>
<td>20</td>
<td>TAG</td>
<td>-1</td>
<td></td>
<td></td>
<td>TAGATGCATAGCTAGCTAGTCATACC</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>ATG</td>
<td>23</td>
<td></td>
<td></td>
<td>ATGCATAGCTAGCTAGTCATACC</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>TAG</td>
<td>23</td>
<td></td>
<td></td>
<td>ATGCATAGCTAGCTAGTCATACC</td>
<td>CATAG</td>
</tr>
</tbody>
</table>
String client example: Gene finding

```java
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 && beg+3 < genome.length()) {
                String gene = genome.substring(beg+3, i+1);
                if (gene.length() % 3 == 0) {
                    StdOut.println(gene);
                }
            }
        }
    }
}
```

Possible memory representation of

```
String genome = "aacaagtgtacaagc";
String s = genome.substring(1, 5);
String t = genome.substring(9, 13);
```

Implications

- s and t are different strings that share the same value "acaa".
- (s.equals(t)) is true (because it compares character sequences).
- Java String interface is more complicated than the API (and not really an ADT).

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.