3.3 Designing Data Types

Object Oriented Programming

Procedural programming. [verb-oriented]
- Tell the computer to do this.
- Tell the computer to do that.

OOP philosophy. Software is a simulation of the real world.
- We know (approximately) how the real world works.
- Design software to model the real world.

Objected oriented programming (OOP). [noun-oriented]
- Programming paradigm based on data types.
- Identify objects that are part of the problem domain or solution.
- Identity: objects are distinguished from other objects (references).
- State: objects in the world know things (instance variables).
- Behavior: objects do things (methods).

Alan Kay

Alan Kay. [Xerox PARC 1970s]
- Invented Smalltalk programming language.
- Conceived Dynabook portable computer.
- Ideas led to: laptop, modern GUI, OOP.

“ The computer revolution hasn’t started yet. ”
“ The best way to predict the future is to invent it. ”
“ If you don’t fail at least 90 per cent of the time, you’re not aiming high enough. ”

— Alan Kay
Encapsulation

Data type. Set of values and operations on those values.
Ex. int, String, Complex, Vector, Document, GuitarString, ...

Encapsulated data type. Hide internal representation of data type.

Separate implementation from design specification.
- Class provides data representation and code for operations.
- Client uses data type as black box.
- API specifies contract between client and class.

Bottom line. You don’t need to know how a data type is implemented in order to use it.

Intuition

Client

API
- volume
- change channel
- adjust picture
- decode NTSC signal

Implementation
- cathode ray tube
- electron gun
- Sony Wega 36XBR250
- 241 pounds

client needs to know how to use API
implementation needs to know what API to implement

Implementation and client need to agree on API ahead of time.

Counter Data Type

Counter. Data type to count electronic votes.

Counter.java

```java
public class Counter {
    public int count;
    public final String name;
    public Counter(String id) { name = id; }
    public void increment() { count++;
    public int value() { return count; }
}
```

Legal Java client.

```java
Counter c = new Counter("Volusia County");
c.count = -16022;
```

Oops. Al Gore receives -16,022 votes in Volusia County, Florida.
Counter. Encapsulated data type to count electronic votes.

```java
public class Counter {
    private int count;
    private final String name;

    public Counter(String id) {
        name = id;
    }

    public void increment() {
        count++;
    }

    public int value() {
        return count;
    }
}
```

Counter Data Type

Does not compile.

Counter c = new Counter("Volusia County");
c.count = -16022;

Benefit. Can guarantee that each data type value remains in a consistent state.

Changing Internal Representation

Encapsulation.
- Keep data representation hidden with `private` access modifier.
- Expose API to clients using `public` access modifier.

```java
public class Complex {
    private final double re, im;

    public Complex(double re, double im) {
    }

    public double abs() {
    }

    public Complex plus(Complex b) {
    }

    public Complex times(Complex b) {
    }

    public String toString() {
    }
}
```

e.g., to polar coordinates

Lesson. By exposing data representation to client, might need to sift through millions of lines of code in client to update.

Internal representation changes.
- [VIN numbers] We’ll run out by 2010.

Time Bombs

Ask, Don’t Touch

Encapsulated data types.
- Don’t touch data and do whatever you want.
- Instead, ask object to manipulate its data.

Lesson. Limiting scope makes programs easier to maintain and understand.

“Ak, don’t touch.”

“A principle of least privilege.”

Adele Goldberg
Former president of ACM
Co-developed Smalltalk
Immutability

Immutable data type. Object’s value cannot change once constructed.

Immutability: Advantages and Disadvantages

Immutable data type. Object’s value cannot change once constructed.

Advantages.
- Avoid aliasing bugs.
- Makes program easier to debug.
- Limits scope of code that can change values.
- Pass objects around without worrying about modification.

Disadvantage. New object must be created for every value.

Final Access Modifier

Final. Declaring an instance variable to be final means that you can assign it a value only once, in initializer or constructor.

public class Counter {
    private final String name;
    private int count;
    ...
}

Advantages.
- Helps enforce immutability.
- Prevents accidental changes.
- Makes program easier to debug.
- Documents that the value cannot not change.
**Spatial Vectors**

**Set of values. Sequence of real numbers.** [Cartesian coordinates]

**API.**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Vector(double[] a)</code></td>
<td>create a vector with the given Cartesian coordinates</td>
</tr>
<tr>
<td><code>Vector plus(Vector b)</code></td>
<td>sum of this vector and b</td>
</tr>
<tr>
<td><code>Vector minus(Vector b)</code></td>
<td>difference of this vector and b</td>
</tr>
<tr>
<td><code>Vector times(double t)</code></td>
<td>scalar product of this vector and t</td>
</tr>
<tr>
<td><code>double dot(Vector b)</code></td>
<td>dot product of this vector and b</td>
</tr>
<tr>
<td><code>double magnitude()</code></td>
<td>magnitude of this vector</td>
</tr>
<tr>
<td><code>Vector direction()</code></td>
<td>unit vector with same direction as this vector</td>
</tr>
</tbody>
</table>

```
Vector x = (0, 3, 4, 0), y = (0, -3, 1, -4)
x + y = (0, 0, 5, -4)
3x = (0, 9, 12, 0)
x ⋅ y = (0 ⋅ 0) + (3 ⋅ -3) + (4 ⋅ 1) + (0 ⋅ -4) = -5
|x| = (0^2 + 3^2 + 4^2 + 0^2)^(1/2) = 5
x = x / |x| = (0, 0.6, 0.8, 0)
```

**Vector Data Type Applications**

**Relevance.** A quintessential mathematical abstraction.

**Applications.**
- Statistics.
- Linear algebra.
- Clustering and similarity search.
- Force, velocity, acceleration, momentum, torque.
- ...

**Vector Data Type: Implementation**

```java
public class Vector {
    private int N;
    private double[] coords;

    public Vector(double[] a) {
        N = a.length;
        coords = new double[N];
        for (int i = 0; i < N; i++)
            coords[i] = a[i];
    }

    public double dot(Vector b) {
        double sum = 0.0;
        for (int i = 0; i < N; i++)
            sum += (coords[i] * b.coords[i]);
        return sum;
    }

    public Vector plus(Vector b) {
        double[] c = new double[N];
        for (int i = 0; i < N; i++)
            c[i] = coords[i] + b.coords[i];
        return new Vector(c);
    }
}
```
This. The keyword this is a reference to the invoking object.

Ex. When you invoke a.magnitude(), this is an alias for a.

3.5 Case Study: Purple America

Data Visualization

Challenge. Visualize election results.

“ If I can’t picture it, I can’t understand it.”
— Albert Einstein

Approach.
- Gather data from data sources on the web; save in local files.
- Build a modular program that reads files and draws maps.
Data Sources

Geometric data.
- www.census.gov/tiger/boundary
- Text files have boundaries of every state and county.
  (format useful for programmers)

Election returns.
- www.uselectionatlas.org
- Web site displays election results.
  (need to screen scrape to extract raw data)

Emerging standard.
- Publish data in text format on the web (like geometric data).
- Write mashup program to produce visuals (like we’re doing)!

Geometric Data: States within the Continental US

USA data file. State names and boundary points.

<table>
<thead>
<tr>
<th>% more USA.txt</th>
</tr>
</thead>
<tbody>
<tr>
<td>-124.731216 24.544102 -66.980385 49.384365</td>
</tr>
<tr>
<td>104 number of regions</td>
</tr>
<tr>
<td>Alabama 498</td>
</tr>
<tr>
<td>-88.200027 34.995548</td>
</tr>
<tr>
<td>-88.202919 35.007942</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>New Jersey 368</td>
</tr>
<tr>
<td>-74.695305 41.357330</td>
</tr>
<tr>
<td>-74.461754 41.250000</td>
</tr>
<tr>
<td>-74.366302 41.202801</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>-74.721313 41.347294</td>
</tr>
</tbody>
</table>

Bounding box: (-124.73, 24.54) to (-66.98, 49.38)

Geometric Data: Counties within a State

State data files. County names and boundary points.

<table>
<thead>
<tr>
<th>% more NJ.txt</th>
</tr>
</thead>
<tbody>
<tr>
<td>-75.560143 38.928589 -73.894402 41.35733</td>
</tr>
<tr>
<td>21 number of regions</td>
</tr>
<tr>
<td>Atlantic 127</td>
</tr>
<tr>
<td>-74.877563 39.608414</td>
</tr>
<tr>
<td>-74.736694 39.729721</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>Mercer 88</td>
</tr>
<tr>
<td>-74.748825 40.424248</td>
</tr>
<tr>
<td>-74.722702 40.375301</td>
</tr>
<tr>
<td>-74.674507 40.384399</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>-74.808403 40.415401</td>
</tr>
</tbody>
</table>

Bounding box: (-75.56, 38.92) to (-73.89, 41.35)
Screen Scraping the Election Returns

Screen scraping. Download html from web and parse.

region name is text between <b> and </b> tags that occurs after width:100px
NJ = FIPS 34

Election Returns: By County

Screen-scraped results. Votes for McCain, Obama, Other by region.

Election Returns: By State

Screen-scraped results. Votes for McCain, Obama, Other by region.

int year = 2008; // election year
String whole = "NJ"; // region name for New Jersey
int fips = 34; // FIPS code for New Jersey

In in = new In(url + "?year=" + year + "&fips=" + fips);
Out file = new Out(whole + year + ".txt");
String input = in.readAll();
while (true) {
    // screen scrape region name
    int p = input.indexOf("width:100px", p);
    if (p == -1) break;
    int from = input.indexOf(< b >, p);
    int to = input.indexOf(</ b >, from);
    String region = input.substring(from + 3, to);
    // screen scrape vote totals for each candidate
    save results to file
    file.println(region + "," + mccain + "," + obama + "," + other + ");
}

Screen Scraping the Election Returns (Java sketch)
Real Data are Messy

Different data sources have different conventions.
- State names: NJ vs. New Jersey vs. FIPS 34.
- County names: LaSalle vs. La Salle, Kings County vs. Brooklyn.

Other annoyances.
- A state can be comprised of several disjoint polygons.
- A county can be entirely inside another county.
- County boundaries change over time.
- Write-in candidates.
- Unreported results.
- Alaska and Hawaii.

Bottom line. Must clean the data (but write a program to do most of it!)

Summary of Data Files

714 data files. \([13 + 1] \times [50 + 1]\)
- Each file represents a "whole" divided into regions.
- One entry per region.

<table>
<thead>
<tr>
<th>whole</th>
<th>part</th>
<th>files</th>
<th>type of data</th>
</tr>
</thead>
<tbody>
<tr>
<td>USA</td>
<td>state</td>
<td>USA, USA2008,</td>
<td>boundary</td>
</tr>
<tr>
<td></td>
<td></td>
<td>usa2004, ...</td>
<td></td>
</tr>
<tr>
<td>state</td>
<td>county</td>
<td>NJ, NJ2008,</td>
<td>election return</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nj2004, ...</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>nj1960, ...</td>
<td></td>
</tr>
</tbody>
</table>

[Similar files for all 50 states]

Modular Programming

Modular programming.
- Model problem by decomposing into components.
- Develop data type for each component.

Region. State or county.
Vote tally. Number of votes for each candidate in a region.
Election map. Map of votes by region in a given election.
Region Data Type

**Region.** A state or county.

Set of values. Sequence of boundary points, name.

Operations. Create and draw.

New Jersey
368 point polygon

Mercer
88 point polygon

Vote Tally Data Type

**Vote tally.** Election returns for one region.

Set of values. Number of votes for each candidate.

Operations.
- Create (whole, region, year).
- Number of votes for Republican, Democrat, and Independent candidates.

public class Region {
    private final String name;    // name of region
    private final int N;          // number of boundary points
    private final double[] x, y;  // the points (x[i], y[i])

    public Region(String name, double[] x, double[] y) {
        this.name = name;
        this.N = x.length;
        this.x = new double[N];
        this.y = new double[N];
        for (int i = 0; i < N; i++) {
            this.x[i] = x[i];
            this.y[i] = y[i];
        }
    }

    public void draw() { StdDraw.filledPolygon(x, y); }
    public String name() { return name; }
}

defensive copy (stay tuned)

Vote Tally Data Type

public class VoteTally {
    private final int rep, dem, ind;

    public VoteTally(String region, String whole, int year) {
        In in = new In(whole + year + ".txt");
        String input = in.readAll();
        int i0 = input.indexOf(region);
        int i1 = input.indexOf(".", i0 + 1);
        int i2 = input.indexOf(".", i1 + 1);
        int i3 = input.indexOf(".", i2 + 1);
        int i4 = input.indexOf(".", i3 + 1);
        rep = Integer.parseInt(input.substring(i1 + 1, i2));
        dem = Integer.parseInt(input.substring(i2 + 1, i3));
        ind = Integer.parseInt(input.substring(i3 + 1, i4));
    }

    public int rep() { return rep; }
    public int dem() { return dem; }
    public int ind() { return ind; }
}
% more NJ2008.txt
---
Hunterdon, 39092, 29776, 1147,
Mercer, 50223, 107926, 2229,  
Middlesex, 123695, 193812, 4283,
Monmouth, 160433, 148737, 4244,
Election map. Map of votes by region in a given election.

```java
class ElectionMap {  
   private final int REGIONS          ...  
      } 
   } 
} 
for each region, set the pen color according to
the vote tallies and draw the region

public ElectionMap(String whole, int year) {  
   In in = new In(whole + ".txt");  
   int N = in.readInt();  
   regions = new Region[REGIONS];  
   votes = new VoteTally[REGIONS];  
   for (int j = 0; j < REGIONS; j++) {  
      String region = in.readLine();  
      int N = in.readInt();  
      double[] x = new double[N];  
      double[] y = new double[N];  
      for (int i = 0; i < N; i++) {  
         x[i] = in.readDouble();  
         y[i] = in.readDouble();  
      }  
      regions[j] = new Region(part, x, y);  
      votes[j] = new VoteTally(region, whole, year);  
   }  
   read in bounding box and rescale coordinates
```

Modular Programming

Modular program. Collection of interacting data types.

```java
public class ElectionMap {  
   private final int REGIONS          ...  
      } 
   } 
}
```
Data Visualization

Red states, blue states. Nice, but a misleading and polarizing picture.

Edward Tufte. Create charts with high data density that tell the truth.

Purple America

Idea. [Robert J. Vanderbei] Assign color based on number of votes.
- \(a_1\) = McCain votes.
- \(a_2\) = Other votes.
- \(a_3\) = Obama votes.

\[
(R, G, B) = \left( \frac{a_1}{a_1 + a_2 + a_3}, \frac{a_2}{a_1 + a_2 + a_3}, \frac{a_3}{a_1 + a_2 + a_3} \right)
\]

Implementation. Change only one method in `ElectionMap.java`.

```java
public Color getColor() {
    int dem = tally.dem();
    int rep = tally.rep();
    int ind = tally.ind();
    int tot = tally.dem + tally.rep + tally.ind;
    return new Color((float) rep/tot, (float) ind/tot, (float) dem/tot);
}
```
Remark. Humans perceive red more strongly than blue.

Remark. Amount of color should be proportional to number of votes, not geographic boundary.

Remark. Project latitude + longitude coordinates to 2d plane.

3D visualization. Volume proportional to votes; azimuthal projection.

Robert J. Vanderbei
www.princeton.edu/~rvdb/JAVA/election2004
Cartograms

**Cartogram.** Area of state proportional to number of electoral votes.

---

**Summary**

**Modular programming.**
- Break a large program into smaller independent components.
- Develop a **data type** for each component.
- **Ex:** Region, VoteTally, ElectionMap, In, Out.

**Ex 1.** Build large software project.
- Software architect specifies API.
- Each programmer implements one module.
- Debug and test each piece independently. [unit testing]

**Ex 2.** Build reusable libraries.
- Language designer extends language with new data types.
- Programmers share extensive libraries.

**Data visualization.** You can do it! [worthwhile to learn from Tufte]