3.3 Designing Data Types

Object Oriented Programming

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

Alan Kay’s 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 things that are part of the problem domain or solution.
- Things in the world know things: instance variables.
- Things in the world 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.

Encapsulation

"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 percent of the time, you’re not aiming high enough."

Bond: What’s your escape route?
Saunders: Sorry old man. Section 26 paragraph 5, that information is on a need-to-know basis only. I’m sure you’ll understand.
Encapsulation

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

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.

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

Legal Java client.
Oops. Al Gore receives -16,022 votes in Volusia County, Florida.

Can substitute better implementation without changing the client.
Counter. **Encapsulated** data type to count electronic votes.

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

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) {
        this.re = re;
        this.im = im;
    }
    public double abs() {
        return Math.sqrt(re*re + im*im);
    }
    public Complex plus(Complex b) {
        return new Complex(re + b.re, im + b.im);
    }
    public Complex times(Complex b) {
        return new Complex(re*b.re - im*b.im, re*b.im + im*b.re);
    }
    public String toString() {
        return "Complex(re = " + re + ", im = " + im + ");";
    }
}
```

**Time Bombs**

**Internal representation changes.**
- **[Y2K]** Two digit years: January 1, 2000.
- **[Y2038]** 32-bit seconds since 1970: January 19, 2038.
- **[VIN numbers]** We'll run out by 2010.

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

**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.
Immutability

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 change.
Spatial Vectors

Vector Data Type

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

API:

- `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);
} 

Relevance. A quintessential mathematical abstraction.

Applications.
- Statistics.
- Linear algebra.
- Clustering and similarity search.
- Force, velocity, acceleration, momentum, torque.
- ...
This. The keyword this is a reference to the invoking object.
Ex. When you invoke a.magnitude(), this is an alias for a.

```
public Vector times(double t) {
    double[] c = new double[N];
    for (int i = 0; i < N; i++)
        c[i] = t * coords[i];
    return new Vector(c);
}
public double magnitude() {
    return Math.sqrt(this.dot(this));
}
public Vector direction() {
    return this.times(1.0 / this.magnitude());
}
...
Data Visualization

**Challenge.** Visualize election results.

> "If I can't picture it, I can't understand it." — Albert Einstein

**Basic approach.**
- Gather data from web sources, save in local file.
- Build modular program that reads files, draws maps.

---

Modular Programming

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

**Polygon.** Geometric primitive.
**Region.** Name, postal abbreviation, polygonal boundary.
**Vote tally.** Number of votes for each candidate.
**Election map.** Regions and corresponding vote tallies for a given election.

---

Data Sources

**Boundary Data: States within the Continental US**

**Geometric data.** [US census bureau]
- www.census.gov/tiger/boundary
- NJ.txt has boundaries of every county in New Jersey.
- USA.txt that has boundary of every state.

**Election results.** [David Leip]
- http://uselectionatlas.org/RESULTS
  - Interactive and graphical.
  - Need to screen-scrape to get data.

**Emerging standard.**
- Publish data in text form on the web (like geometric data).
- Write programs to produce visuals (like we’re doing!)
- Mashups.
Boundary Data: States within the Continental US

USA data file. State names and boundary points.

```
% more USA.txt
-124.731216 24.544102 -66.980385 49.384365
104
Alabama
498
-88.200027   34.995548
-88.202919   35.007942
New Jersey
368
-74.695305   41.357330
-74.461754   41.250000
-74.366302   41.202801
-74.721313   41.347294
```

Number of regions: 104
Bounding box: (-124.73, 24.54) to (-66.98, 49.38)

368 points (longitude, latitude)

Pitfalls: Pieces and Holes

Pieces. A state can be comprised of several disjoint polygons.

Holes. A county can be entirely inside another county.

Boundary Data: Counties within a State

State data files. County names and boundary points.

```
% more NJ.txt
-75.560143 38.928589 -73.894402 41.35733
21
Atlantic
NJ
127
-74.877563   39.608414
-74.736694   39.729721
Mercer
NJ
88
-74.748825   40.424248
-74.722702   40.375301
-74.674507   40.384399
-74.808403   40.415401
```

Number of regions: 21
Bounding box: (-75.56, 38.92) to (-73.89, 41.35)

88 points (longitude, latitude)

Screen Scraping the Election Returns

Screen scrape. Download html from web and parse.

```
```

County name is text between `<b>` and `</b>` tags that occurs after `width:100px`
Election Scraper (sketch)

```java
int year = 2008; // election year
String usps = "NJ"; // United States postal code for New Jersey
int fips = 34; // FIPS code for New Jersey
String url = "http://uselectionatlas.org/RESULTS/datagraph.php";
In in = new In(url + "?year=" + year + "&fips=" + fips);
Out file = new Out(usps + year + ".txt");
String input = in.readAll();
while (true) {
    // screen scrape county 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 county = input.substring(from + 3, to);
    // screen scrape vote totals for each candidate
    // save results to file
    file.println(county + "," + bush + "," + kerry + "," + nader + ",");
}
```

Data sources have different conventions.
- FIPS codes: NJ vs. 34.
- County names: LaSalle vs. La Salle, Kings County vs. Brooklyn.

Plenty of other minor annoyances.
- Unreported results.
- Third-party candidates.
- Changes in county boundaries.

Bottom line. Need to clean up data (but write a program to do it!)

More Pitfalls

Polygons and Regions

Polygon. Closed, planar path with straight line segments.
Simple polygon. No crossing lines.
**Polygon Data Type: Java Implementation**

```java
public class Polygon {
    private final int N;          // number of boundary points
    private final double[] x, y;  // the points (x[i], y[i])

    // read from input stream
    public Polygon(In in) {
        N = in.readInt();
        x = new double[N];
        y = new double[N];
        for (int i = 0; i < N; i++) {
            x[i] = in.readDouble();
            y[i] = in.readDouble();
        }
    }

    public void fill() { StdDraw.filledPolygon(x, y); }
    public boolean contains(double x0, double y0) { return poly.contains(x0, y0); }
    public String toString() { … }
}
```

**Region Data Type**

```java
public class Region {
    private final String name;  // name of region
    private final String usps;  // postal abbreviation
    private final Polygon poly; // polygonal boundary

    public Region(String name, String usps, Polygon poly) {
        this.name = name;
        this.usps = usps;
        this.poly = poly;
    }

    public void draw() { poly.fill(); }
    public boolean contains(double x0, double y0) { return poly.contains(x0, y0); }
    public String toString() { … }
}
```

**Region Data Type: Java Implementation**

**Election Returns**

**Region.** Represents a state or county.

Mercer, NJ
88 point polygon

New Jersey, USA
368 point polygon
Election Returns: By State

Screen-scraping results. Number of votes for Bush, Kerry, Nader by region.

Election Returns: By County

Screen-scraping results. Number of votes for Bush, Kerry, Nader by region.

Vote Tally Data Type

VoteTally. Represents the election returns for one region.

Vote Tally Data Type: Java Implementation

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

    public VoteTally(String name, String usps, int year) {
        In in = new In(usps + year + ".txt");
        String input = in.readAll();
        int 10 = input.indexOf(name);
        int 11 = input.indexOf(",", 10+1);
        int 12 = input.indexOf(",", 11+1);
        int 13 = input.indexOf(",", 12+1);
        int 14 = input.indexOf(",", 13+1);
        rep = Integer.parseInt(input.substring(11+1, 12+1));
        dem = Integer.parseInt(input.substring(12+1, 13+1));
        ind = Integer.parseInt(input.substring(13+1, 14+1));
    }

    public Color getColor() {
        if (rep > dem) return StdDraw.RED;
        if (dem > rep) return StdDraw.BLUE;
        return StdDraw.BLACK;
    }
}
```
Election Map

Election Map Data Type: Java Implementation

```java
public class ElectionMap {
    private final int N;
    private final Region[] regions;
    private final VoteTally[] votes;

    public ElectionMap(String name, int year) {
        In in = new In(name + ".txt");
        // read in bounding box and rescale coordinates
        N = in.readInt();
        regions = new Region[N];
        votes = new VoteTally[N];
        for (int i = 0; i < N; i++) {
            String name = in.readLine();
            String usps = in.readLine();
            Polygon poly = new Polygon(in);
            regions[i] = new Region(name, usps, poly);
            votes[i] = new VoteTally(name, usps, year);
        }
    }

    public void show() {
        for (int i = 0; i < N; i++) {
            StdDraw.setPenColor(votes[i].getColor());
            regions[i].draw();
        }
    }
}
```

Modular Programming

**Modular program.** Collection of data types.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>ElectionMap</td>
<td>Represents the election map for a given election.</td>
</tr>
<tr>
<td>Region</td>
<td>Collection of data types.</td>
</tr>
<tr>
<td>int</td>
<td>Collection of data types.</td>
</tr>
<tr>
<td>VoteTally</td>
<td>Use polygon, region, and vote tally data types to build map.</td>
</tr>
</tbody>
</table>

Election Map

client

```java
public static void main(String[] args) {
    String name = args[0];
    int year = Integer.parseInt(args[1]);
    ElectionMap election = new ElectionMap(name, year);
    election.show();
}

% java ElectionMap USA 1968
% java ElectionMap NJ 2008
```

use polygon, region, and vote tally data types to build map
Data Visualization

Visual Display of Quantitative Information

Red states, blue states. Creates 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.
  - $a_1$ = Bush votes.
  - $a_2$ = Nader votes.
  - $a_3$ = Kerry 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 one method!

```java
public Color getColor() {
    int tot = dem + rep + ind;
    return new Color((float) rep/tot, (float) ind/tot, (float) dem/tot);
}
```

Purple New Jersey

% java ElectionMap NJ 2004
% java ElectionMap NJ 2008
Data Visualization: Design Issues

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

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

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

![Cartogram of state areas proportional to electoral votes.](image1)

**Cartogram.** Area of country proportional to population.

![Cartogram of country areas proportional to population.](image2)

---

**Summary**

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

**Data visualization.** You can do it! (worthwhile to learn from Tufte)