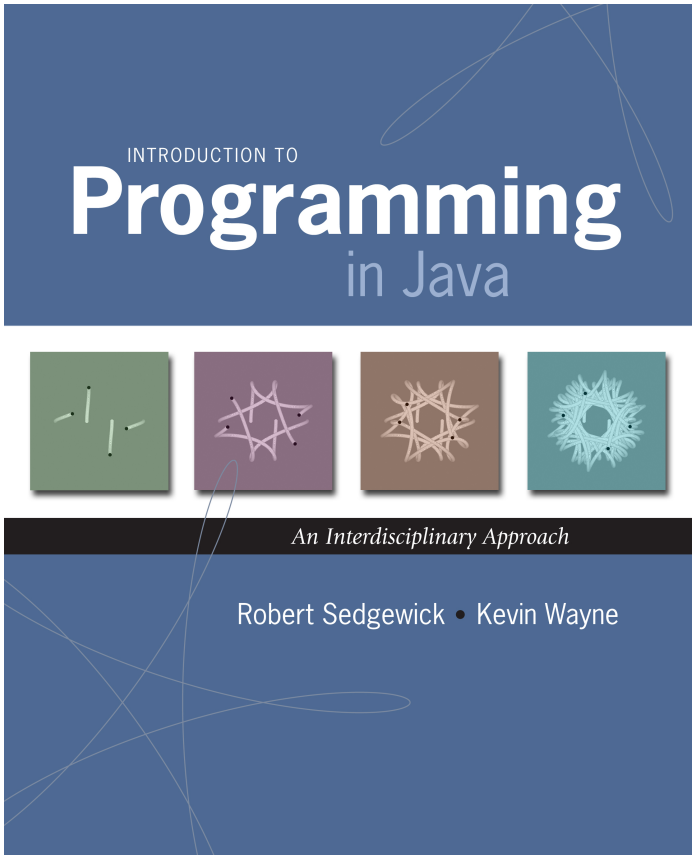
3.3 Designing Data Types



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.

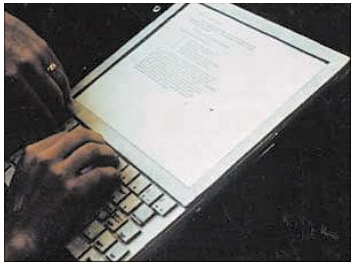
Objected oriented programming (OOP). [noun-oriented]

- 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 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*

Alan Kay
2003 Turing Award

Encapsulation

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`, ...

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.

Intuition

Client

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

Implementation
- gas plasma monitor
- Samsung FPT-6374
- wall mountable
- 4 inches deep

client needs to know how to use API

implementation needs to know what API to implement

Can substitute better implementation without changing the client.

Counter Data Type

Counter. Data type to count electronic votes.

```
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.

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

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

Counter Data Type

Counter. Encapsulated data type to count electronic votes.

```
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;  }

}
```

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.

```
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

Advantage. Can switch internal representation without changing client.

Note. All our data types are already encapsulated!

Time Bombs

Internal representation changes.

- [Y2K] Two digit years: January 1, 2000.
- [Y2038] 32-bit seconds since 1970: January 19, 2038.

www.cartoonstock.com/directory/m/millenium_time-bomb.asp

Lesson. By exposing data representation to client, might 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.

"Ask, don't touch."

Adele Goldberg
Former president of ACM
Co-developed Smalltalk

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

"principle of least privilege"

Immutability

Immutability

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

mutable	*immutable*
`Picture`	`Charge`
`Histogram`	`Color`
`Turtle`	`Stopwatch`
`StockAccount`	`Complex`
`Counter`	`String`
Java arrays	primitive types

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;
...
}
```

this value doesn't change once the object is constructed

this value changes by invoking instance method

Advantages.

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

Spatial Vectors

Vector Data Type

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

API.

public class Vector	
Vector(double[] a)	*create a vector with the given Cartesian coordinates*
Vector plus(Vector b)	*sum of this vector and* b
Vector minus(Vector b)	*difference of this vector and* b
Vector times(double t)	*scalar product of this vector and* t
double dot(Vector b)	*dot product of this vector and* b
double magnitude()	*magnitude of this vector*
Vector direction()	*unit vector with same direction as this vector*

$$x = (0, 3, 4, 0), \quad y = (0, -3, 1, -4)$$

$$x + y = (0, 0, 5, -4)$$

$$3x = (0, 9, 12, 0)$$

$$x \cdot y = (0 \cdot 0) + (3 \cdot -3) + (4 \cdot 1) + (0 \cdot -4) = -5$$

$$|x| = (0^2 + 3^2 + 4^2 + 0^2)^{1/2} = 5$$

$$\vec{x} = x / |x| = (0, 0.6, 0.8, 0)$$

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

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

instance variables

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

constructor

```
   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);
   }
```

methods

Vector Data Type: Implementation

```
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());
}
...
```

This. The keyword `this` is a reference to the invoking object.

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