

3.3: Encapsulation and ADTs



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.

Abstract Data Types

Data type: set of values and operations on those values.

Ex: int, String, Complex, Card, Deck, Wave, Tour, . . .

Abstract data type.

- Data type whose representation is hidden.

Separate implementation from design specification.

- **CLASS:** provides data representation and code for operations.
- **CLIENT:** uses data type as black box.
- **INTERFACE:** contract between client and class.

Intuition



Client



Interface

- volume
- change channel
- adjust picture
- decode NTSC, PAL signals



Implementation

- cathode ray tube
- electron gun
- Sony Wega 36XBR250
- 241 pounds, \$2,699

client needs to know how to use interface

implementation needs to know what interface to implement

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

Intuition



Client



Interface

- volume
- change channel
- adjust picture
- decode NTSC, PAL signals



Implementation

- gas plasma monitor
- Pioneer PDP-502MX
- wall mountable
- 4 inches deep
- \$19,995

client needs to know how to use interface

implementation needs to know what interface to implement

Can substitute better implementation without changing the client.

ADT Implementation in Java

Java ADTs.

- Keep data representation hidden with `private` access modifier.
- Define interface as operations having `public` access modifier.

```
public class Complex {
    private double re;
    private double im;

    public Complex(double re, double im) { . . . }
    public double abs() { . . . }
    public String toString() { . . . }
    public Complex conjugate(Complex a) { . . . }
    public Complex add(Complex a) { . . . }
    public Complex multiply(Complex a) { . . . }
}
```

Advantage: can switch to polar representation without changing client.

Note: all of the data types we have created are actually ADTs!

5

Y2K And Other Time Bombs

Time bombs.

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

```
public class Date {
    int seconds;

    public int getSecond()
    public int getMinute()
    public int getHour()
    public int getDay()
    public boolean after(Date d)
}
```

```
Date d1 = new Date();
d1.seconds = 31334534;
```

legal (but bad) Java client

```
public class Date {
    private int seconds;
    ↑
    public int getSecond()
    public void setSecond()
    public int getMinute()
    public void setMinute()
    public boolean after(Date d)
}
```

```
Date d1 = new Date();
d1.seconds = 31334534;
```

illegal Java client

6

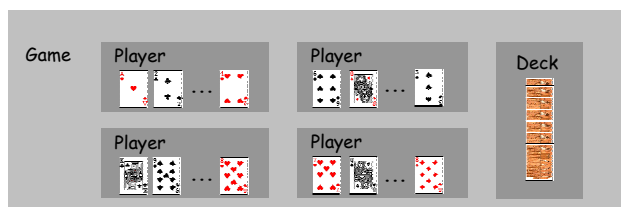
Modular Programming and Encapsulation

ADTs enable modular programming.

- Split program into smaller modules.
- Separate compilation.
- Different clients can share the same ADT.

ADTs enable encapsulation.

- Keep modules independent (include `main` in each class for testing).
- Can substitute different classes that implement same interface.
- No need to change client.



7

ADT Advantages

Modular programming and encapsulation.

- Essential for many real applications.
- Crucial software engineering principle.

Ex: building large software project.

- Software architect specifies design specifications.
- Each programmer implements one module.

Ex: build libraries.

- Language designer extends language with ADTs.
- Programmers share extensive libraries.

8