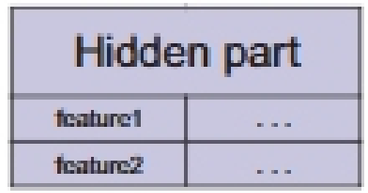
Object Oriented Programming

- Primary object-oriented language concepts
  - dynamic lookup
  - encapsulation
  - inheritance
  - subtyping
- Program organization
  - Organize concepts into objects and relationships between them
  - Build program to be *extensible*
    - Both *open* and *closed*
- Comparison
  - Objects as closures?

Objects

- An object consists of
  - hidden or private part
    - *instance variables / data members*
    - possibly, also some private functions
  - public part
    - *public operations*, or *member functions*, or *methods*, or *"messages"*
    - possibly, also some public instance variables
- In Eiffel terminology: *features*



- OO program
  - Creates a "*root object*"
    - which may create other objects
    - which, in turn, may create other objects ...
  - Objects
    - *send messages to / invoke methods on / apply features of* other objects

What's interesting about this?

- Universal encapsulation construct
  - Flight simulator
  - File system
  - Database
  - Window
  - List
  - Integer
- Metaphor usefully ambiguous
  - sequential or concurrent computation
  - distributed, synchronous or asynchronous communication

Why study Eiffel?

- *Completely* object-oriented approach from start to finish
- Design by contract (DBC) method
- Language mechanisms supporting OO and DBC
  - Classes, objects, assertions, contracts, ...
  - Single & multiple inheritance with
    - redefinition
    - undefinition
    - renaming
  - Genericity: contrained & unconstrained
  - Uniform type system: safe covariance
  - Agents (power of functional programming in OO)
  - "Once" mechanisms
  - Conversions
  - Unrestricted streaming: files, databases, networks, ...
  - Full garbage collection

Goals for Eiffel

- Productivity: faster time to market, fewer developers
- Reliability: fewer bugs
- Extensibility: be responsive to customer needs
- Reuse: stand on the shoulder of giants
- Efficiency: make the best use of hardware resources
- Maintainability: spend your time on new developments

(Courtesy of http://se.ethz.ch/teaching/2008-1/eiffel-02910)

Some Eiffel principles

- Abstraction
- Information hiding
- Seamlessness
- Reversibility
- Design by contract
- Open-closed principle
- Single choice principle
- Uniform access principle
- Command-query separation principle
- Option-operand separation principle
- Strong style rules
- . . .

(Courtesy of http://se.ethz.ch/teaching/2008-1/eiffel-02910)

Dogmatism

- Dogmatic where it counts
  - Information hiding, e.g., not allowed: `x.a := b`
  - Overloading
  - Style rules
  - "One good way to do anything."
- Purposely left out
  - Goto
  - Functions as arguments
  - Pointer arithmetic
  - Cryptic symbols, e.g., `x++`, `++x`
- Flexible when it makes no point to harass programmers
  - Syntax, e.g., optional semicolon's

(Courtesy of http://se.ethz.ch/teaching/2008-1/eiffel-02910)

Libraries

- Fundamental data structures and algorithms
- Portable graphics
- Internet, Web
- Lexical analysis, parsing
- Database interfaces
- . . .

(Courtesy of http://se.ethz.ch/teaching/2008-1/eiffel-02910)

Eiffel lifecycle model

- Focus on abstractions
- Reuse (consumer and producer)
- Seamless development
- Reversibility
- Single model principle
  - The software is the model
  - The model is the software
  - The software includes *everything* relevant to the project
  - Tools automatically extract views
    - analysis model, design, program, documentation, ...
- Contracts as a guide to
  - analysis, design, implementation, maintenance, management, ...

(Courtesy of http://se.ethz.ch/teaching/2008-1/eiffel-0291/)

Object-Oriented Software Construction

All you need is code

Code is all you need

–The bOOtles

(Courtesy of http://se.ethz.ch/teaching/2008-1/eiffel-0291/)

Seamless development

Single notation, tools, concepts, principles

Continuous, incremental development

Model, implementation, documentation are always consistent

Reversibility

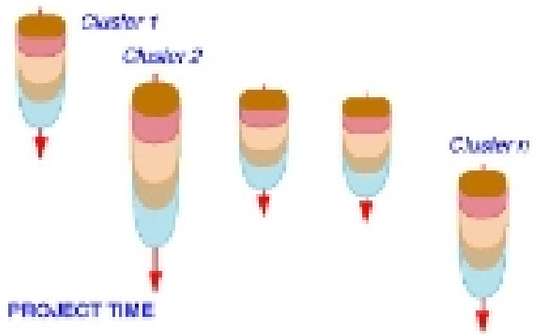
Example classes at each stage:

```
PLANE, ACCOUNT,...
STATE, COMMAND,...
HASH_TABLE,...
TEST_DRIVEN,...
TABLE, ...
```

(Courtesy of http://se.ethz.ch/teaching/2008-1/eiffel-0291/)

The cluster method

- Clusters (set of related classes)
- Start with foundational clusters
- Mini lifecycles, each covering a cluster
- Always a "demo-able" version at any point



(Courtesy of http://se.ethz.ch/teaching/2008-1/eiffel-0291/)