

TinySelf: Tiny object-oriented language

History and foundations

Tomas Petricek, Charles University

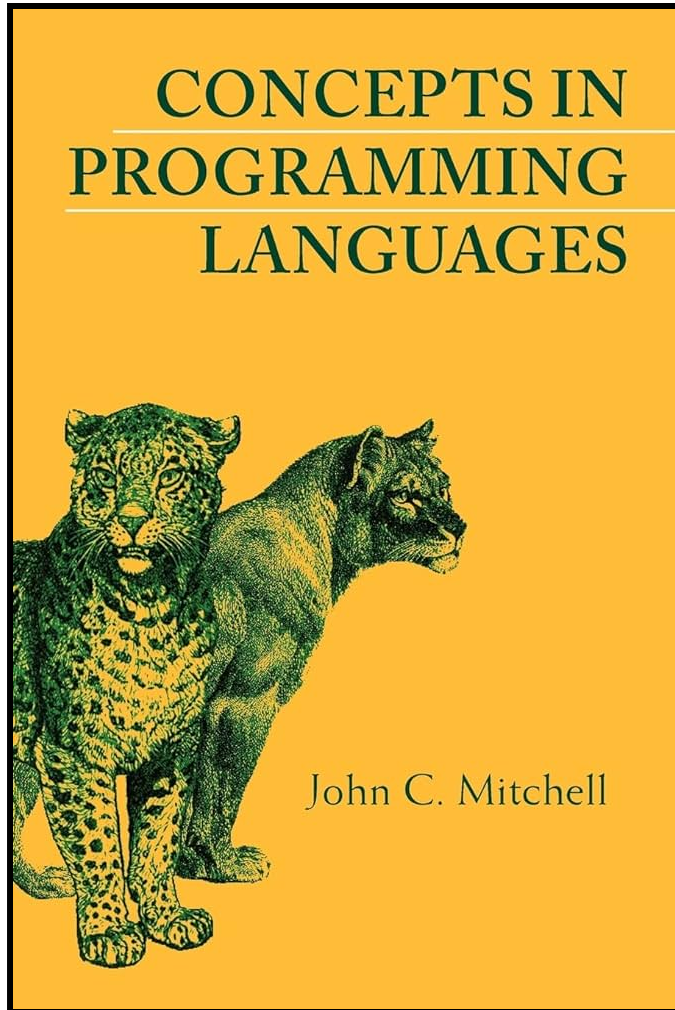
✉ tomas@tomasp.net

🦋 [@tomasp.net](https://tomasp.net)

🌐 <https://tomasp.net>

🌐 <https://d3s.mff.cuni.cz/teaching/nprg077>





Object-orientation

Dynamic lookup - object chooses how to respond

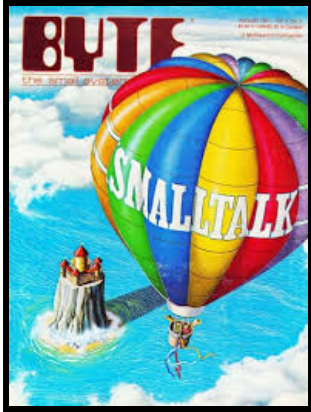
Abstraction - object state can be hidden from user

Subtyping - any compatible object can be used

Inheritance - reuse to implement a new object

Brief history

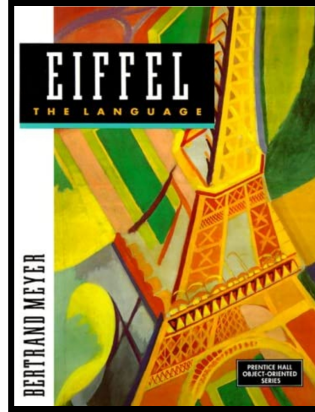
1960s-70s



Algol-based and scientific Simula

Tools for thought and messaging in Smalltalk

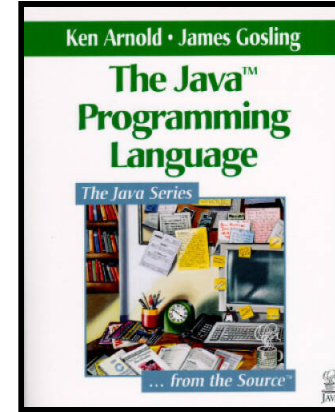
1980s



Rigorous Eiffel and "serious" C++

Prototypes and materialized objects in Self

1990s-2000s



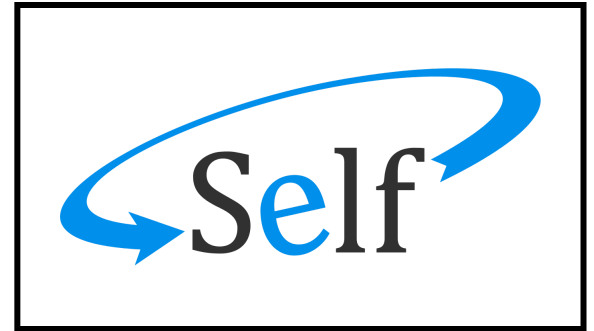
Class-based safe Java and C#

Prototypes in JavaScript and typed TypeScript

Why TinySelf?

"Pure" object-orientation

- Simple, uniform system
- Everything is an object (for real)
- Simpler than class-based Smalltalk



Shows the potential of objects

- Not Java-style organization of code
- Objects for code, state and execution!
- Objects with introspection and debugging!
- Objects and graphical interfaces!

Self & Morphic user interface framework

Visual programming

Programming by graphically manipulating objects on screen

Direct programming

Objects on screen *are* objects in the system

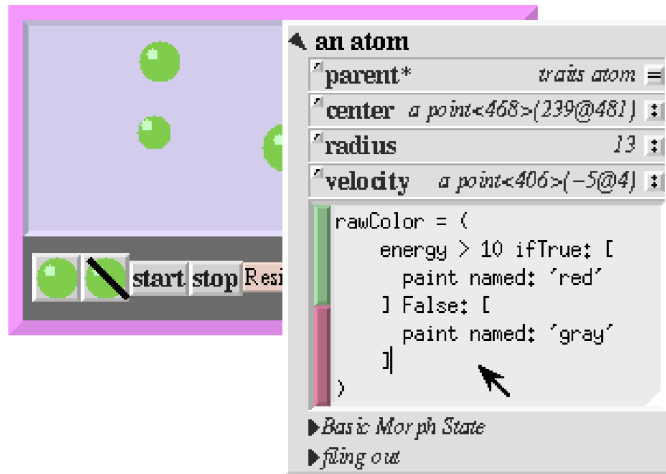


Figure 9. The user has selected one atom on which to experiment. The user changes the “rawColor” slot from a computed to a stored value by editing directly in the atom’s outliner.

Demo

(Not so) Tiny Smalltalk

TinySelf

Scope of the implementation

-  Prototype-based multiple inheritance
-  Explaining basic runtime structures
-  Methods with simple interpreted code
-  Inaccurate interpreter in "Self style"