Object-Z Highlights

Object-Z

- **is an *extension* of Z**
- **associates operations with a single state schema**
- **introduces the *class* concept as consisting of a state schema together with its associated operations, and used as a template for objects**
- **permits a class to be used as a type**
- **supports inheritance of classes, with adaptation**
- **originated the "history view" of object semantics, and introduced *history invariants***

Class Structure

A *class* is a template for objects — each object of the class has a state that conforms to the class' state schema, and is subject to state transitions that conform to the class' operations.

Classes as Types

If C is a class, the declaration x: C establishes x as a variable whose value is a reference to (i.e., the identity of, or a pointer to) an object of class C. Distinct references denote distinct objects. Hence as a type, C denotes the set of references to objects of class C.

The usual O-O "dot notation" is used in Object-Z. If C is a class-name and x: C a declaration, then x.y denotes the value of the state variable y of C, and x.op(...) denotes the application of operation op to x (and may be referred to as "sending x the op(...) message").

Inheritance

Inheritance is a mechanism for incremental class specification. New classes may be *derived* **from one or more existing classes, a conceptual counterpart to a schema importing others in Z. Definitions of the derived class are united with those of the inherited class. Inheritance permits the option of renaming attributes so name clashes may be resolved by renaming. The state and initialization schemas are conjoined with those of the derived class.**