MASSACHVSETTS INSTITVTE OF TECHNOLOGY
Department of Electrical Engineering and Computer Science
6.037—Structure and Interpretation of Computer Programs
IAP 2019

Object-Oriented Programming

Problems

1. Write a `food` class
   - Input state is the `name`,`nutrition` value, and `good-until` time.
   - Additional state is the `age` of the food, initially 0.
   - Methods are:
     - `NAME` - returns the name of the food
     - `AGE` - returns the age of the food
     - `SIT-THERE` - takes an amount of time, and increases the age of the food by the amount.
     - `EAT` - return the nutrition if the food is still good; 0 otherwise.
2. Write an `aged-food` class
   - Input state is the same as the `food` class, with an additional parameter, which is the `good-after` time.
   - Should inherit from the `food` class.
   - Methods are:
     - `SNIFF` - returns `#t` if it has aged enough to be good.
     - `EAT` - returns 0 if the food is not good yet; otherwise behaves like normal food.
3. Extend the object system to support dynamic mixin classes. A "mixin" is when one class, after being defined, can be modified to include methods definitions from some other class[1]. This effectively allows a class to inherit from multiple classes, and is also sometimes called a role or an abstract base class. [2]
4. Further extend the system to support mixins on *instances*, in addition to classes. That is, some particular instance of `aged-food` (a `stinky-cheese-wheel`, for instance) might mix in the methods of the `round` trait to get the `ROLL` method.

[1] Technically, since this is only adding the methods of the other class, and not its state, this is a "trait" and not a mixin

[2] Mixins actually first appeared in an object system for Lisp Machine Lisp in 1982; the name was inspired by Steve's Ice Cream Parlor in Somerville, which allowed toppings to be mixed into their ice cream.