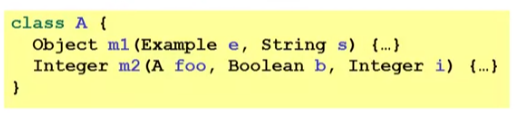
**Statically-Typed OOP**

* Now contrast multiple inheritance and mixins with Java/C#-style interfaces
* Important distinction, but interfaces are about static typing, which Ruby does not have
* So will use Java [pseudo]code after quick introduction to static typing for class-based OOP…
  + Sound typing for OOP prevents “method missing” errors

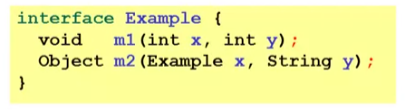
**Classes as Types**

* In Java/C#/etc. each class is also a type
* Methods have types for arguments and result



* If c is a (transitive) subclass of D, then C is a subtype of D
  + Type-checking allows subtype anywhere supertype allowed
  + So can pass instance of C to a method expecting instance of D

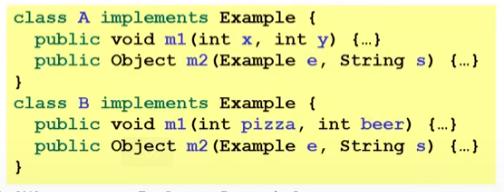
**Interfaces are Types**



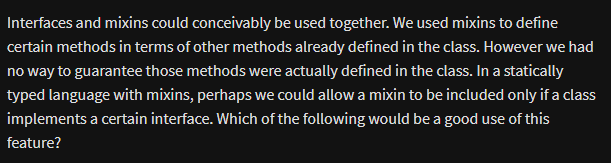
* An interface is not a class; it is only a type
  + Does not contain method *definitions*, only their *signatures* (types)
    - Unlike mixins
  + Cannot use *new* on an interface
* Like an interface of TypeScript

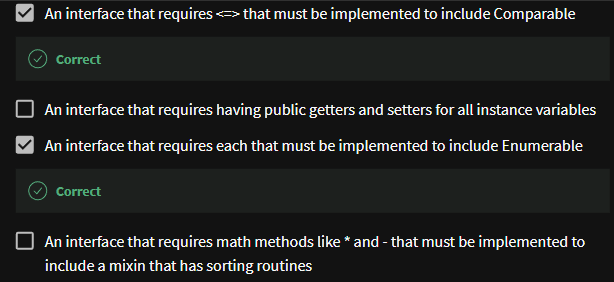
**Implementing Interfaces**

* A class can explicitly implement **any number** of interfaces
  + For class to type-check, it must implement every method in the interface with the right type
    - More on allowing subtypes later!
  + Multiple interfaces no problem; just implement everything
* If class type-checks, it is a subtype of the interface



* Implements is like extends in TypeScript





**Multiple interfaces**

* Interface provide no methods or fields
  + So, no questions of method/field duplication when implementing multiple interfaces, unlike multiple inheritance
* What interfaces are for:
  + “Caller can give any instance of any class implementing I”
    - So callee can call methods in I regardless of class
  + So much more flexible type system
* Interfaces have little use in a dynamically typed language
  + Dynamic typing already much more flexible, with trade-offs we studied