Programación orientada a objetos. Primer parcial 1999.

Lunes 3 de mayo de 1999.

1. El software es inherentemente complejo. La complejidad excede nuestra capacidad de compresión debido a nuestras limitaciones cognoscitivas. La complejidad es una propiedad esencial de los grandes sistemas de software. La tarea del equipo de desarrollo de software es dar la ilusión de simplicidad. ¿Cuáles son los fundamentos del paradigma de descomposición orientada a objetos con los que es posible dar la ilusión de simplicidad y cómo están puestos en práctica en Smalltalk? Justifiquen sus respuestas.
2. En la descomposición orientada a objetos vemos el mundo como una colección de agentes autónomos que colaboran para exhibir algún nivel más elaborado de comportamiento. La colaboración aparece cuando un objeto solicita un servicio de otro objeto. ¿De qué forma un objeto cliente solicita un servicio a un objeto servidor? ¿De qué forma un objeto servidor satisface la solicitud de un objeto cliente en Smalltalk?
3. A continuación aparece el método `=` de la clase `IndexedCollection`.
   a. Analicen este método y expliquen cómo funciona.
   b. El método retorna `true` o `false` según diversas condiciones; den un ejemplo de código Smalltalk que se pueda evaluar con ShowIt en el Transcript para cada una de esas condiciones.

```
= aCollection
        "Answer true if the elements contained by
         the receiver are equal to the elements
         contained by the argument aCollection."
    | index |
    self == aCollection
        ifTrue: [^true].
    (self class == aCollection class)
        ifFalse: [^false].
    index := self size.
    index ~= aCollection size
        ifTrue: [^false].
    [index <= 0]
        whileFalse: [
            (self at: index) = (aCollection at: index)
                ifFalse: [^false].
            index := index - 1].
    ^true
```

4. Un objeto tiene estado, comportamiento e identidad. Definan estos términos e indiquen cómo se ponen en práctica en Smalltalk.