Ejercicio 2 - Preguntas para después de hacer el ejercicio

Abstracción de los tests 01 y 02

En los test 01 y 02 hay código repetido. Cuando lo extrajeron crearon algo nuevo. Eso es algo que estaba en la realidad y no estaba representado en nuestro código, por eso teníamos código repetido. ¿Cuál es esa entidad de la realidad que crearon?

En los tests 01 y 02 ciertamente había código repetido. Cuando aplicamos el algoritmo para quitarlo nos dimos cuenta de que lo que se repetía explícitamente era la medición del tiempo que llevaba en hacer una tarea. En el mundo real, esa entidad se la conoce como un cronómetro. Originalmente, el tiempo empezaba a correr, se realizaba la acción (ya sea añadir o remover algun cliente del libro) y luego se lo detenía. Por último se lo comparaba con un tiempo límite.

En primer lugar, realizamos la abstracción de este código repetido y a continuación lo hicimos más general, por ejemplo, el tiempo límite o la tarea que se quería realizar.

Investigando un poco el ambiente nos dimos cuenta que `CustomerBookTest` es una subclase de `TestCase` . Lo que no es detalle menor ya que allí encontramos implementaciones de mensajes que necesitabamos. Hicimos uso del mensaje `should: notTakeMoreThan:` que tenía la misma funcionalidad de nuestra abstracción (contaba el tiempo de una acción). Descartamos lo nuestro y le dimos uso a ese mensaje y a muchos más en otros tests.

Cómo representar en Smalltalk

¿Cuáles son las formas en que podemos representar entes de la realidad en Smalltalk que conocés? Es decir, ¿qué cosas del lenguaje Smalltalk puedo usar para representar entidades de la realidad?

En SmallTalk hay varias formas de representar entes de la realidad. Uno de ellos son los objetos pero también existen las clases (junto con las instancias).

Trabajar con objetos significa la representación de un ente “específico” de la realidad, los objetos se los modela particularmente para que puedan responder una serie de mensajes y así representar el comportamiento deseado (similar al ente de la realidad).

Por otro lado, las clases son más “generales”, poseen una relación con superclases y subclases. Al utilizar alguna de esta se trabaja con instancias. Al hacer esto, el implementador define mensajes para darle forma y caracter a la instancia (se podría decir que la instancia es un caso “particular” de cada clase). Una vez hecho esto, las instancias pueden responder sus propios mensajes y conjuntamente los que definen a sus superclases y subclases. Algo que no pueden hacer los objetos, ya que estos últimos tienen una relación padre-hijo. Si el hijo no sabe responder un mensaje, se lo envía al padre, pero si el padre no sabe responder un mensaje (por más que su hijo sí sepa hacerlo) no podrá responderlo.

Teoría de Naur

¿Qué relación hay entre sacar código repetido (creando abstracciones) y la teoría del modelo/sistema (del paper de Naur)?

A lo largo del paper, Naur detalla y refuerza la importancia de la construcción de teoría por parte del programador:

> (...) la calidad de la teoría construida por el programador dependerá en gran parte de la
> familiaridad del programador con las soluciones modelo a problemas típicos, con
> técnicas de descripción y verificación, y con principios de estructuración de sistemas
> formados por muchas partes en interacciones complicadas.

Esto quiere decir que para poder construir un buen modelo se requiere partir de una buena base de conocimiento. La calidad de la abstracción será determinada por la familiaridad y la relación que contenga el programador con lo que se quiere representar.

La relación que existe entre sacar código repetido y la teoría del modelo es que para poder crear abstracciones debemos construir y apropiarnos de una teoría que nos permita entender la forma en que las leyes centrales aplican a ciertos aspectos de la realidad. Es decir, adquirir ciertos conocimientos que nos permita entender y explicar cómo la solución a nuestro problema se relaciona con los problemas del mundo real.

Sin la construcción de una teoría no seríamos “(...) capa[ces] de reconocer fenómenos similares en el mundo, para poder utilizar correctamente las reglas de la teoría expresadas” debido a que no conoceríamos el dominio del problema real en su totalidad. Realizar abstracciones conlleva conocimiento de lo que se quiere abstraer para así reducir por no decir eliminar completamente el código repetido. En otras palabras es necesaria “(...) la comprensión de ciertos tipos de similitudes entre situaciones y acontecimientos del mundo real (...)” para poder aplicarlas a otros aspectos y abstraer lo de mayor importancia.