1. Concepto de abstracción

Introducción

♦ Problemas complejos: Modularidad + Abstracción

• Modularidad: permite dividir el problema en componentes que pueden ser combinados para resolver el problema original.

• Abstracción: ayuda a realizar una buena elección de dichos componentes, separando el propósito del módulo de su implementación.

• Modularidad y abstracción se complementan mutuamente.

♦ **Abstracción**:

• Permite simplificar el análisis y resolución de un problema, diseñando soluciones a alto nivel.

• Especifica claramente qué hace cada componente o módulo, antes de su implementación en un lenguaje de programación.

• La especificación clarifica el diseño de la solución, porque se centra en la funcionalidad a alto nivel, sin entrar en los detalles de implementación.

• Permite modificar partes de una solución sin afectar significativamente a otras partes.

• Es esencial para equipos de proyectos, donde habrá que usar métodos escritos por otros, sin conocer a menudo los algoritmos utilizados.

2. Abstracción de datos

♦ **Abstracción de datos** o **tipo abstracto de datos** (**TAD**): nuevo tipo de dato más un conjunto de operaciones que permiten manipular los objetos de dicho tipo

Abstracción

Shaw define abstracción como "*una descripción simplificada o especificación de un sistema que enfatiza algunos de los detalles o propiedades del mismo mientras suprime otros. Una buena abstracción es aquella que enfatiza detalles significativos al lector o usuario y suprime detalles que son, al menos por el momento, irrelevantes o causa de distracción*". Una abstracción denota las características esenciales de un objeto que lo distinguen de todos los demás tipos de objeto y proporciona así fronteras conceptuales respecto a la perspectiva del observador.

La **abstracción** es una técnica de programación que permite definir nuevos tipos de datos por el usuario. La esencia de la abstracción es similar al uso de un tipo de dato de un lenguaje de programación, cuyo uso se realiza sin saber cómo está representado o implementado.

Las abstracciones deben ser completas, es decir, los objetos deben abstraer y encapsular tanto los datos como sus procesos (de hecho, la abstracción y la encapsulación están íntimamente ligadas). Las *cosas* se perciben no sólo a través de sus propiedades, sino a través de su comportamiento: se conocerán por su estado.

Una abstracción se centra en la vista externa de un objeto, de modo que sirva para separar el comportamiento esencial de un objeto de su implementación. Definir una abstracción significa describir una entidad del mundo real, no importa lo compleja que pueda ser, y a continuación utilizar esta descripción en un programa.

Veamos varias formas de explicar la abstracción

A) Shaw[8] define abstracción como "*una descripción simplificada o especificación de un sistema que enfatiza algunos de los detalles o propiedades del mismo mientras suprime otros. Una buena abstracción es aquella que enfatiza detalles significativos al lector o usuario y suprime detalles que son, al menos por el momento, irrelevantes o causa de*

distracción". Una abstracción denota las características esenciales de un objeto que lo distinguen de todos los demás tipos de objeto y proporciona así fronteras conceptuales respecto a la perspectiva del observador.

La **abstracción** es una técnica de programación que permite definir nuevos tipos de datos por el usuario. La esencia de la abstracción es similar al uso de un tipo de dato de un lenguaje de programación, cuyo uso se realiza sin saber cómo está representado o implementado.

Las abstracciones deben ser completas, es decir, los objetos deben abstraer y encapsular tanto los datos como sus procesos (de hecho, la abstracción y la encapsulación están íntimamente ligadas). Las *cosas* se perciben no sólo a través de sus propiedades, sino a través de su comportamiento: se conocerán por su estado.

Una abstracción se centra en la vista externa de un objeto, de modo que sirva para separar el comportamiento esencial de un objeto de su implementación. Definir una abstracción significa describir una entidad del mundo real, no importa lo compleja que pueda ser, y a continuación utilizar esta descripción en un programa.

El uso de un objeto por vía de su especificación como tipo abstracto o clase abstracta significa que, si su aspecto interno experimenta modificaciones, otras partes del sistema no se verán afectadas.

8 Shaw, M. (1984), "Abstraction Techniques in Modern Programming Languages," *IEEE Software*, Vol. 1, No. 4,

2007- Metodología y tecnología de programación
Universidad de Vigo.

B)

Abstracción

VMINO NUEVO

La abstracción se refiere a quitar las propiedades y acciones de un objeto para dejar sólo aquellas que sean necesarias. ¿Qué significa esto último?

Diferentes tipos de problemas requieren distintas cantidades de información, aun si estos problemas pertenecen a un área en común. En la segunda fase de la creación de la clase Lavadora, se podrían agregar más atributos y acciones que en la primera fase. ¿Vale la pena?

Valdría la pena si usted pertenece al equipo de desarrollo que generará finalmente la aplicación que simule con exactitud lo que hace una lavadora. Un programa de este tipo (que podría ser muy útil para los ingenieros de diseño que actualmente estén trabajando en el diseño de una lavadora) deberá ser tan completo que permita obtener predicciones exactas respecto a lo que ocurriría cuando se fabrique la lavadora, funcione a toda su capacidad y lave la ropa. De hecho, para este caso podrá quitar el atributo del número de serie, dado que posiblemente no será de mucha ayuda.

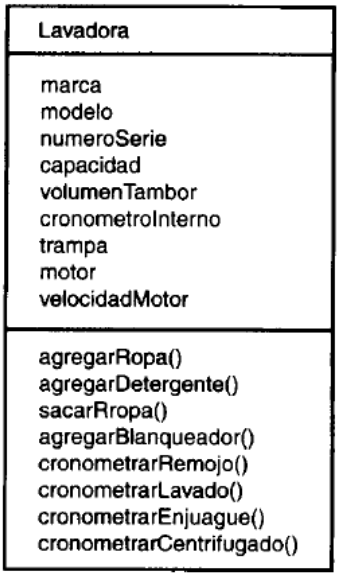
Por otra parte, si va a generar un software que haga un seguimiento de las transacciones en una lavandería que cuente con diversas lavadoras, posiblemente no valdrá la pena. En este programa no necesitará todos los atributos detallados y operaciones del párrafo anterior, no obstante, quizá necesite incluir el número de serie de cada objeto Lavadora.

En cualquier caso, con lo que se quedará luego de tomar su decisión respecto a lo que incluirá o desechará, será una abstracción de una lavadora.

Libro: Aprendiendo UML en 24 horas.
Joseph Schmuller

Ejemplo de abstracción

FIGURA 2.2
La adición de atributos y acciones al modelo lo acerca a la realidad.



3 ¿Qué Significa POO?

La filosofía de la POO (*Object Oriented Programming, Programación Orientada a Objetos)* rompe con este esquema, dando lugar a una nueva idea, *el objeto*.

El objeto es una abstracción en la que se unen sentencias y datos, de tal forma que a un objeto sólo lo van a poder tratar los métodos definidos para él, y estos métodos están preparados para trabajar específicamente con él. Este grado de compenetración evita que un método pueda tratar datos no apropiados, o bien que unos datos puedan ser tratados por un método no adecuado, ya que la llamada a cualquier método ha de ir siempre precedida del objeto sobre el que se quiere actuar, y éste sabe si ese método se ha definido o no para él.

C++ es un lenguaje que contiene estos y otros conceptos de POO.

Libro: PROGRAMACIÓN ORIENTADA A OBJETOS CON C++ -
Enrique Alba Torres
Andrés Rubio del Río

---

4. PROGRAMACIÓN CON ABSTRACCIÓN

Para realizar un programa complicado, debemos dividirlo en partes que resuelvan subproblemas del problema original. En este caso el objetivo debe ser la descomposición del programa en módulos que sean ellos mismos programas de tamaño pequeño y que interactúen unos con otros de forma sencilla y bien definida.

En programación estructurada se recomienda usar la abstracción como forma de descomponer un problema. Al llevarla a cabo se ignoran ciertos detalles de manera que el

problema original se vea en forma más simple al atender solo a que operaciones hay que realizar, pero no a cómo hay que realizarlas.
El método de descomposición en base a abstracciones se continúa aplicando hasta llegar a operaciones lo bastante sencillas como para ser programadas de forma directa.

5. TAD.
La denominación TAD engloba dos clases de abstracciones:
Abstracciones de datos.
Abstracciones funcionales.
Las primeras aparecen al abstraer el significado de los diferentes tipos de datos significativos que intervienen en el problema. Permiten definir nuevos tipos de datos especificando sus posibles valores y las operaciones que los manipulan.
Las segundas surgen al plantearse de una manera abstracta las operaciones significativas del problema. Son una herramienta muy poderosa que permite dotar a una aplicación de operaciones que no están definidas directamente en el lenguaje en el que estamos trabajando.
En resumen, un TAD es un conjunto de valores y unas operaciones definidas sobre esos valores. Cada vez que deseemos emplear el TAD solo lo podemos hacer con las operaciones definidas, incluso no sabiendo cómo están implementadas.
5.1 Los TAD permiten:
• **Encapsular información**:
− Es posible localizar la definición del tipo y el código de todas las operaciones sobre ese tipo en una determinada sección del programa
− Un cambio en el TAD permite revisar una pequeña sección del programa (no hay detalles en otras partes que puedan ocasionar errores relacionados con ese tipo de datos)
− Forma de organizar los programas en unidades lógicas que en muchos lenguajes de programación pueden compilarse separadamente
• **Ocultar la representación**:
− El código de las unidades de programa que usan el tipo no dependen de su implementación
− La implementación puede cambiar sin afectar a las unidades de programa que usan el tipo
− Se garantiza la fiabilidad de las unidades de programa que usan un TAD y la integridad de los objetos de ese tipo (las unidades de programa no pueden cambiar directamente los objetos, sino sólo a través de las operaciones que proporcionan los tipos)

5.2 Características de los TAD.
× Cada modulo es una abstracción del problema, no conocemos el detalle.
× Se puede realizar una especificación breve y precisa del TAD.
× Con esta especificación se puede usar correctamente el TAD.
× Si realizamos, no se va a alterar considerablemente la especificación.
× Cada modulo TAD va a poder ser compilado por separado.
× Las comunicaciones entre los módulos van a ser pocas y claras.
× Cada modulo debe tener un tamaño razonable.

Libro: 1991: Estructuras de datos y algoritmos
Miguel Angel Díaz