UNDERSTANDING THE TERMS “DATA STRUCTURES” & “ADT”

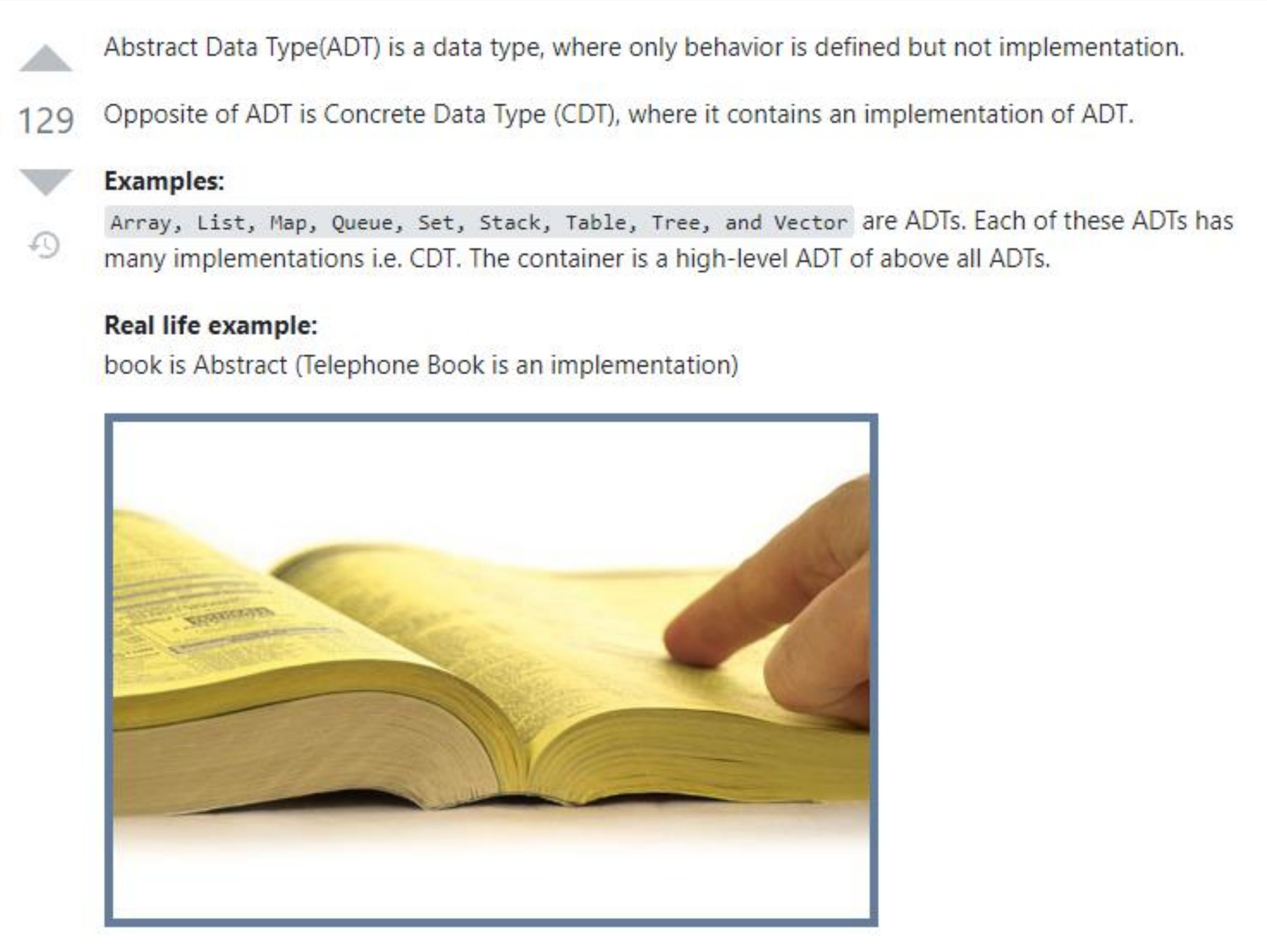
UNDERSTANDING THE TERMS “DATA STRUCTURE” & “ADT”

BACS2063

Data Structures and Algorithms

What is an Abstract Data Type (ADT)?

Definition 1



Abstract Data Type(ADT) is a data type, where only behavior is defined but not implementation.

Opposite of ADT is Concrete Data Type (CDT), where it contains an implementation of ADT.

Examples:

`Array, List, Map, Queue, Set, Stack, Table, Tree, and Vector` are ADTs. Each of these ADTs has many implementations i.e. CDT. The container is a high-level ADT of above all ADTs.

Real life example:

book is Abstract (Telephone Book is an implementation)

Source:
https://stackoverflow.com/questions/10267084/what-is-adt-abstract-data-type

What is an Abstract Data Type (ADT)?

Definition 2

An abstract data type is an abstraction of a data structure that provides only the interface to which the data structure must adhere. The interface does not give any specific details about something should be implemented or in what programming language.

Source: https://www.javatpoint.com/abstract-data-type-in-data-structure

What is Data Structure? (*From Chapter 1 Notes*)

The implementation of ADT is often referred to as data structure, using some constructs and primitive data types.

A representation of data and the operations allowed on that data (Weiss, 2010)

How do you express an ADT in Java?

In Java, an ADT can be expressed by an **interface**, which is simply a list of method declarations, where each method has an empty body.

An ADT is realized by a concrete data structure, which is modeled in Java by a **class**. A class defines the data being stored and the operations supported by the objects that are instances of the class. Also, unlike interfaces, classes specify **how** the operations are performed in the body of each method.

Source: https://www.cpp.edu/~ftang/courses/CS240/lectures/adt.htm

How do you express an ADT in Java?

Thus, a Java class is said to implement an interface if its methods include all the methods declared in the interface, thus providing a body for them.

However, a class can have more methods than those of the interface. Also, the complier or run-time system requires that the types of parameters that are actually passed to methods rigidly conform with the type specified in the interface. This requirement is known as **strong typing**. It helps catch programming errors that would otherwise go unnoticed.

Source: https://www.cpp.edu/~ftang/courses/CS240/lectures/adt.htm

Recommended Websites for further reading

OpenDSA Data Structures and Algorithms Modules Collection

https://opendsa-server.cs.vt.edu/ODSA/Books/Everything/html/index.html

JavaTPoint

https://www.javatpoint.com/abstract-data-type-in-data-structure