1b: ADT (Abstract data type)

There are some basic definitions we need to get out of the way before we dive into more complicated stuff.

Type: a collection of values. For example, a boolean type is collection of two values – true and false. A simple type (such as boolean or integer) will not have subparts, whereas an aggregate or composite type will contain several pieces of information.

Data item: a data item is a member of the type i.e. a piece of information derived from a type.

Data type: This is a combination of a type (the values) and the operations that can be applied to that type. For example, given the possible values of an integer (-2^{31} – 2^{31} -1), possible operations include addition, subtraction, division, etc.

There is a difference between the logical concept of a data type and its physical implementation (even if the line is blurred). Consider an array: Its logical concept is that of potentially homogeneous data items stored in a collection where each item is accessed using an index number. However, its physical implementation is a block of contiguous memory locations. The distinction between the two needs to be made because there are cases where arrays (particularly multi-dimensional arrays) have been designed with the same logical idea, but different physical implementation (i.e. the data items are not in contiguous memory locations) but that doesn't make them any less of an array than a typical array.
Another example is the list. Its logical concept is a collection of items, but its physical implementation could be either an array or a linked list. Both are lists, but they have very different physical implementations.

Abstract data type: When the idea of a data type is realised as a software component i.e. the type and operations are realised in code. This is typically represented by an interface. It does NOT specify how the data type is implemented. In fact, such details are typically hidden from the user of the data type or any outside access.

Data structure: The complete implementation of an ADT. It typically takes the form of a class where the operations of the ADT are captured by member functions of that class.

Food for thought:
an integer as well as the operations that an integer can be put through make up the integer data type. The java int variable is a physical representation of the abstract integer. This variable, alongside the java int operations make up an ADT. But there is difference between the two (i.e. a difference between a java int ADT and the abstract integer). The java int ADT can only support values within a certain range and we know that real integers do not have that limitation. If this limitation is a deal breaker for you, then you will have to figure out a different ADT (with different implementations) to represent your integers.

Ref: A Practical Introduction to Data Structures and Algorithm Analysis, Clifford A. Shaffer.