CS 214 – DATA STRUCTURES FALL 2015

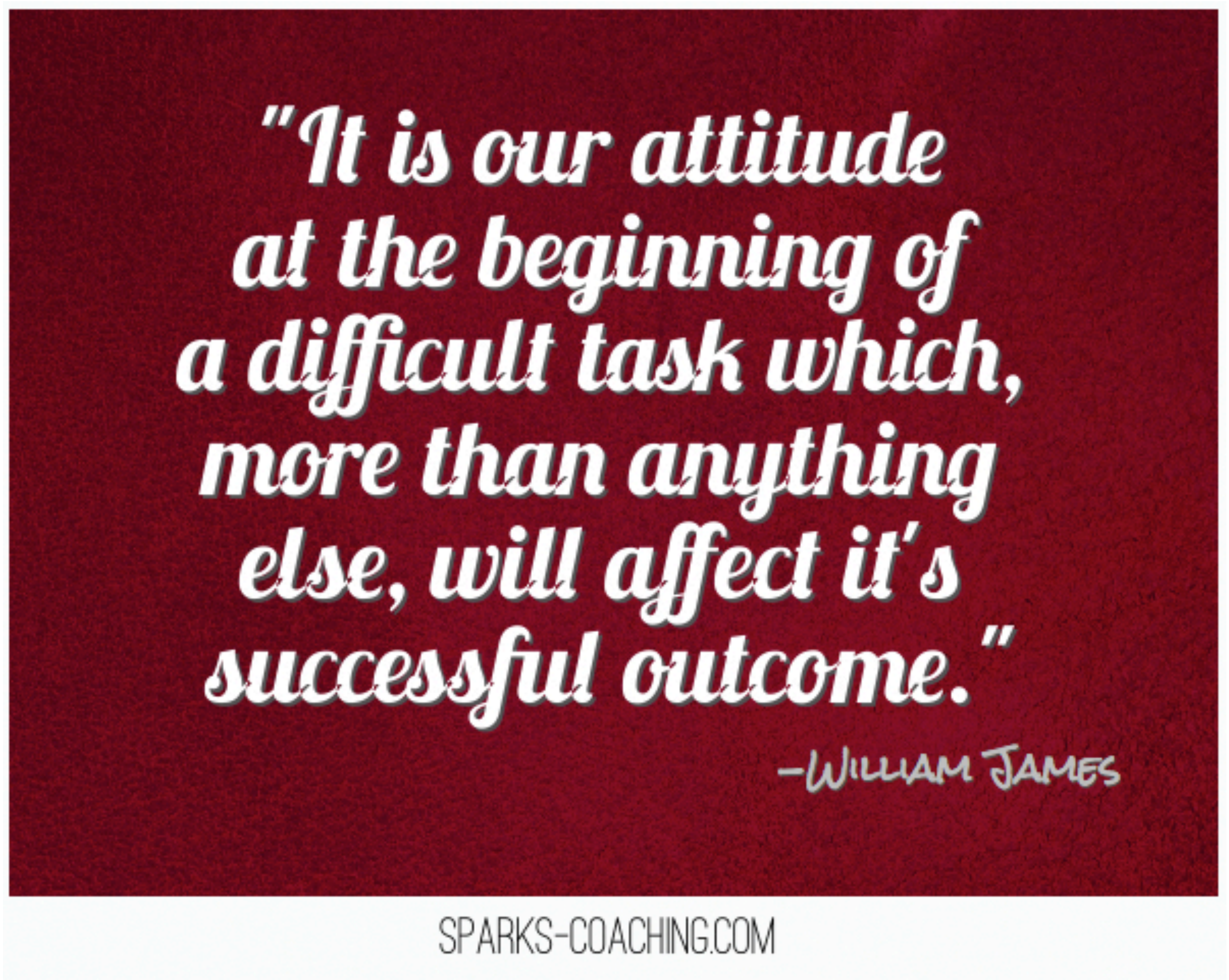
Lecture 1 – ADT

Group **B**

18th September 2017

Dr. Mai Hamdalla

mai@fci.helwan.edu.eg



"It is our attitude at the beginning of a difficult task which, more than anything else, will affect it's successful outcome."

–William James

SPARKS-COACHING.COM

House Keeping

- Piazza:
  http://piazza.com/fci_helwan_university/fall2017/cs214
  access code: **CS214 Fall 2017**
  - Office Hours
  - Announcements
  - Suggested Books
  - Syllabus
- Email: mai@fci.helwan.edu.eg

House Keeping

- Lecture Attendance
- Sections - Labs
    - Attendance
    - Quiz
- Developing Environment
    - Code Blocks

WHAT IS A DATA STRUCTURE?

What is a Data Structure?

- **Data** is the basic entity or fact that is used in calculation or manipulation process.

- **Data structure** is a way of organizing data items by considering its relationship to each other.

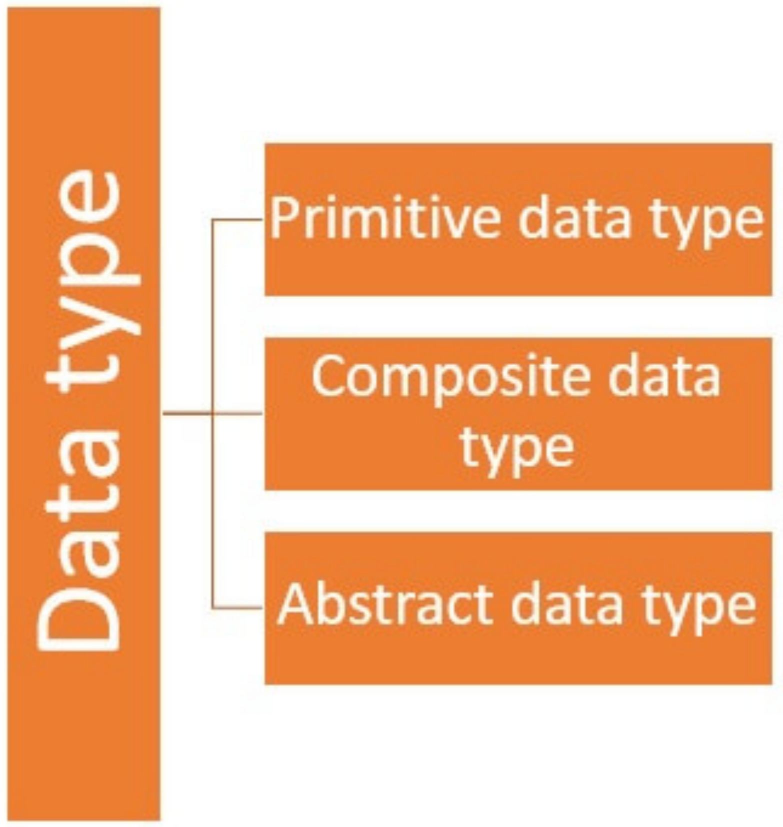
What is a Data Structure?

- A data structure is a way of **organizing data** that considers not only the **items stored**, but also their **relationship** to each other.

- Advance knowledge about the relationship between data items allows designing of efficient algorithms for the manipulation of data.

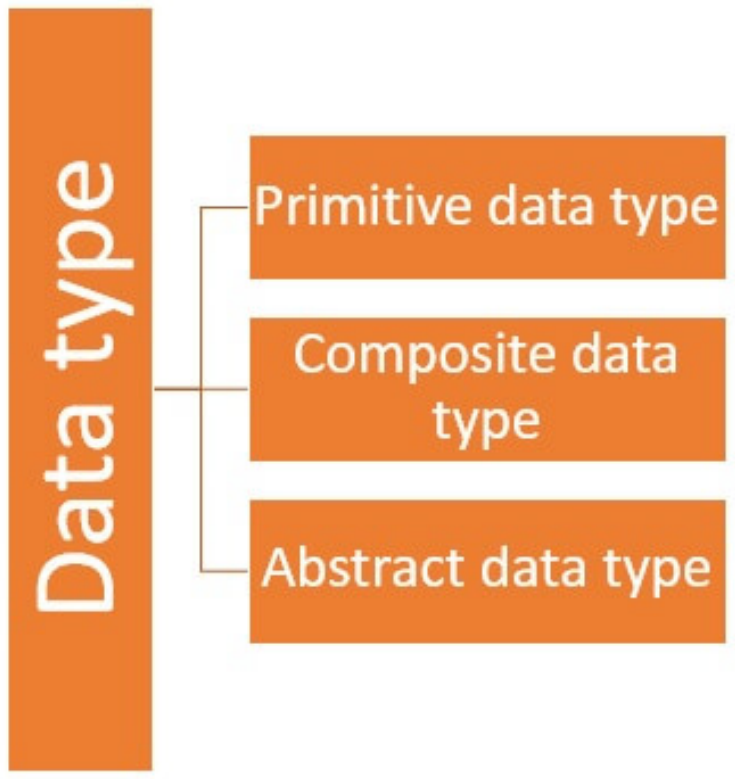
What is a Data Type?

- Is the organization of data that can store definite type of information.



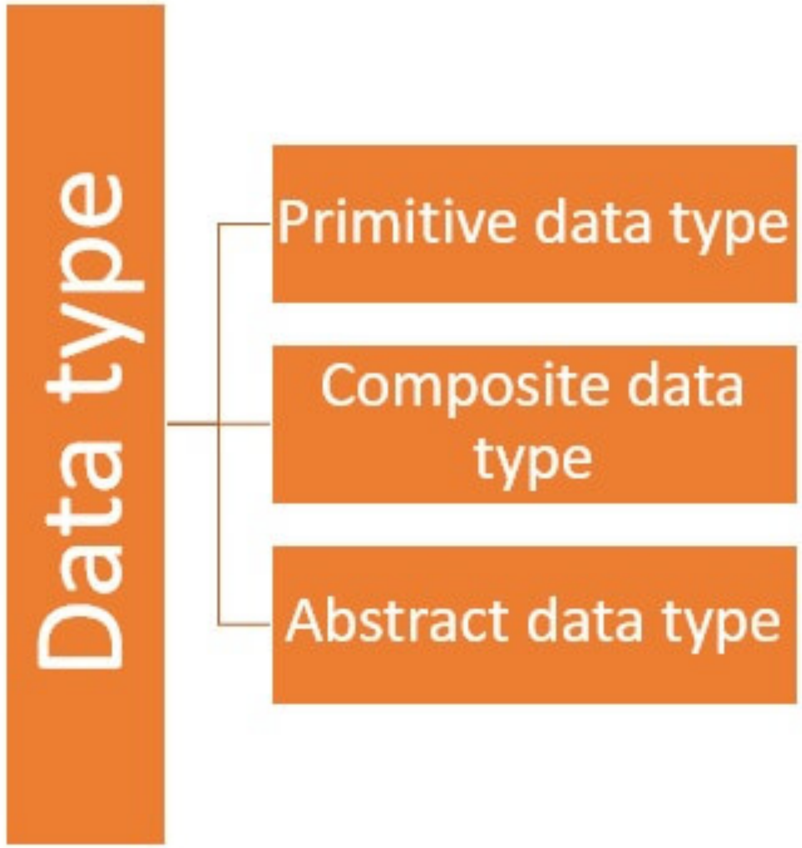
Primitive Data Type

- Is basic entity of a language and are used while creating variables in a program.
- Include: floating-point, integer, double.



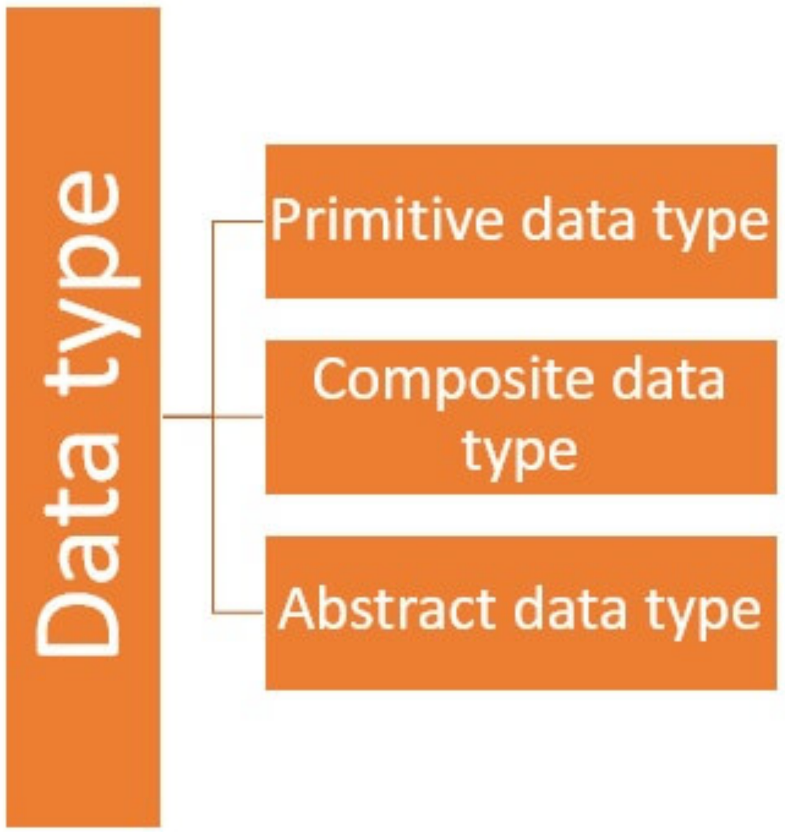
Composite Data Type

- Is the management of multiple related data as a distinct datum.
- Include: array, union, record.



Abstract Data Type

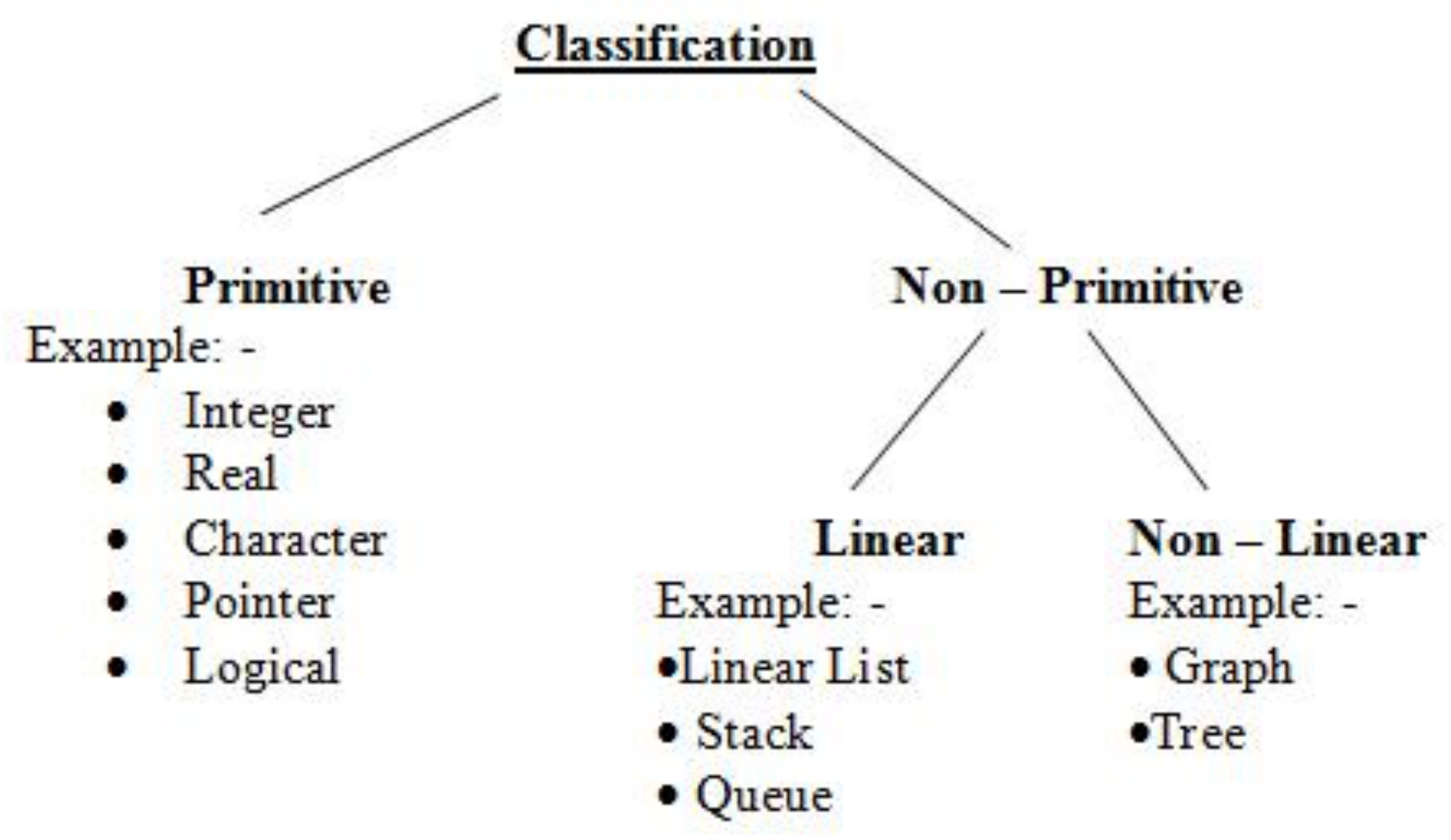
- The separation of data usage and detail of its execution is the reason for use of abstract data type.
- Include: stack, queue, graph, tree… etc.



Data Structure Classification

- Data structure are divided into two main categories:
  - **Primitive data structure**
  - **Non-primitive data structure**

Data Structure Classification



Data Structure – Why?

- The selection of a particular data structure will help the programmer design more efficient programs.

- The efficiency of the program depends on two measurements :
  1. Space complexity
  2. Time complexity

Data Structure – Why?

- The time complexity can now be expressed as function of number of key operations performed.

- One solution may require more space but less time while the other requires less time space but takes more time. Thus, we may have to sacrifice one at the cost of the other.

Memory Space Needed by a Program

- Include:
  - Instruction space
  - Data space
  - Environment stack space:

    The data saved on the environment stack is:

    (a) Return address

    (b) Values of all lead variables and
      the values of formal parameters in
      the function being invoked .

Memory Management

- There are two types of memory allocations :
  1. **Static memory** allocation or Compile time

     ```
     float a[5], f;
     ```

  2. **Dynamic memory** allocation or Run time

```
ptr = (int *) malloc (10 * sizeof (int));
```

ABSTRACT DATA TYPES (ADT)

Abstraction

- Abstraction allows us to consider the high-level characteristics of something without getting bogged down in the details.

- ***Process abstraction***, for example, allows one to ignore the details of portions of a large program, and to be able to assign portions of a large program to different teams who work independently.

Abstract Data Type (ADT)

- Data abstraction allows us to design and use a *data structure without* considering how it is implemented.

- Abstract Data Type (ADT) – An organized data object and a set of operations for manipulating it

ADT = Organized data + Operations

Abstract Data Type (ADT)

- ADT is defined in terms of the operations that can be performed on instances of the type rather than by how those operations are actually implemented.
- In other words, an ADT defines the *interface* of the objects.
- The interface is considered a kind of contract between the implementers and the users of the ADT.

Information Hiding (Encapsulation)

- The hiding of the data structure implementation inside the ADT is referred to as ***encapsulation*** *or* ***information hiding***.

- We refer to a program that uses an ADT as a **client** (or **user**) and a program that specifies the ADT as an **implementation**

Information Hiding (Encapsulation)

- You use the structure at the “User Level” without caring about the details at the “Implementation Level”.

- Your program, i.e., the user level, does not change even if the implementation of the used structure is changed.

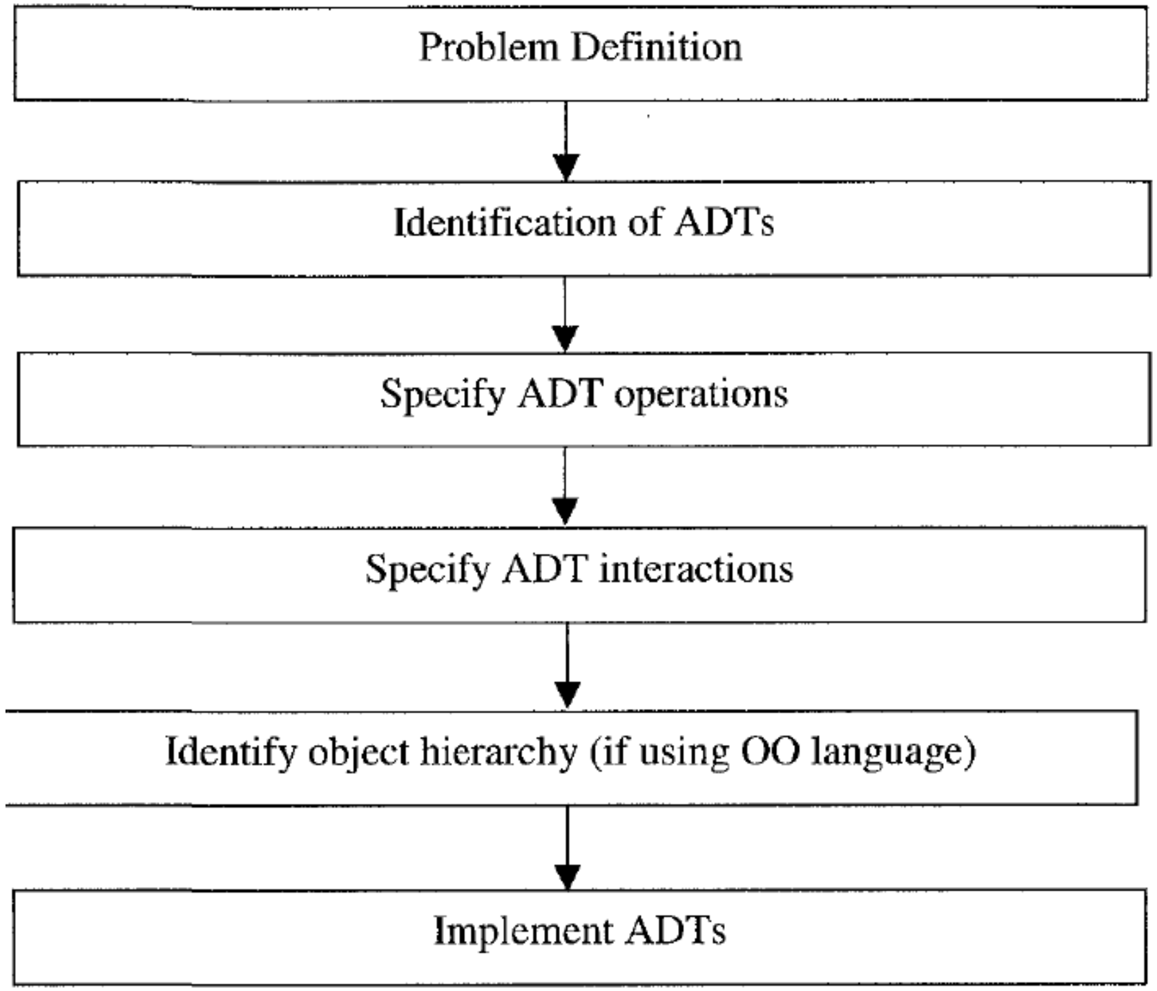
Why use ADT?

- Rather than having to understand the detailed implementation of the set operations, the user only has to study the interface at a much higher level so much time can be saved.
- The ADT can be used in a variety of different programs

Why use ADT?

- The implementation of the component can be changed without affecting some other component.

System design with ADTs



YOUR RESPONSIBILITY

Your Responsibility

- Sign up on Piazza
- Install Code Blocks
- Review C programming basics
- Watch online tutorials
- Get EXCITED about Data Structures!

CS 214 – DATA STRUCTURES FALL 2015

Lecture 1 – ADT

Group **B**

18th September 2017

Dr. Mai Hamdalla

mai@fci.helwan.edu.eg