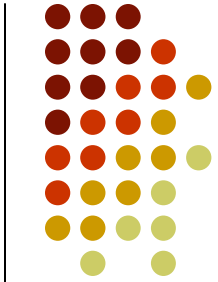
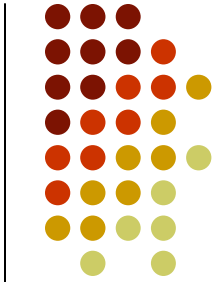
Design



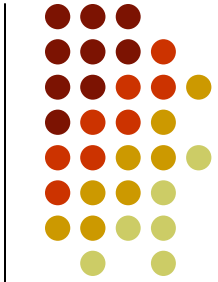
- The goal is to design a *modular* solution, using the techniques of:
  - Decomposition
  - Abstraction
  - Encapsulation
- In Object Oriented Programming this is done by:
  - Identifying the objects in the problem
  - Identifying what actions can be performed on objects, and
  - Determining how the objects interact with each other

Modular Design: Decomposition



- Simplify a complex problem by dividing it into smaller, simpler parts (modules)
  - Modules should be loosely coupled and
  - Highly cohesive
  - The purpose, inputs and outputs of each module and methods should be specified
  - This specification should not include a description of the implementation
- Different programmers can work on different modules

Modular Design: Abstraction



- Abstraction separates the purpose of a module from its implementation
  - The implementation details are ignored (or suppressed) to concentrate on the purpose of a module
- Procedural abstraction
  - Separates the purpose of a method from its implementation
  - Can be specified in pre and post conditions
- Data abstraction
  - Specifies *what* can be done to data in a data collection, not *how* the data is stored or *how* it is done
  - A collection of data specified in this way is called *an abstract data type* (**ADT**)

Procedural Abstraction

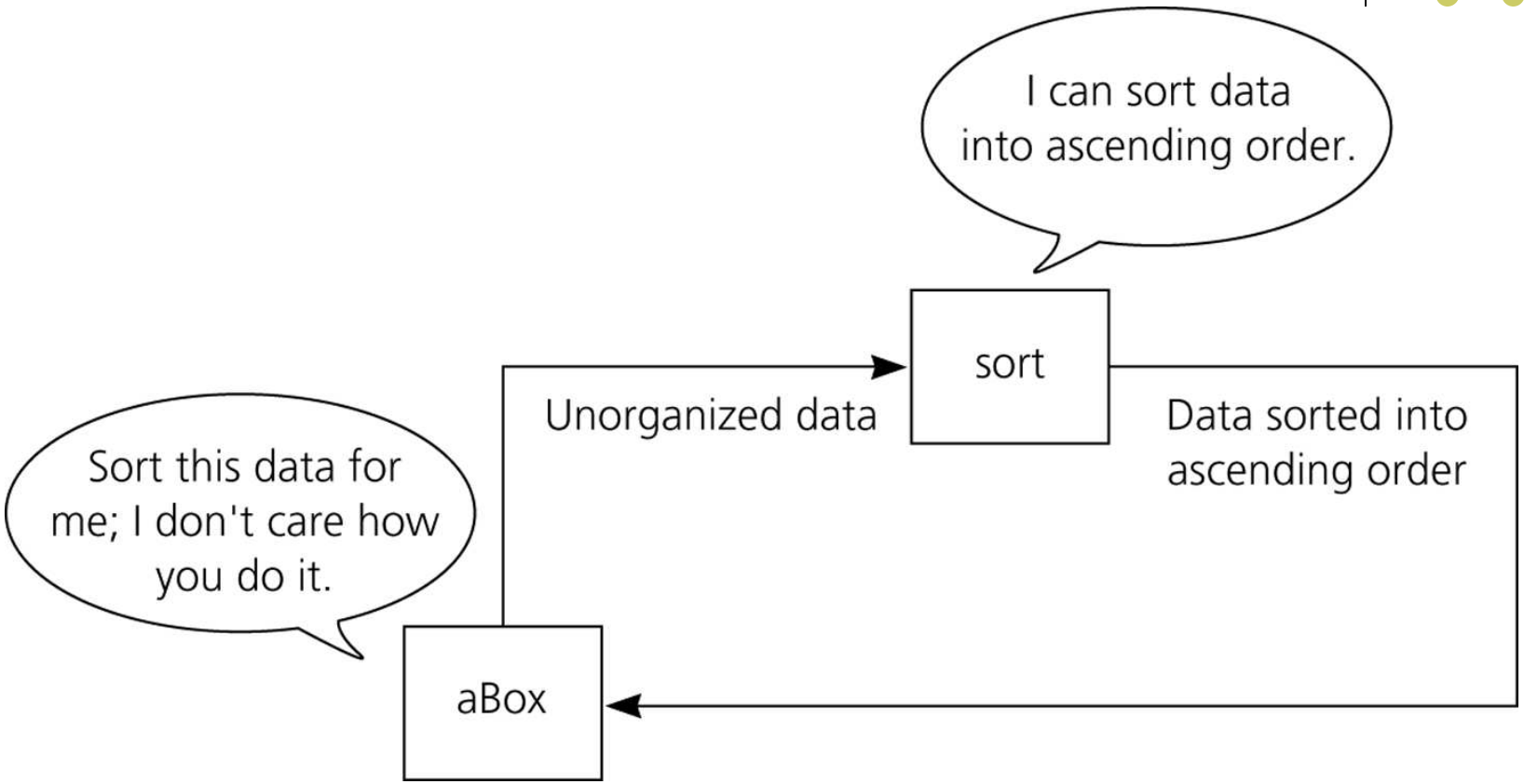
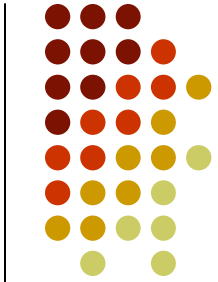


Figure 2-2

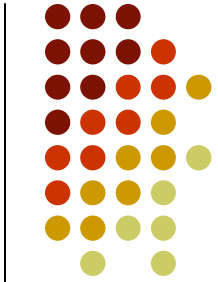
The details of the sorting algorithm are hidden from other parts of the solution.

Modular Design: Encapsulation



- A decomposition technique where a complex object is decomposed into modules which have a separate and distinct purpose
  - i.e. highly cohesive
  - Because a module is a complete entity with one purpose it can be re-used in another application
- This allows each module to be relatively independent from the others
  - i.e. the modules are loosely coupled
  - This also makes it easy for different people to work on different modules at the same time
- Because modules are independent their inner details can be hidden from each other
  - Referred to as *information hiding*
  - To use a module one is only required to learn its interface, rather than its entire implementation

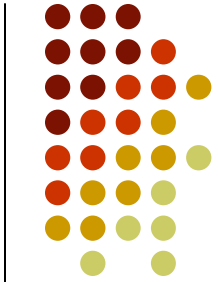
Modular design (summary)



- Decomposition: dividing modules to smaller simpler submodules
- Abstraction: separation of the purpose of module from details (implementation)
- Encapsulation: packing related data together, providing interface to data, while hiding actual data

Note: The last two items and information hiding are highly-related concepts (can be easily confused). For details see: http://www.itmweb.com/essay550.htm

Advantages of modular design



- Helps to implement large programs
- Isolates errors: helps in debugging
- Easy to read
- Easy to modify (changes are localized in few modules)