Data Structures -- Introduction

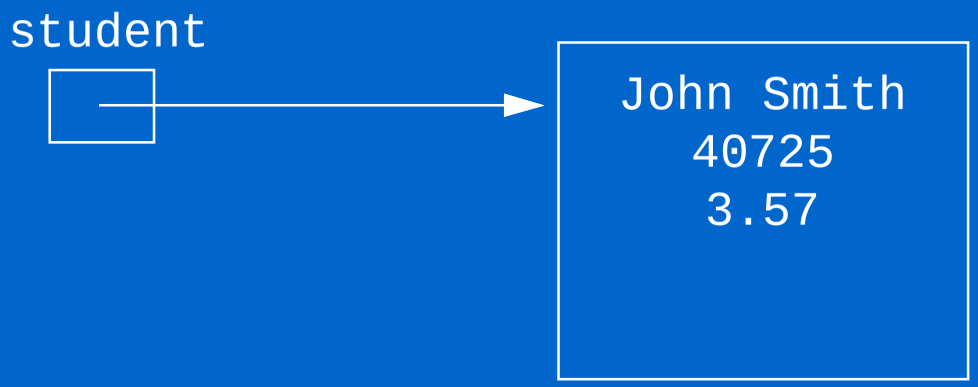
- Let’s explore some advanced techniques for and managing information
- Chapter 16 focuses on:
  - dynamic structures
  - Abstract Data Types (ADTs)
  - linked lists
  - queues
  - stacks

Static vs. Dynamic Structures

- A static data structure has a fixed size
- This meaning is different than those associ static modifier
- Arrays are static; once you define the num elements it can hold, it doesn’t change
- A dynamic data structure grows and shrinks by the information it contains

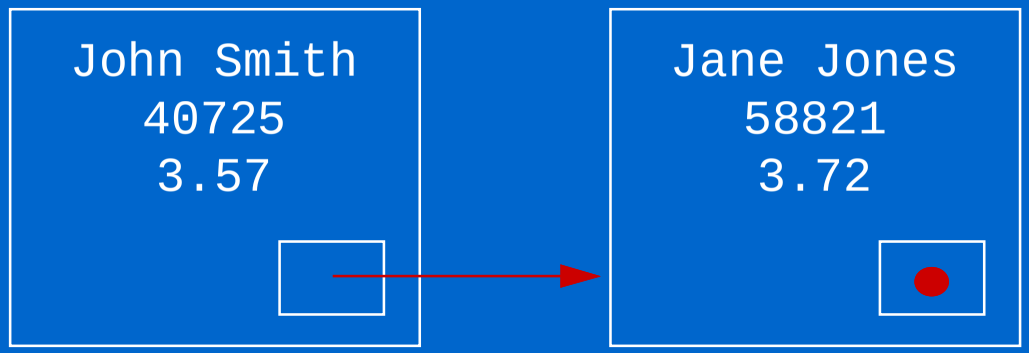
Object References

- Recall that an object reference variable that st
  the address of an object
- A reference can also be called a pointer
- They are often depicted graphically:
-



References as Links

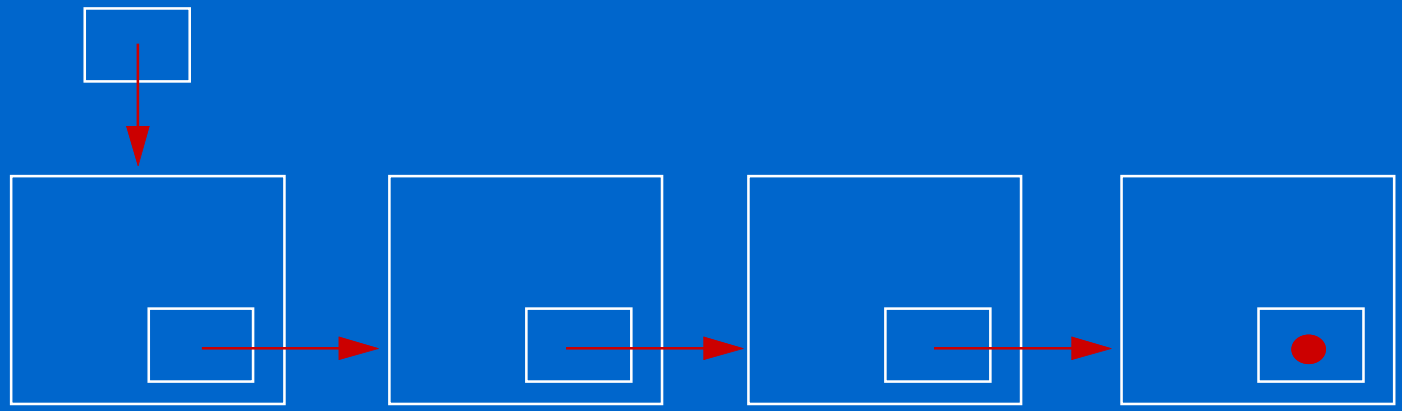
- Object references can be used to create links between objects
- Suppose a Student class contained a reference to another Student object
-



References as Links

- References can be used to create a variety structures, such as a linked list

- studentList



- See Library.java

Abstraction

- Our data structures should be abstractions
- That is, they should hide details as approp
- This helps manage complexity
- Our original Library solution is not abstr
- One problem is that we use a public variabl links
-

Abstract Data Types

- An abstract data type (ADT) is an organized colle of information and a set of operations used that information
- The set of operations define the interface
- As long as the ADT accurately fulfills the the interface, it doesn't really matter how implemented
- Objects are a perfect programming mechanism ADTs because their internal details are enc
-

Coupling and Cohesion

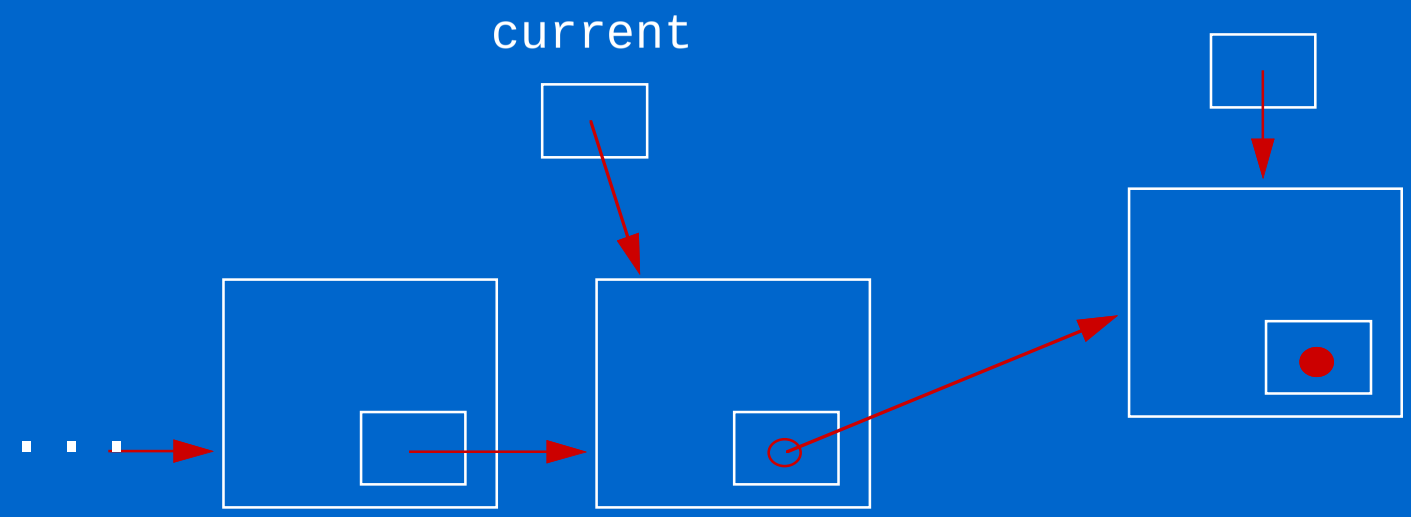
- A well-defined ADT attempts to minimize cou
  while maximizing cohesion
- Coupling is the strength of the relationshi
  components
- Cohesion is the strength of the relationshi
  parts of one component
- We want to formally specify a simple relati
  between the ADT and the outer world, and pu
  things that relate to the ADT management in

Library Revisited

- A better version of the Library solution wo book_list object that provides services suc
  - add a book to the list
  - print the book list
  -
- The book_listobject in turn interacts with Book objects
- Each Bookobject governs its own references
- See Library2.java

Adding a Node

- To add a new node to the end of a linked li
  list looking for the last node, then add th
- The last node is the one with a reference
-

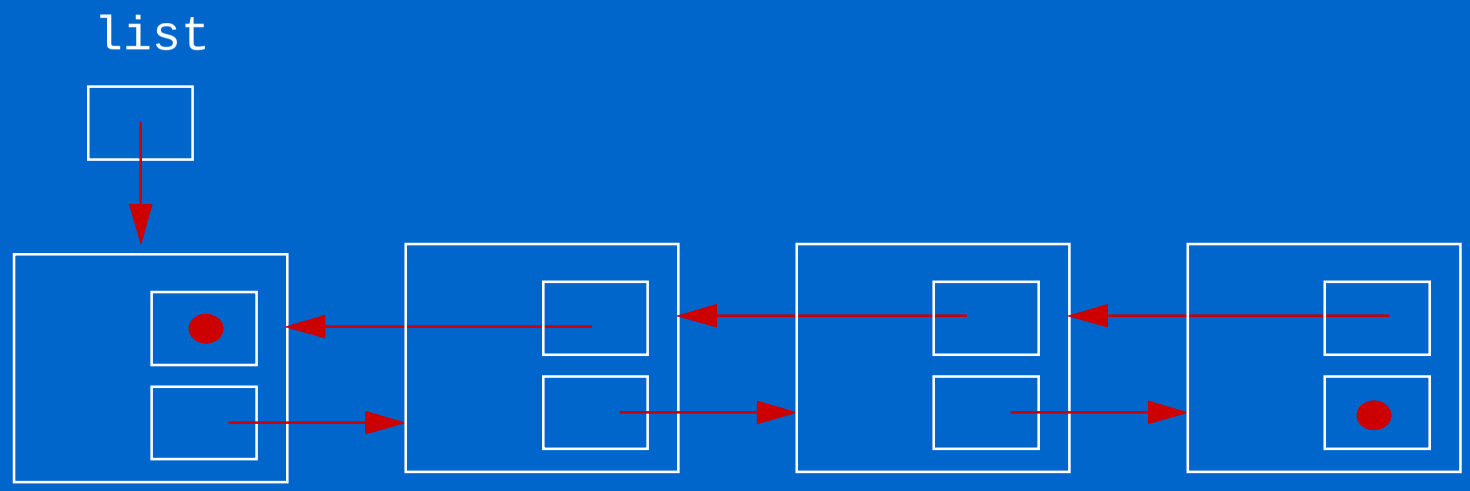


Other List Operations

- You may also want to perform such operation
  - add a node to the front of the list
  - add a node somewhere in the middle of the list
  - delete a node from the list
  -
- Each operation can be defined separately as method
- How an operation is implemented depends on underlying representation of the data struc

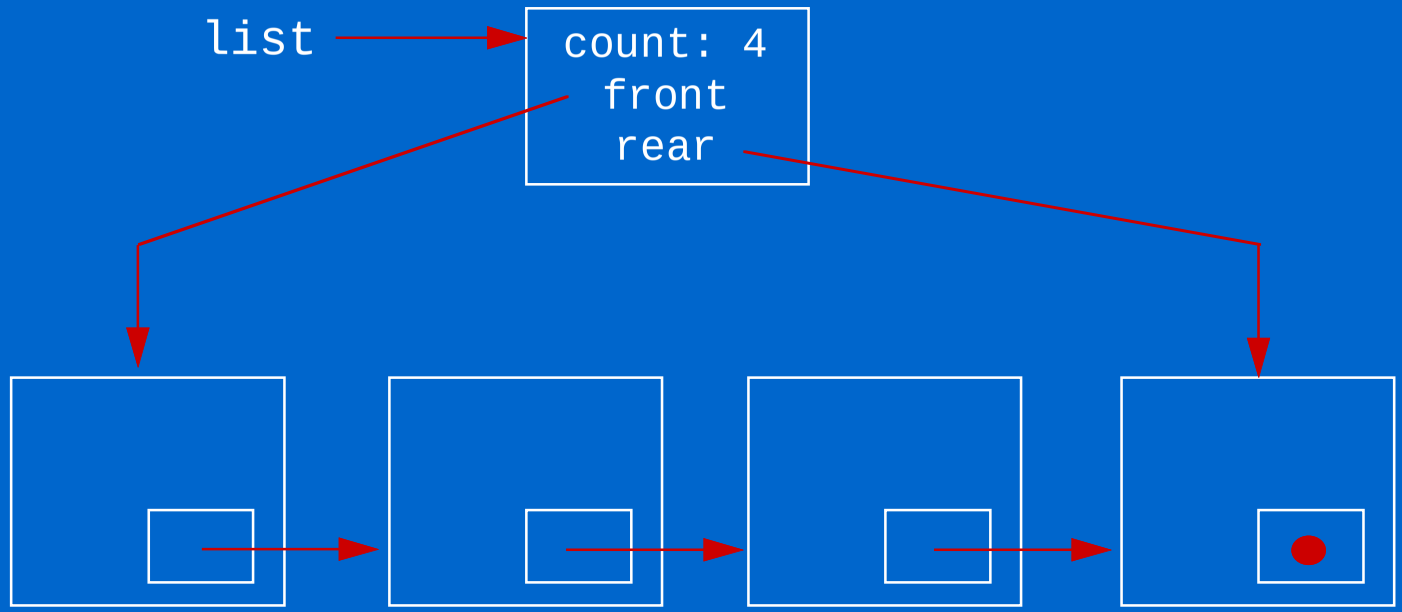
Other Dynamic List Implementations

- It may be convenient to implement a doubly linked list, with next and previous referen



Other Dynamic List Implementations

- It may also be convenient to use a separate
  with references to both the front and rear

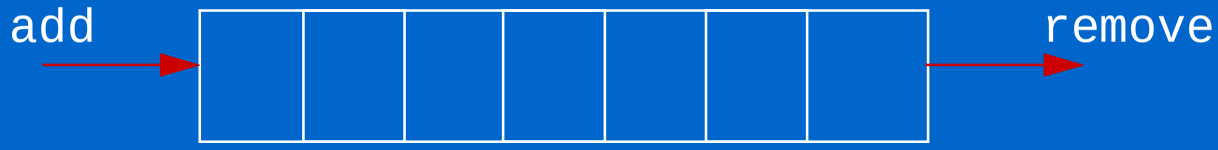


Library Revisited

- Another problem with the existing Library s
  that the Book class contains the reference to
  object in the list
- Ideally, we would like to separate the data
  management from the information it holds
- The objects we want to store in the list sh
  to be involved with the list references
- As we explore other data structures, we wil
  separation

Queues

- A queue ADT is similar to a list but adds i the end of the list and removes them from t
- It is called a FIFO data structure: First-
- Analogy: a line of people at a bank teller



Queues

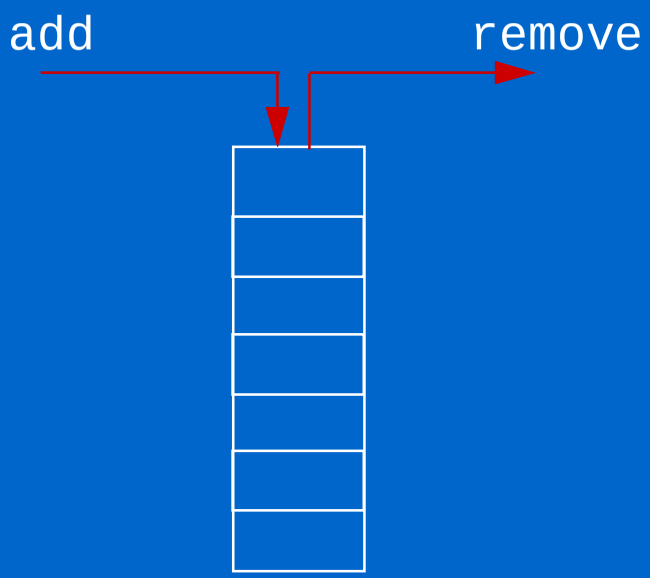
- We can define the operations on a queue as
  - enqueue - add an item to the rear of the queue
  - dequeue - remove an item from the front of the qu
  - empty - returns true if the queue is empty
  -
- By operating on the Object, any object can stored in the queue
- See QTrek.java

Stacks

- A stack ADT is also linear, like a list or
- Items are added and removed from only one e stack
- It is therefore LIFO: Last-In, First-Out
- Analogy: a stack of plates

Stacks

- Stacks are often drawn vertically:



Stacks

- Some stack operations:
  - push - add an item to the top of the stack
  - pop - remove an item from the top of the stack
  - peek - retrieves the top item without removing it
  - empty - returns true if the stack is empty
  -
- The java.utilpackage contains a Stackclass, which is implemented using a Vectorclass
- See Decode.java