FIT 100 Collections

- Collections are data structures that let you track multiple related objects
- Collections use one piece information (a Key) in order to access another piece of information (a Value)
- A Collection is like a table made of 2 columns
  - One column holds the keys
  - The other column holds values
- Keys in a Collection must be unique

FIT 100 Collections: Why use them?

- You might need to keep track of a group of related things
- You might also need to search and find specific information about those related things
- Using a Collection and Iteration will allow you to cycle through many, many rows of information and pick out the one row that is necessary

FIT 100 Collections: How do I use them?

- Collections in VB are created just like variables
  - Dim phoneList As New Collection
  - If declared GLOBALLY, then the whole Form has access to that information
- Once a Collection is created, it is initialized
  - In Collections, initializing happens every time another row is added to the Collection table

    phoneList.Add <value>, <key>
- So what are keys and values?
  - Keys are the unique pieces of data that give access to the Value

    phoneList.Add “922-8909”, “Grace Whiteaker”

FIT 100 Collections: How do I use them?

- Once you have added information to your Collection, you can access it if you know the key

  x = phoneList.Item (“Grace Whiteaker”)

What is in x? “922-8909”

- Think about this:

  If you had to add a large number of items to a collection, and the pattern of those items was very specific (say it was just a series of numbers or letters in sequence), how would you do it?

FIT 100 Actions You Take On Collections

- When you have a collection of things, you usually perform some pretty standard operations on it:
  - Add
  - Remove
  - Select one or more items from it
  - Count the number of items in it
- Collections in VB are similar to Collections in real life
  - You can add to them (but the keys must be unique):
    collectionName.Add <value>, <key>
  - You can remove items from them:
    collectionName.Remove <key>
  - You can find a specific item's value:
    collectionName.Item <key>

FIT 100

Topics In Programming

Why Algorithms Matter: Search Collections

FIT 100 Searching A List

- If there's no order to the list (like the deck of cards)...
  - best you can do is start at the beginning
  - This is called sequential or linear search
- Binary search is a simple, common sense way to search through an *ordered* set of items.
  - Questions, often referred to as queries or probes, are asked to *find if the desired item is smaller or larger.*
  - If the question is chosen from the middle of the sequence, ½ the possibilities are eliminated with each answer.
  - It's a bit like 20 questions, but MUCH more specific.

FIT 100 How Good is a Particular Algorithm?

- You might think we can't answer this question without programming a computer and trying it.
- Amazingly, it is possible to make very good comparisons between algorithms without programming them!
- Basic idea: estimate the number of "steps" each algorithm needs to solve the problems.
- This gives us an abstract, mathematical way to compare the speed of different algorithms

Algorithm vs. Program

- Remember that an algorithm is an abstraction.
- We can apply it, at least mentally, to a variety of situations, even without a computer
- A program incorporates all the details needed for a computer to perform the algorithm
- A program for search will encode the algorithm for a specific situation, in a specific language, with specific assumptions

Battle of the Algorithms

- Binary Search: Each question allows you to throw out half of the unexamined items (throw half of the phone book away each time)
- Linear Search: Each question lets you tear out only one page, or throw out one card

Do The Math for Searching 200 Items

	linear	binary
step 0	200 remaining	200
step 1	199	100
step 2	198	50
step 3 see where it's going?	197	25

Bottom Line

- It can be shown mathematically when a sorted list of N items is to be searched:
- Linear sort needs on average about N/2 steps
- Binary sort needs on average about Log_2N steps
  - No, you don't have to be able to compute Log_2N !
  - Just remember this, the bigger N is, the bigger the improvement.

FIT 100 Trade-Offs

- If we know algorithm A has a better formula than algorithm B:

Would we ever still want to use algorithm B??

FIT 100 Searching a small set of things: 20

	linear	binary
step 0	20 remaining	20
step 1	19	10
step 2	18	5
step 3 Could you tell the difference in time if a computer does the search?	17 …	3 …