FIT100

Lab 13: "Hotel Guest Manager"

Spring 2002

Introduction:

This activity will spread over two lab sections. In it, you will continue to get practice with VB features you already have seen (such as collections), and learn some new ones which will be useful for the last part of Project 2. In particular, the "for each" and "for next" loops.

Collections

A collection is a *data structure* that lets you work with multiple related objects. In this lab we will use a collection to hold the names of all the people checked into a hotel and the rooms that they are checked into. To create a collection, you use a dim statement:

```
Dim colGuests as New Collection
```

You can think of a collection as a table. Each row in the table has two entries a *key* and a *value*. For our collection, there may be several people checked into different rooms in the hotel. Each person has their own row. The room they're staying in is the key, and the person's name is the value.

Key	Value
Room 2	Grace
Room 13	Caro

To insert a new row into the table, you use the Add method. It looks like this:

```
colGuests.Add "Adam", "Room 5"
```

Note that the value comes first and the key comes second. Also, if you try to add an entry to a collection that already has a row with the same key (i.e. if there's already a "Room 5" entry), that's an error. You should write an error trap to handle this situation in procedures that use the Add method.

To remove an entry from a table, you use the Remove method. It looks like this:

```
colGuests.Remove "Room 13"
```

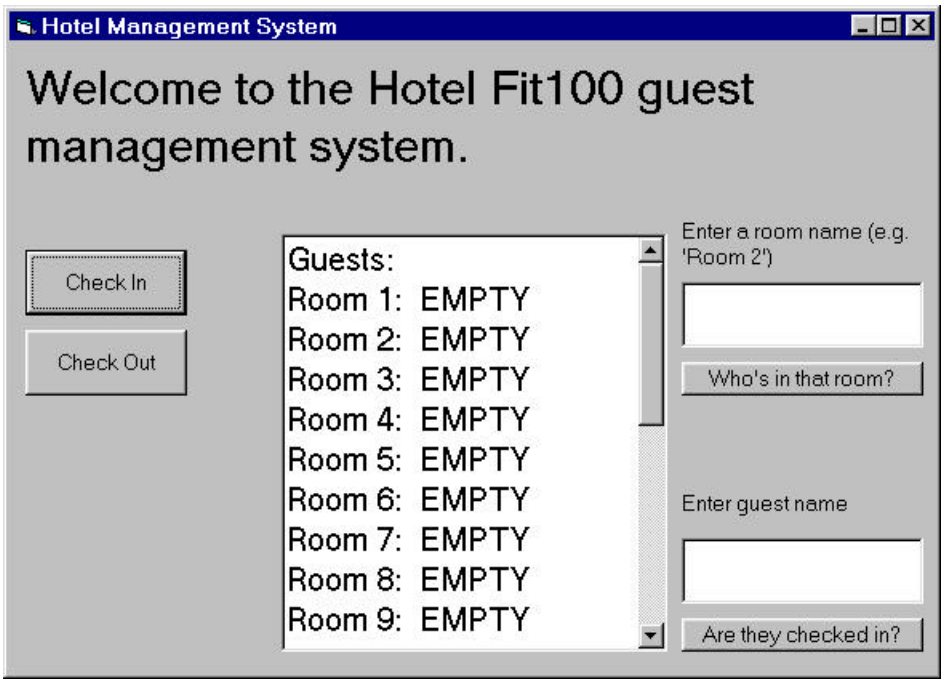
Note that the remove method only uses the key, not the value.

Finally, to look up an entry in the table (also called a "query"), you use the Item method, like this:

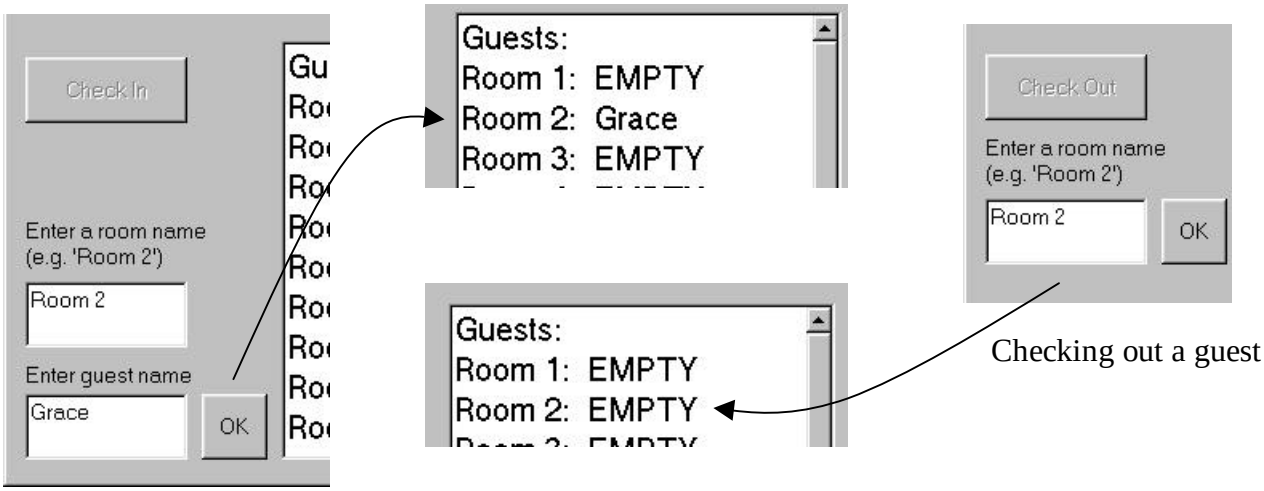
```
colGuests.Item("Room 2")
```

Like the remove method, the Item method only works with keys. It returns the value for the appropriate entry or an error if the key doesn't exist in the table.

Before we get started, here's an example of the program in action… The program is one an employee at a hotel might use to keep track of which guests are checked in, find out who's in a room and handle check-ins and check-outs.



In the middle of the screen is a list of all the rooms and who occupies each one. On the left hand side are buttons for checking in and checking out a guest. When you click on the "Check In" button two text boxes appear, one for the room name, and one for the guest name. You must enter the room name exactly (e.g. "Room 7", not "room 7" or "7".) When you click on the OK button, the display in the middle of the window is updated to show the new guest. When you click on the "Check Out" button, a single text box appears where you specify the room name being vacated. Again, when you click on the OK button, the window is updated to reflect the change.



Checking in a guest

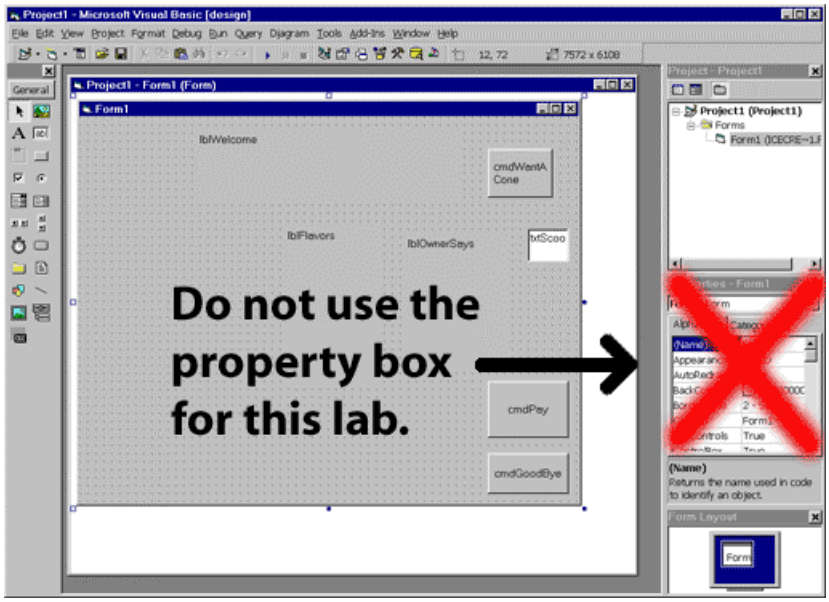
Checking out a guest

On the right hand side are two ways to search the data. On the top is a text box where the user can enter a room name (again, it must match exactly.) When the "Who's in that room?" button is pressed the program says who's in the specified room. Below that is another textbox where the user can enter a guest's name. The "Are they checked in?" button will show a message saying whether that person is checked into any room.

1. Play around with this program, until you figure out what it does.

2. Your next job is to build a VB program that works more or less like that one. To save time getting started, you can make local copy of this form, and upload it to your Dante account for safekeeping, too. The form already has all the user interface elements you'll need for this lab, including some and text boxes on the right hand side that start out invisible, but which you'll activate later on. In addition there is already a lot of visual basic code to handle things like setting up the initial contents and visibility of the various labels, text boxes, etc. Create a VB project that incorporates this form.

3. Here's an important ground rule for this lab: ***you can't make any changes to the objects by using the VB property boxes!*** That means that to change captions, colors, fonts, etc. you have to write VB program statements, in some appropriate SUB.



Most of the initial setting up of captions, etc. should be done in the form load event handling. Begin with this event handler, and write as many statements as are needed to get the form looking approximately correct.

The program will use a collection to store information about which guests are in which rooms. There are several ways to do this, but we're going to set up the collection so that the keys are the names of the room ("Room 1", "Room 2", etc.) and the values are the names of the people in the rooms. Note that there is a row for every room even if that room is empty. When that happens, the value of the row is the string "EMPTY". If we wanted to find out who was in room 1, we'd use the statement colGuests.Item("Room 1"). If we want to change who is in room 1, we must first remove the current entry for that room, with the statement colGuests.Remove "Room 1", then we can add a new entry for that room using colGuests.Add "Adam", "Room 1".

Key	Value
Room 1	Grace
Room 2	Empty
Room 3	Caro
Room 4	Empty
...	...

Make your changes a few at a time, and test carefully after each few changes. Here's a suggested order for proceeding:

4. Get the basic form working. You'll need to do a little bit of setup before the form will run.

   a. Create a collection called colGuests

      - Add the line
        `Dim colGuests as New Collection`
        at the top of your code

   b. Do any setup required in the Form_Load sub

      - Set the form caption
      - Set the lblWelcome message
      - Set the txtRoomList font size to 12
      - Set the lblSearchByRoom and lblSearchByGuest captions
      - Clear out the txtSearchByRoom and txtSearchByRoom text boxes

For Next Loops

A For Next loop allows you to iterate through a sequence of numbers. For example, the following code would print the numbers from 1 to 10 on the form:

```
Dim i as Integer

For i = 1 to 10
   Print i
Next i
```

In your code, you will need to construct a room name, like "Room 13" from the loop variable, i. To do this, you should dim another variable, roomName as a string. Then, inside the loop put the line:

```
   roomName = "Room " & CStr(i)
```

This will put into roomName the word "Room " concatenated with the string representation of the number i. (CStr converts a number to a string.)

      - Use a For Next loop to add 20 entries to the colGuests collection. Each entry has the key "Room i" (where i is a number from 1 to 20) and the value "EMPTY"

      - Test the program. It should show 20 rooms in the center textbox, all empty.

5. First let's make the check in and check out buttons actually do something. There is already code in the check in and check out buttons to do any required user interface manipulation. These buttons just make the appropriate text boxes and OK button appear. It's in the CheckIn and CheckOut OK button handlers that guests are actually checked in and out.

   a. First write the ChangeRoomOccupant procedure. This procedure has two parameters, the room name and the new occupant name. It modifies the colGuests collection by removing the old value of the entry whose key matches the room and adding a new entry with the same key but the new occupant's name as the value.

- Use colGuests.Remove to delete the old entry for room
- Use colGuests.Add to insert a new entry for room with value name

b. Modify the cmdCheckInOK_Click() event handler.

- Insert code to call your ChangeRoomOccupant procedure. The two parameters should be the values of the txtRoomName and txtGuestName text boxes.
- Add a call to ShowGuests
- Test your code. Are you able to check in new guests? What if you enter an invalid room name?

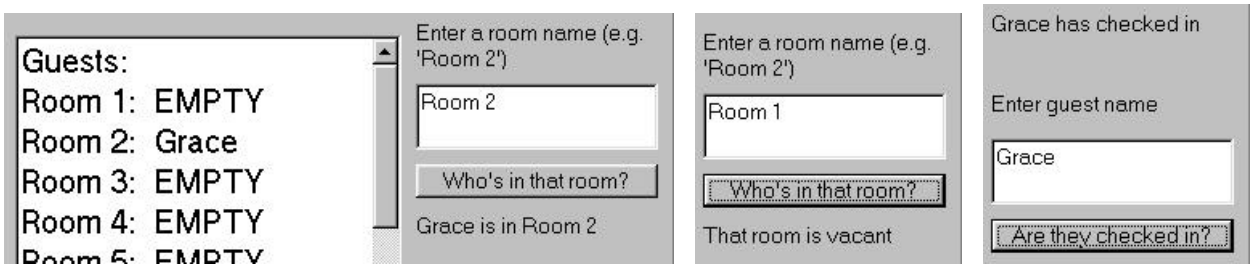
c. Add error handling code to ChangeRoomOccupant

- Add an On Error Goto statement at the beginning. Whatever word comes after the "Goto" is the *target* of the goto. So if you write "On Error Goto InvalidRoom", then "InvalidRoom" is the target.
- At the end of the procedure, write your error handling code. First write a line that has the goto target followed by a ':'. For example, InvalidRoom:
- On the line before the target, put an Exit Sub statement
- On the line after the target, put a msgbox statement displaying an "Invalid room" error message.

d. Modify the cmdCheckOutOK_Click() event handler

- Insert code to call your ChangeRoomOccupant procedure. The two parameters should be the values of the txtRoomName text box and the string "EMPTY".
- Add a call to ShowGuests
- Test your code. Are you able to check out guests?

6. Now we'll add code to make the query buttons do something. We want to allow the user to find out who is in a particular room and to see if a particular guest has checked in.



Query an occupied room… an empty room or by guest name

a. Write code to handle the "Who's in that room?" button.

- Create a variable to hold the room name and set that variable to the contents of the txtSearchByRoom textbox
- Using the Item method of the colGuests collection, look up who is checked in under the room name entered.
- Set the contents of lblResponse appropriately

- If you want to get fancy, you can use an If Then statement to determine if the occupant of the room is "Empty", if so, them message you print might look a little different than if it's a real person.

b. Test the code you wrote in 6a. Does it work properly when the user enters a valid room name, like "Room 1"? How about if they enter an invalid room name? Add an error trap to avoid the error message that appears when an invalid room is entered.

- Add an OnErrorGoto statement at the beginning of the procedure. (If you say "On Error Goto NoSuchRoom", then the target of the error handler will be "NoSuchRoom".)
- Add a goto target at the end of the procedure. (This will be the name of the target, followed by a :, like "NoSuchRoom:")
- On the line before the goto target, put an Exit Sub statement.
- On the line after the goto target, put a message in the lblResponse label saying that the room name was invalid.
- Now test your code again and see what happens when you try to search on a non-existent room.

c. Write code to handle the "Are they checked in?" button. This procedure will require a For Each loop. Read the description below to learn more about this type of loop.

For Each Loops

A For Each loop allows you to iterate through all the values in a collection. For example, if a collection colNames contained the values "Grace", "Caro" and "Adam", then the following code would print each of those names on the form:

```
Dim name

For Each name in colNames
   Print name
Next name
```

Remember that every element in a collection has two parts, a key and a value. The For Each loop only gives you the values, not the keys. In this example, we used the statement "Print name" in the loop to print each name. In your program, you'll use other code there to check if the guest's name in the collection is the same as the name the user typed into the text box.

- Create two variables, GuestSearchName and GuestName. GuestSearchName should be a string, but don't specify a type for GuestName
- Set the value of GuestSearchName from the contents of the txtSearchByGuest textbox
- Use a For Each loop to look at all the guests in the colGuests collection. For each guest, check if their name is equal to GuestSearchName. If it is, then set the contents of lblResponse appropriately

Lab Questions

1. Consider the following code:

```
dim myCollection as new collection

myCollection.Add "Jazz" "Miles Davis"
myCollection.Add "Alternative" "Beck"
myCollection.Add "Jazz" "John Coltrane"
myCollection.Add "Rock" "Jimi Hendrix"
```

What will each of the following lines of code do?

a. Print "John Coltrane is a " & myCollection.Item("John Coltrane") & " musician."

b.
```
Dim style
For Each style in myCollection
    print style
Next style
```

c. myCollection.Remove "Miles Davis"

d. myCollection.Add "Jazz", "Jimi Hendrix" * This is a trick question, give it extra thought

2. What is the difference between a for next loop and a for each loop?