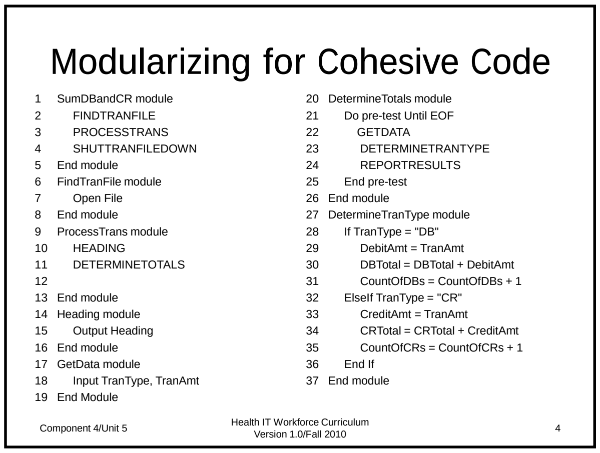
Component 4/Unit 5-5

Topic 5 Additional Programming Concepts

- Modularity and strong cohesive code
- Conditional and Unconditional Branching
- Classes
- Instantiation
- Objects
- Attributes
- Methods

Modularity

- Code within a program is often broken up into modules for many reasons
  - Manageable “chunks” of code
  - Subtasks
  - Structured design
  - Strong cohesive code with loose coupling

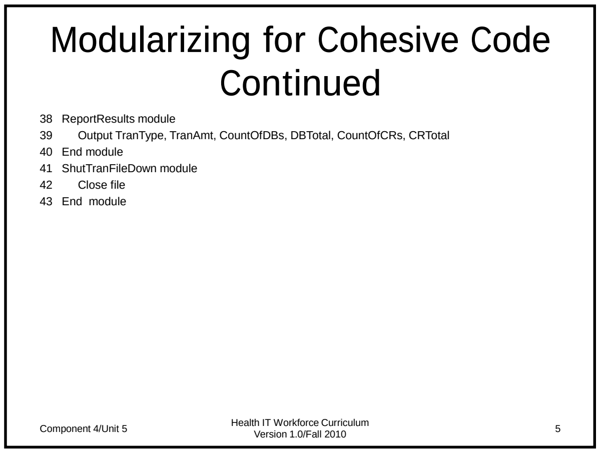


Modularizing for Cohesive Code

```
SumDBandCR module
    FINDTRANFILE
    PROCESSTRANS
    SHUTTRANFILEDOWN
End module
FindTranFile module
    Open File
End module
ProcessTrans module
    HEADING
    DETERMINETOTALS

End module
Heading module
    Output Heading
End module
GetData module
    Input TranType, TranAmt
End Module
DetermineTotals module
    Do pre-test Until EOF
        GETDATA
        DETERMINETRANTYPE
        REPORTRESULTS
    End pre-test
End module
DetermineTranType module
    If TranType = "DB"
        DebitAmt = TranAmt
        DBTotal = DBTotal + DebitAmt
        CountOfDBs = CountOfDBs + 1
    ElseIf TranType = "CR"
        CreditAmt = TranAmt
        CRTotal = CRTotal + CreditAmt
        CountOfCRs = CountOfCRs + 1
    End If
End module
```

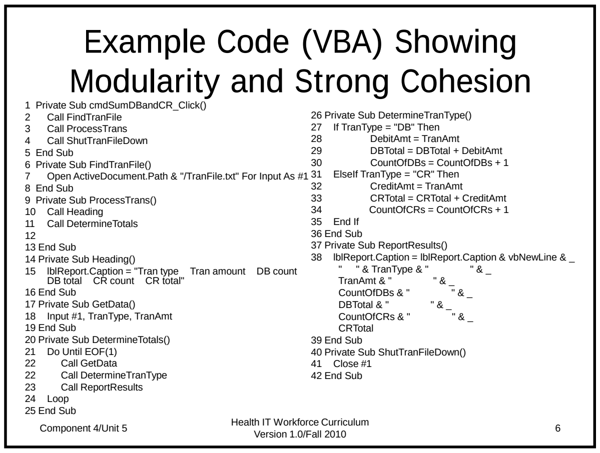
Component 4/Unit 5 Health IT Workforce Curriculum Version 1.0/Fall 2010 4



Modularizing for Cohesive Code Continued

```
ReportResults module
    Output TranType, TranAmt, CountOfDBs, DBTotal, CountOfCRs, CRTotal
End module
ShutTranFileDown module
    Close file
End  module
```

Component 4/Unit 5 Health IT Workforce Curriculum Version 1.0/Fall 2010 5



Example Code (VBA) Showing Modularity and Strong Cohesion

```
Private Sub cmdSumDBandCR_Click()
    Call FindTranFile
    Call ProcessTrans
    Call ShutTranFileDown
End Sub
Private Sub FindTranFile()
    Open ActiveDocument.Path & "/TranFile.txt" For Input As #1
End Sub
Private Sub ProcessTrans()
    Call Heading
    Call DetermineTotals

End Sub
Private Sub Heading()
    lblReport.Caption = "Tran type    Tran amount    DB count
    DB total    CR count    CR total"
End Sub
Private Sub GetData()
    Input #1, TranType, TranAmt
End Sub
Private Sub DetermineTotals()
    Do Until EOF(1)
        Call GetData
        Call DetermineTranType
        Call ReportResults
    Loop
End Sub
Private Sub DetermineTranType()
    If TranType = "DB" Then
            DebitAmt = TranAmt
            DBTotal = DBTotal + DebitAmt
            CountOfDBs = CountOfDBs + 1
    ElseIf TranType = "CR" Then
            CreditAmt = TranAmt
            CRTotal = CRTotal + CreditAmt
            CountOfCRs = CountOfCRs + 1
    End If
End Sub
Private Sub ReportResults()
    lblReport.Caption = lblReport.Caption & vbNewLine & _
      "     " & TranType & "            " & _
      TranAmt & "            " & _
      CountOfDBs & "            " & _
      DBTotal & "            " & _
      CountOfCRs & "            " & _
      CRTotal
End Sub
Private Sub ShutTranFileDown()
    Close #1
End Sub
```

Component 4/Unit 5 Health IT Workforce Curriculum Version 1.0/Fall 2010 6

Structured Design

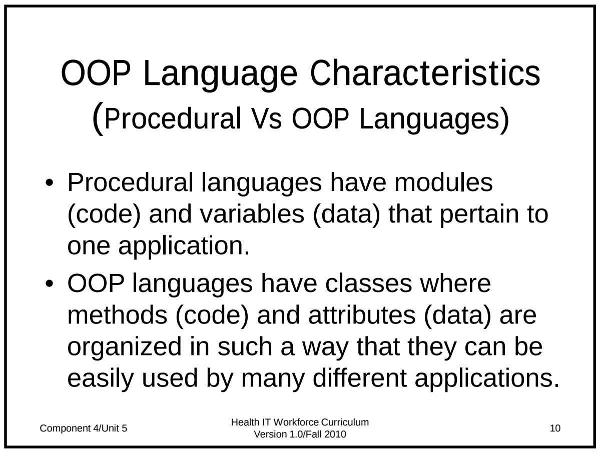
- Spaghetti code
  - Arrows used in early flowchart diagrams where unconditional branching was used looked like spaghetti.
  - The term now is synonymous with an unstructured solution.
- Structured solution
  - Disallowing unconditional branching
  - The design of the program is based on the structure of the data being processed and the output information being generated.
  - There is one entry point into the program and in some definitions of structured programming, one exit point.
- Top Down Design
  - Starting at the highest level of a process and identifying sublevel processes as black boxes.

Conditional and Unconditional Branching

- Conditional Branching (the requirement to come back)
  - Considered a structured programming tool
- Unconditional Branching (no strings attached)
  - GoTo logic – violates good code structure
  - Can create spaghetti code

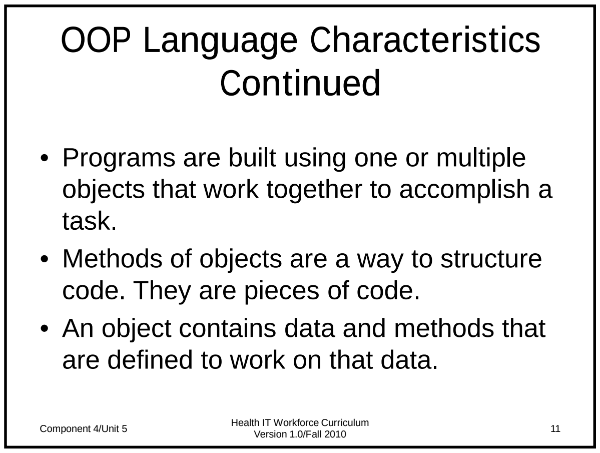
Object Oriented Programming

- OOP stands for Object Oriented Program.
- C++, Java, VB.NET and C# are a few programming languages that are considered object-oriented.
- OOP languages all have certain characteristics that qualify them for being OOP languages.



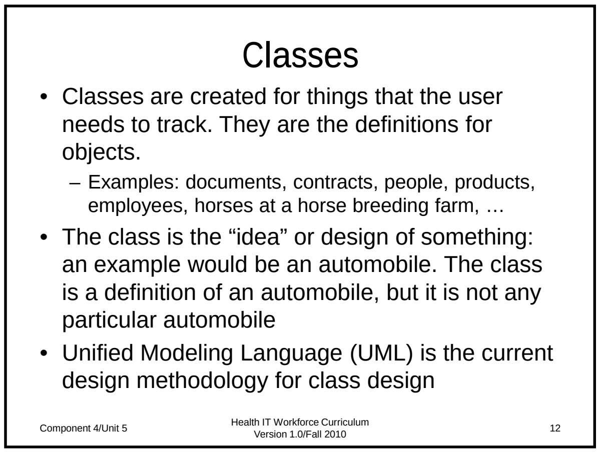
OOP Language Characteristics (Procedural Vs OOP Languages)

- Procedural languages have modules (code) and variables (data) that pertain to one application.
- OOP languages have classes where methods (code) and attributes (data) are organized in such a way that they can be easily used by many different applications.



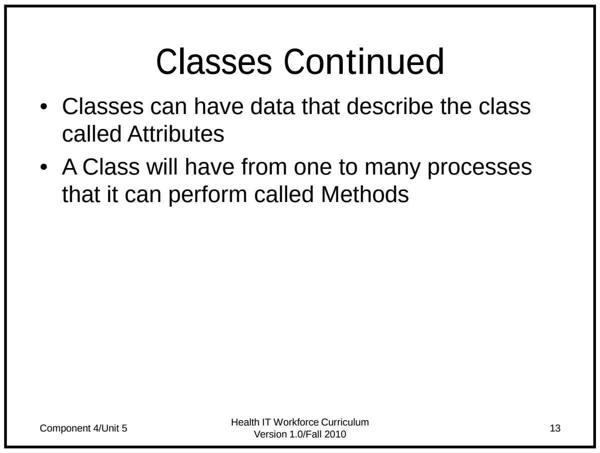
OOP Language Characteristics Continued

- Programs are built using one or multiple objects that work together to accomplish a task.
- Methods of objects are a way to structure code. They are pieces of code.
- An object contains data and methods that are defined to work on that data.



Classes

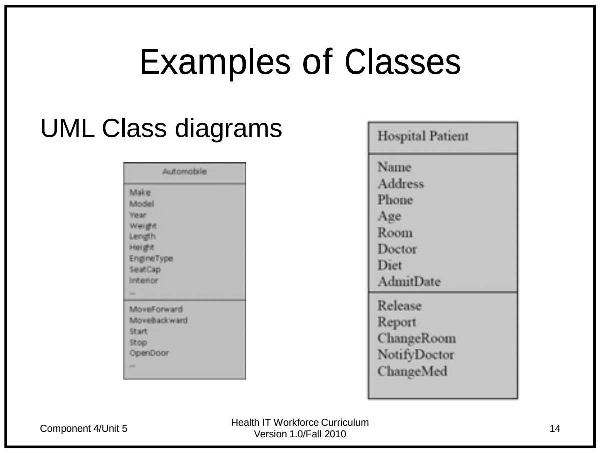
- Classes are created for things that the user needs to track. They are the definitions for objects.
  - Examples: documents, contracts, people, products, employees, horses at a horse breeding farm, …
- The class is the “idea” or design of something: an example would be an automobile. The class is a definition of an automobile, but it is not any particular automobile
- Unified Modeling Language (UML) is the current design methodology for class design



Classes Continued

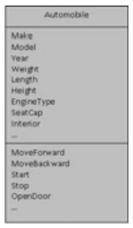
- Classes can have data that describe the class called Attributes
- A Class will have from one to many processes that it can perform called Methods

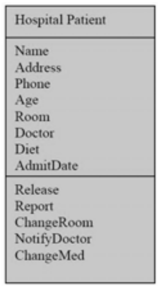
Component 4/Unit 5 Health IT Workforce Curriculum Version 1.0/Fall 2010 13



Examples of Classes

UML Class diagrams





Component 4/Unit 5 Health IT Workforce Curriculum Version 1.0/Fall 2010 14