

VITAL INFORMATION

Subject(s):	Careers, Computer Fundamentals 1-2
Topic or Unit of Study:	Integrated Unit
Grade/Level:	9-12
Objective:	<p>At the conclusion of this lesson students will be able to:</p> <ol style="list-style-type: none">1. Define a Module and explain cohesion.2. Create a Module.3. Add code to a Module.4. Properly call code that is located in a different module.
Summary:	<p>Students add a Module to their Computer Integration project and populate it with a single subroutine which was "for some reason" absent from the Module already present in their project. To write the correct subroutine for Friday, they have to first understand how Thursday works, and it has to be added to a new Module because old a new code are separated by company policy.</p>

IMPLEMENTATION

Learning Context:	<p>Students have just added a CheckBox to the program for their Computer Integration activity. The user interface for the ScheduleTabPage is completed (except for some optimizations) with this activity. Students are thus nearly ready to add the actual content which describes Computer Integration into their other coursework.</p>
Procedure:	<ol style="list-style-type: none">1. The activity is written up in fairly fine detail on a web page which is printed and attached to this lesson plan. It specifies how to add a new item to a project, how to name the new Module, where to find the incomplete code, where to place the new code, and how to invoke functions that reside in different modules. It does not review the big picture, so the stage should be reset using some of the learning context from the introduction to the integrated unit.2. Ensure that students can find the web page, ask for questions, and have students start.3. Since this is a long, sequential unit, more attention needs to be paid to holding it all together and keeping it synchronized. Visit students early and often to ensure that they are keeping up. It may be a good

idea to have students demonstrate their progress beginning five minutes before the bell rings.

Differentiated Instruction:	There is little differentiation in instruction and for Modules there is little that can differ in the deliverable.
Sample Student Products:	The outward appearance of the program should not change in this lesson, although the project structure will.
Collaboration:	Students will work individually.
Time Allotment:	1 class period. 55 Min. per class.
Author's Comments & Reflections:	Reflections will follow in a diary entry.

MATERIALS AND RESOURCES

Instructional Materials: The activity page from the class web site is printed and attached as is the Visual Basic file which is edited in this activity. Students do also work with the Designer, but that work is summarized in screen shots.

Attachments


1. [modSchedule](#)
2. [Module](#)

Resources:

- Technology resources:
Visual Basic

STANDARDS & ASSESSMENT

Standards:

-  **AZ- Career and Technical Education Programs**
 - **Level :** Career Preparation (Grades 10 - 12)
 - **Program :** Information Technology CIP No. 15.1200
 - **Option :** Software Development - Option C
 - **Competency :** *3.0 DEVELOP APPROPRIATE WORK HABITS FOR SUCCESSFUL EMPLOYMENT IN INFORMATION TECHNOLOGY
 - **Indicator :** 3.3 Complete tasks accurately
 - **Indicator :** 3.4 Complete tasks with minimal supervision
 - **Competency :** 27.C DEMONSTRATE PROGRAM ANALYSIS AND DESIGN
 - **Indicator :** 27.6c Use stepwise refinement to improve design
 - **Competency :** 28.C USE SOFTWARE TO CREATE PROGRAMS
 - **Indicator :** 28.1c Enter and modify code using a program editor
 - **Competency :** 29.C TEST AND DEBUG TO VERIFY PROGRAM OPERATION
 - **Indicator :** 29.1c Test individual program modules
 - **Competency :** 28.C USE SOFTWARE TO CREATE PROGRAMS
 - **Indicator :** 28.2c Compile and execute programs
 - **Indicator :** 28.5c Use recognized conventions for naming identifiers and formatting code
 - **Competency :** 38.C EMPLOY OBJECT-ORIENTED PROGRAMMING TECHNIQUES
 - **Indicator :** 38.4c Write appropriate statements to invoke an object's accessor method(s)

- **Competency** : 32.C WRITE CODE USING CONDITIONAL STRUCTURES
 - **Indicator** : 32.4c Construct decision statements such as if/else, if, switch case
- **Competency** : 31.C EMPLOY MODULARITY IN WRITING PROGRAMS
 - **Indicator** : 31.2c Utilize parameters to pass data into program modules
- **Competency** : 32.C WRITE CODE USING CONDITIONAL STRUCTURES
 - **Indicator** : 32.2c Evaluate Boolean expressions
- **Competency** : 31.C EMPLOY MODULARITY IN WRITING PROGRAMS
 - **Indicator** : 31.4c Write and use modules that return values
 - **Indicator** : 31.5c Write cohesive units with minimal coupling
- **Competency** : 34.C USE SIMPLE DATA TYPES AND STRINGS
 - **Indicator** : 34.4c Write assignment statements for initializing and modifying variables

Assessment/Rubrics: By the end of the class period, students should have added and named the Module modNewSchedule.vb, written a subroutine modNewSchedule.fillFriday which invokes functions still in modSchedule, and rerouted the call in frmMain from the old module to the new one. Each activity contributes a few criteria to the larger rubric. This activity adds points for project structure, subroutine code, and rewiring as described by the Module rubric.

Rubrics

1. Module
