Lesson - Week 4:

Course Objectives Alignment:

<<* The Course Objectives (CO##) are listed in the Course Outline *>>

- CO01 - The authors show you many UML diagrams for Classes and objects. You are encouraged to use flowcharts and pseudocode in creating your first project.
- CO02 - You are encouraged to review/edit sample programming code (within the chapters) in a file creation and editing simulation environment called TutorialsPoint and/or the NetBeans IDE. You will use NetBeans for the Programming Assignment.
- CO03 - You will continue to work with NetBeans to create the Programming Assignment.
- CO04 - Chapter 10 advances the topic of objects as we look at Inheritance.
- CO05 - We are starting new material for the semester (no longer review). We learn more about Object-Oriented programming.
- CO06 - We will start to analyze a programming project this week and start to code it into next week.
- CO07 - Key terms and the page numbers are found in the.

Read in Your Textbooks:

- Chapter 10

It is imperative that you have a good handle on what an object is and how to use them in a program. Chapter 9 was the introduction to User-Defined Classes and Objects. Chapter 10 just gets deeper into Abstraction and Encapsulation and we learn about Polymorphism – program behavior is depending on data types (overloading methods is an example). You will learn how to think in terms of objects. You will learn about the Class relationships of Association, Aggregation, and Composition and how to represent them in a UML diagram. We learn about the base class of Object.

Things to Look Out For:

- Object-oriented design involves understanding terminology. There are several goals of using object-oriented programming over standard programming using primitive variables. The first one… modeling data storage as attributes having values to describe objects as we view them in the real world which we learned in Chapter 9 (or in your CST112 course). In our course textbook in Chapter 10, we then we go further and use object-oriented design to reduce the development time of programs by abstracting object design to use inheritance to build base designs which include common attributes and then model the unique attributes to similar objects as child objects. We then delve further into data security to make

it harder for a malicious programmer to merge their program with your program that may be running at the time in memory.
- Section 10.2 mentions package-protected … think of it as classes within the same folder (package folder) trust each other.
- We see why NetBeans adds @overrides to our programs.
- Understand how to access methods and instance variables of the base class. The discussion of accessing base methods within a child class starts in the section of 10.2 labeled "Calling a base class method".

Discussion Questions:

Muddiest Points Forum - Post to this forum as often as you need to when you run into trouble understanding a concept, something in the book, or just have a question. You may assist your classmates by answering. Emailing me is the fastest way to get a response from your instructor but your classmates may be able to offer you help when they are in the course. If your questions are of a personal nature, please email me instead of using this forum.

zyBooks Participation Assignment:

See the Assignments tab in your zyBooks textbook for the assignment "Participation Assignment for Week 04 – Chapter 10". You are reading the textbook but also doing the Participation activities within the chapter (the Challenge activities are not counted as part of the assignment). Note that doing the participation activities while you are reading the textbook is an important part of the learning process in this course. I have made the assignment due by 10/15. If you prefer to delay the work within the chapter, keep in mind two other chapters of work will be assigned between Week 04 and Week 07.

Project for Week 04 (Due in two weeks on 10/1):

We are going to use the solution to the Employees program - example zyDE 10.13.2 to create a College Courses program. There are a lot of instructions for this problem but most of them are iterating single tasks such as copy and paste. Read through the sample problem after reading the chapter thoroughly. Ask questions if you have trouble understanding what is going on. Then read through the entire problem below and then follow the individual steps. Again, I have broken down the big problem into finite tasks.

We will change the names of the classes and the instance variables of the Employees program and adapt it to a program that uses similar people objects but the data attributes we need to track are different. I provided renamed class source code files from the textbook under Week 04's Content folder in BrightSpace but left the class definitions and code as is from the Employees program. Your job will be to modify the programs according to the specifications of the project which will be detailed below.

The source files linked in the Week 04's content folder have the matching classes in the 10.13.2 problem:

CollegeCoursesMain -> EmployeeMain
CoursePerson -> EmployeePerson
CourseStudent -> EmployeeStaff
CourseTeacher -> EmployeeManager

I just changed the file names of the Authors' files but kept the contents intact. We are going to first build a new project in NetBeans according to the steps below and then we will create additional class files and do some copying and pasting before we make additional modifications to the original code.

1. Create a project in NetBeans called ProjectForWeek04 with the same package name (but lower case) and specify that the main class file is named CollegeCoursesMain. We will modify this Class below.
2. Create a new Java file in the same package called CoursePerson.
3. Open the file in Week 04's Content folder in BrightSpace called CoursePerson. Copy all the code between **but not including** the { and } of the class block (the class statement references EmployeePerson). Paste the copied block inside the { and } in the class file that is in the NetBeans editor. Again, you will get lots of errors which will be fixed later. Choose the Save file option in NetBeans before moving on. We will come back to this class file later.
4. Repeat steps 3 and 4 for CourseStudent and CourseTeacher so that your project has 4 class files in it.

Before we code the new Instance Variable names and update the methods of our classes to fit a College Course system, we need to make CoursePerson abstract and then CourseStudent and CourseTeacher as being extended classes of CoursePerson (the same way the Authors coded the Employees problem.) As part of the NetBeans project creation wizard and the new class file creation wizard, standard class statements are coded for you as the basis for you to make changes such as making a class abstract or extending a class from another class. **Please try to keep the concepts of inheritance and abstraction separate from the implementation of those concepts in Java coding and creating projects in NetBeans (or other IDE such as Eclipse).**

5. Open the CoursePerson class if not already open in the NetBeans editor and add "abstract" after "public" and before "class". Save the changes.
6. Open the CourseStudent class if it is not already open in the NetBeans editor and add " extends CoursePerson" after the "public class CourseStudent" code but before the {. Save the changes.

7. Open the CourseTeacher class if it is not already open in the NetBeans editor and add " extends CoursePerson" after the "public class CourseTeacher" code but before the {. Save the changes.

Program modifications for Course Program:

The Employee program allows us to set a manager to one department and then an employee to one department. We will do something different for teacher and students by adding an additional Class called Course. Our object variable names in the main() method of CollegeCoursesMain Class will assign a teacher and students to a particular course.

Make the following changes and additions to the existing Classes and create on additional Class:

CoursePerson Class will have the Instance Variables - fullName, phoneNumber, and userName all as protected Strings.

- Add the missing Instance Variables.
- Remove the Instance Variables not needed for our problem (that came from the Employee Program).
- Update the 2 Constructors for the new Instance Variables and the new name of our Class.
- Remove the the setData() method.
- Create get methods for the new Instance Variables.
- The printInfo() abstract method stays the same.
- Change getAnnualBonus() abstract method to getEmailAddress() abstract method and the type should be changed from int to String.

CourseStudent Class will have the Instance Variables - matriculated as type Boolean, fullTime as Boolean, and programOfStudy as type String.

- Add the new Instance Variables.
- Remove the instance variable managerName.
- Update the two Constructors for the new Instance Variable names and the new name of our Class.

  Your Constructor that has no parameter variables should be coded with:

  super();

  Then have assignment statements for empty or false values as default for Instance Variables that belong to CourseStudent.

Your Constructor that has the parameter variables should be coded with the parameter list:

CourseStudent(String theFullName, String thePhoneNumber, String theUserName, boolean theMatriculated, boolean theFullTime, String theProgramOfStudy) {

- We will call the Constructor for CoursePerson to update the Instance Variables with the values sent to the CoursePerson Constructor. To do this, add to the top of the Constructor with parameter variables:

  super(theFullName, thePhoneNumber, theUserName);

  Now add the Instance Variable assignments for the Instance Variables defined just in the CourseStudent Class.

- Create get methods for the new Instance Variables and remove the getManagerName() method.
- Change the printInfo() method to include the new Instance Variable names and corresponding labels of the CoursePerson Class using super.get methods and the Instance Variables we just changed in bullet one above for CourseStudent, and then the email address.
- Change the name of getAnnualBonus() to getEmailAddress and change the type from int to String. Return username concatenated with "@mail.sunysuffolk.edu".

CourseTeacher Class will have the Instance Variable fullTime as type Boolean.

- Add the new Instance Variable.
- Remove the numManaged Instance Variable.
- Update the two Constructors for the new Instance Variable names and the new name of our Class.

  Your Constructor that has no parameter variables should be coded with:

  super();

  Then have an assignment statement to set your one Instance Variable that belongs to CourseTeacher to false as a default value.

  Your Constructor that has the parameter variables should be coded with the parameter list:

  CourseTeacher(String theFullName, String thePhoneNumber, String theUserName, boolean theFullTime) {

- We will call the Constructor for CoursePerson to update the Instance Variables with the values sent to the CourseTeacher Constructor. To do this, add to the top of the Constructor with parameter variables:

  super(theFullName, thePhoneNumber, theUserName);

  Now add the Instance Variable assignments for the Instance Variables defined just in the CourseTeacher Class.

- Create a get method for the new Instance Variable and remove the getNumManaged() method.
- Change the printInfo() method to include the new Instance Variable names and corresponding labels of the CoursePerson Class with super.get methods and the Instance Variable we just changed in bullet one above for CourseTeacher, and then the email address.
- Change the name of getAnnualBonus() to getEmailAddress and change the type from int to String. Return userName concatenated with "@sunysuffolk.edu".

MAKE A NEW CLASS:

Course will have Instance Variables courseNumber as type String, CRN as type String, and numberOfCredits as int. This is the 5th class file in your project now.

- Create the new Java Class file in NetBeans called Course in the same package folder.
- Add the Instance Variables above.
- Create two Constructors like CourseStudent and CourseTeacher but for Course.
- Create get methods for the Instance Variables.
- Create a printInfo() method that simply outputs the Instance Variables from the Course class with appropriate labels.

CollegeCoursesMain:

- Erase all the current code between the { and } of the main() method.
- Create a new Course object variable called courseCST141 with the data "CST141", "95537", 4.
- Create a new CourseTeacher object variable called courseCST141Teacher with the data "Prof. H.", "631-548-2691", "hassile", true.
- Create an Array of type CourseStudent with the size of 5 called courseCST141Student.
- Define 5 array elements with pretend student info in it.
  - Index 0: "Sally", "631-555-1111", "sally01", true, true, "CST"
  - Index 1: "Mohammed", "631-555-2222", "moham56", true, true, "CST"

    - Index 2: "Alex", "631-555-3333", "alex93", false, false, "N/A"
    - Index 3: "Barbara", "631-555-4444", "barba28", true, false, "GenStud"
    - Index 4: "Monique", "631-555-5555", "moniq71", true, true, "CST"
- Output a heading for "Course Info:".
- Invoke courseCST141.printInfo().
- Output a heading skipping a line for "Teacher Info:".
- Invoke courseCST141Teacher.printInfo().
- Output a heading skipping a line for "Students in course:".
- Using a loop, output "Student #" and then the loop index, then invoke courseCST141Student[loop variable name].printInfo().

You will use a Project Assignment Template file for submitting your work. You will name your Project Assignment Document as "yourlastname-CST141FA23-ProjectForW04" as either a .doc, .docx, or .pdf. I do not want .rtf because they are quite large. Ex. Smith-CST141FA23-ProjectForW04. Submit the file under the Project for Week 04 Assignment folder in BrightSpace. The assignment is due by 11:59PM on 10/1.