W. P. CAREY
SCHOOL of BUSINESS
ARIZONA STATE UNIVERSITY

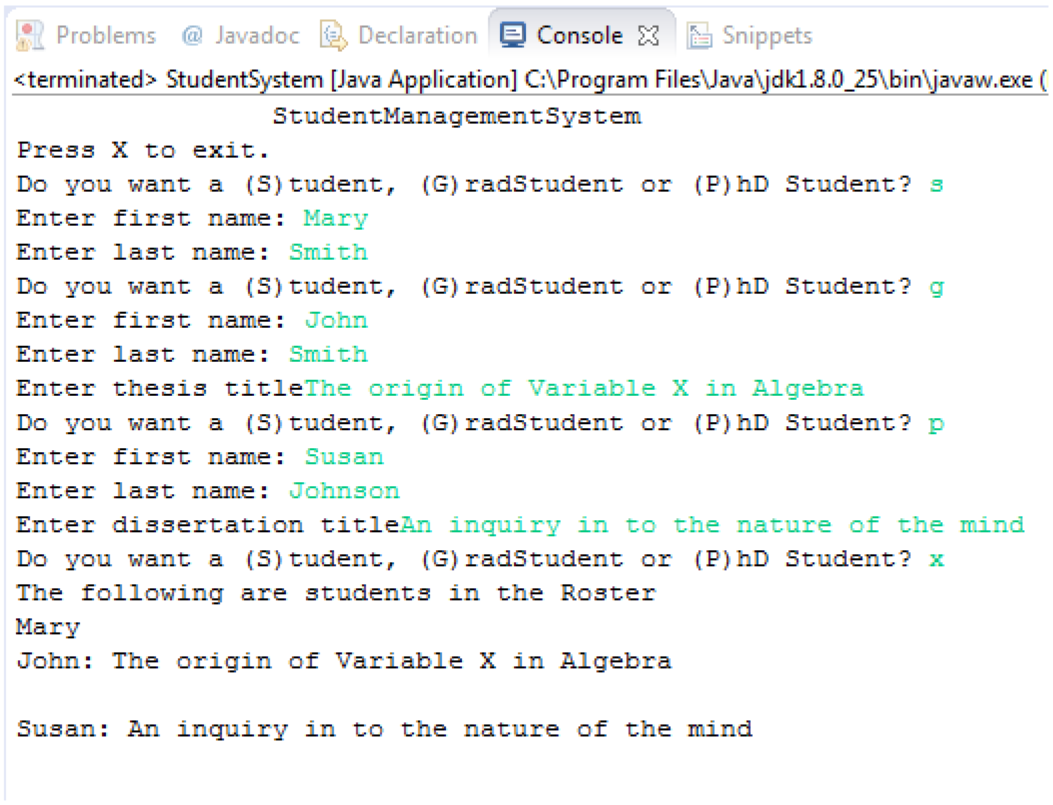
CIS 494 – Business Systems Development with Java – Spring 2015

In-class Exercise 3: Polymorphism

Due on Blackboard: Tonight, February 2 by 11:59 PM

This purpose of this assignment is to give you more practice in implementing inheritance and polymorphism.

Sample Output



```
<terminated> StudentSystem [Java Application] C:\Program Files\Java\jdk1.8.0_25\bin\javaw.exe (
                StudentManagementSystem
Press X to exit.
Do you want a (S)tudent, (G)radStudent or (P)hD Student? s
Enter first name: Mary
Enter last name: Smith
Do you want a (S)tudent, (G)radStudent or (P)hD Student? g
Enter first name: John
Enter last name: Smith
Enter thesis titleThe origin of Variable X in Algebra
Do you want a (S)tudent, (G)radStudent or (P)hD Student? p
Enter first name: Susan
Enter last name: Johnson
Enter dissertation titleAn inquiry in to the nature of the mind
Do you want a (S)tudent, (G)radStudent or (P)hD Student? x
The following are students in the Roster
Mary
John: The origin of Variable X in Algebra

Susan: An inquiry in to the nature of the mind
```

Use the provided template project. Import the project into your Workspace. ***Create variables of appropriate data types as you need to!***

1) Add Class GraduateStudent

Create a Class GraduateStudent that inherits from the Student class. Graduate students have a Master's thesis. Create an attribute and get/set methods for the thesis. You may write those methods yourself or use the Eclipse IDE to generate those methods for you.

2) Add Class PhDStudent

Create a Class PhDStudent that inherits from the Student class. Graduate students have a Dissertation. Create an attribute and get/set methods for the dissertation. You may write those methods yourself or use the Eclipse IDE to generate those methods for you.

3)

3.1) Within the StudentSystem class, the method processStudents() asks the user if they want to create a generic student, a graduate Student, or a PhD student. After the line of code that stores the first and last name, add a switch statement to process userInput.

Within the *switch* statement compare user input and based on its value, declare **and** create either an instance of a Student, GraduateStudent or PhDStudent by calling their respective ***constructors***. You may want to have "Student" as the default. *You should have 3 cases in your switch, only one of which will be executed based on user selection.*

```
Case  "G":
        Ask for and store thesis title
        Create an instance of GradStudent and store it in an appropriate variable.
        Call set methods to set first name, last name, thesis title

Case "P":
        Ask for and store dissertation title
        Create an instance of PhDStudent and store it in an appropriate variable.
        Call set methods to set first name, last name, dissertation title

Default:
        Create an instance of Student and store it in appropriate variable.
```

3.2) Go to the 3 *cases* you just created within the Switch statement. Within each *case*, call the addStudent() method of the StudentRoster class.

Note the type of argument the method expects: Student.

Both GraduateStudent and PhDStudent are specialized cases of Student. If someone wants a "Student", you may give them a graduate student because all graduate students are students! So, pass the variable of type Student, GraduateStudent, or PhDStudent as appropriate. *This is polymorphism in action.*

The following is what you are ***adding*** to the *existing* cases in the switch:

Case "G":

    Call the AddStudent method of theRoster and pass to it your GradStudent.

Case "P":

    Call the AddStudent method of theRoster and pass to it your PhDStudent.

Default:

    Call the AddStudent method of theRoster and pass to it your Student.

4) Go to the printStudentList() method with the StudentRoster class.

This method loops through all the Students in the array and prints out their first and last name.

However, ***some*** Students are specialized. They have a thesis or a dissertation. Those two attributes need to be printed for those specialized students. However, all Student do not have a thesis or a Dissertation. That is why we cannot access that information for all Students through the Student class. Implement the following solution:

4.1) ***TRY*** accessing the thesis information through the array of Students. Note that you are not able to do so. This is because the Student class does not provide getThesis().

4.2) After the name information has been printed, write an *if* statement and test if the current element of the array i.e. roster[i] is a GradStudent by using the keyword "instanceof"

```
If (roster[i] instanceof GradStudent)
    Declare a tempGradStudent variable; assign to it the current array element (use Casting).
    Print the tempGradStudent’s Thesis title to the console
Else if (roster[i] instanceof PhDStudent)
    Declare a tempGradStudent variable; assign to it the current array element (use Casting).
    Print the tempGradStudent’s Thesis title to the console
```

Additional Polymorphism Learning Task

You were asked to write a call to the AddStudent() method three times within the switch statement. That is somewhat inefficient. Make your code much more efficient by utilizing polymorphism:

- At the top of the processStudents method, declare a tmpStudent variable, which is of type Student.
- In each of the cases, ***comment out*** the lines that call the set methods for first name and last name. ***Comment out*** the line that calls the AddStudent method.
- In each of the cases, take the variable that is of either type Student, GradStudent, or PhDStudent and assign it to your tmpStudent variable.
- After the switch statement block ends:
    - Add a line that calls the set methods for first name and last name for tmpStudent.
    - Call the AddStudent method and pass it your tmpStudent.

This methodology of processing students treats ***all*** students as *just* students. Since ***all*** students have a first name and a last name, there is no need to do the processing of first name and last name while working with the subtypes of Student. That creates additional lines of code. If we need to work with a series of instances of Students and need to work only with those variables and methods that are found in the Student class, then it is most efficient to do that while they are cast as Students.

<u>Submitting Files</u>

Submission should be made using a zip file that contains the entire Eclipse project folder. You will need to ***zip the entire project folder***. The folder will automatically contain the class source files as well as the compiled .class files.

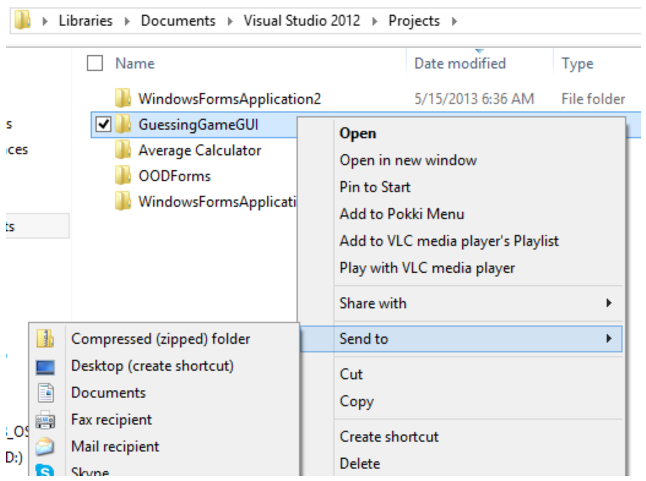
ZIP file should be named: **AX.zip or IC.zip**

> where X is the in-class assignment number, e.g., A1.zip is the submission file for Assignment 1 and IC1.zip is the submission file for InClass1.

Note: *The ZIP filename is <u>independent</u> of the Project name. Do not name your project A1.*

How to Properly ZIP and Submit Your Project Files:

- Go to the folder within {Eclipse Workspace}\
- ZIP the entire top-level folder for your project by right-clicking your project folder and selecting Send to | Compressed (zipped) folder.
- Finally, submit the ZIP file using the submission link on Blackboard by the due-date and time listed on the assignment. Upload the ZIP file.



Using built-in windows zip tools: http://windows.microsoft.com/en-us/windows/compress-uncompress-files-zip-files

Verify your files BEFORE and AFTER submission:

- Check for actual class files being present in the folder before you zip it.
- ***Ensure that you are not zipping a short-cut to the folder.***
- After zipping, check file size. A file size under 4K likely does not contain all the files.
- Unzip, extract all files, and verify you see actual files, not a solitary short-cut.
- Uncompress your zip file before submitting and verify that files are present.
- Make sure you have ***submitted*** your file and not just saved a draft on Blackboard. ***A blue clock indicates a submission in progress, i.e. a draft, not a submission. The draft is accessible only to you. You will get a ZERO if you only ever save a draft on Blackboard and never submit your files.***

- Download your zip file after submitting, uncompress, and again verify that your files are present. Test your files in Visual Studio after uncompressing.

This takes an extra couple of minutes. Please do it if your grade is important to you. If you do this, you will not end up submitting a bad file. If you submit an empty file, or one containing only a shortcut, or a bad zip file, or a bad project file, you will receive a score of zero and your only recourse will be to do the makeup assignment at the end of the semester.