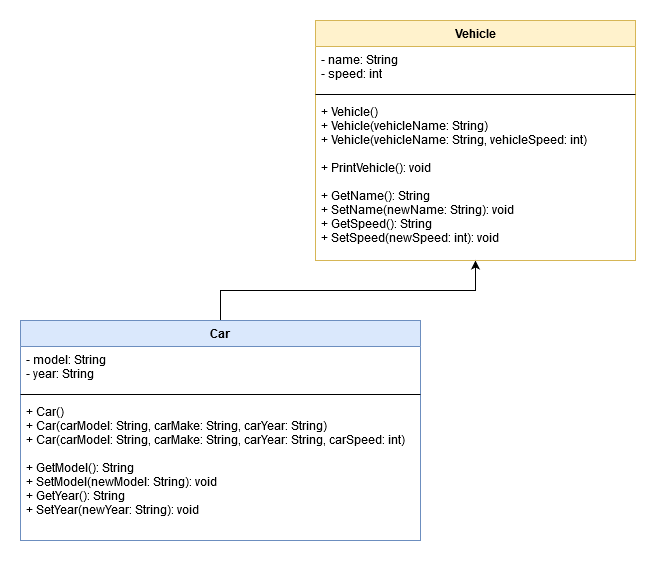
* Given what you know from what we have studied this far, give an example of an inheritance relationship that we have not already discussed in class. Draw a UML *class* diagram below to illustrate your example. Give each class at least two attributes and two methods. Complete the UML class diagram with attributes and methods. (15pts.)
* Code this relationship in Netbeans. Use the **csu.csci325** package. Call the project **TestPrep**. Be sure to code the constructors and the methods. (10pts)
* Add the names of all of the people in your group to the comments at the top of all .java files as co-authors. (5pts)
* For the parent class, add an additional constructor that takes different parameters. (For reference, see the ConstructorFun code where we created a Student class. See where we coded one Student constructor that expected both Name & ID and one that expected only a name.) (10pts)
* Could we also do this type of thing with a regular method? If so, add code to do just that. If not, explain below why this is not possible. (5pts)

Yes.

public static int Add(int a, int b)

{

return a + b;

}

public static int Add(int a, int b, int c)

{

return a + b + c;

}

* When coders do what you did in steps #4-#5, what is it called?(5pts) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Function overloading
* Take one of the methods that you coded in your parent class and code a method in the child class with the same name and the same parameters (in Netbeans.) Have the logic in the child’s version of the method behave differently than the method of the same name in the parent class. Write the name of the method in the space below: (10pts)

PrintVehicle()

* What is this called when we do what you did in step #7? (5pts) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Function overriding
* What is encapsulation? Did you use encapsulation in this exercise? If so, how?(5pts)

Encapsulation is hiding certain data from external use while having unhidden functions that work with that hidden data.

* What test-cases should be run to test the code you have written? Identify at least 8 test cases. Write a short description of each test case below: (10pts)

\_(a) Create a Vehicle using the constructor that takes in a name and speed

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_(b) Create a Car using the constructor that takes in the car make, model, year, and speed

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_(c) Call PrintVehicle() on the Vehicle that we made and see if it only outputs the vehicle's name and speed.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_(d) Call PrintVehicle() on the Car that we made and see if it outputs the car's model and year on top of its name and speed that we got from the super (Vehicle)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_(e) Call PrintVehicle() on our Vehicle and then call SetSpeed() and set the Vehicle's speed to something. Then call PrintVehicle() again and see if the Vehicle's speed has changed

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_(f) Call PrintVehicle() on our Car and then call SetModel() and set the Vehicle's model to something. Then call PrintVehicle() again and see if the Car's model name has changed

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_(g) Make another Car using the default constructor (the one with no parameters) and ensure that all the member's are initialized to 0 if a numeric value or "None" if a String

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_(h) Make another Vehicle using the default constructor (the one with no parameters) and ensure that name is initialized to "None" and speed is initialized to 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* What is polymorphism? (5pts)

Polymorphism is when the program can determine its behavior based on the types that it is working with. Example: function overloading. Two functions with the same name but the program will know which one to use because of the parameters being given. Example: function overriding. One class overriding a function from its parent class and the program will run the overrided version of the function because the object's type is the child class not the parent.

* Pick one of the test cases identified in step #10. Create a driver (main) class and code the test case using polymorphism. Place comments in your code to show where you are using polymorphism. (E.G.,

//The line(s) of code below show(s) the use of polymorphism since the variable blah, blah is blah, blah. ) (5pts)

* Code the rest of the test cases in your driver class and execute them. (10pts)

What to submit:

* The team must create a video demonstration of a walk-through of the working code. In the video, *show and explain where you implemented #1-13 above in your code*.
* Upload the video to YouTube. You can make the video unlisted if you’d like.
* **EACH team member should submit the following to Blackboard**:
* A copy of the final code
* A copy of this worksheet that contains all of the team members’ names
* A link to the YouTube video.
* A comment in the BlackBoard submission that contains the names of the 2 people in the team.