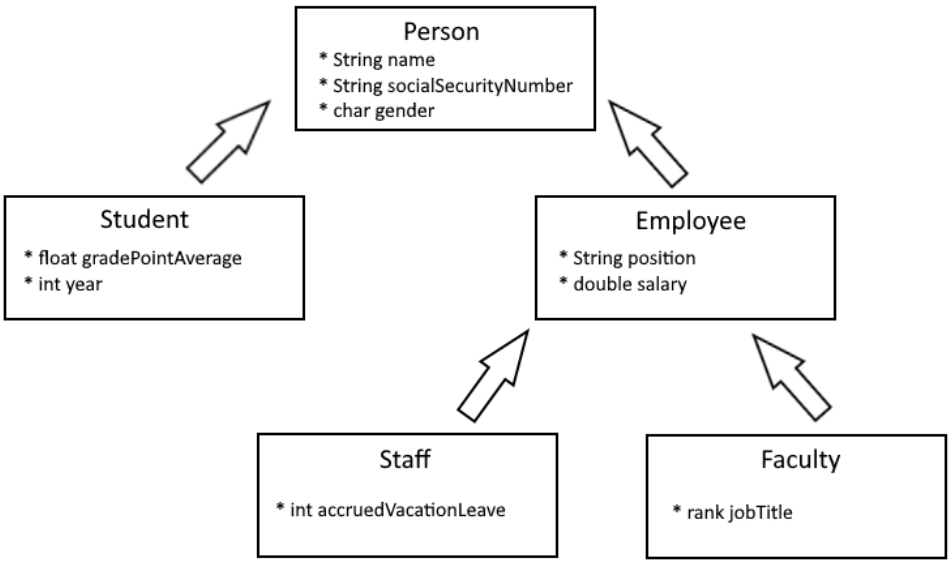
Logan Tillman

1. These questions all concern types:
    a. What is a type?
        - A type is a set of values and a set of operations that manipulate those values
    b. What is a subtype?
        - A type that permits the same values, same operations, and one or more additional operations as another type
    c. What is the relationship between an interface and a type?
        - Interfaces represent abstract types
    d. What is the relationship between a class and a type?
        - Classes allow you to declare and implement types
    e. What is the relationship between a subclass and a type?
        - Subclasses represent subtypes of their respective superclasses

2. Consider the following code:
    a. What is the output of fruit's main method? Explain why you get each of the two output lines.
        - The output is:
            1. orange
            2. fruit
        - You get both of the outputs because the printType() in orange overrides the printType() in fruit.
    b. What is the output of fruit's main method? Explain why you get each of the two output lines.
        - When you try to recompile the classes, you will get an error. This is because the final keyword will make the printType() method in fruit non-virtual. Which means it cannot be overridden in the orange class.

3. Suppose I am designing a personnel database for a university. The university has three types of personnel: students, staff, and faculty. Here are the characteristics of the three groups:
    a. Design and draw a class hierarchy for the above objects, based on the given properties. In your class hierarchy place variable names next to each class where you declare them. You will have to create additional superclasses since you will need to factor out some common information.

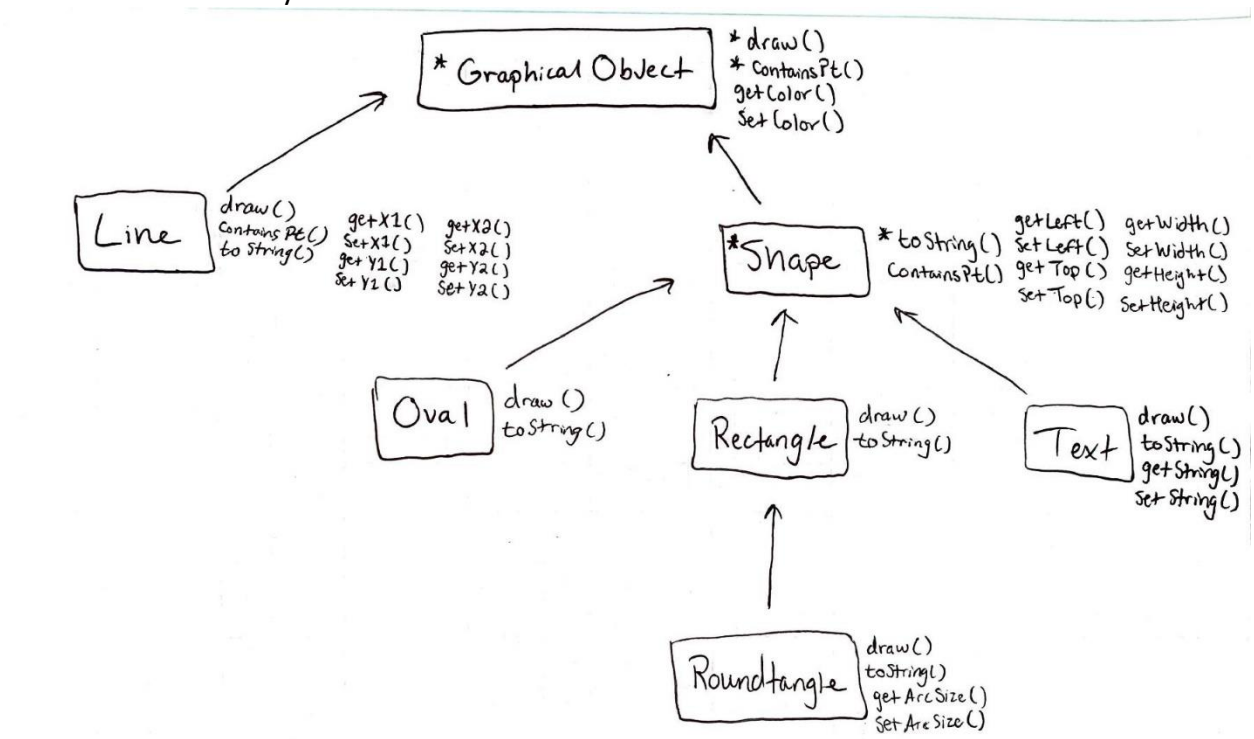


    b. Which of the above classes should be abstract classes?
        - The person class and employee class should be abstract
    c. Should the getter and setter methods for the properties be declared as virtual or non-virtual? Why?
        - The getters and setters should be non-virtual because the implementation does not change from class to class

5. Here are the questions you need to answer for this problem:

    a. Design and draw a class hierarchy for the above objects, based on the given properties. You may need to introduce one or more additional classes over and above the ones associated with the objects described above. My driver program in fact assumes that you have an additional class named GraphicalObject. In your class hierarchy place method names next to each class where you declare them and put an asterisk (*) next to each method that you declare abstract. For example, each of your leaf classes will

define a toString method, so the word "toString" should be placed next to each of them. Since each of them provides a method body for the toString method, you would not put an asterisk next to any of them.



a. Based on your class hierarchy, can the top element in the hierarchy be implemented as an interface, or must it be implemented as a class. Why or why not?
    - The top element in my hierarchy must be implemented as a class because it defines the functionality of the getters and setters for the color attribute