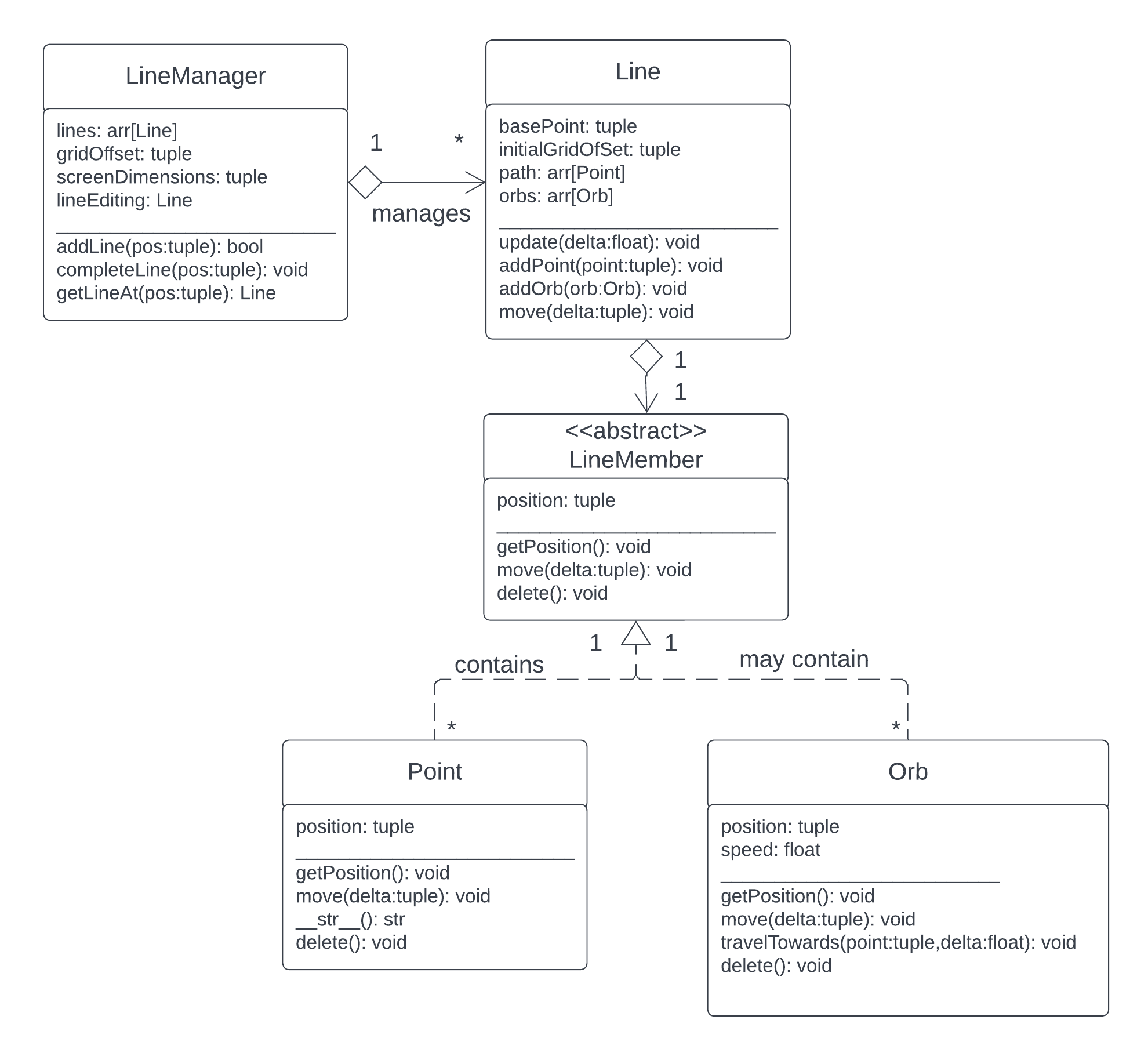
Acies- Group 5

Design Pattern: Composite

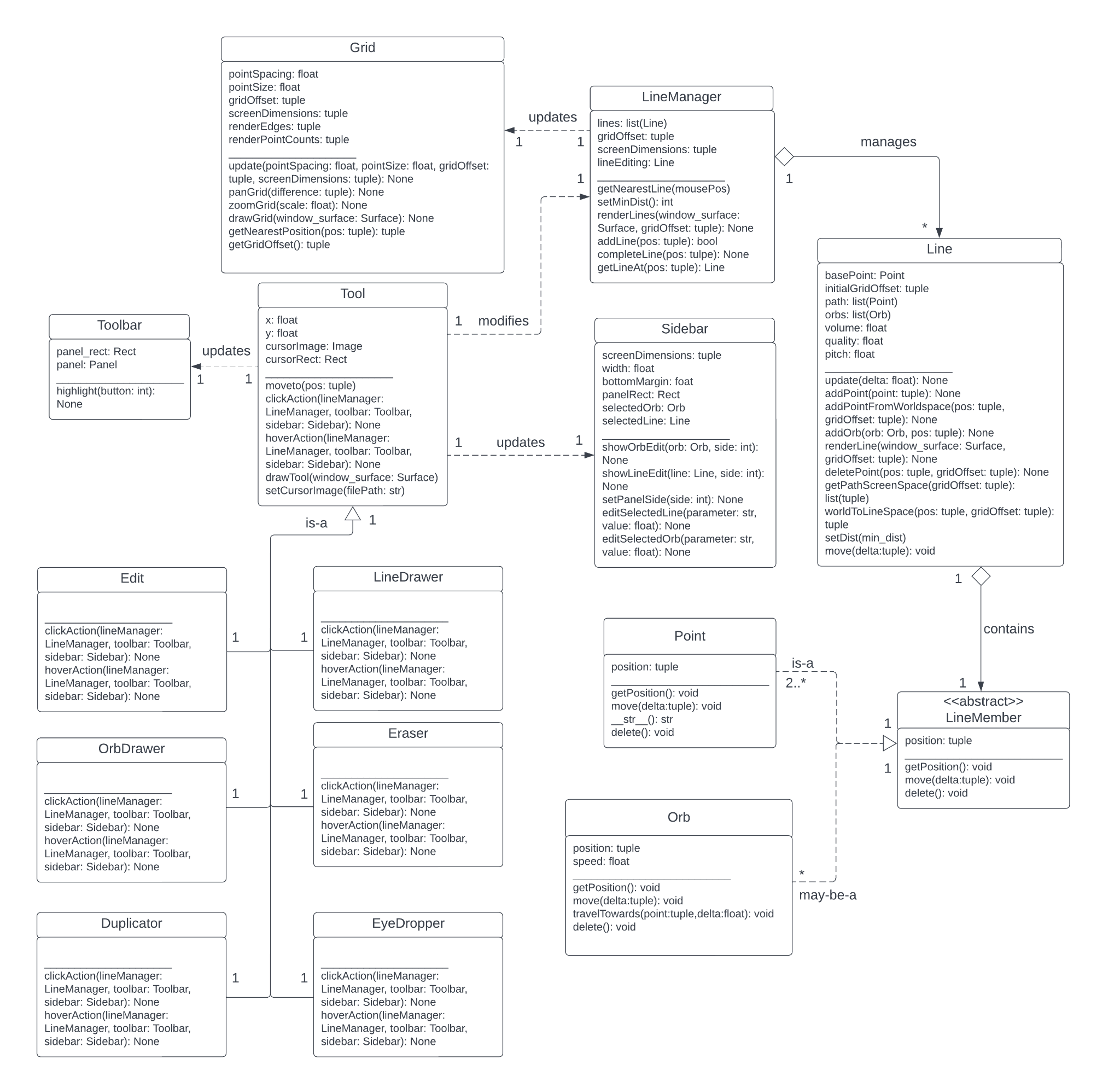


Description:

Composite is a structural design pattern. Composite is used when you want to treat a group of individual objects as a whole. For a real-life example, if you were a principal who wanted a classroom to go to lunch, you would simply tell the teacher that it was time to go to lunch, and they would instruct all of their students, no matter what type of students they were, to go to lunch. In this way, the Composite design pattern abstracts over the individual objects that may differ from one another. Using the controller class, one can control part or all of the objects under its domain. Individual objects are created from an abstract base class called *Component* that lists the methods that each of the concrete classes must define.

This design pattern fits well to Acies’ line system. In our application, each line on the grid is a number (two or more) of Points as well as a number of (0 or more) Orbs. Points are used to make sure that lines don’t intersect and that they are not diagonal, while Orbs are the things that traverse the line and make music. This design fits each criteria of the composite pattern. The Point and Orb objects are stored in arrays as part of the Line, as a group of objects. Thus, they can be referenced as a whole (apply an action to every element in the array) or as a part (apply an action to the two middle elements in the array, for example). They are only referenced through the Line class that they are on. Both the Point and Orb objects inherit from an abstract class LineMember, which mandates that each instantiation has a tuple holding its position, as well as the functions getPosition(), move(), and delete(). The move() function, in the future, would modify the tuple coordinates by a certain amount. If you moved a whole line, then you would normally have to change the position of each Point and Orb. Instead of doing this, however, you would simply call the Line move() function which would iterate through the Point and Orb arrays and use the move() function on each object. The same idea could be used for delete(). In this way, it allows us to treat the line as a composite, instead of interacting with each Point or Orb object.

Design Class Diagram:



Notes on the Design Class Diagram:

Many getters and setters were omitted from the design class diagram as in general they are implied or unimportant. Additionally many functions are named differently from our design sequence diagrams or have slightly different arguments due to unexpected requirements to make the implementation functional. Finally, the design class diagram does not include most UI elements as in most other frameworks they would not be part of the code at all.