Mark Pratt

CST-135

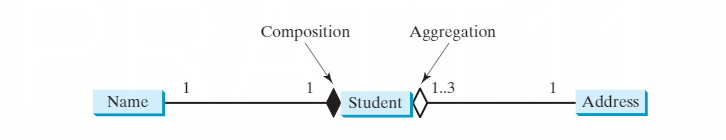
8/20/2020

Professor Elchouemi

Activity 1 Key Learning

In terms of variable visibility, there are three main types: Public, Private, and Protected. A variable visibility modifier effects what can see or use a variable. You can also use these on classes as well. If I have a class scoped variable “public int Age;” then when I create an instance of that class anywhere else, I can access that variable. We do this through the use of getter and setter methods. Chapter 9 goes in depth about the private modifier and the protected modifier is introduced in chapter 11. “The Private modifier makes methods and data fields accessible only from within its own class.” (Page 367 Electronic). For example: “private void doSomething();” would only be accessible from within whatever class it is in. If I created an instance of that class somewhere else and tried to invoke class.doSomething(); nothing would happen. It would throw an error, actually, since the modifier does not allow access. There are no restrictions on accessibility of methods or fields from within a class, however. One thing to note, “Using modifiers public and private on local variables would cause a compile error.” (Page 368 Electronic). These modifiers are only for methods or fields. They do not work, nor would they be necessary, for local variables.

When looking at class relationships, we are introduced to the term Association. Association “is a general binary relationship that describes an activity between two classes.” (Page 397 Electronic) We are shown two main forms of association: aggregation and composition. Aggregation is a form of association that models the “*has-a”* relationship. This put simply is that one class instance *has a* reference to an instance of another class. The example using in the book for this was a student and their address. The student *has an* address, therefore it has a *“has a”* relationship meaning it is association. The other form of association that we were introduced to was composition. This is slightly different than association because of the idea of dependency. With association, the class will have an idea about the other class, but it does not depend on it. With composition, there still is the *“has a”* relationship, but one of the parts of the relationship depends on the other. The example used in the book is a student and their name. The student *has a* name and the name belongs to the student. Though the name depends on the existence of the student. If we delete the student, the name is also deleted. If we delete the student, their address is not necessarily deleted as well. This is the main difference between aggregation and composition. When we denote these within diagrams, we use a black line between classes to show association. To show composition, we put a filled-in black diamond. To show aggregation, we use an empty black diamond. Below is a screenshot of the book to show what I mean.



Here we see the same example as described above. The student and address have an aggregated relationship, and the student and the name have a composition relationship.