**CSCI 2302**

**OOT Chapter – Composition Lab**

Table of Contents

[Intro 1](#_Toc82358629)

[Learning Goals 1](#_Toc82358630)

[Notes 1](#_Toc82358631)

[Has-a 2](#_Toc82358632)

[Task 2](#_Toc82358633)

[Submit 3](#_Toc82358634)

Intro: Understanding the concepts of Object-Oriented Programming is the purpose of this course. The overall approach of using an OOP approach is to focus on objects verses a procedural approach. By focusing on objects, we can encapsulate states/fields/data and behaviors/methods to create an object that has a tightly coupled data aspect, high cohesion within itself and low cohesion with the program using the object.

Objects can interact with other object and objects be made up of other objects. Understanding the relationship of objects – between them and comprising of them is a core concept of Object-Oriented Thinking, which is the purpose of this chapter. As such, this lab focuses on how an object can be made up of other objects. In specific, we will be looking at the composition relationship.

Learning Goals: This lab’s goal is to understand the concept of composition objects. To demonstrate this concept, we will create a composition object, thus we are applying this concept; using the composition object, thus we are understanding and analyzing this concept.

Notes: A composition relationship is a special form of aggregation, it is a *has-a* relationship where the object that is comprised of other objects (the owner of the object / the parent) owns the objects (child objects) that comprise it. The composite object owns the object it is comprised of and it (the child object) cannot exist without the owner object (the parent object).

* Composition implies a relationship where the child cannot exist independent of the parent. Example: House (parent) and Room (child). Rooms don't exist separate to a House.
* the relationship between Class A and Class B is defined as a strong lifecycle dependency between the two, meaning that when Class A is deleted then Class B is also deleted as a result

Definition: The nature of something's ingredients; the way in which a whole or mixture is made up.

## Has-a

Composition is *has-a* relationship where an object is ***exclusively owned by*** a composite object and cannot exist on its own. It is a special form of an aggregating object.

Task: Complete the steps outlined below in a file named CompositionObject.java.

1. Download the CompositionObject.java from the Dropbox in Brightspace by D2L and rename with yourMySFAusernamer\_CompositionObject.java.
2. Analyze and understand the object, LiveRedOakLeaf, it has the following UML diagram.

|  |
| --- |
| **LiveRedOakLeaf** |
| -leafType: String = “Deciduous”  -leafShape: String = “Pointed Lobed” |
| +LiveLeaf()  +getLeafType(): String  +getLeafShape(): String  +toString(): String |

A LiveRedOakLeaf cannot exist on its own, it has to have an object to help support the vascular system or it will die.

1. Complete the composition object TexasRedOakTree, in the CompositionObject file based on the following UML diagram. This applies your concepts of object and composition of objects.

|  |
| --- |
| **TexasRedOakTree** |
| -leaf: LiveRedOakLeaf  -rootType: String = “Taproot”  -bark: String = “dark gray to black with platelike scales” |
| +TexasRedOakTree()  +getLeaf(): String  +getRootType(): String  +getBark(): String  +toString(): String |

1. Apply, analyze, and understand composition objects by complete the following in the CompositionObject file:
   1. Instantiate a LiveRedOakLeaf object using the identifier of lrol.
   2. Invoke the lrol toString method.
   3. Instantiate a TexasRedOakTree object using the identifier of trot.
   4. Invoke the trot toString method.

Submit: Submit yourMySFAusernamer\_CompositionObject.java in the Dropbox in Brightspace by D2L.