https://youtu.be/iUkRGy6kK4A

Composition

Composition is a closely related alternative to inheritance. Composition involves constructing ("composing") classes from other classes, instead of inheriting traits from a parent class.

A common way to distinguish "composition" from "inheritance" is to think about what an object can do, rather than what it is. This is often expressed as **"has a"** versus **"is a"**.

From the standpoint of composition, a cat "has a" head and "has a" set of paws and "has a" tail.

From the standpoint of inheritance, a cat "is a" mammal.

There is **no hard and fast rule** about when to prefer composition over inheritance. In general, if a class needs only extend a small amount of functionality beyond what is already offered by another class, it makes sense to **inherit** from that other class. However, if a class needs to contain functionality from a variety of otherwise unrelated classes, it makes sense to **compose** the class from those other classes.

In this example, you'll practice working with composition in C++.

Instructions

In this exercise, you will start with a `LineSegment` class and create a `Circle` class.

Note that you will compose `Circle` from `LineSegment`, instead of inheriting `Circle` from `LineSegment`. Specifically, the `length` attribute from `LineSegment` will become the circle's radius.

1. Create a class `LineSegment`.
2. Declare an attribute `length` in class `LineSegment`.
3. Define pi (3.14159) with a `macro`.
4. Create a class `Circle`, composed of a `LineSegment` that represent's the circle's radius. Use this radius to calculate the area of the circle (area of a circle = πr^2).
5. Verify the behavior of `Circle` in `main()`.

Saving Graffiti Recording. Please wait...