

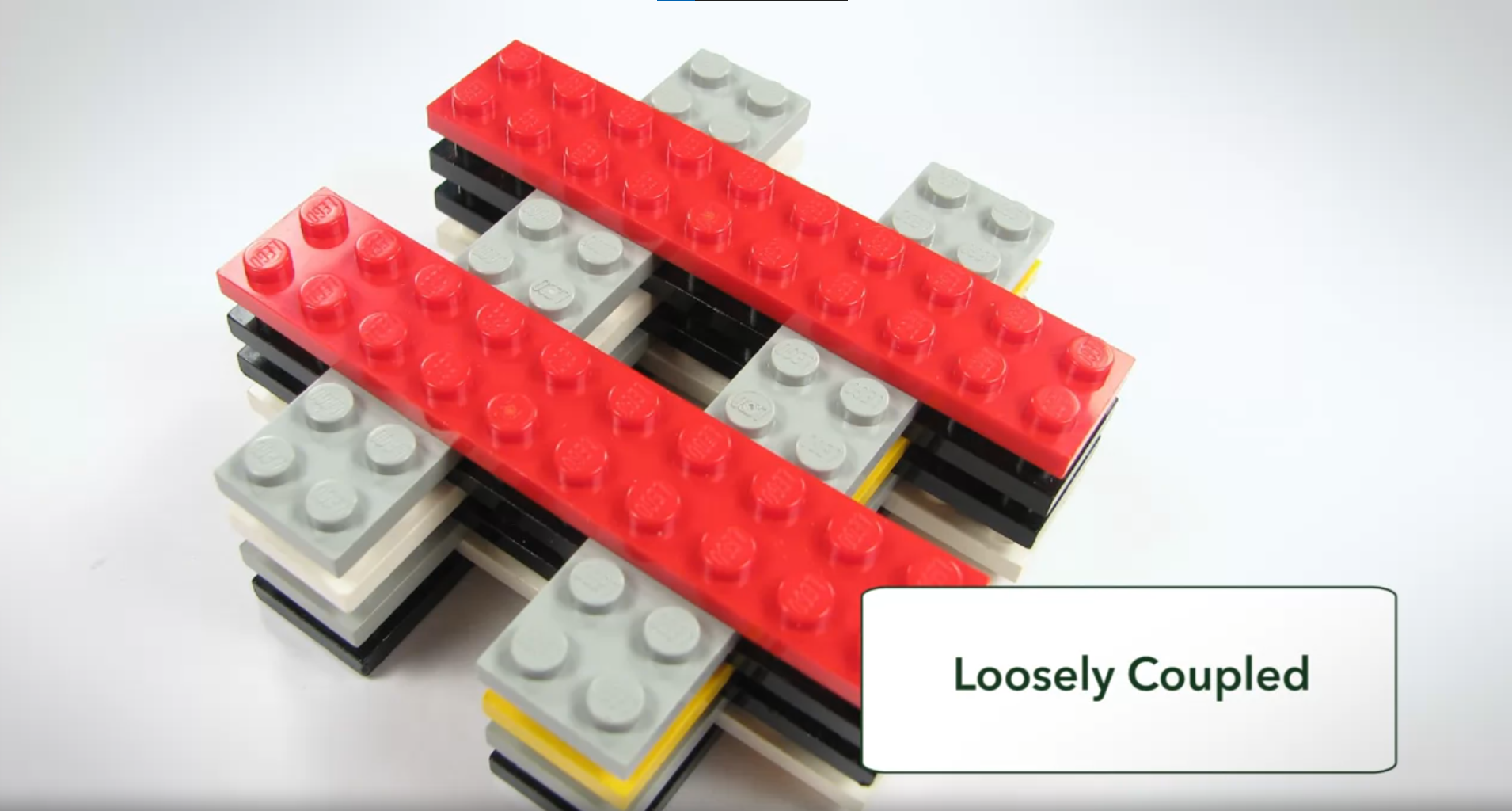
Coupling – complexity module to module (connection)

Cohesion – complexity within a module

Coupling



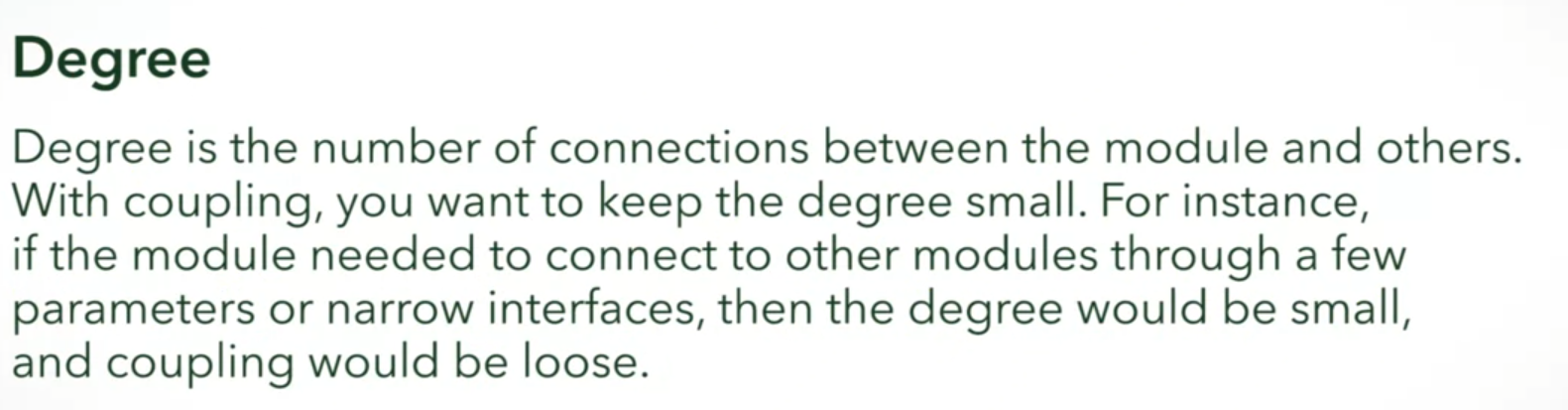
* Highly reliant to one module



* Easy to connect to other modules
* You want this!

When evaluating coupling of a module you need to consider:

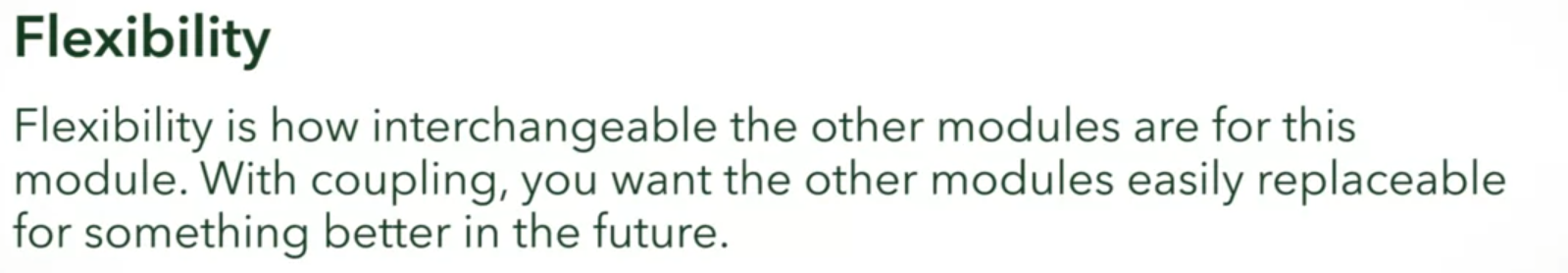
* Degree



* Ease



* Flexibility



**What aspects of a lego piece makes it loosely coupled to other lego pieces?**



Corresponding pieces are hard to find.



It can be connected to any lego piece.

Correct

Absolutely! This is the reusability aspect of coupling.



Connecting two legos work flawlessly.

Correct

Great job. This describes the flexibility that you get from loose coupling.



Pieces can be interchanged.

Correct

Awesome! This also describes the flexibilty you get from loose coupling.

**What aspects of a puzzle piece make it tightly coupled to other puzzle pieces?**



It can only be connected to specific puzzle pieces.

Correct

It can only be connected to specific puzzle pieces, making them, inflexible.



Pieces are interchangeable.



Corresponding pieces are hard to find.

Correct

A corresponding piece is hard to find, giving it poor ease.

Cohesion





* One task, clear purpose
* You want this!



* Tries to encapsulate more than one purpose or has non-clear purpose

**What aspects of a lego piece makes it high cohesion?**



A lego’s task is clear, “to connect to other lego”.

Correct

Absolutely! A cohesive "piece" will have a clear purpose. That's high cohesion!

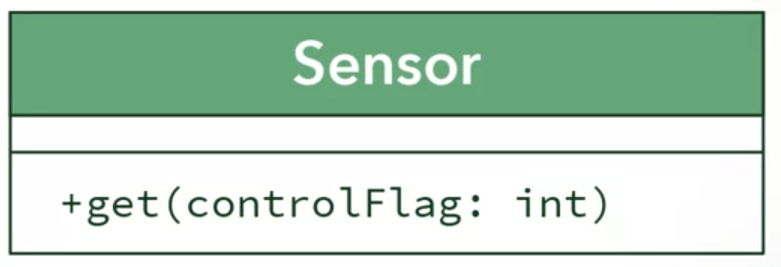
**What aspects of a puzzle piece makes it low cohesion?**

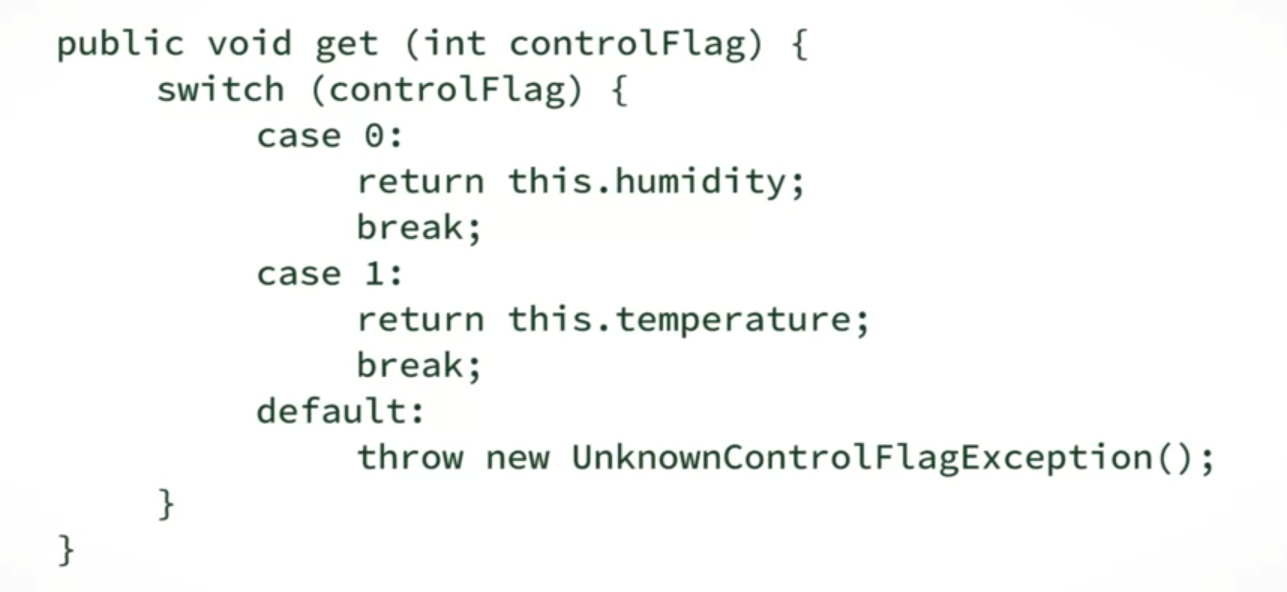


Its responsibility is unclear and varies in the pieces.

Correct

This is a true statement that addresses low cohesion.

****

****

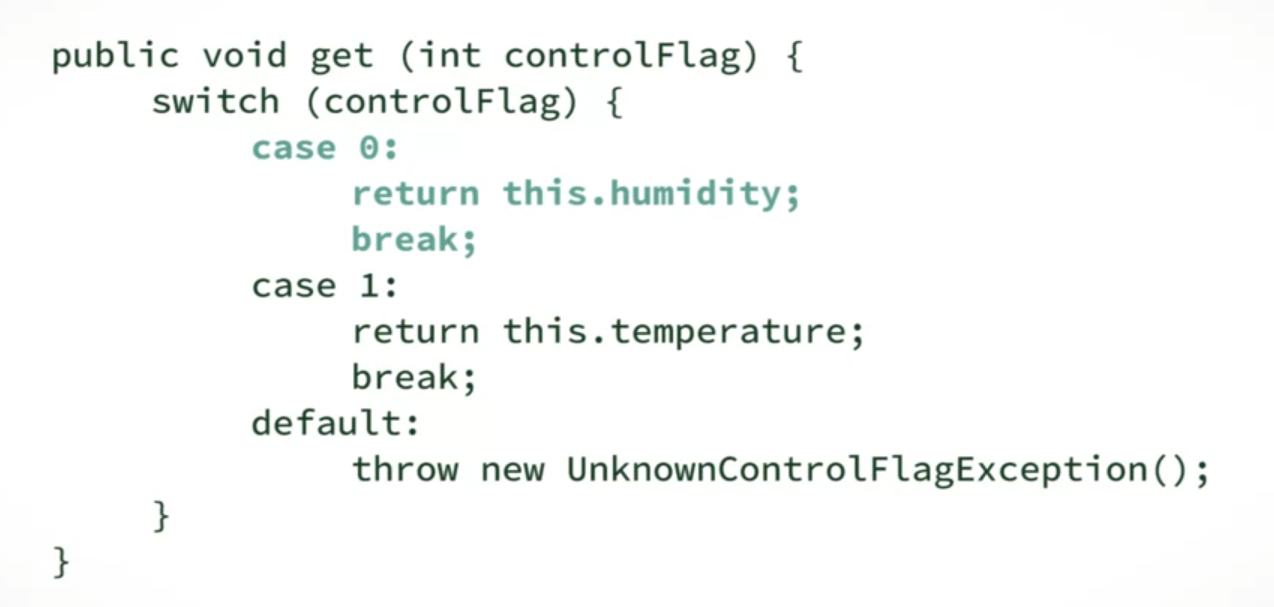
**Which of the following is true about Sensor class and its cohesion?**



Sensor class has low cohesion, since it has two purposes and therefore unclear.

Correct

Correct! One purpose is enough for one class. Any more than that takes away from cohesion.



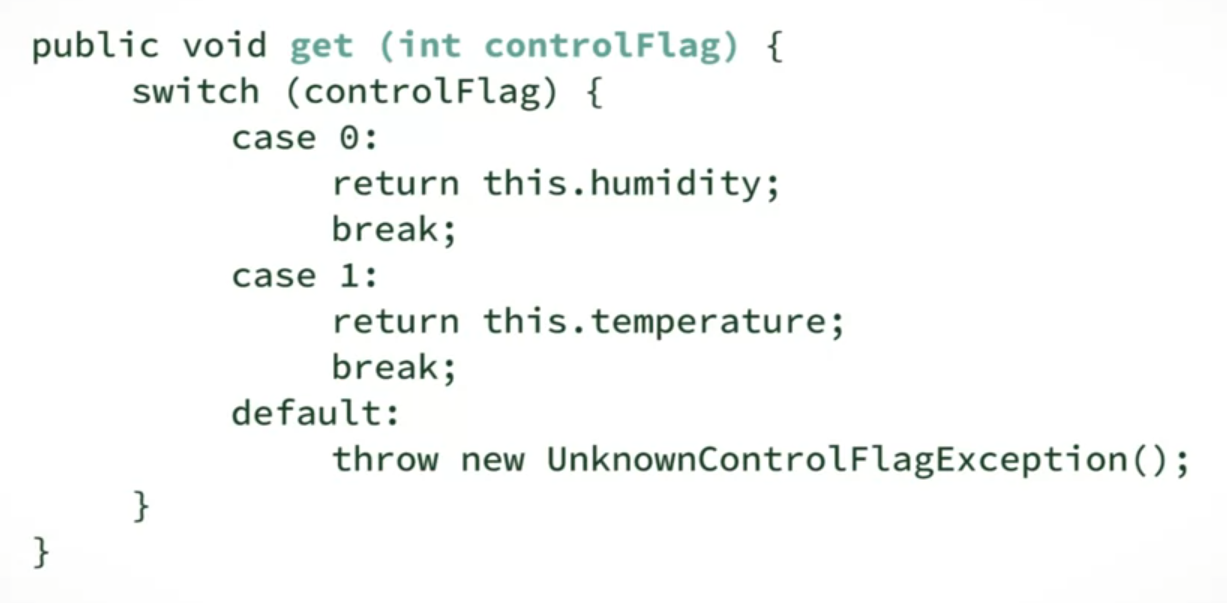
**Which of the following is true about coupling to the Sensor class?**



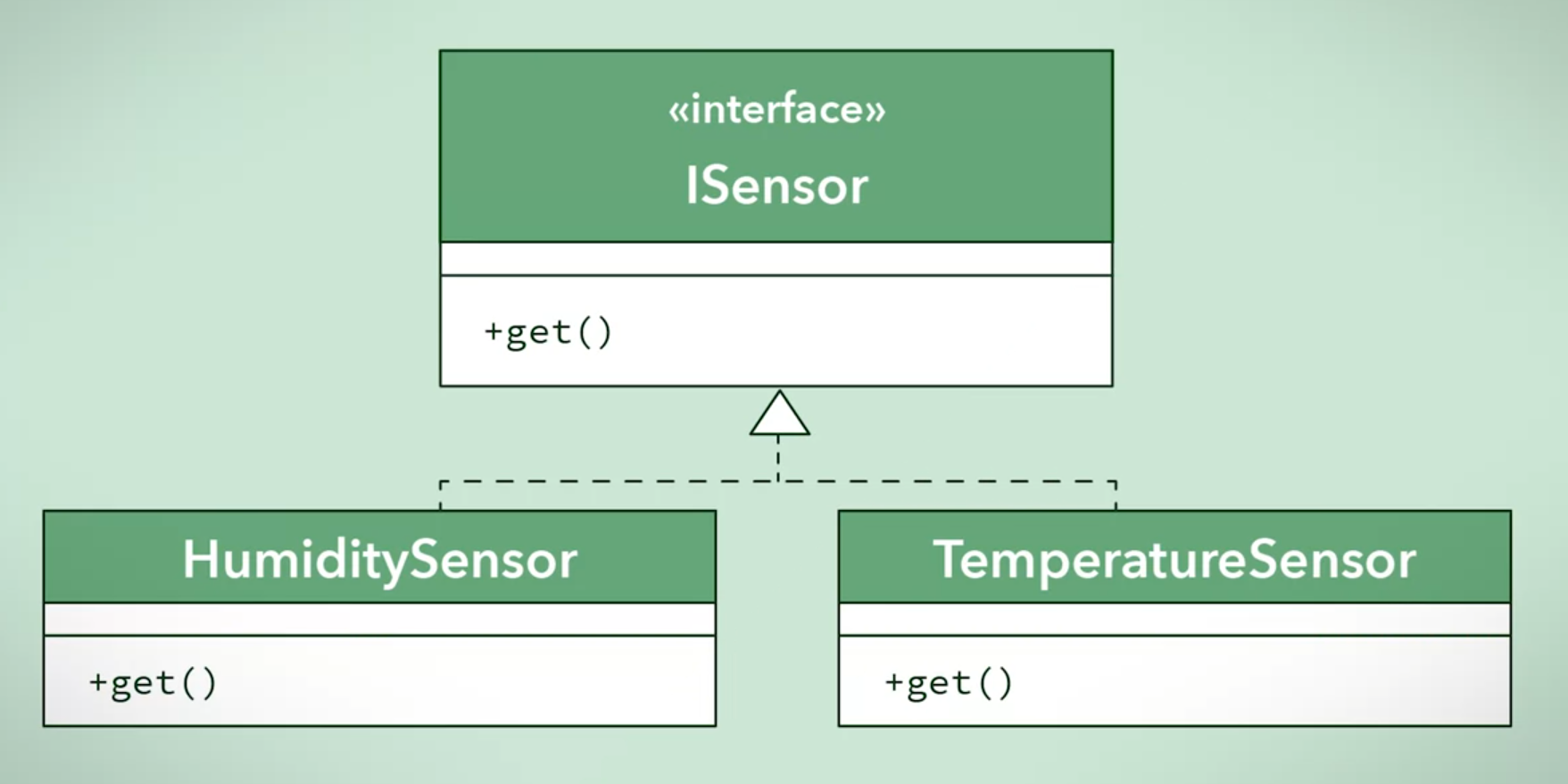
Sensor class has tight coupling, since its **get** method is not straightforward.

Correct

Correct! A user of the get method will have to open up Sensor to see how it works. That's not loose coupling!

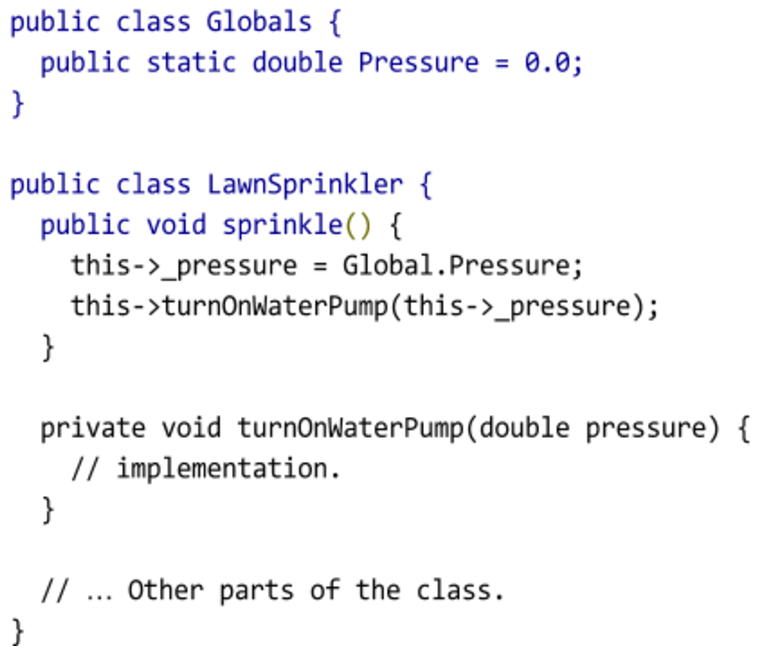
****

Let’s replace the system:



* Highly cohesive
  + Don’t have to break into encapsulation to look inside the method
  + Clear purpose
* Tightly coupled
  + Method is straightforward, you can assume that HumiditySensor returns humidity, TemperatureSensor returns temperature

Suppose you are working on a company that manufactures an automated watering program. You are asked to evaluate the complexity of the following code:



**What can you say about LawnSprinkler’s cohesion?**



It has one clear purpose, to sprinkle the lawn. Therefore it is highly cohesive.

Correct

Great job! The class is highly cohesive because it is completely focused on Sprinkling. turnOnWaterPump() is part of that responsibility, although if it was outsourced to another class, then users of sprinkle() would have no idea!

**What can you say about the coupling of LawnSprinkler’s sprinkle method?**



Despite having no parameter, it lacks ease due to its use of globals. Therefore tightly coupled.

Correct

The reason why you should avoid using globals is that you never know what other modules use or modify them. This makes methods that use them have low ease which makes them tightly coupled.