Object Oriented Programming

OOP Is NOT Class Oriented Programming

When writing, and reading, code, we see a class construct, but a class is merely a template for objects. When code runs, it runs on objects. Always think about objects.

Objects Are Living Things

Think of an object is a living thing. In an executing program, objects do have a lifetime: they are created, they do things, and then they are destroyed. Living things can do things, they have capabilities; living things also have characteristics, they have attributes. In a true OOP program, objects are living things.

All Attributes Should Be Private (or at least protected)

An object's attributes are its own data, there is no good reason other code should have access to them. The object itself should use them, and then provide its clients with derived results. Sometimes subclasses may need access.

Getters Are Somewhat Evil; Setters are More Evil

Getters and setters are just a lazy way to give access to object attributes without directly violating the rule above. At least getters only give read access; setters allow external code to assign values! This completely violates OOP principles. The object itself should manage its own attributes.

Use Implementation Inheritance Wisely

Implementation inheritance (e.g., Java extends) is strong coupling between classes. Use it carefully. It is a great and powerful tool, and is often overused. Be careful.

Use Interfaces And Interface Inheritance Liberally

Interfaces act as abstractions, and computer science loves abstractions. Make lots of small interfaces that capture your needed abstractions, and then program to those abstractions. Create classes that implement those interfaces (e.g., Java implements).

Unless You Are Creating a Math Library, Don't Use Static

Use a Singleton Pattern instead, if you need a class of just one object.