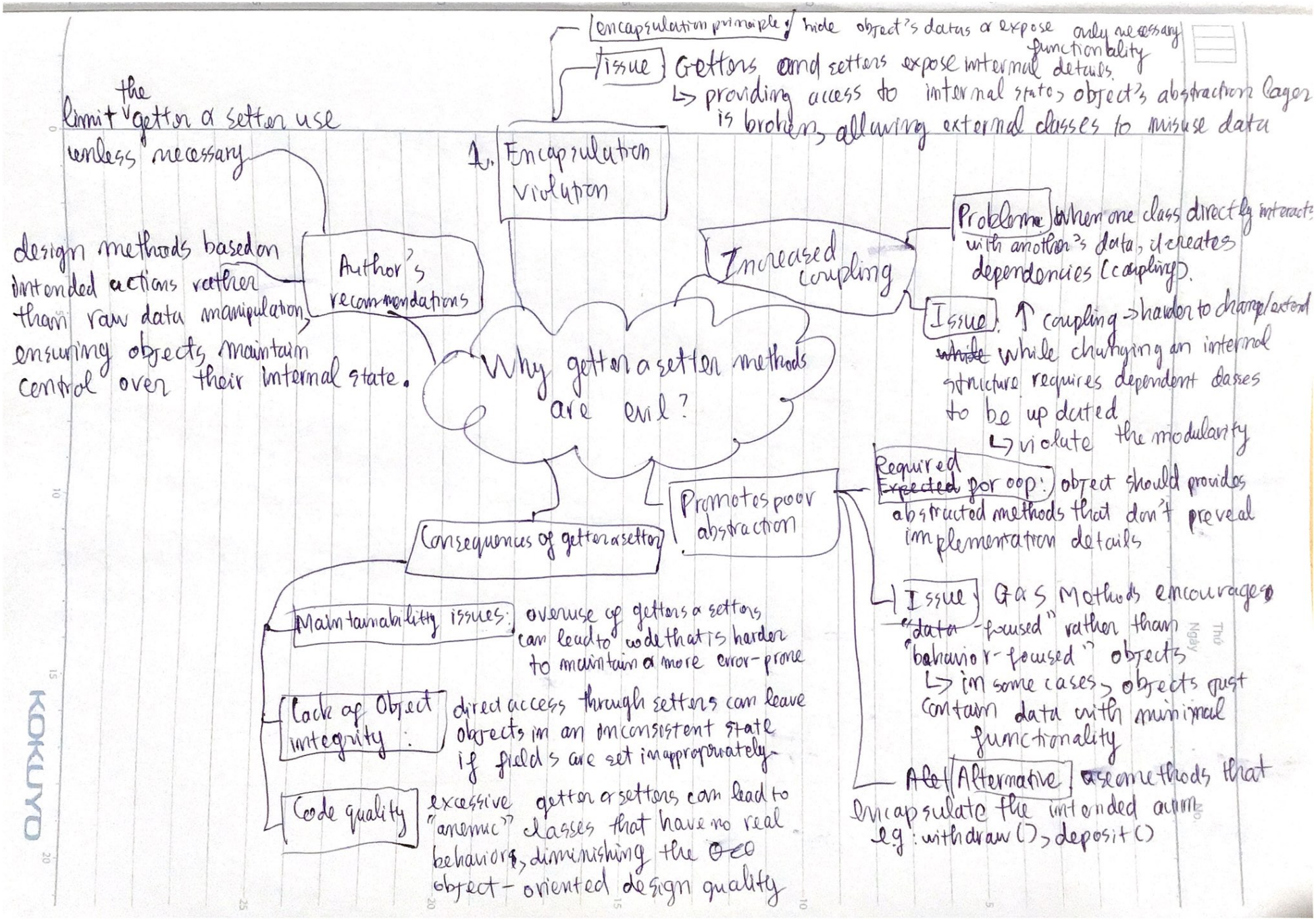
Why getter a setter methods are evil?
1. Encapsulation violation
encapsulation principle: hide object's datas or expose only necessary functionality
Issue: Getters and setters expose internal details.
↳ providing access to internal state, object's abstraction layer is broken, allowing external classes to misuse data
limit the getter a setter use unless necessary
Author's recommendations
design methods based on intended actions rather than raw data manipulation, ensuring objects maintain control over their internal state.
Increased coupling
Problème: when one class directly interacts with another's data, it creates dependencies (coupling).
Issue: ↑ coupling → harder to change/extend while changing an internal structure requires dependent classes to be updated
↳ violate the modularity
Promotes poor abstraction
Required (Expected) por oop: object should provides abstracted methods that don't reveal implementation details
Issue: Gas Methods encourages "data-focused" rather than "behavior-focused" objects
↳ in some cases, objects just contain data with minimal functionality
Alternative: use methods that encapsulate the intended action
e.g: withdraw(), deposit()
Consequences of getter a setter
Maintainability issues: overuse of getters a setters can lead to code that is harder to maintain a more error-prone
Lack of Object integrity: direct access through setters can leave objects in an inconsistent state if fields are set inappropriately.
Code quality: excessive getter a setters can lead to "anemic" classes that have no real behaviors, diminishing the object-oriented design quality