Design Principles

Thierry Sans

with slides from Anya Tafliovich

SOLID

Single responsibility principle

Open/closed principle

Liskov substitution principle

Interface segregation principle

Dependency inversion principle

Single Responsibility Principle

Every class should have **a single responsibility** and that responsibility should be entirely encapsulated by the class

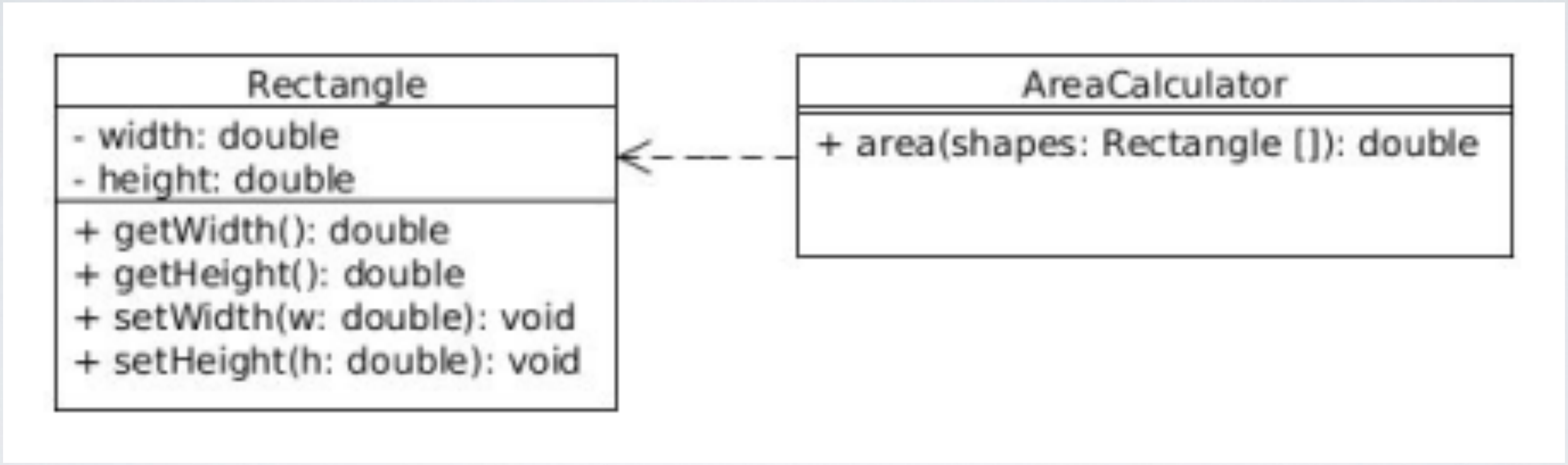
➡ Also referred as the cohesion principle

Open/Closed Principle

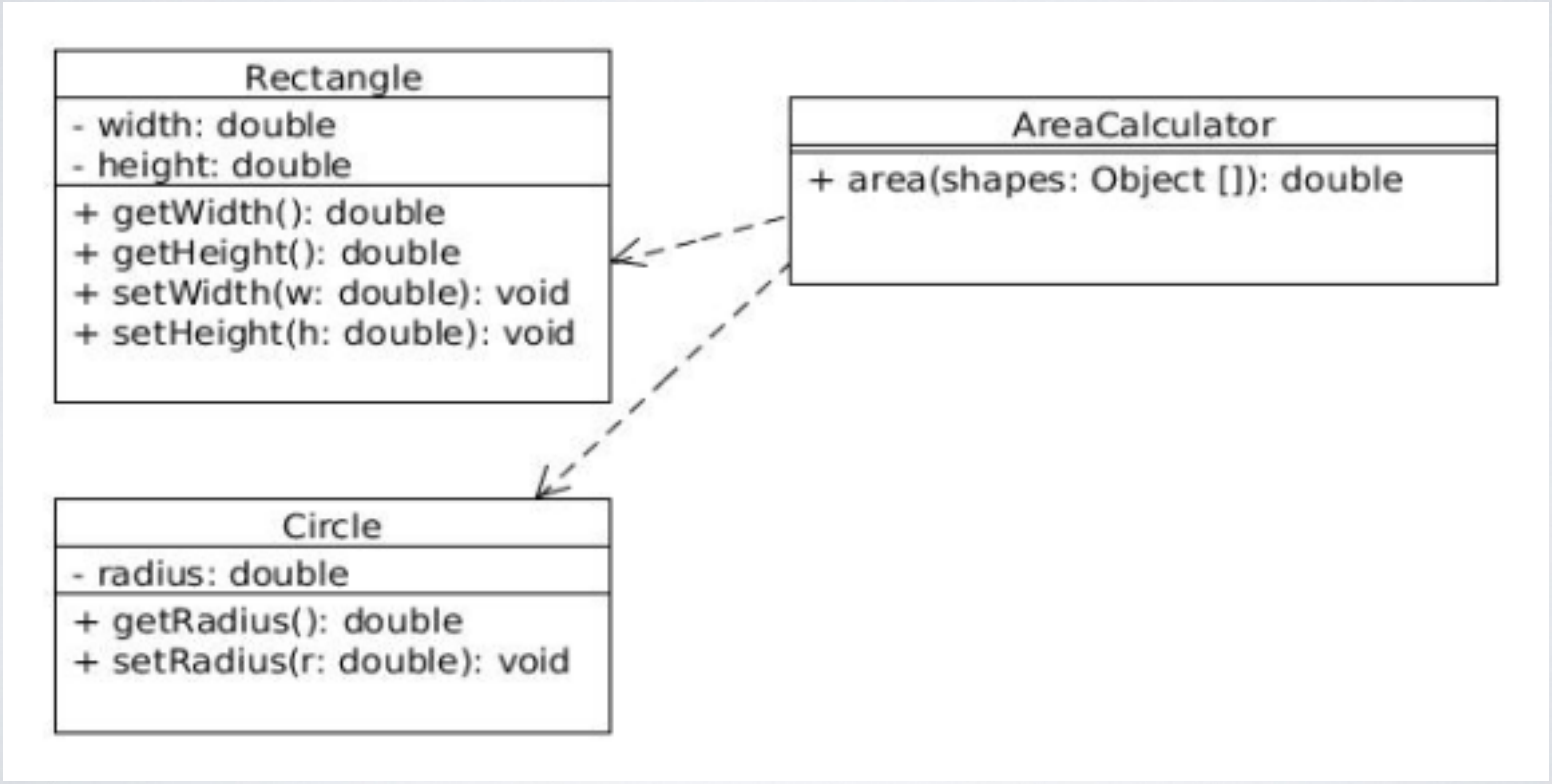
Software entities (classes, modules, functions, etc.) should be **open for extension**, but **closed for modification**

➡ Also referred as the information hiding principle

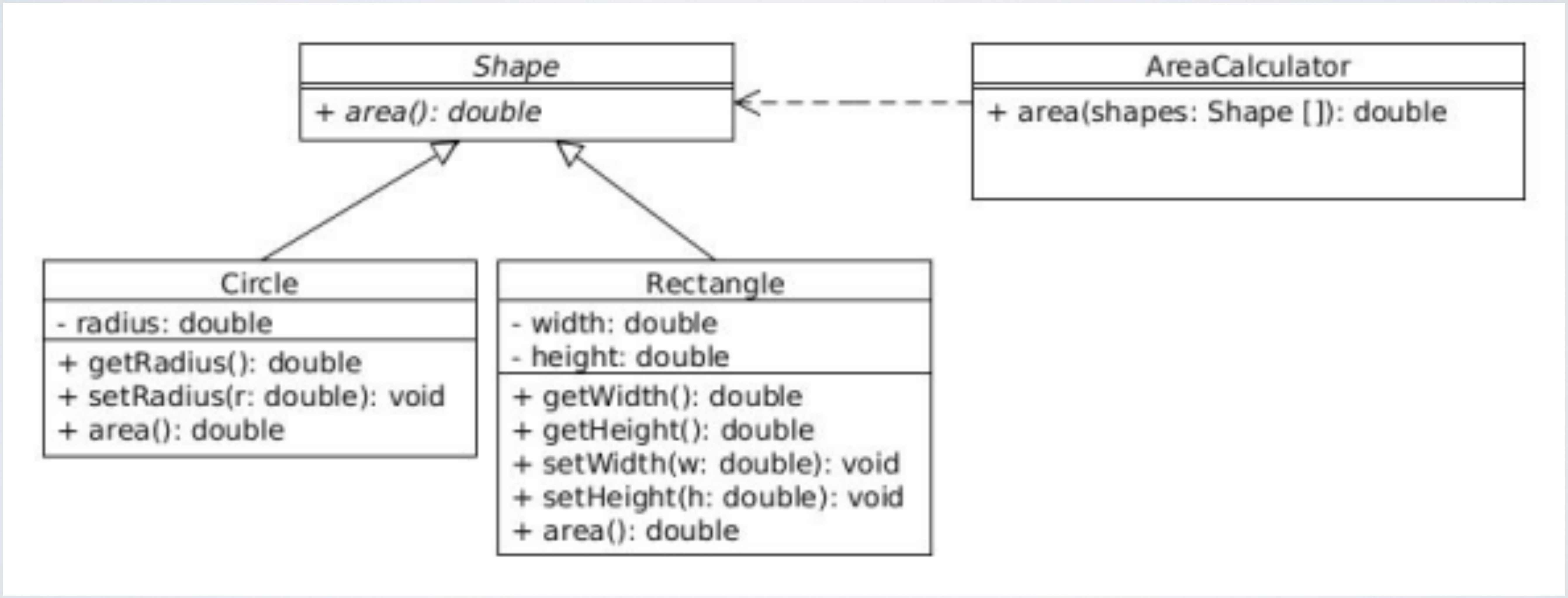
An example of bad design



An example of a bad solution



A good solution

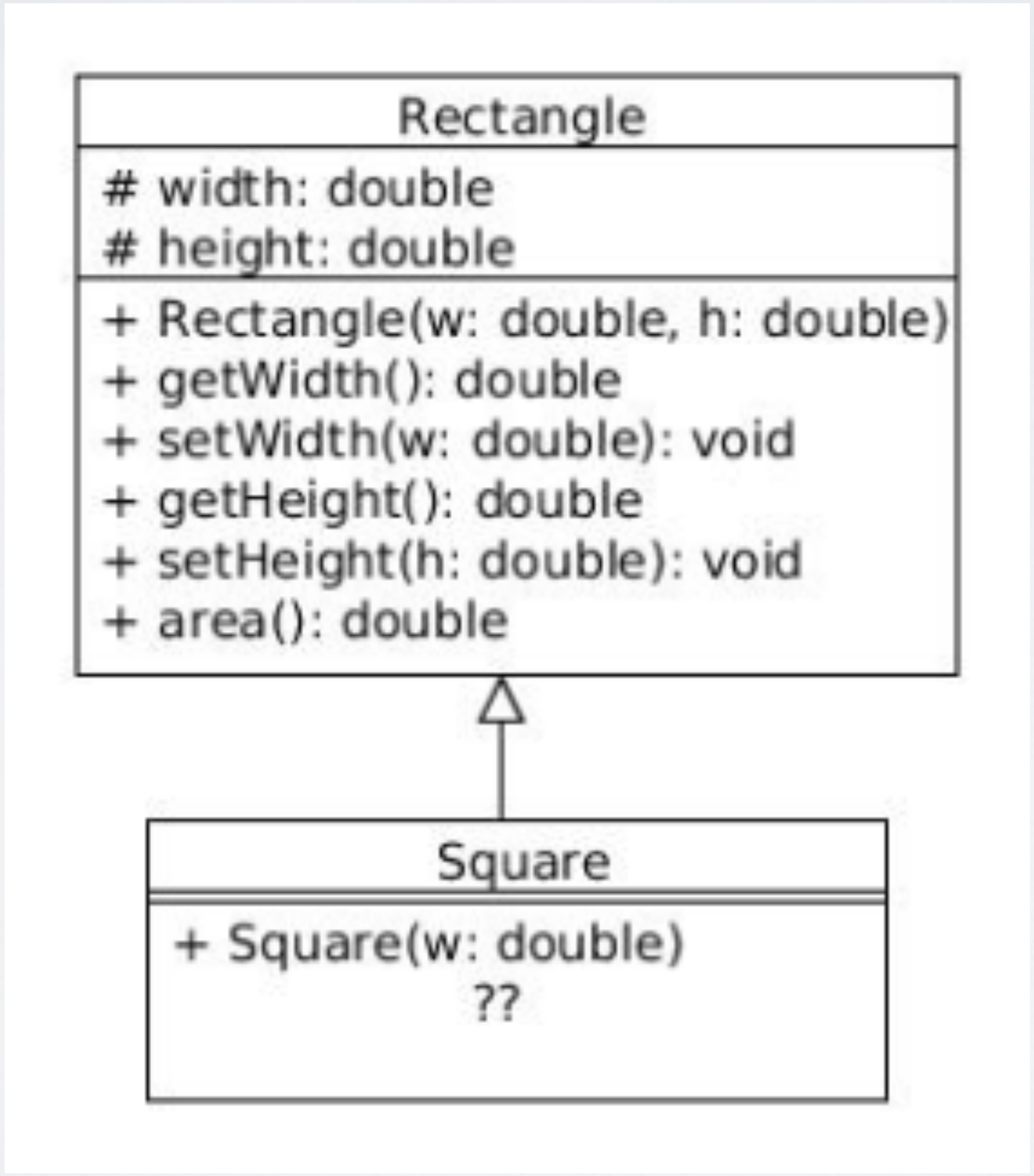


Liskov Substitution Principle

If S is a subtype of T, then objects of type **S may be substituted for objects of type T,** without altering any of the desired properties of the program

➡ Also referred as the strong behavioral subtyping principle

An example of bad design



Interface Segregation Principle

No client should be forced to depend **on methods it does not use**

➡ Also referred as the high cohesion principle

Dependency inversion principle

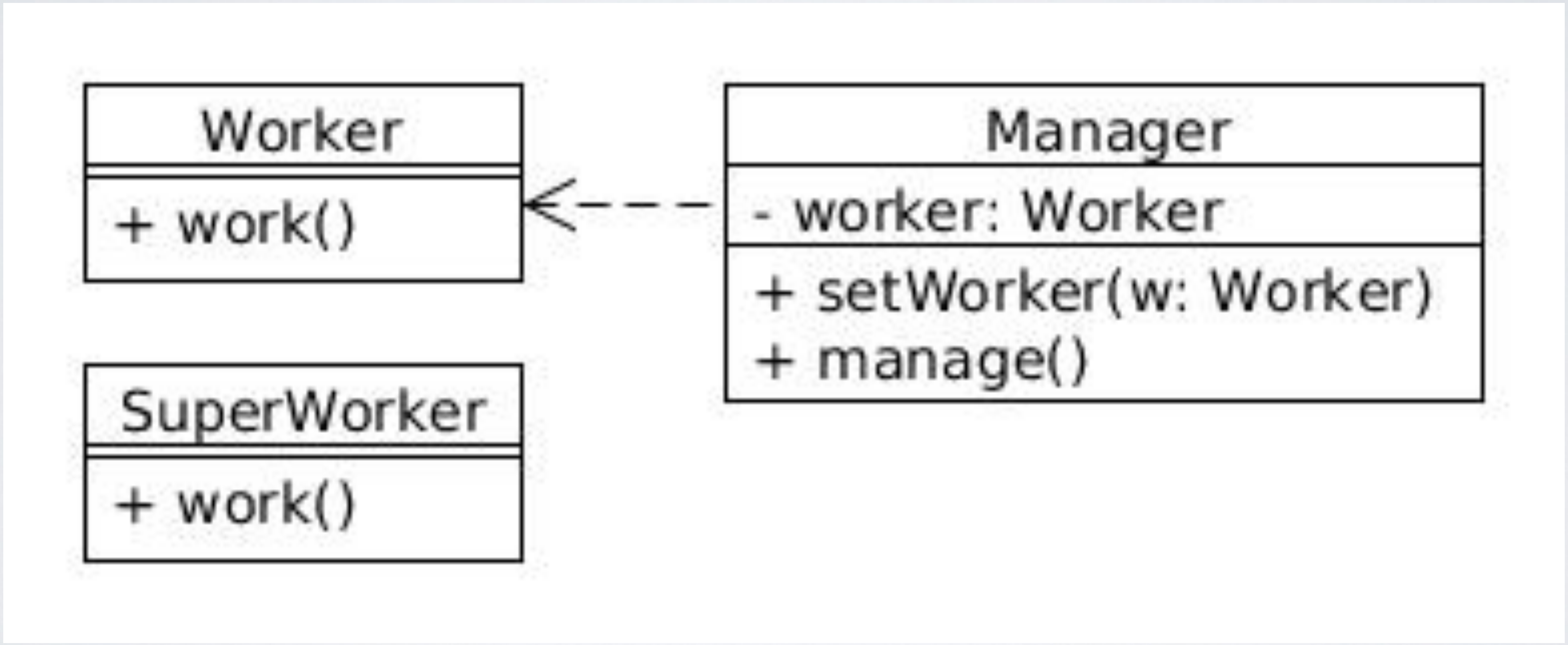
Dependency relationship between high-level module and low-level module are reversed

- High-level modules should not depend on low-level modules Both should depend on **abstractions**

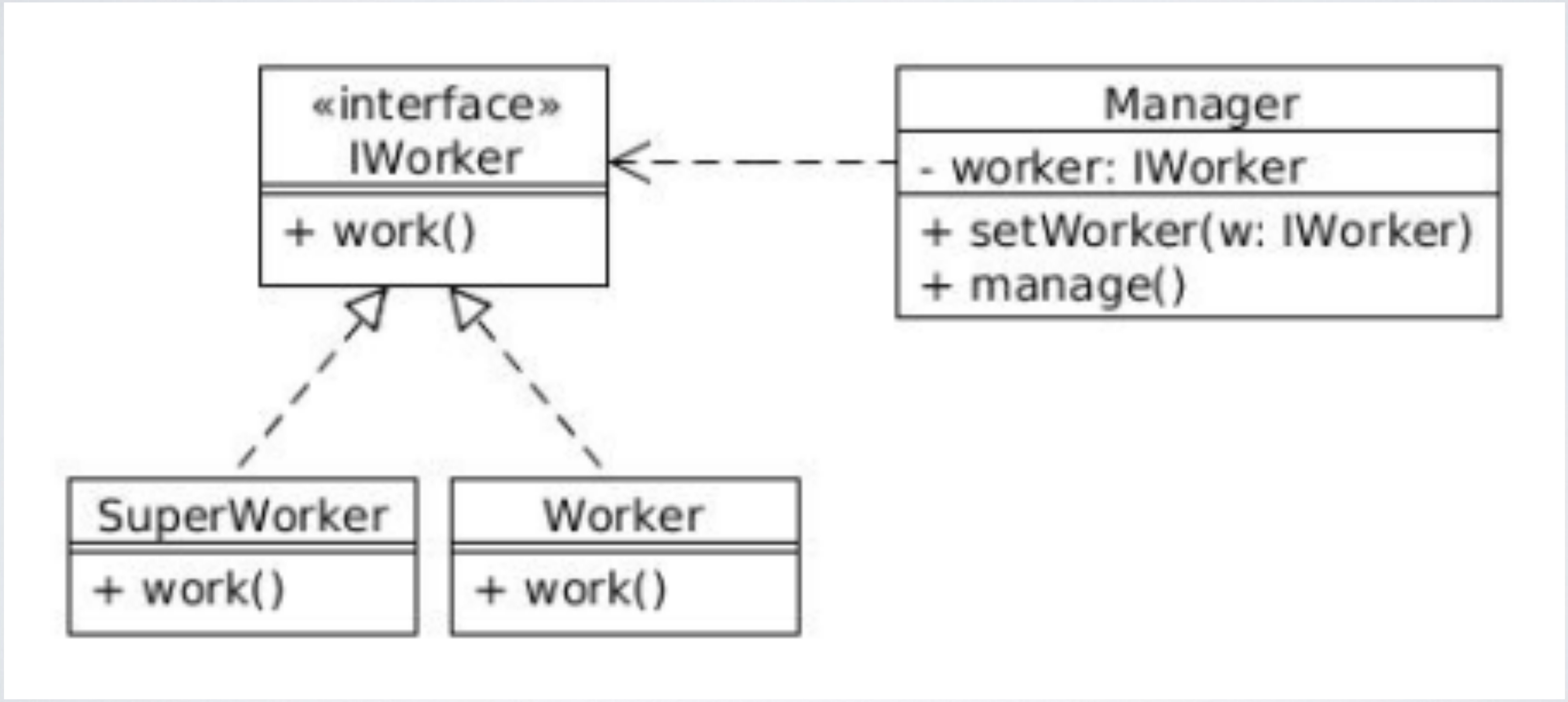
- Abstractions should not depend on details. Details should depend on abstractions

➡ Also referred as the <u>decoupling</u> principle

An example of bad design



An example of good design



Coming next

Many **Design Patterns** follow the SOLID principles