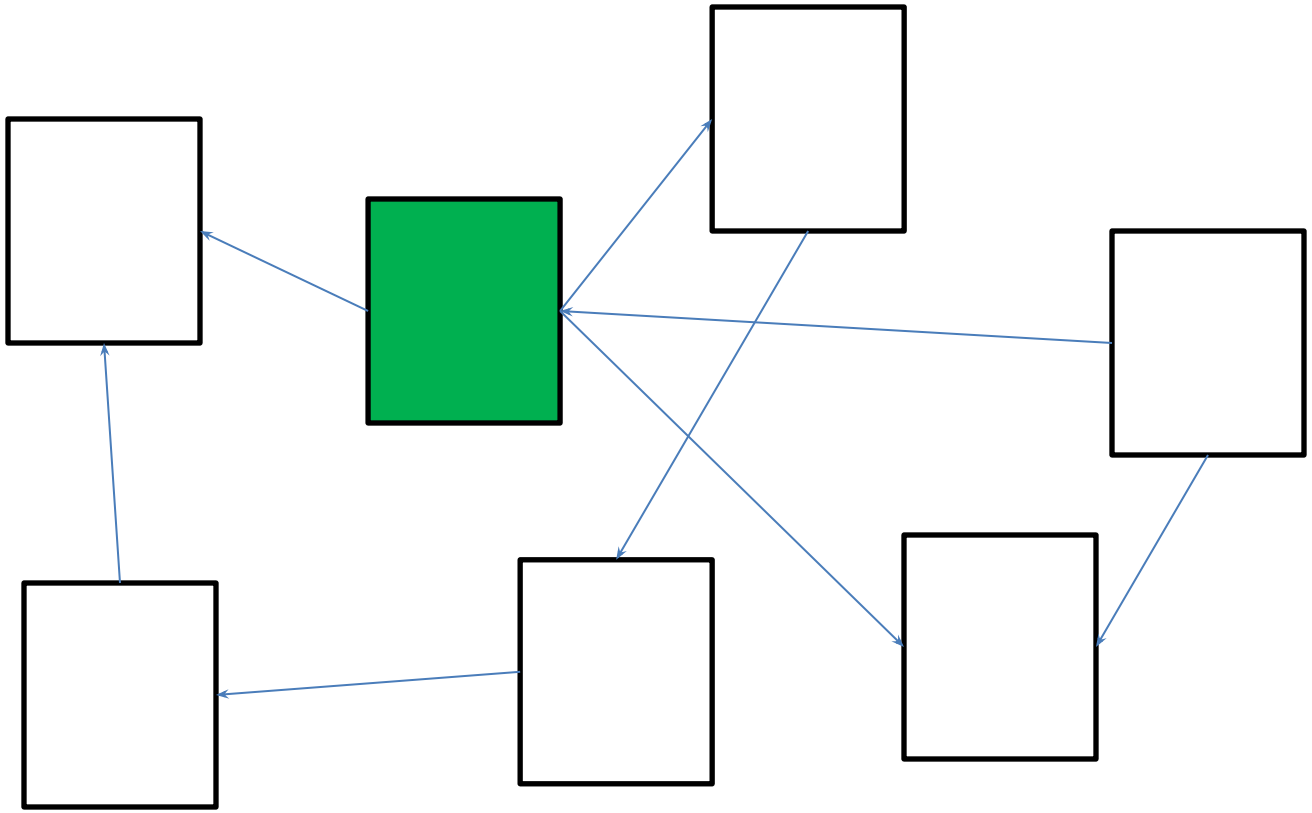
SOLID

Java program is a set of objects interacting with each other

Good program design -> good classes design



SOLID

SOLID is a mnemonic acronym introduced by Robert C. Martin in the early 2000s which stands for five basic principles of object-oriented programming and design.

Single responsibility
Open-closed
Liskov substitution
Interface segregation
Dependency inversion

S

Single Responsibility Principle

"An object should have only a single responsibility, and that responsibility should be entirely encapsulated by the class."

O

Open/closed principle

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

Once completed, the implementation of a class could only be modified to correct errors; new or changed features would require that a different class be created

L

Liskov substitution principle

"objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program"

In a computer program, if **S** is a subtype of **T**, then objects of type **T** may be replaced with objects of type **S** (i.e., objects of type **S** may be substituted for objects of type **T**) without altering any of the desirable properties of that program (correctness, task performed, etc.)

I

Interface segregation principle

"many client-specific interfaces are better than one general-purpose interface."

Once an interface has become too 'fat' it needs to be split into smaller and more specific interfaces so that any clients of the interface will only know about the methods that relate to them. In a nutshell, no client should be forced to depend on methods it does not use.

D

Dependency inversion principle

"Depend upon Abstractions. Do not depend upon concretions."

A. High-level modules should not depend on low-level modules. Both should depend on abstractions.

B. Abstractions should not depend upon details. Details should depend upon abstractions