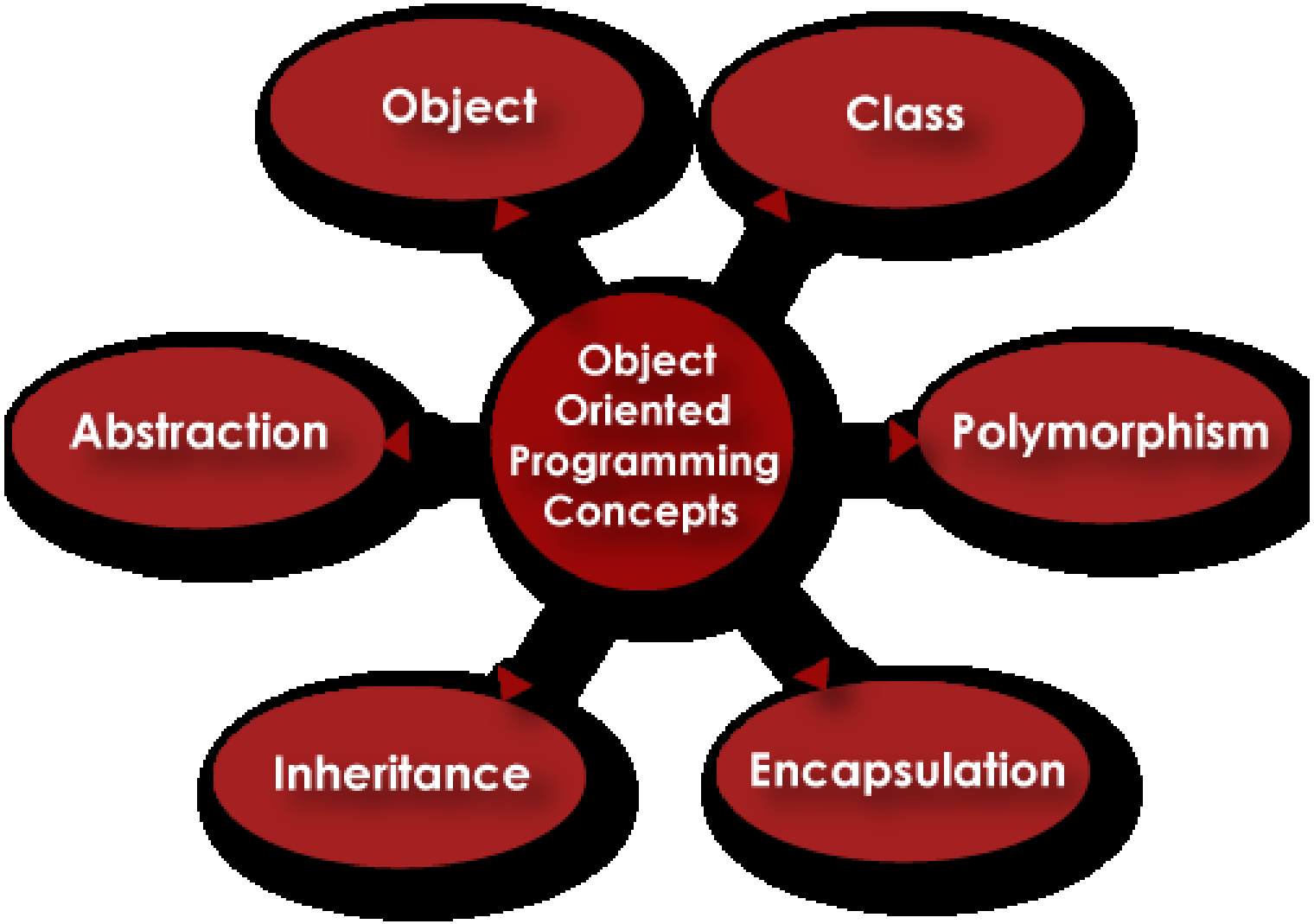
Object-oriented programming

Second semester

Lecture №1

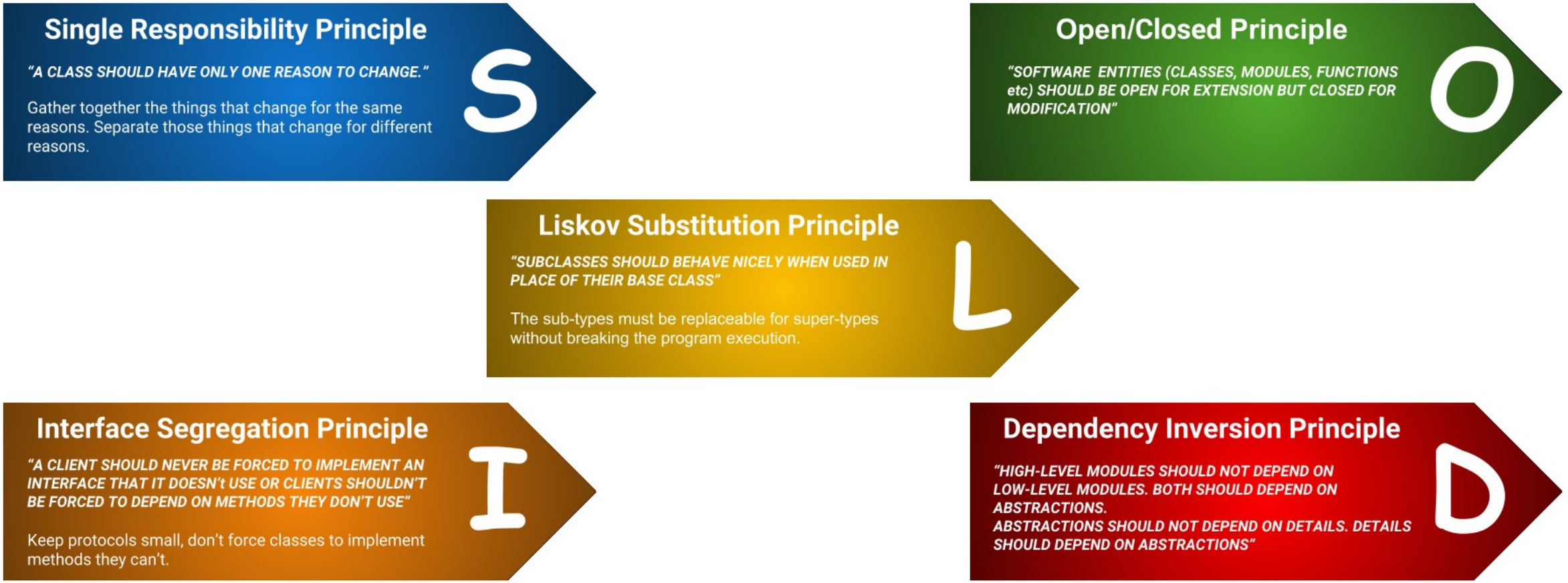
Total Recall, SOLID principles of OOP

PhD, Alexander Vlasov



- Contract
- Modularity
- Typing
- Concurrency
- Persistence

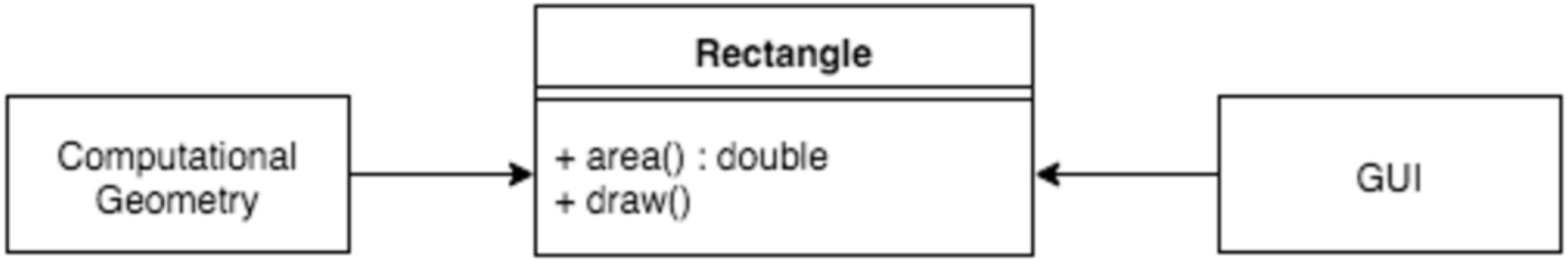
S.O.L.I.D


Single Responsibility Principle
"A CLASS SHOULD HAVE ONLY ONE REASON TO CHANGE."
Gather together the things that change for the same reasons. Separate those things that change for different reasons.
S
Open/Closed Principle
"SOFTWARE ENTITIES (CLASSES, MODULES, FUNCTIONS etc) SHOULD BE OPEN FOR EXTENSION BUT CLOSED FOR MODIFICATION"
O
Liskov Substitution Principle
"SUBCLASSES SHOULD BEHAVE NICELY WHEN USED IN PLACE OF THEIR BASE CLASS"
The sub-types must be replaceable for super-types without breaking the program execution.
L
Interface Segregation Principle
"A CLIENT SHOULD NEVER BE FORCED TO IMPLEMENT AN INTERFACE THAT IT DOESN't USE OR CLIENTS SHOULDN'T BE FORCED TO DEPEND ON METHODS THEY DON'T USE"
Keep protocols small, don't force classes to implement methods they can't.
I
Dependency Inversion Principle
"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"
D

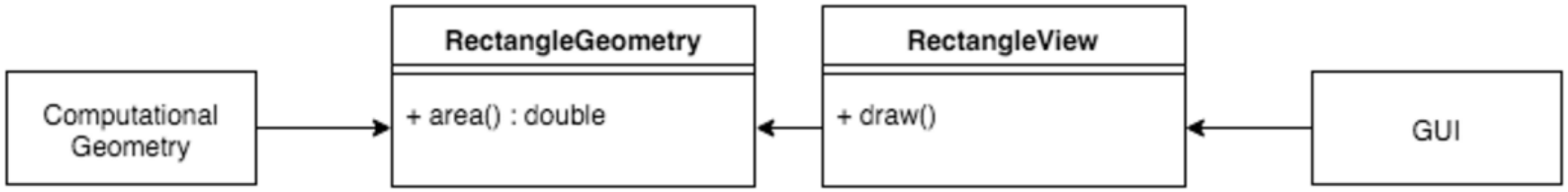
Single Responsibility Principle (SRP)

- The Single Responsibility principle states that every module, class, or function should have responsibility over a single part of the functionality provided by the software, and that responsibility should be entirely encapsulated by the class, module or function.
- **A class should have only one reason to change**
- This means that every class should have a single responsibility or single job or single purpose.

Single Responsibility Principle (SRP)



Wrong approach



SRP approach

Open/Closed Principle (OCP)

- The Open/Closed principle states that you should be able to extend a class behavior, without modifying it, that is we can extend the behavior of software entity without touching the source code of the entity.
- **Software entities … should be open for extension, but closed for modification**
- Open-Closed principle can be achieved using abstraction and inheritance.

Liskov’s Substitution Principle (LSP)

- The Liskov’s Substitution principle ensures that any class that is the child of a parent class should be usable in place of its parent without any unexpected behaviour.
- **Derived or child classes must be substitutable for their base or parent classes**
- Liskov’s notion of a behavioural subtype defines a notion of substitutability for objects; that is, if S is a subtype of T, then objects of type T in a program may be replaced with objects of type S without altering any of the desirable properties of that program.

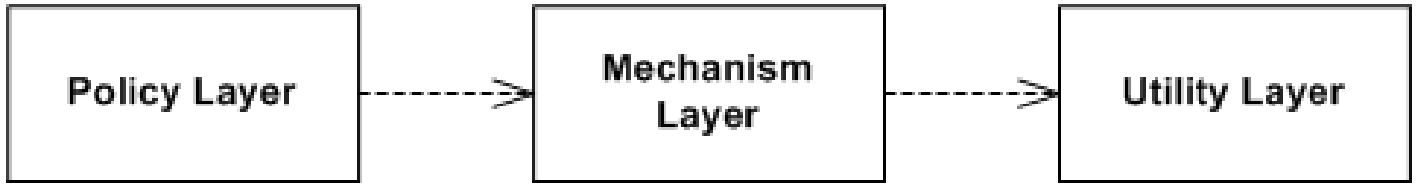
Interface Segregation Principle (ISP)

- The Interface Segregation principle states that no client should be forced to depend on methods it does not use. Interfaces that are very large should be splitted into smaller and more specific ones so that clients will only have to know about the methods that are of interest to them.
- **Objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program**
- You should prefer many client interfaces rather than one general interface and each interface should have a specific responsibility.

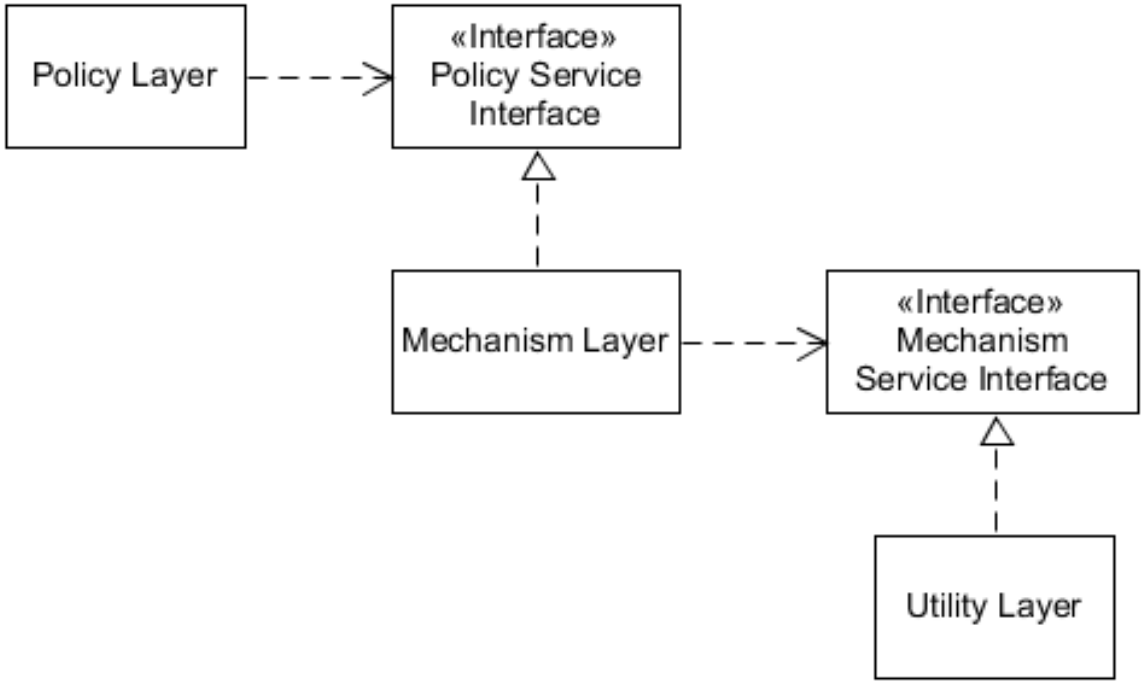
Dependency Inversion Principle (DIP)

- The Dependency Inversion principle states that high-level modules, which provide complex logic, should be easily reusable and unaffected by changes in low-level modules, which provide utility features.
- **High-level modules should not depend on low-level modules. Both should depend on abstractions (e.g. interfaces).**
- **Abstractions should not depend on details. Details (concrete implementations) should depend on abstractions.**
- The main motive of this principle is decoupling the dependencies so if class A changes the class B doesn’t need to care or know about the changes.

Dependency Inversion Principle (DIP)



Wrong approach



DIP approach