# ООП ПРОЕКТИРОВАНИЕ

ДИАГРАММЫ UML

- Принципы SOLID (2 часа)
- ДЗ: Мини проект
- Диаграммы UML (2 часа)
- ДЗ: UML диаграмма мини проекта

# ПРИНЦИПЫ SOLID

| | Название сокр. | Описание принципа |
|---|---|---|
| S | SRP | Принцип единственной обязанности (Single responsibility principle) |
| O | OCP | Принцип открытости/закрытости (Open/closed principle) |
| L | LSP | Принцип подстановки Барбары Лисков (Liskov substitution principle) |
| I | ISP | Принцип разделения интерфейса (Interface segregation principle) |
| D | DIP | Принцип инверсии зависимостей (Dependency inversion principle) |

# ПРИНЦИП ЕДИНСТВЕННОЙ ОБЯЗАННОСТИ (SINGLE RESPONSIBILITY PRINCIPLE)

- На каждый объект должна быть возложена одна единственная обязанность. Обязанность - это набор методов, служащих одному действующему лицу. Для обязанности действующее лицо - единственный источник изменений (Роберт Мартин)

# ПРИНЦИП ОТКРЫТОСТИ/ЗАКРЫТОСТИ (OPEN-CLOSED PRINCIPLE)

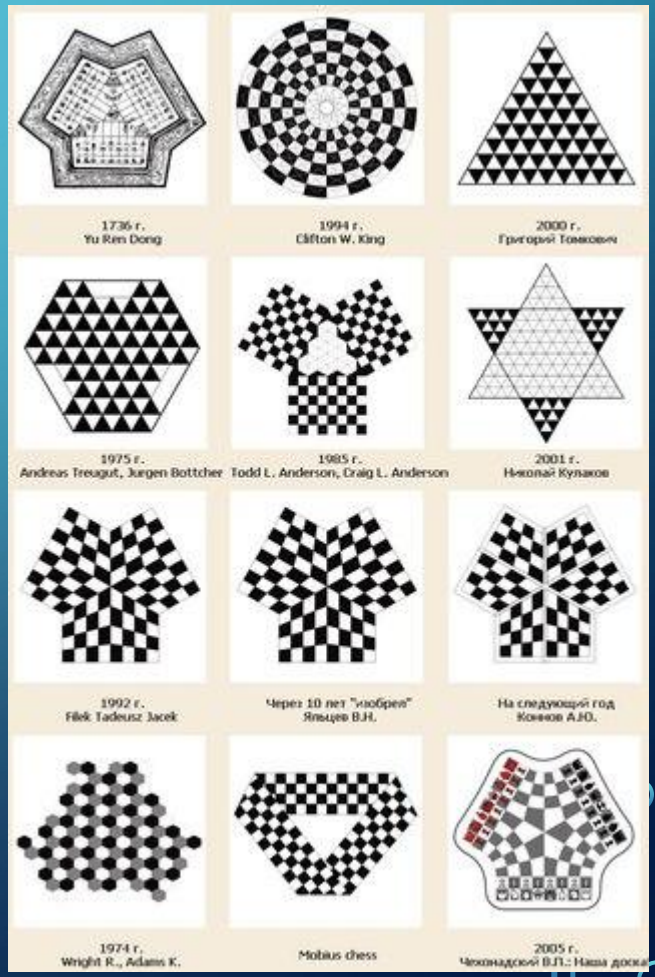
- Программные сущности (классы, модули, функции и т.п.) должны быть открыты для расширения, но закрыты для изменения. Достоинством применения такого подхода следующие: ● Не нужно пересматривать уже существующий код, не нужно менять уже готовые для него тесты при доработке проекта. ● Если нужно ввести какую-то дополнительную функциональность, то это не должно коснуться уже существующих классов или как-либо иначе повредить уже существующую функциональность.

# ПРИНЦИП ПОДСТАНОВКИ БАРБАРЫ ЛИСКОВ (LISKOV SUBSTITUTION PRINCIPLE, КРАТКО - LSP)

- Объекты в программе могут быть заменены их наследниками без изменения свойств программы. Иными словами поведение наследуемых классов не должно противоречить поведению, заданному базовым классом, то есть поведение наследуемых классов должно быть ожидаемым для кода, использующего переменную базового типа.
- *Классический пример: класс прямоугольник и его наследник квадрат*

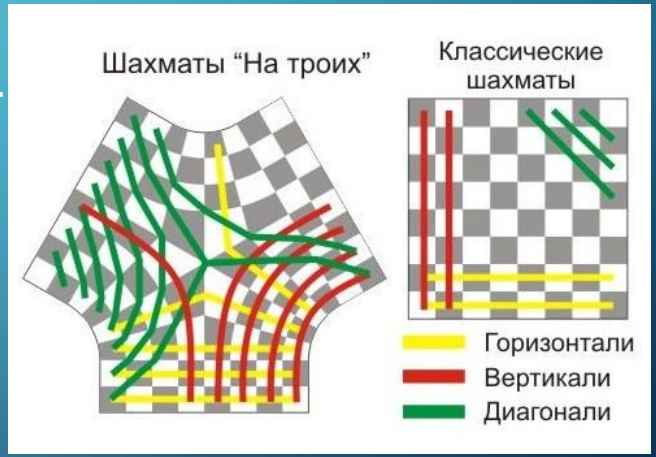
# ПРИНЦИП РАЗДЕЛЕНИЯ ИНТЕРФЕЙСА (INTERFACE SEGREGATION PRINCIPLE)

- Много специализированных интерфейсов лучше, чем один универсальный.
- Принцип разделения интерфейсов состоит в том, что слишком универсальные интерфейсы необходимо разделять на более маленькие и специфические, чтобы при использовании маленьких интерфейсов потребителю необходимо было бы реализовать только методы, необходимы им в работе.



# ПРИНЦИП ИНВЕРСИИ ЗАВИСИМОСТЕЙ (DEPENDENCY INVERSION PRINCIPLE)

- Зависимости внутри системы строятся на основе абстракций. Модули верхнего уровня не зависят от модулей нижнего уровня. Абстракции не должны зависеть от деталей. Детали должны зависеть от абстракций. Иными словами, следует разрабатывать ПО таким образом, чтобы различные модули были автономными, и соединялись друг с другом с помощью абстракции.



# РАЗБОР ПРИМЕРА ПРОЕКТИРОВАНИЯ КЛАССА ДРОБЬ.