Liskov Substitution Principle

Лисков, Барбара

Материал из Википедии — свободной энциклопедии

[править | править код]

В Википедии есть статьи о других людях с фамилией Лисков.

Барбара Лисков (англ. *Barbara Liskov*, урождённая *Барбара Джейн Губерман — Barbara Jane Huberman*; род. 7 ноября 1939) — американский учёный в области информатики, исследователь проблемы абстракции данных, руководитель группы разработки языка программирования Клу, лауреат премии Тьюринга 2008 года.

Член Национальной инженерной академии США (1988)[1], Национальной академии наук США (2012)[2].

Содержание [скрыть]

1 Биография
2 Награды
3 Библиография
4 Примечания
5 Ссылки

Биография [править | править код]

Родилась в Калифорнии, где поселились её бабушка и дедушка по отцовской линии — эмигранты из Российской империи Лев Губерман и Роза Марголис. Получила степень бакалавра по математике в Калифорнийском университете в Беркли в 1961 году, после чего продолжила обучение в Стэнфорде, где в 1968 году стала первой женщиной в США, получившей степень доктора по информатике с диссертацией о программной реализации игры в шахматный эндшпиль (*A program to play chess endgames*).

С 1972 года работает и преподаёт в Массачусетском технологическом институте.

Руководила разработкой таких языков программирования как Клу и Argus в 1970-х и 1980-х годах, а также объектно-ориентированной системы управления базами данных Thor. Вместе с Дженнет Уинг разработала в 1987 году принцип подстановки — концепцию определения подтипа. Возглавляет группу по методологии программирования в Массачусетском технологическом институте, в настоящее время уделяя особое внимание BFT-отказоустойчивости и распределенным вычислениям.

Действительный член Американской академии наук и искусств и Ассоциации вычислительной техники.

Лисков всегда поощряла студенток, оказывала им поддержку, уделяет много внимания тому, чтобы сделать информатику более дружелюбной областью, в особенности для женщин. Привлекая к работе больше женщин и младший профессорско-преподавательский состав, Лисков помогает им в построении карьеры и дальнейшем продвижении. Сегодня Массачусетский технологический институт значительно отличается от того места, где она начала свою карьеру в начале 1970-х. Тогда на факультете работала лишь небольшая группа сотрудников женского пола[3].

Муж — Натан Лисков (*Nathan Arthur Liskov*, поженились в 1970 году) и сын Мозес Лисков (*Moses Liskov*, 1975) — также учёные в области информатики.

Награды [править | править код]

- 2004 — медаль Джона фон Неймана

Барбара Лисков

Barbara Jane Liskov

Дата рождения	7 ноября 1939 (81 год)
Место рождения	Лос-Анджелес, Калифорния, США
Страна	США
Научная сфера	информатика
Место работы	Массачусетский технологический институт
Альма-матер	Калифорнийский университет в Беркли, Стэнфордский университет
Учёная степень	доктор
Научный руководитель	Джон Маккарти
Известна как	создатель Клу, исследователь абстракции данных
Награды и премии	Премия Тьюринга
Сайт	pmg.csail.mit.edu/~lisko…

Медиафайлы на Викискладе

Принцип подстановки Барбары Лисков

Liskov Substitution Principle (LSP)

Функции, работающие с базовым классом, должны иметь возможность работать с подклассами не зная об этом.

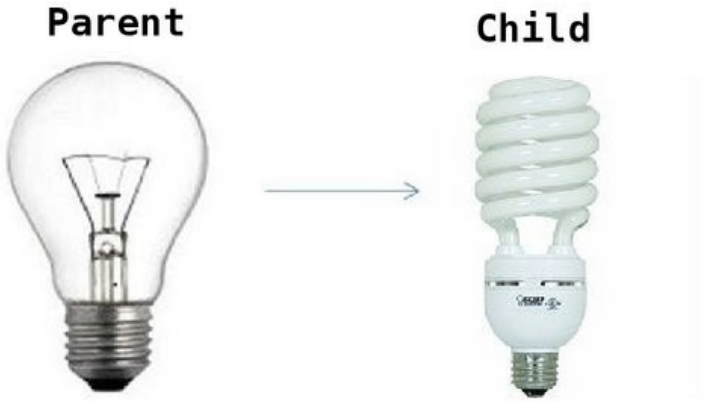
Этот принцип является важнейшим критерием при построении иерархий наследования.

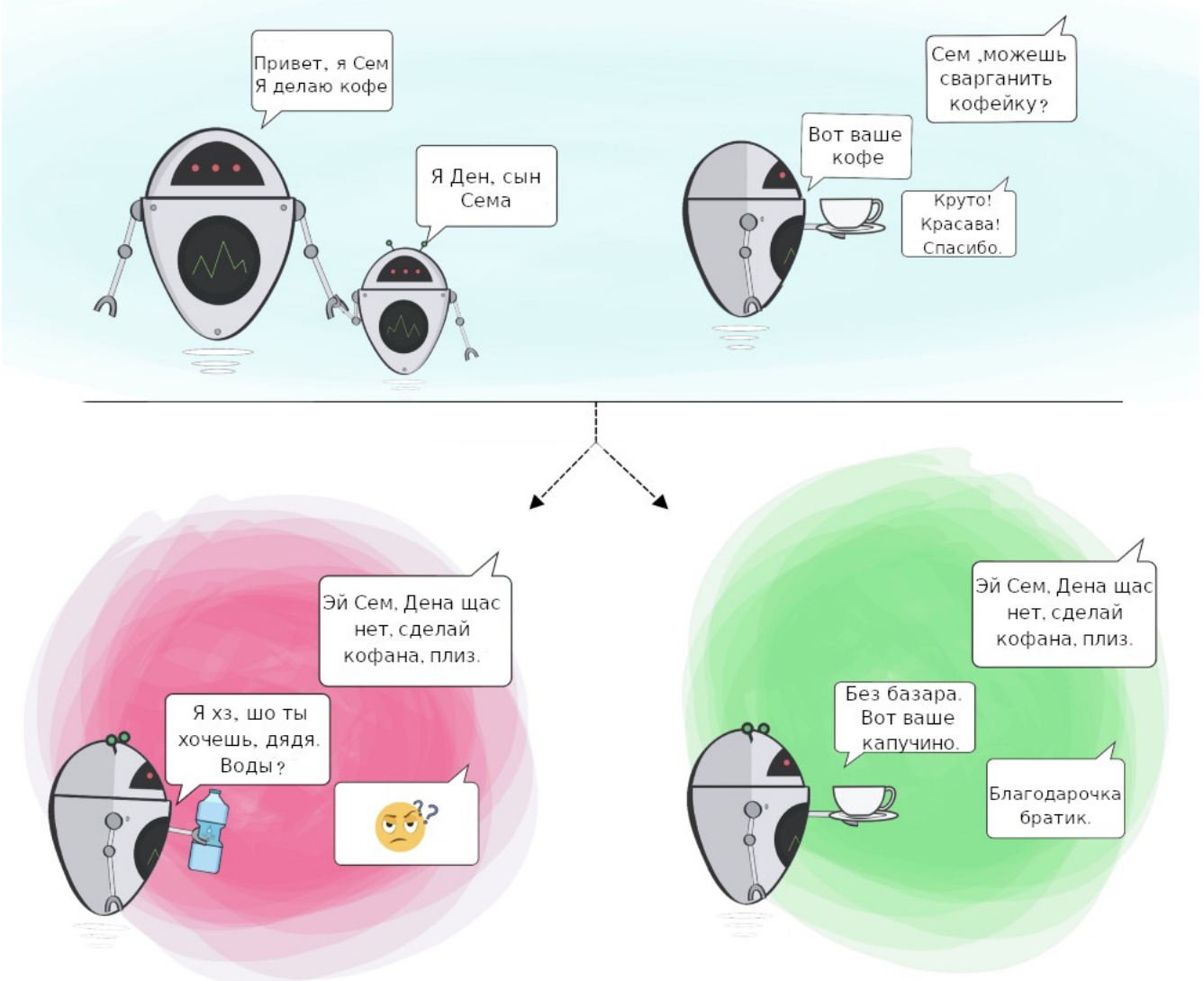
Другие формулировки

- Поведение наследуемых классов не должно противоречить поведению, заданному базовым классом.
- Подкласс не должен требовать от вызывающего кода больше, чем базовый класс, и не должен предоставлять вызывающему коду меньше, чем базовый класс

LSP – принцип подстановки Барбары Лисков

Смысл LSP: «вы должны иметь возможность использовать любой производный класс вместо родительского класса и вести себя с ним таким же образом без внесения изменений».





Liskov Substitution

Принцип подстановки Барбары Лисков

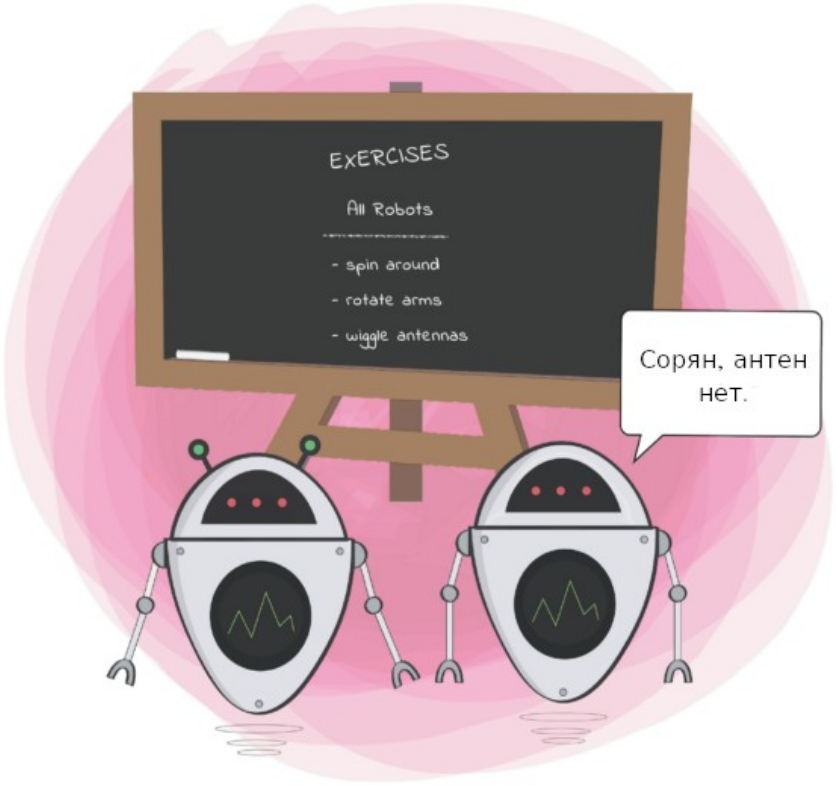
Interface segregation principle

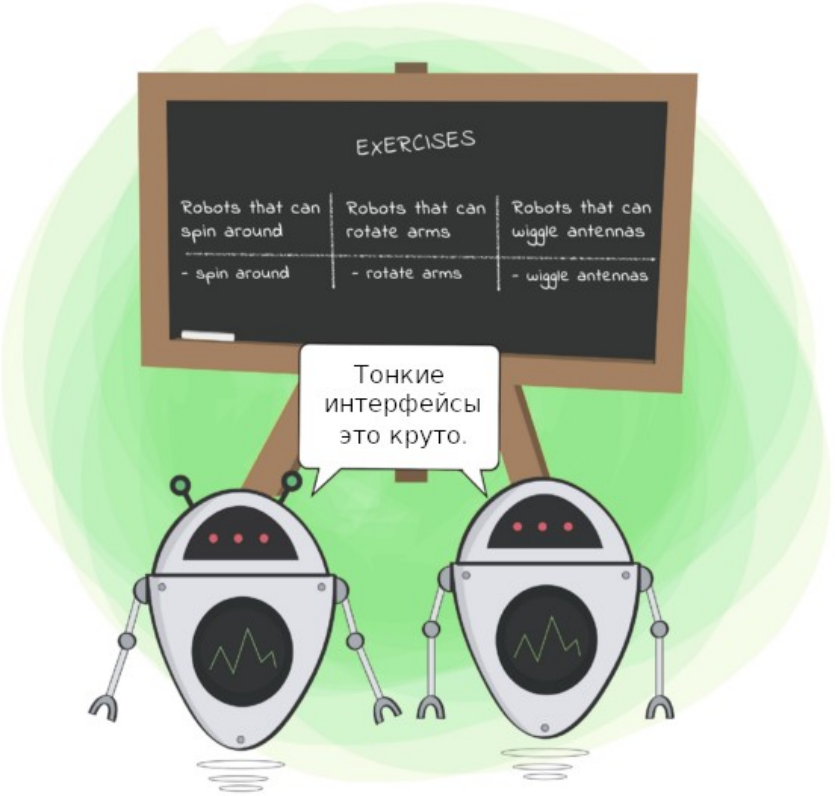
- Принцип разделения интерфейсов
  - Клиенты не должны зависеть от методов, которые они не используют
  - Интерфейсы должны быть сфокусированными
- Большие интерфейсы должны разделяться на более мелкие и узкоспециальные
- Такие интерфейсы (полностью абстрактные типы) скорее имеют роль тегов, чем организуют свою иерархию наследования

Interface segregation

Принцип разделения интерфейса

- Слишком «толстые» интерфейсы необходимо разделять на более маленькие и специфические, чтобы клиенты маленьких интерфейсов знали только о методах, которые необходимы им в работе.





Interface Segregation

Принцип Сегрегации интерфейсов

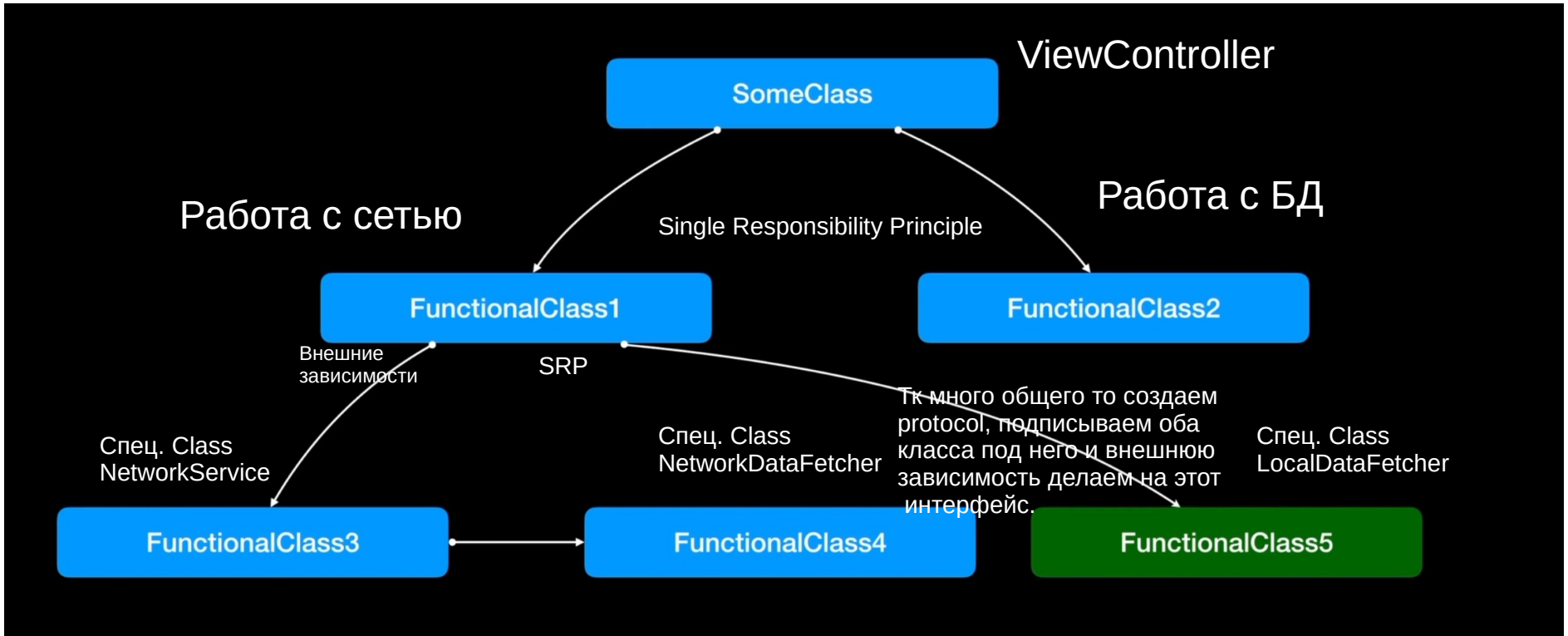
Dependency Inversion Principle

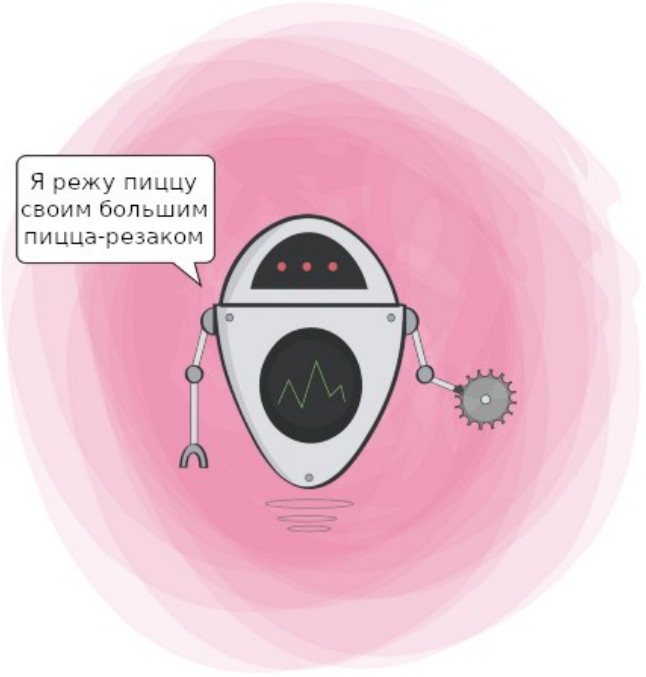
Принцип инверсии зависимости

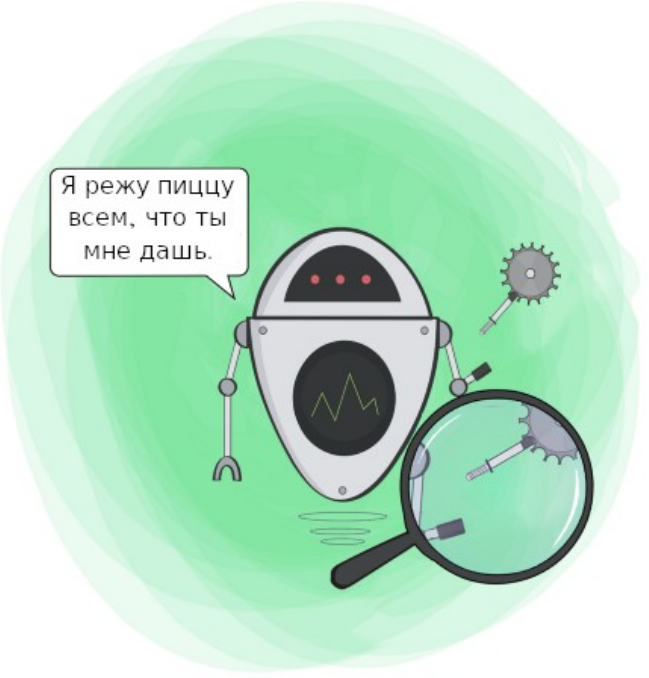
- Модули верхнего уровня не должны зависеть от модулей нижнего уровня. И те, и другие должны зависеть от абстракций.
- Абстракции не должны зависеть от деталей. Детали должны зависеть от абстракций.

Dependency Inversion Principle

С абстракциями мы уже работали (Open close principle).
В OCP мы делали универсальные ф-ции.
DIP отвечает за взаимодействие классов.







Dependency Inversion

Принцип инверсии зависимостей

Summarize

Не переусердствуйте с SOLID принципами

- SOLID принципы это лишь принципы, не правила
- Всегда руководствуйтесь здравым смыслом когда применяете SOLID
- Избегайте излишних зависимостей в вашем коде при использовании SRP и SOLID
- Ваша цель упростить ваш код для понимания, а не достичь 100% реализации SOLID

Summarize

Самые финальные мысли

- Главная цель SOLID принципов сделать ваш код более читабельным и легко поддерживаемым
- Позволяет вам тратить больше времени на написание кода, чем на его чтение
- SOLID принципы это всего лишь принципы, не правила
- Не спешите использовать солид принципы всегда и везде, сначала оцените ситуацию здравым смыслом
- SOLID принципы это ваш инструмент, а не цель