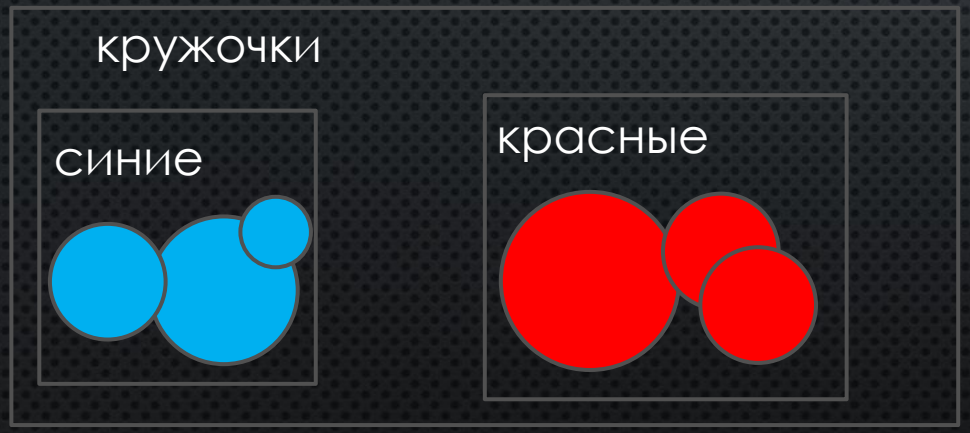
ООП

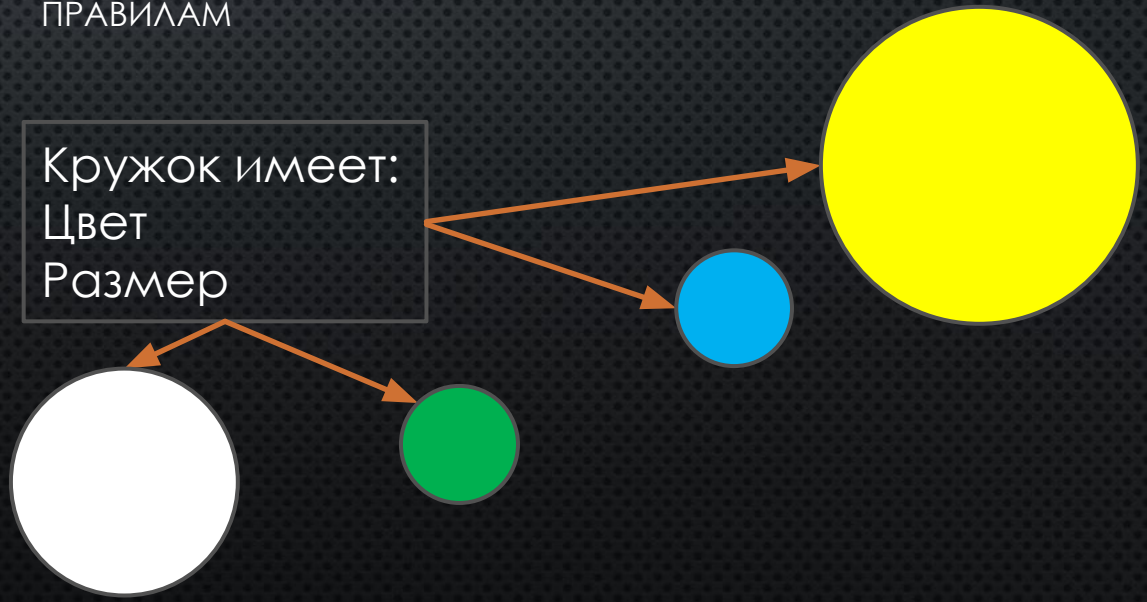
ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Вспоминая первую лекцию

Люди разделяют на классы уже существующие объекты, находят общие свойства у объектов, и дают этой группе похожих объектов имя.



В ООП процесс обратный. Сначала определяется некий класс, описывающий некоторый объект, определяющий свойства и поведение объекта, а уже потом создаются новые объекты, работающие по уже заданным правилам



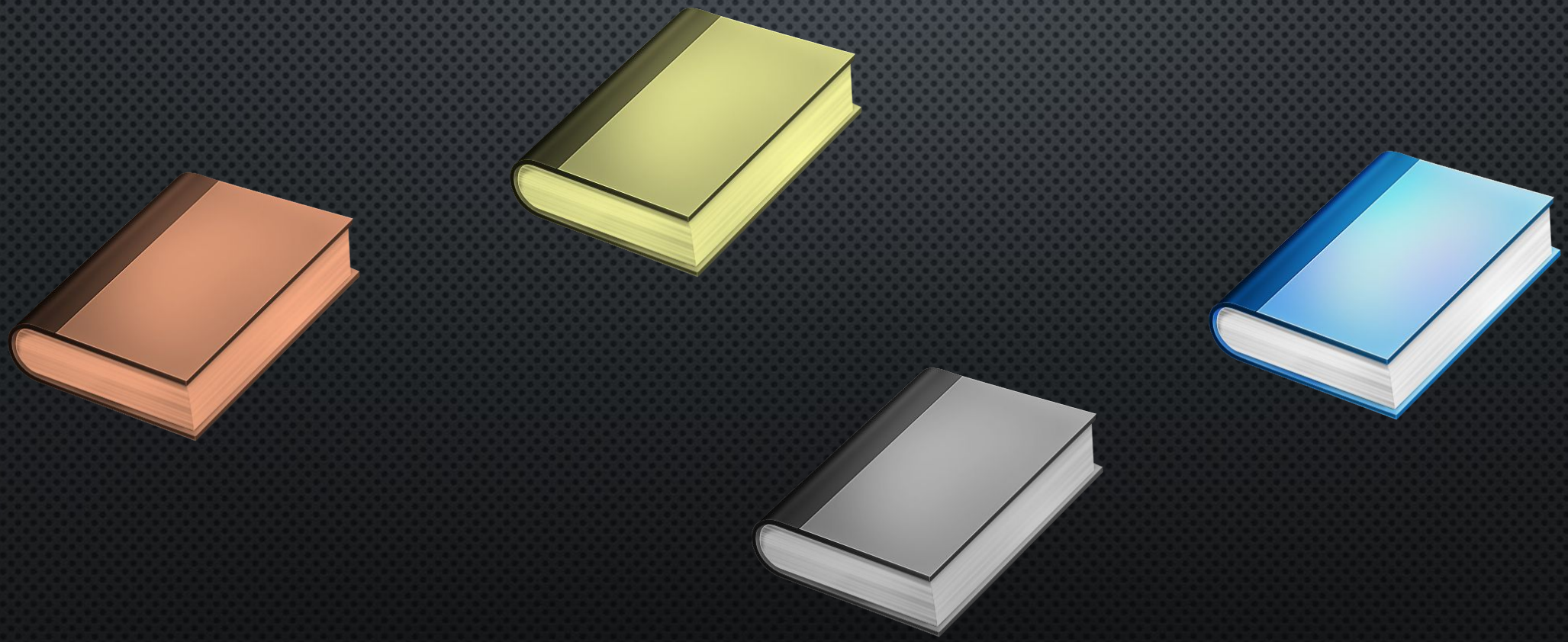
Простые понятия

Объект - Любое существующее нечто, с чем мы можем как-то взаимодействовать. Что угодно. Вообще что угодно. Ну совсем.

Класс – описание некоторого семейства объектов, правила работы этого объекта, его внутреннее устройство, чертеж,

УМНОЕ ХРАНИЛИЩЕ КНИГ

КНИГА – ЭТО ОБЪЕКТ, ОНА СУЩЕСТВУЕТ



Хранилище книг – тоже объект

Само хранилище же существует, и имеет внутри себя много книг, других объектов, на минуточку

Но программа понятия не имеет ни про книгу, ни про хранилище тем более

- Надо объяснить, что такое книга и что такое хранилище.

Класс как раз и определяет модель книги

У нашей книги есть несколько параметров

1. Название
2. Имя автора
3. Сам текст, который содержится

Собственно, это исчерпывающая информация для определения книги в нашей задаче. Мы опустим тип переплета, обложку, форматирование текста на страницах и прочее

Теперь мы можем создать новую книгу с тремя параметрами

Класс хранилища

Хранилище должно содержать в себе

1. Список книг

Оно должно уметь

1. Добавлять новые книги в хранилище
2. Удалять определенную книгу
3. Произвести выборку по какому-то критерию

А как мы можем узнать, что находится в книге?

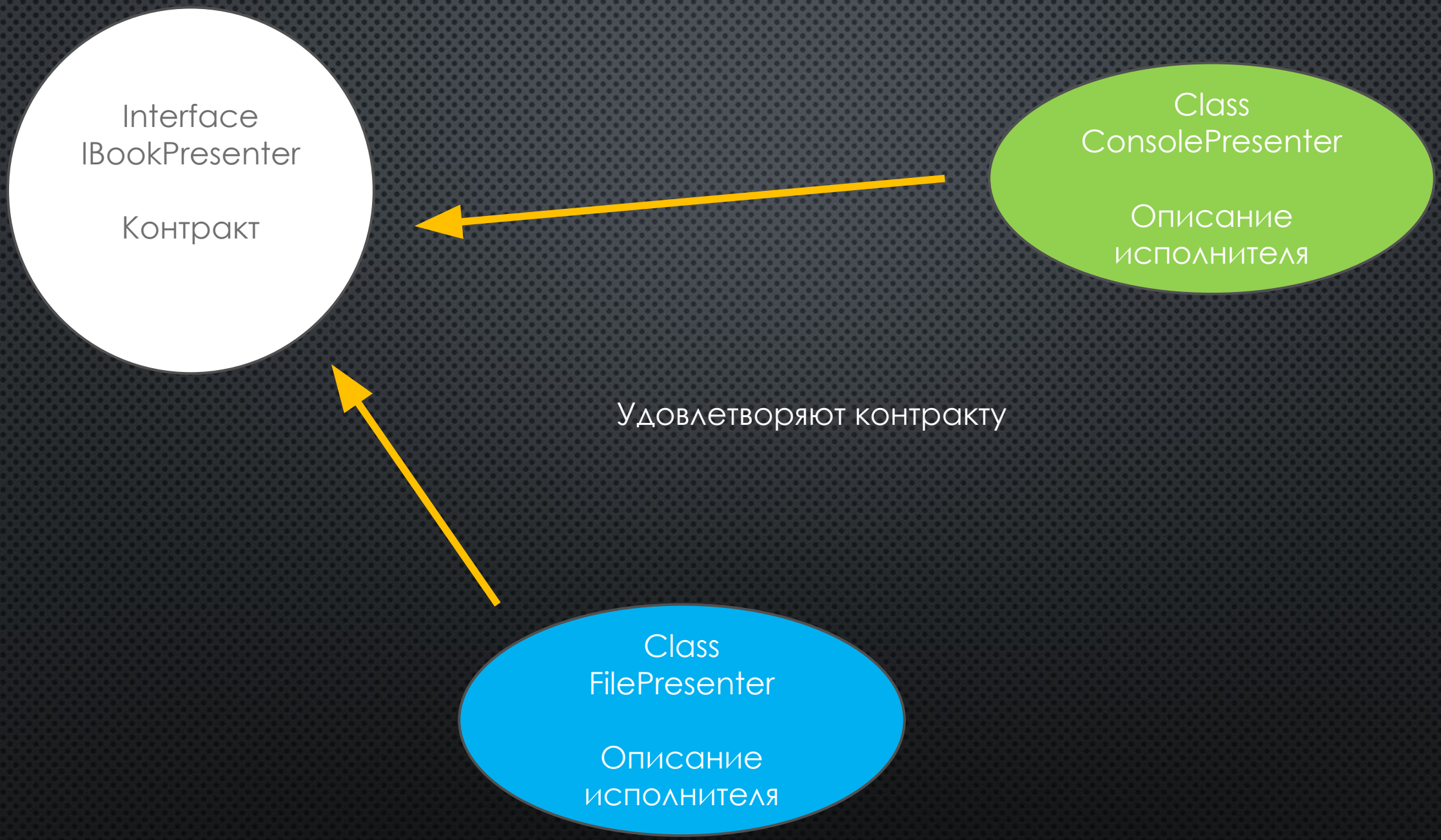
- Вывести на консоль
- Отправить информацию по почте
- Сохранить информацию о книге в файл
- Отобразить информацию на WEB странице

- И нам необходим такой механизм, при помощи которого способ представления изменить можно очень легко и просто.

Решение – интерфейс(interface)

Наш интерфейс должен только говорить, что такой-то объект умеет представлять книгу. Не более.

- Интерфейс лишь описывает способ и правила взаимодействия с некоторым объектом, но никак не конкретные действия, которые нужно совершить.


Interface
IBookPresenter
Контракт
Class
ConsolePresenter
Описание
исполнителя
Удовлетворяют контракту
Class
FilePresenter
Описание
исполнителя

А откуда у нас возьмутся новые книги?

Создаем интерфейс IBooksFabrik

Он будет иметь так же всего одно действие, а именно GetNewBook, которое будет возвращать новый экземпляр книги

Имея интерфейс, мы сможем без труда реализовать множество разных источников книг. Будут их писать нам коллеги по интернету, присылая на почту, вводим ли мы руками в консоль всю информацию – не важно. Реализация действия может быть любая, а действие – получить новую книгу – уже описано, и все знают, что его можно вызвать и не думать про реализацию

Мы попробуем два способа получения новой книги

1. Ввод своими руками в консоль.
2. Рандомная генерация книги из изначально заданного набора возможных слов.

КОДИМ

А теперь мы хотим чтобы наши книги не пропадали при закрытии программы!

Но как-бы поведение, когда книги хранятся у нас в оперативе тоже нужно…

Абстрактный класс!!!

Абстрактный класс это такой класс, такое описание объекта(сущности) который не знает, как конкретно должен делать некоторые свои действия

То есть он имеет в себе логику, реализацию некоторых действий, но некоторые специфические действия он оставляет на доработку своим последователям

Получится такая история

Абстрактный класс BookStorage имеет в себе логику выборки по некоторому критерию, И логику отображения книг с использованием IPResenter

И он имеет абстрактные методы для добавления удаления и получения книг.

А уже конкретные классы RuntimeStorage и InFileStorage будут реализовывать методы хранения книг.

КОДИМ