SOLID

S single responsibility
Классы соответсвует 1 набору функций
(Нож должен резать)))
O open-closed
Открыты для расширения (доступ через публичные методы) get-set
Закрыты для модификации (поля приватные)
L liskov substitution
Если в родителе определено какое то поведение объектов то переопределение этого поведения в наследнике не должно руинить определение родителя
I interface segregation
1 интерфейс = 1 назначение
100500 интерфейсов с одним назначением - тож плохо
D dependency inversion
Автомобиль зависит от колеса
Автомобиль должен позаботиться о том чтобы у него было колесо🙃

Класс сам должен заботится о наличии объектов, в котором он нуждается

X-классы

Вложенные классы (non-static)

Когда в классе объявлен другой класс
С их объектами нужно работать из вне
Car.Wheel wheel = car.new Wheel();
Как сломать машину из колеса?
Car.this.crash();

Внутренние классы (static)

Самостоятельный тип данных закреплённый за этим классом
Доступ через класс а не через объект
Car.Wheel wheel = new Car.Wheel();
Создание статического колеса

Локальные классы

Область видимости класса ограничена блоком кода в который его впихнули

- модификатор доступа не указывается
- Статические методы нельзя
- Статические константы можна
- Внешние локальные переменные (но только effectively final (переменная или final) можна
- Не могут быть статичными

Анонимные классы

Если лень создавать класс который реализует интерфейс
start (new Runnable(){
//code
});

- любой анонимный класс преобразуется в локальный (ну жава преобразует)
- Не обязательно интерфейс: вместо него можно написать класс и на лету переопределить его

Исключения

При ошибках жава порождает объекты-исключения
А мы можем их обрабатывать и создавать свои
Обработка
try{
//code
} catch (PszhException e){
//обработка
} finally {
//обрабатывается всегда
}

- catch может быть несколько при чем нужно прописывать сначала более общие

Классификация исключений

Все наследуется от throwable
-Error (unchecked)
-Exception (checked)

- RunTimeException (unchecked)
- Other Exception (Checked)

Создание своих исключений
Class PszhException extends Exception

PszhExeption PSZH = new PszhException();
throw PSZH;

throw null кинет NullPointerException

- Если исключение checked, то в методе надо писать throws Exception и мы обязаны обернуть потом метод в try catch
- обработаны ли Checked Exception-ы проверяются компилятором
- Unchecked Exception-ы будут скомпилированы в любом случае, компилятору плевать: обработаны они или нет
- Error - исключение системной ошибки и ее вызов это смэрц

Multiple catch
catch (Ex 1 | Ex 2 ex) - исключения разных ветвей наследования (один из них - не предок второго)

Функци Анальное Программирование

Лямбда выражения

```
for (Person p: persons){
    printPerson(p, pn-> pn.getAge() > 18 && pn.IsCool());
//p - объект person
//pn - тоже объект person и для него мы пишем функцию
//трэш...
}
```

Ещё преколы

```
for (Person p: persons){
    printPerson(p, Main::implCheck);
//в качестве аргумента мы кинули ссылку на метод
//Main - имя класса
//implCheck - метод
}
```