Java

Занятие 2. Java и ООП.

План занятия

- Классы в Java
- Наследование
- Инкапсуляция
- Полиморфизм
- Абстракция
- Потоки ввода/вывода

Классы в Java

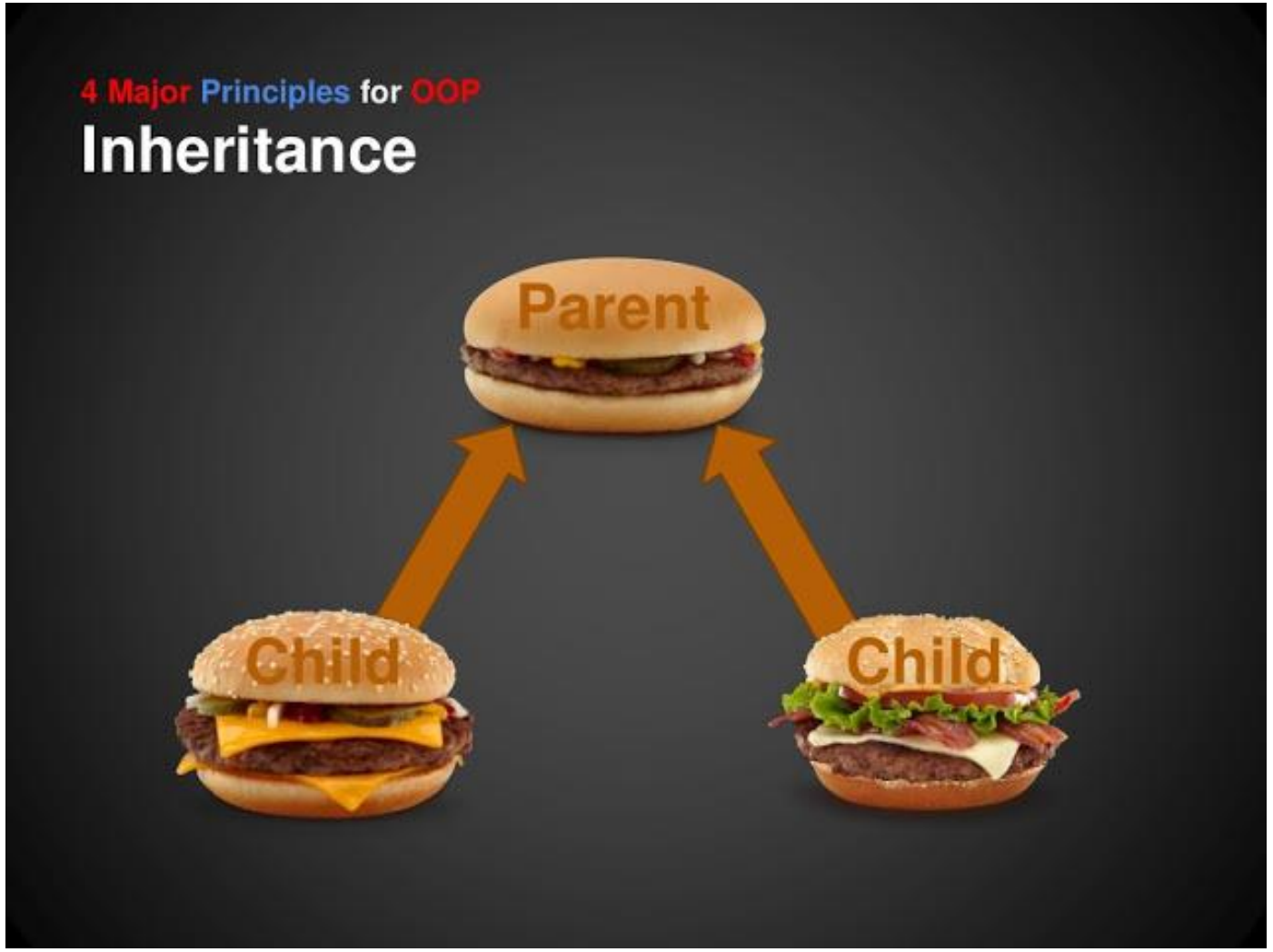
- Класс — объект реального мира
- Класс описывает свойства объекта (поля) и действия над ним (методы)
- Объект — конкретный экземпляр класса. То есть набор свойств общий, а значения у каждого свое

static

- Модификатор static в Java напрямую связан с классом, если поле статично, значит оно принадлежит классу, если метод статичный, аналогично — он принадлежит классу. Исходя из этого, можно обращаться к статическому методу или полю используя имя класса. Например, если поле count статично в классе Counter, значит, вы можете обратиться к переменной запросом вида: Counter.count.


Абстракция
Инкапсуляция
ООП
Наследование
Полиморфизм

Наследование



Наследование

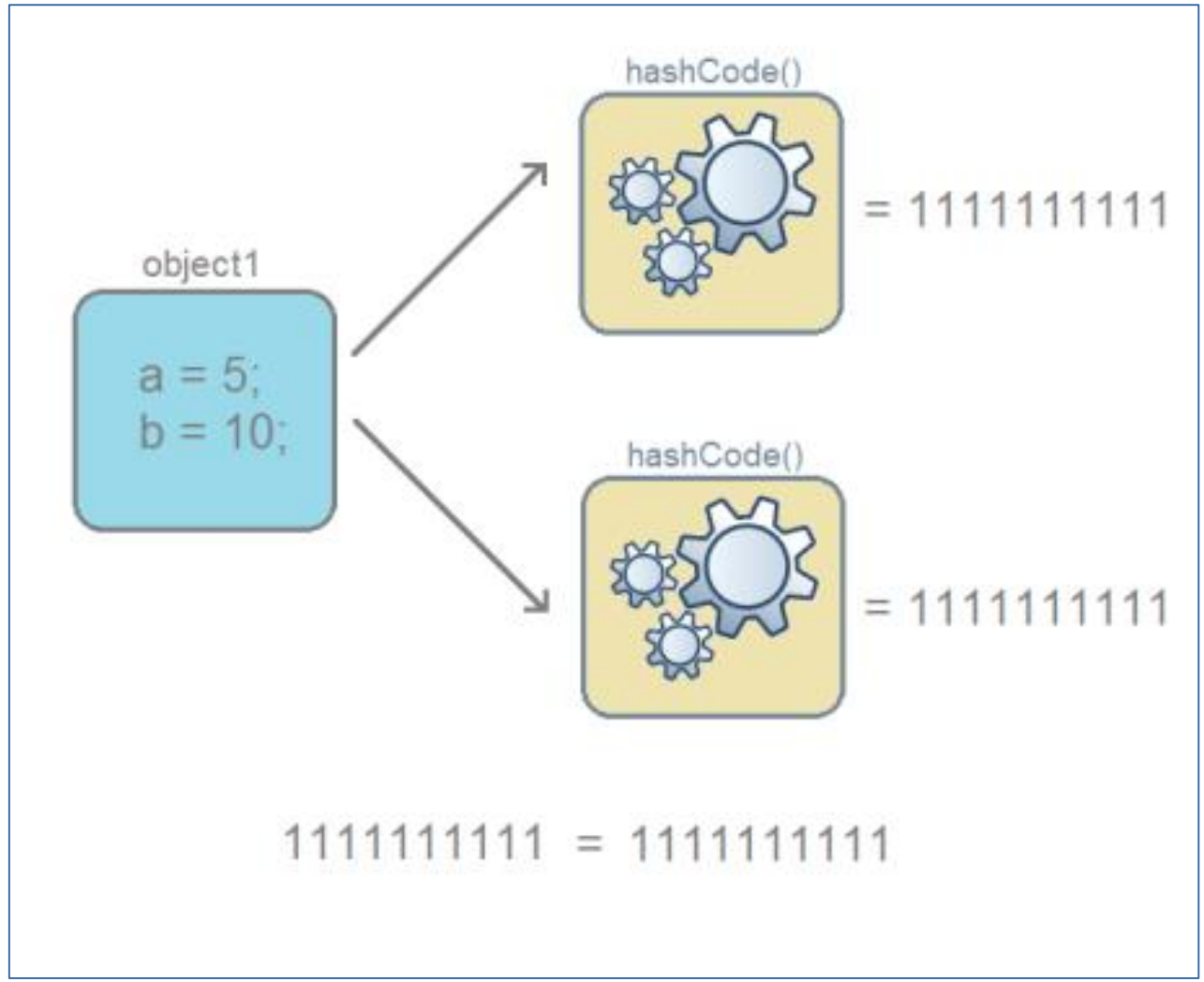
- Переиспользование части кода
- Выделил общее поведение
- Класс-потомок можно использовать вместо класса-родителя в коде
- Класс-потомок может переопределить поведение родителя
- Нет множественного наследования

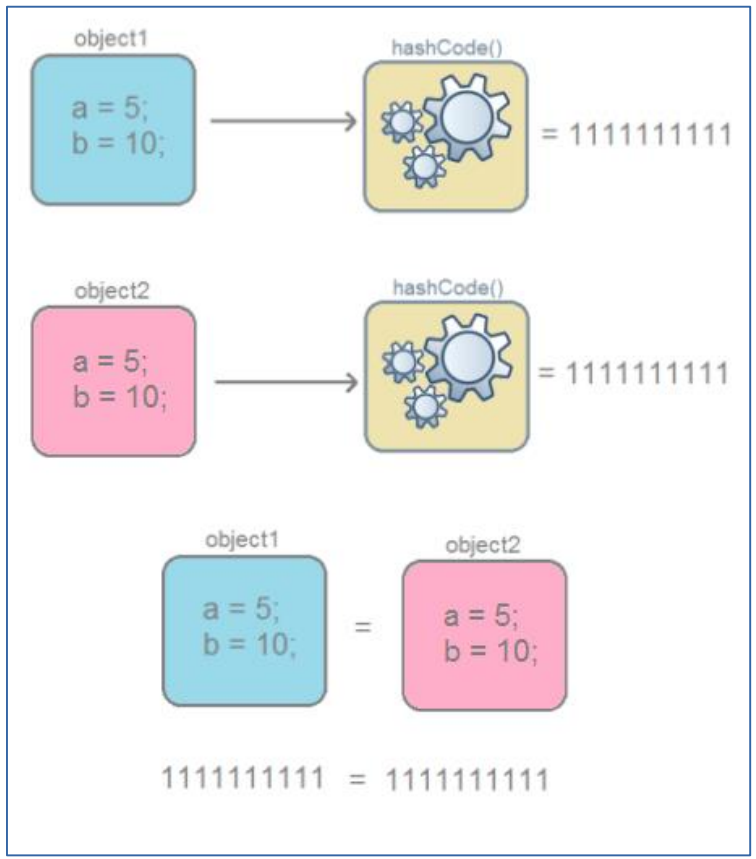
Object

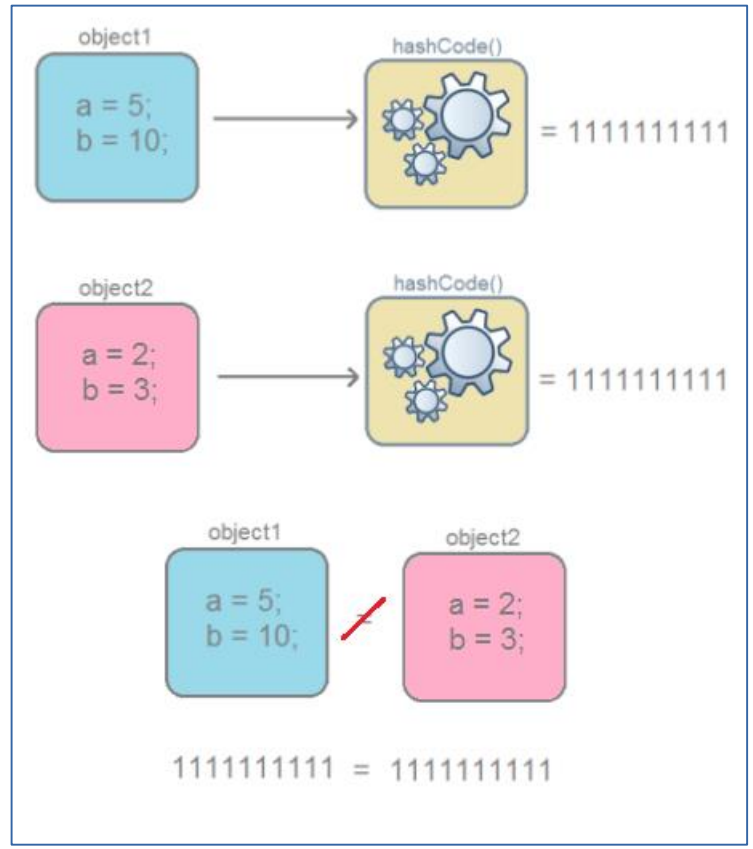
-
- В java всё - объект.
- Все объекты неявно наследуются от
- java.lang.Object

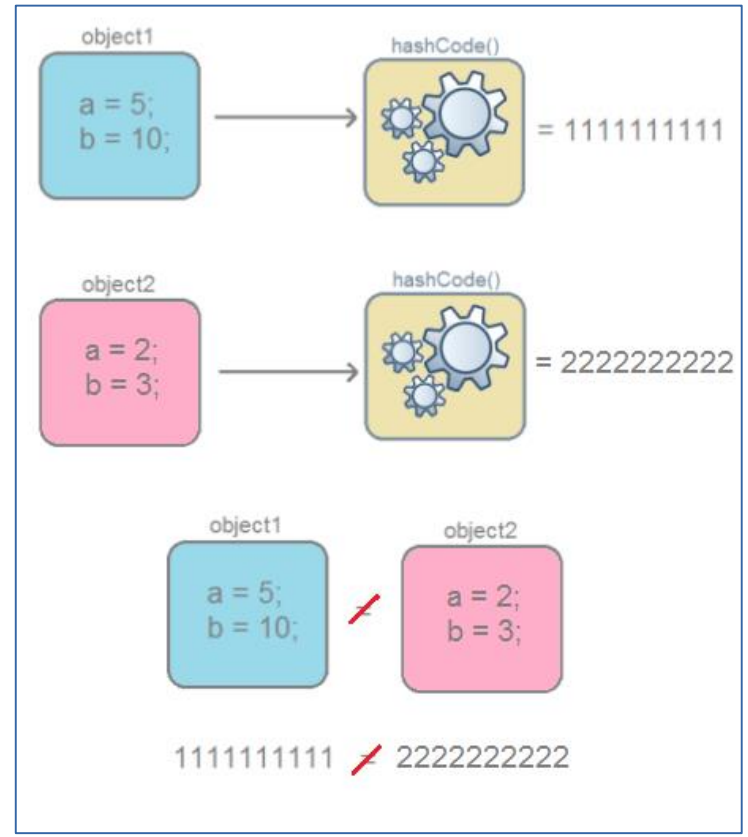
hashCode

- Если очень просто, то хеш-код — это число. Если более точно, то это битовая строка фиксированной длины, полученная из массива произвольной длины (Объект)
-
- В итоге, в терминах Java, хеш-код — это целочисленный результат работы метода, которому в качестве входного параметра передан объект.

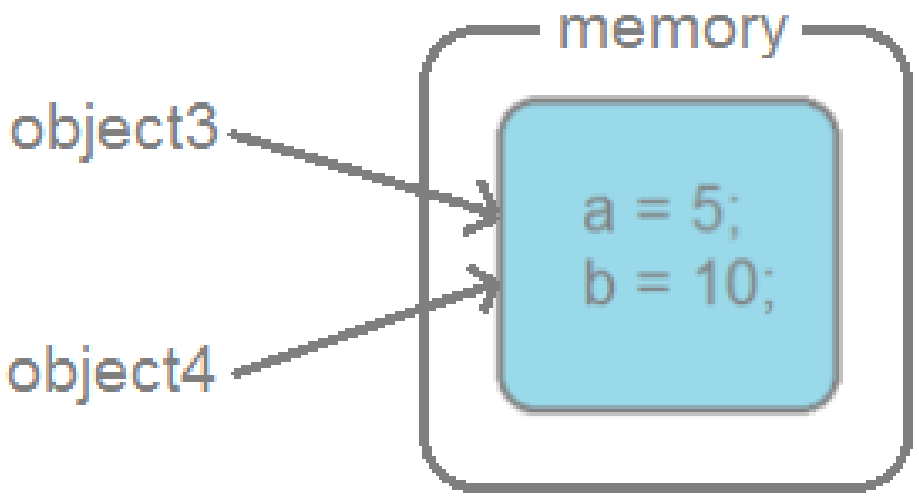

hashCode()
= 1111111111
object1
a = 5;
b = 10;
hashCode()
= 1111111111
1111111111 = 1111111111


object1
a = 5;
b = 10;
hashCode()
= 1111111111
object2
a = 5;
b = 10;
hashCode()
= 1111111111
object1
a = 5;
b = 10;
=
object2
a = 5;
b = 10;
1111111111 = 1111111111

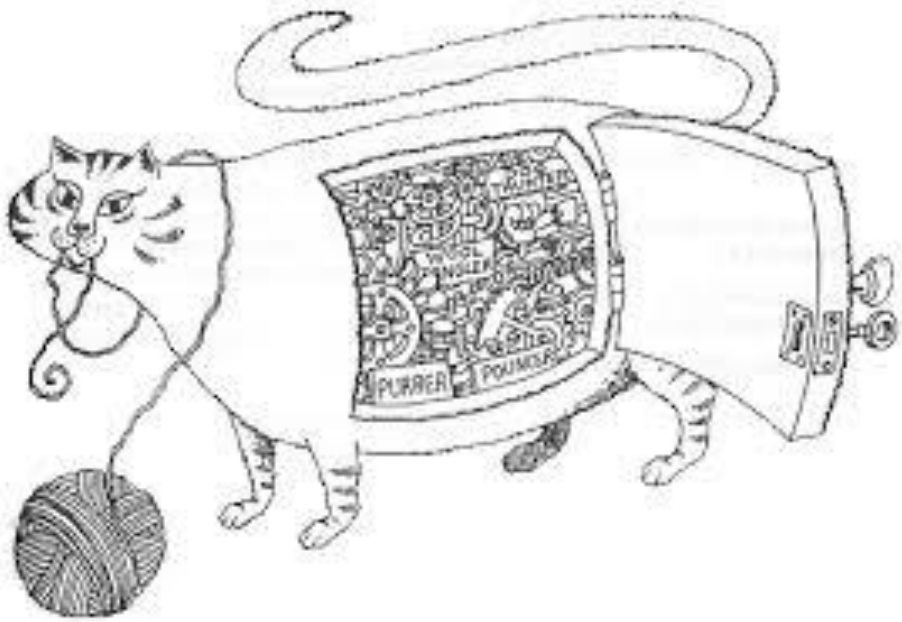

object1
a = 5;
b = 10;
hashCode()
= 1111111111
object2
a = 2;
b = 3;
hashCode()
= 1111111111
object1
a = 5;
b = 10;
≠
object2
a = 2;
b = 3;
1111111111 = 1111111111


object1
a = 5;
b = 10;
hashCode()
= 1111111111
object2
a = 2;
b = 3;
hashCode()
= 2222222222
object1
a = 5;
b = 10;
≠
object2
a = 2;
b = 3;
1111111111 ≠ 2222222222

equals



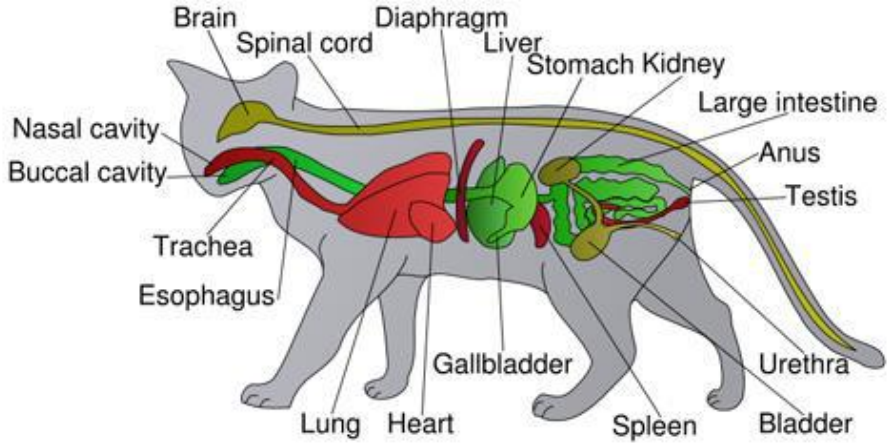
Инкапсуляция



Инкапсуляция

понятно, что делать с объектом
(кормить, что же еще)

не понятно, как именно взаимодействовать с объектом
(слишком много деталей)



Инкапсуляция

- Контроль доступа — это поля можно только читать, или его вообще нельзя видеть
- Контроль целостности — это внутреннее поле класса, только владелец знает, как его корректно менять
- Возможность изменения реализации — например, в методах get/set можно менять логику

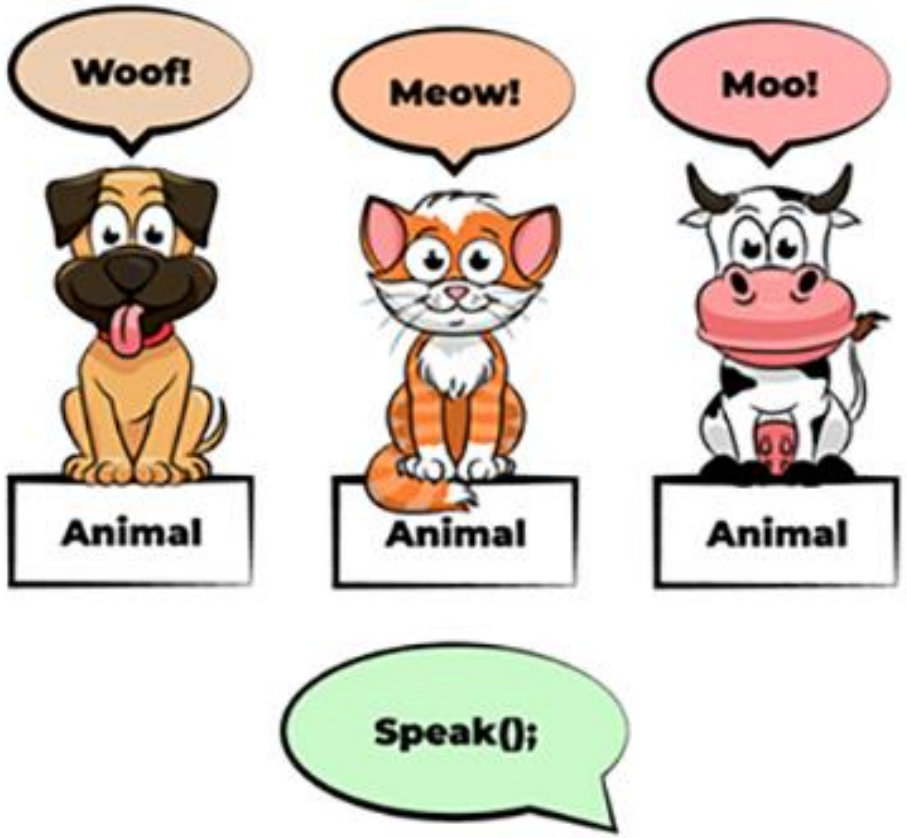
Инкапсуляция

- Контроль доступа
- Контроль целостности
- Возможность изменения реализации
- Контракт для разработчика

Инкапсуляция

	default	private	protected	public
Same Class	Yes	Yes	Yes	Yes
Same package subclass	Yes	No	Yes	Yes
Same package non-subclass	Yes	No	Yes	Yes
Different package subclass	No	No	Yes	Yes
Different package non-subclass	No	No	No	Yes

Полиморфизм



Полиморфизм

- Дочерний класс может быть использован везде, где используется родительский
- Если дочерний класс приведен к родительскому, то доступны только методы родительского класса (по типу ссылки)
- Вызывается реализация по реальному типу объекта (@Override)

Полиморфизм

- Дочерний класс может быть использован везде, где используется родительский

```
class Admin extends User;


1. User admin = new Admin(); // ссылка имеет родительский тип


// можно передать дочерний тип, там где ожидается родительский
2. foo(User u) { return u.getName();}
3. Admin admin = new Admin();
4. foo(admin);
```

Полиморфизм

- Во время компиляции проверяется, что такой метод есть у объекта заданного типа.

```
class User {                          User u = new Admin();
   getName() {return “U”;};           u.getName();
}                                     u.ban(); // Compile-time err

class Admin extends User {            Admin a = new Admin();
   void ban(User user) {...};         a.getName();
}                                     a.ban(); // ok
```

Полиморфизм

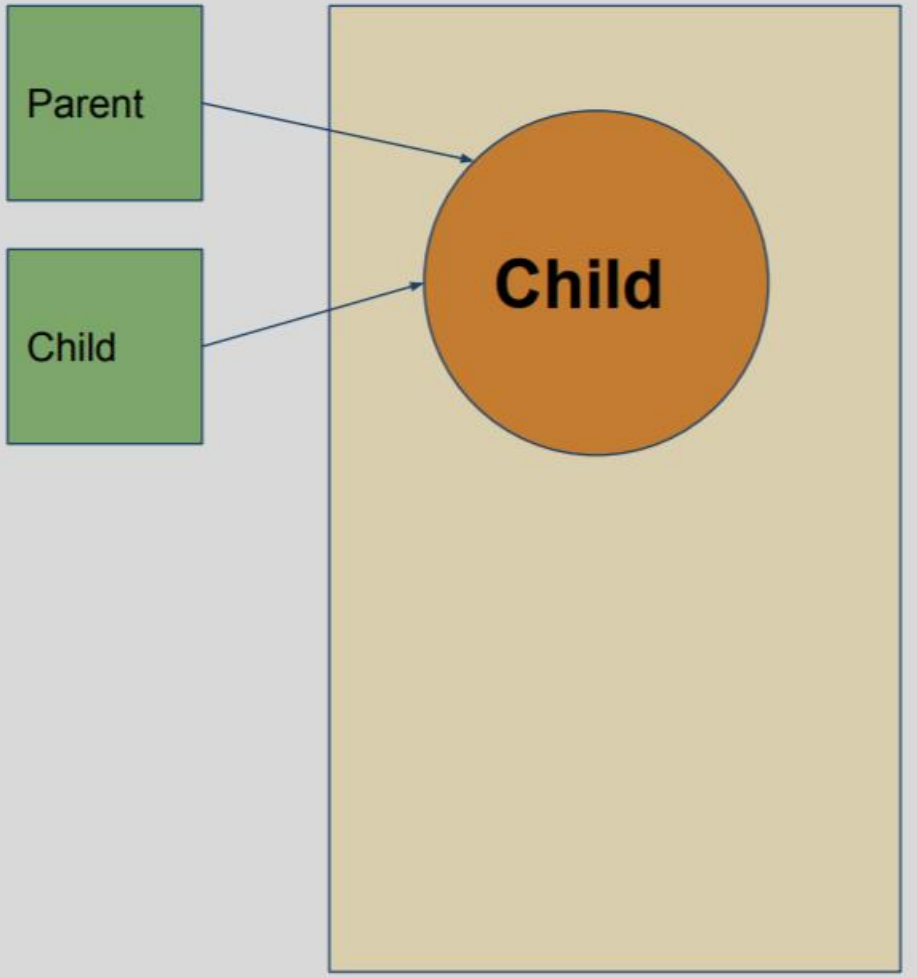
- Вызывается реализация по реальному типу объекта @Override)

```
class User {
   getName() {return “U”;};
}

class Admin extends User {
   void ban(User user) {...};
   getName() {return “A”;};
}
```

```
Admin a = new Admin();
a.getName();// ?

User u = new User();
u.getName();// ?

User u = new Admin();
u.getName();// ?
```

Полиморфизм

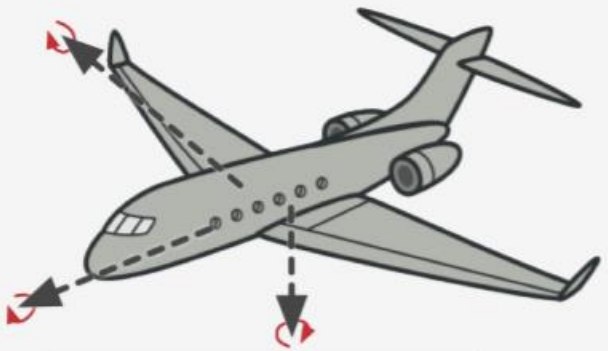
```
class Parent {
   test(){print("P");}
}
class Child ext. Parent {
   @Override
   test(){print("C");}
}

Child child = new Child();
Parent pChild = child;
child.test();
pChild.test();
```

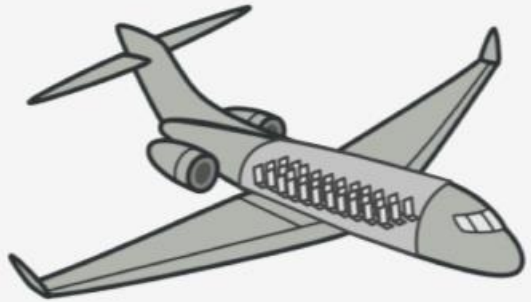


Что еще?

- Наследование
- Полиморфизм
- Инкапсуляция
- ?

Абстракция





Airplane
- speed - altitude - rollAngle - pitchAngle - yawAngle
+ fly()

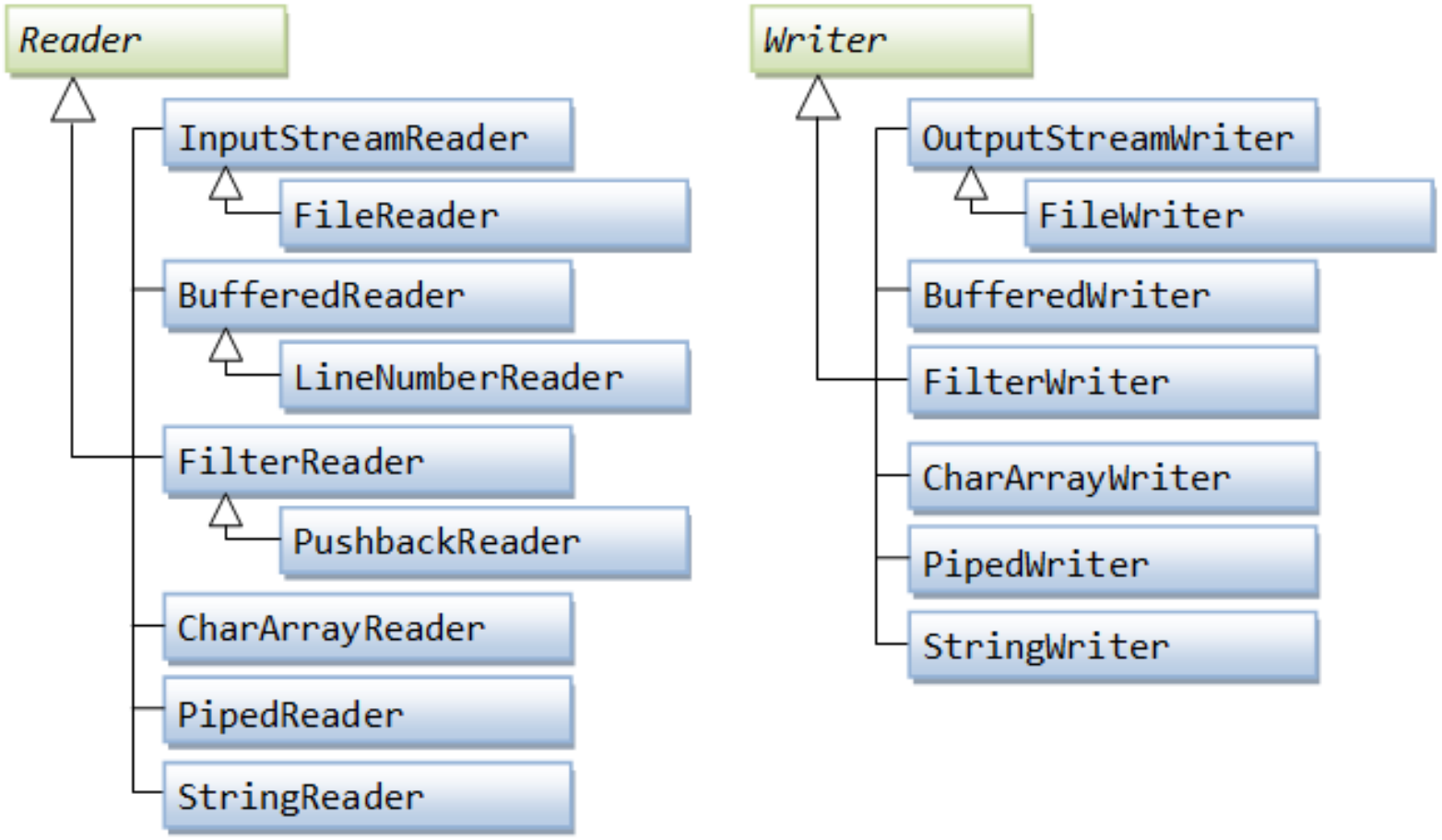
Airplane
- seats
+ reserveSeat(n)

Разные модели одного и того же реального объекта.

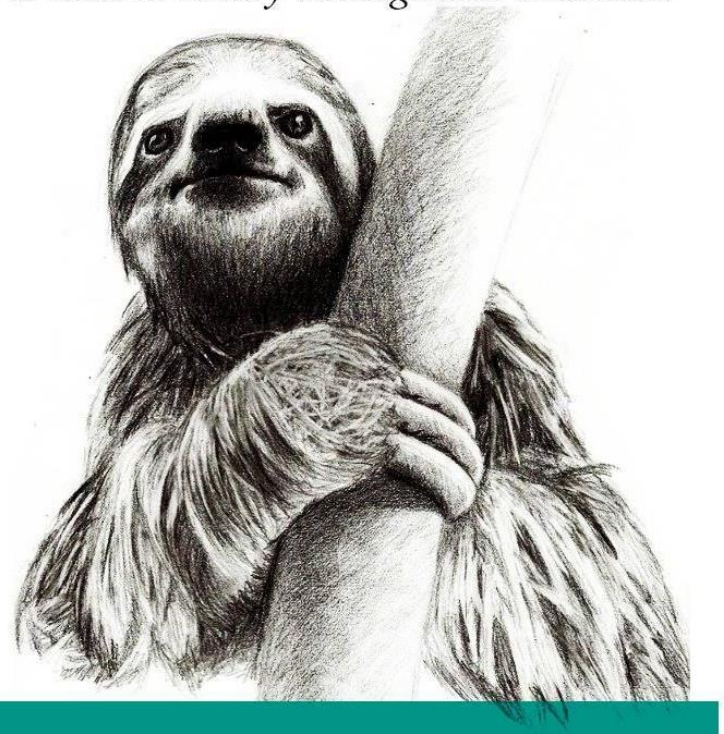
Интерфейс

- Определяет, что можно сделать с классом
- Не определяет, как это делать
- Класс может реализовывать несколько интерфейсов
- Абстракция от реализации
- Обобщение по свойству
- Реализацию можно изменить
- Контракт

Консольный ввод/вывод



Cutting corners to meet arbitrary management deadlines



Essential

Copying and Pasting from Stack Overflow

O'REILLY®

The Practical Developer
@ThePracticalDev

Does it run? Just leave it alone.

Writing Code that Nobody Else Can Read

The Definitive Guide

O RLY?

@ThePracticalDev

Полезные ссылки

https://javarush.ru/groups/posts/1966-principih-obhhektno-orientirovannogo-programmirovanija - хорошая статья про ООП.
https://javarush.ru/groups/posts/principy-oop - хорошая статья про ООП в java.
https://javarush.ru/groups/posts/1989-kontraktih-equals-i-hashcode-ili-kak-ono-vsje-tam - статья про equals и hashCode.
https://javarush.ru/groups/posts/konstruktory-v-java - про конструкторы
https://google.github.io/styleguide/javaguide.html - гайдлайн от Гугл. Читать пока что будет сложно и не нужно. Но знать о его существовании полезно.
http://developer.alexanderklimov.ru/android/java/mememe.php - сильно упрощенный гайдлайн. Стоит прочитать, материал несложный.
https://javarush.ru/groups/posts/1919-schitihvanie-s-klaviaturih--riderih - про ввод и вывод
https://javarush.ru/groups/posts/593-bufferedreader-i-bufferedwritter - тоже про ввод и вывод