http://0861.ru

Парадигмы программирования

Лекция 4
Объектно-ориентированное программирование

ст. препод. каф. ПОВТиАС
Голубничий Артем Александрович
artem@golubnichij.ru

Абакан, 2019

Структура занятия

- понятие ООП;
- история ООП;
- принципы ООП (по Алану Кэю);
- основные понятия;
- UML.

Понятия ООП

Объектно-ориентированное программирование (ООП) – методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования.

- ООП помогает справиться с нелинейно растущей сложностью программ при увеличении их объема.

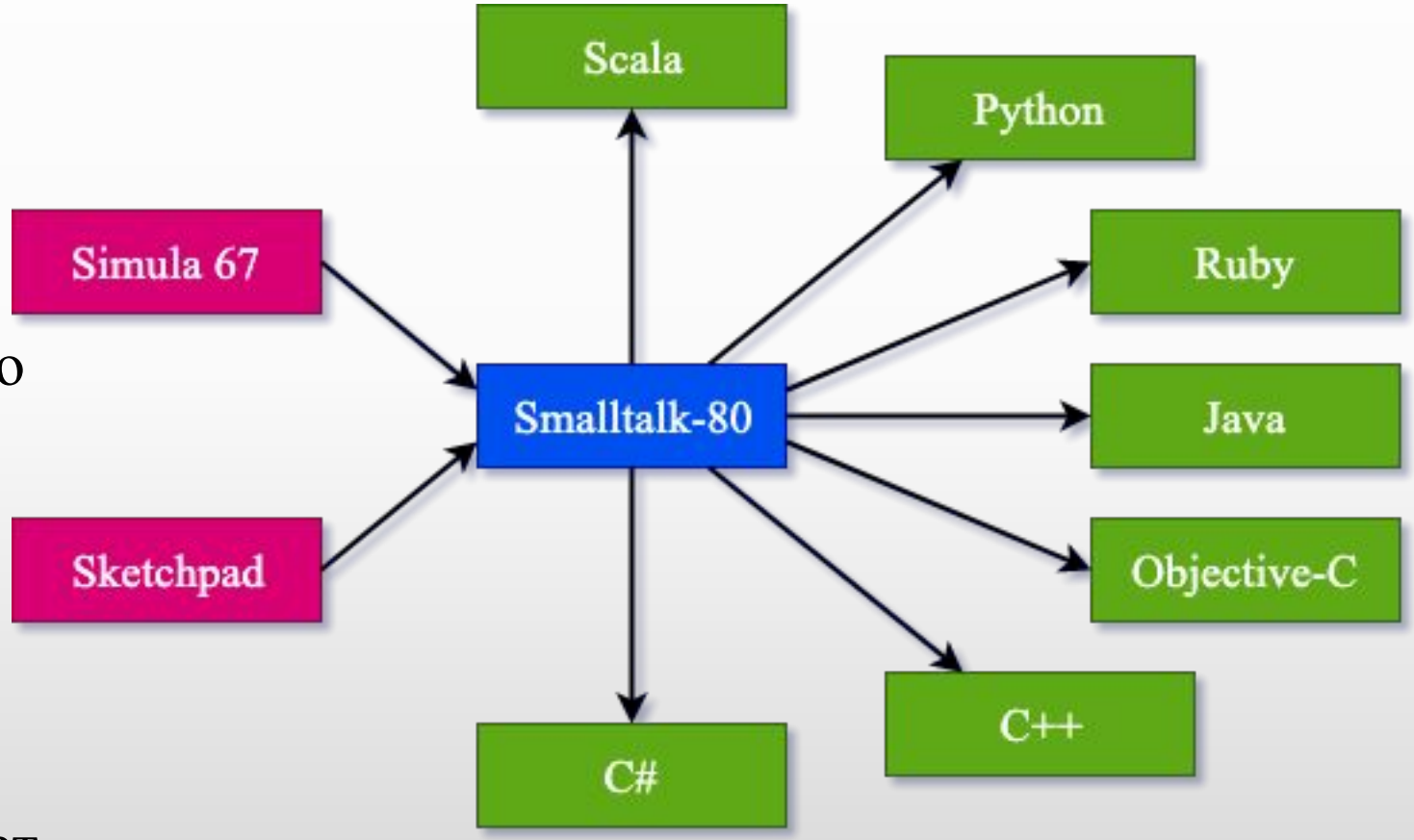
Класс – универсальный, комплексный тип данных, состоящий из тематически единого набора «полей» (переменных более элементарных типов) и «методов» (функций для работы с этими полями).

История ООП

Smalltalk – объектно-ориентированный язык программирования с динамической типизацией, основанный на идее посылки сообщений.

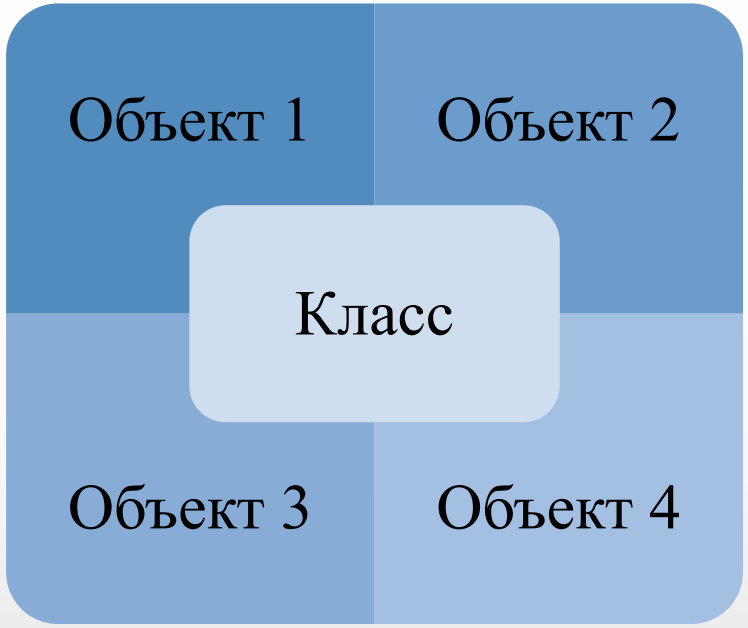
Sketchpad – прообраз будущих CAD систем. Программа, построенная на принципах объектно-ориентированного программирования с высоким (на тот момент) уровнем человеко-машинного взаимодействия.

Simula – язык общего назначения, традиционно не считается объектно-ориентированным языком, представляет расширение алгола «объектной» нотацией.

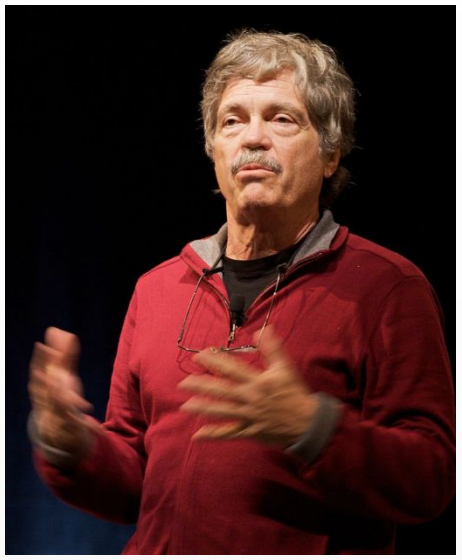


Принципы ООП

Принцип 1. Все является объектом.



объект – абстракция данных;
объект – это отдельный представитель класса, имеющий конкретное состояние и поведение, полностью определяемое классом;



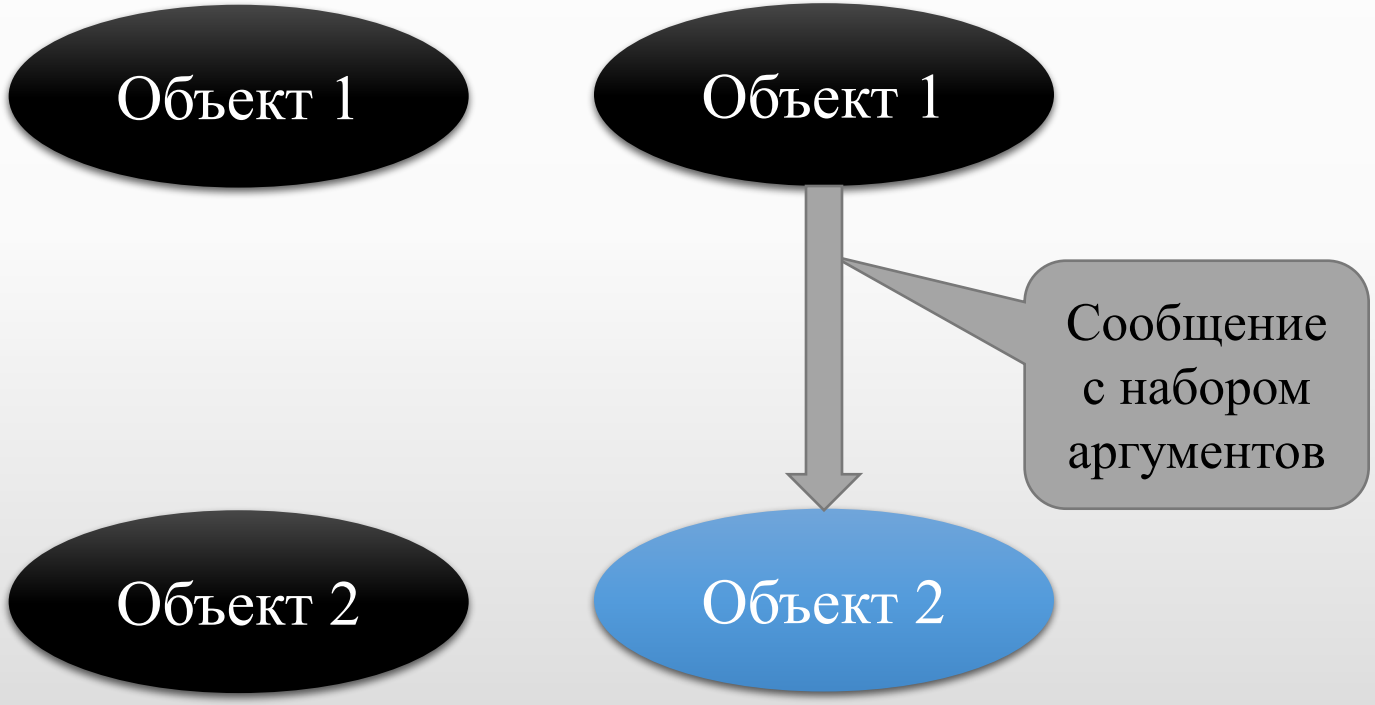
Алан Кёртис Кэй

объект – сущность в адресном пространстве вычислительной системы, появляющаяся при создании экземпляра класса (например, после запуска результатов компиляции и связывания исходного кода на выполнение).

Принципы ООП

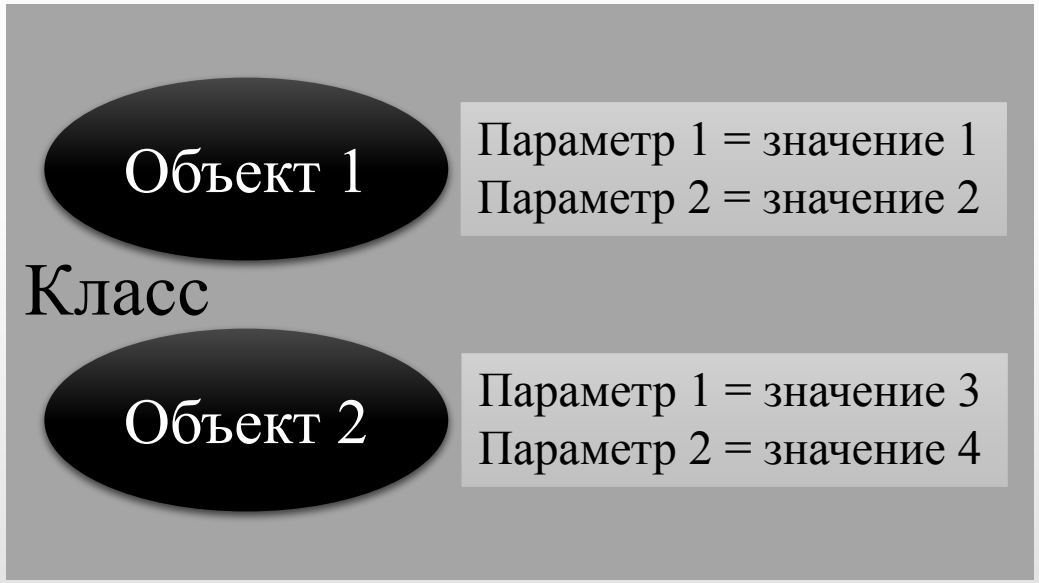
Принцип 2. Вычисления осуществляются путем взаимодействия (обмена данными) между объектами, при котором один объект требует, чтобы другой объект выполнил некоторое действие.

- объекты взаимодействуют, посылая и получая сообщения;
- **сообщение** – это запрос на выполнение действия, дополненный набором аргументов, которые могут понадобиться при выполнении действия.



Принципы ООП

Принцип 3. Каждый объект имеет независимую память, которая состоит из других объектов.



Принципы ООП

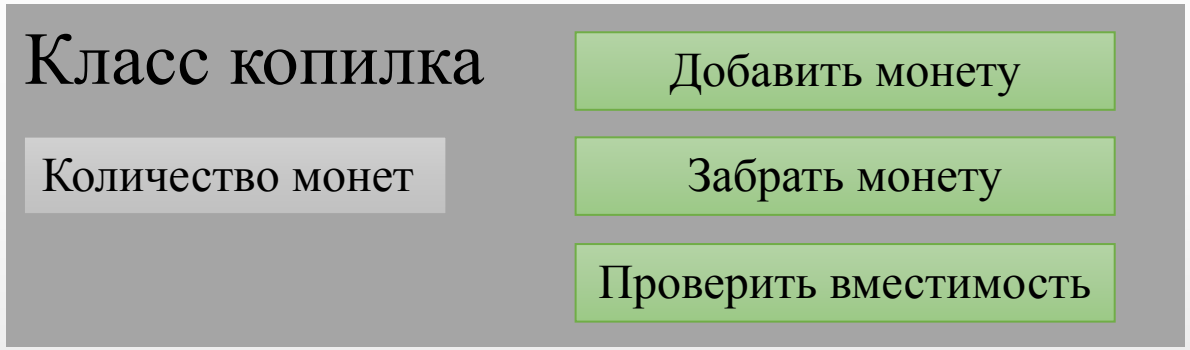
Принцип 4. Каждый объект является представителем класса, который выражает общие свойства объектов данного типа.

Кластеры	Производительность Количество узлов Размер кэша …

Принципы ООП

Принцип 5. В классе задается функциональность (поведение объекта).

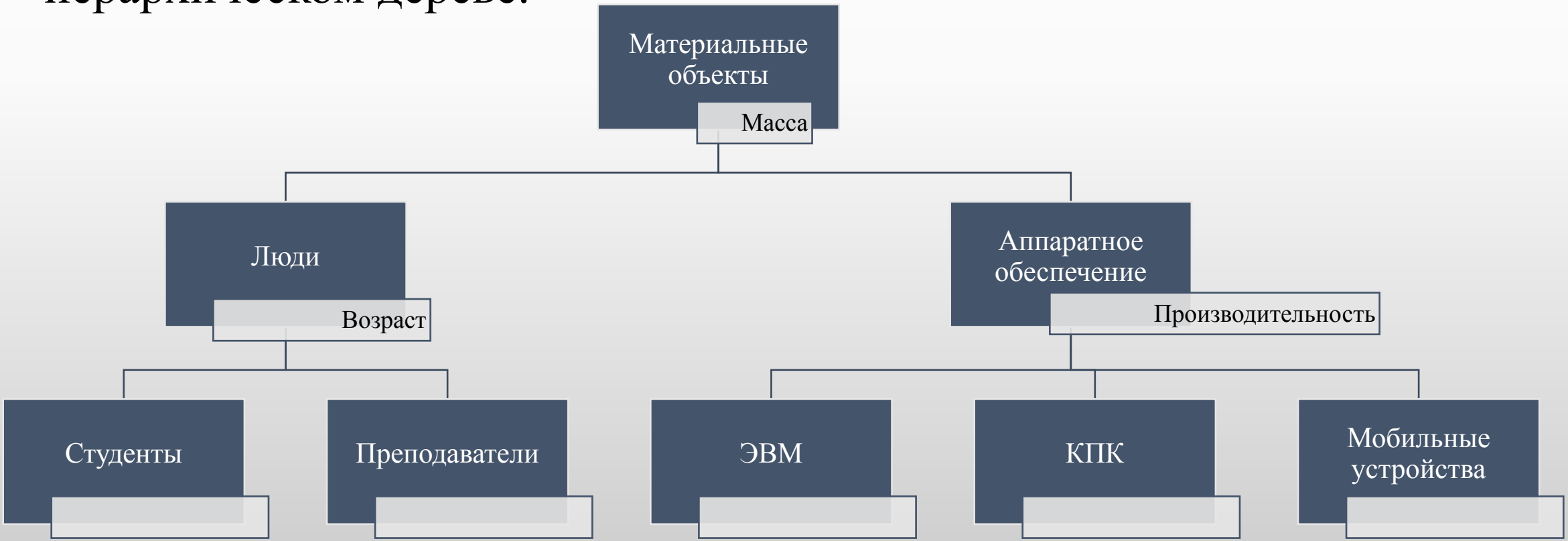
- все объекты, которые являются экземплярами одного класса, могут выполнять одни и те же действия.



Принципы ООП

Принцип 6. Классы организованы в единую древовидную структуру с общим корнем, называемую иерархией наследования.

- память и поведение, связанное с экземплярами определенного класса, автоматически доступны любому классу, расположенному ниже в иерархическом дереве.



Основные понятия: абстрагирование

Абстрагирование – выделение значимой информации и исключение из рассмотрения незначимой.

Значимая информация	Не значимая информация
• Производительность • Количество узлов • Размер кэша	• Место расположения • Версия ОС • Год сборки

Основные понятия: инкапсуляция и сокрытие

Инкапсуляция – упаковка данных и функций в единый компонент;
Инкапсуляция – позволяет объединить данные и методы, работающие с ними, в классе.

- механизм языка, позволяющий ограничить доступ одних компонентов программы к другим;
- языковая конструкция, позволяющая связать данные с методами, предназначенными для обработки этих данных.

Сокрытие – принцип проектирования, заключающийся в разграничении доступа различных частей программы к внутренним компонентам друг друга.

Отождествление понятий	Отсутствие сокрытия
C++, Java, Ruby	Smalltalk, Python

Инкапсуляция на примере Java

```
class A {
 private int a;
 private int b;

 private void doSomething() { //скрытый метод
  //actions
 }

 public int getSomething() { //открытый метод
  return a;
 }
}
```

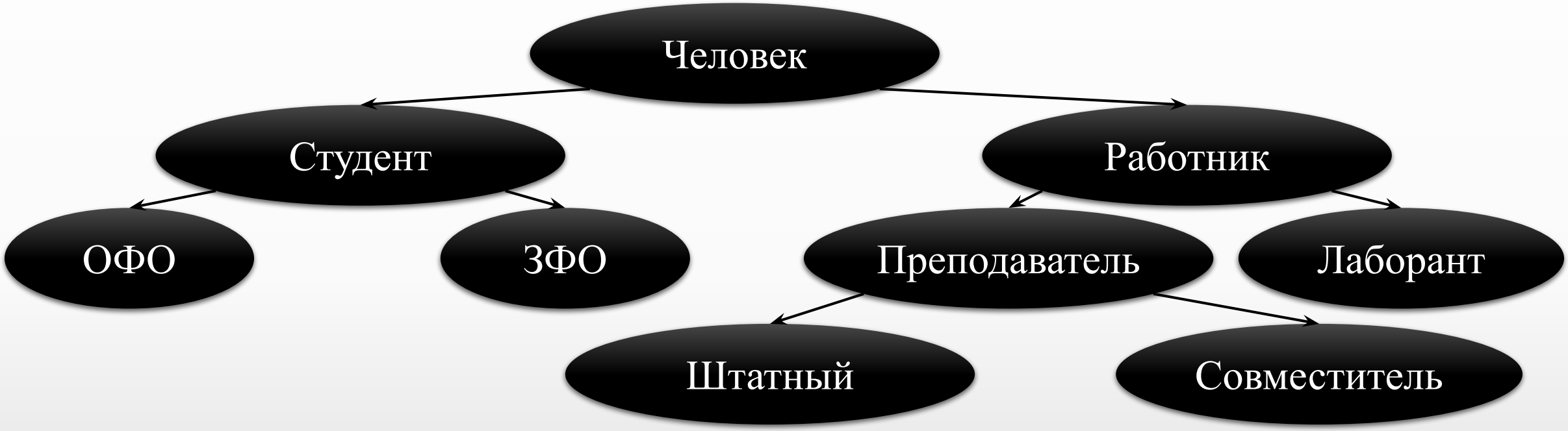
Основные понятия: наследование

Наследование – свойство системы, позволяющее описать новый класс на основе уже существующего с частично или полностью заимствующейся функциональностью

Наследование – концепция ООП, согласно которой абстрактный тип данных может наследовать данные и функциональность некоторого существующего типа, способствуя повторному использованию компонентов программного обеспечения.

- класс, от которого производится наследование, называется базовым, родительским или суперклассом;
- класс, создаваемый на основе другого класса называется потомком, наследником, дочерним или производным классом.

Пример наследования



Основные понятия: полиморфизм

Полиморфизм – свойство системы, позволяющее использовать объекты с одинаковым интерфейсом без информации о типе и внутренней структуре объекта.

Полиморфизм – свойство родственных объектов решать схожие по смыслу задачи разными способами.

- ***вычисление площади*** – метод, применимый к абстрактному классу.

UML в ООП

UML (унифицированный язык моделирования) – язык графического описания для объектного моделирования в области разработки программного обеспечения, моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

Диаграмма классов (Class diagram) – статическая структурная диаграмма, описывающая структуру системы, демонстрирующая классы системы, их атрибуты, методы и зависимости между классами.

UML в ООП

Структура диаграммы класса:

- **Имя**. Имя класса выравнивается по центру и пишется полужирным шрифтом. Имена классов начинаются с заглавной буквы. Если класс абстрактный – то имя пишется полужирным курсивом.
- **Атрибуты**. Они выровнены по левому краю и начинаются с маленькой буквы.
- **Методы**. Они также выровнены по левому краю и пишутся с маленькой буквы.

BankAccount
owner : String balance : Dollars = 0
deposit (amount : Dollars) withdrawal (amount : Dollars)

Видимость в UML

Для задания видимости членов класса (т.е. атрибутов или методов), эти обозначения должны быть размещены перед именем участника

+	Публичный (Public)
-	Приватный (Private)
#	Защищенный (Protected)
/	Производный (Derived) (может быть совмещен с другими)
~	Пакет (Package)

public – внутри класса и вне класса;
private – внутри класса;
protected – внутри самого класса или внутри производных классов;
package – видимость внутри пакета.

Взаимосвязи в UML

Объекты классов	• зависимость • ассоциация • агрегация • композиция
Классы	• наследование • реализация