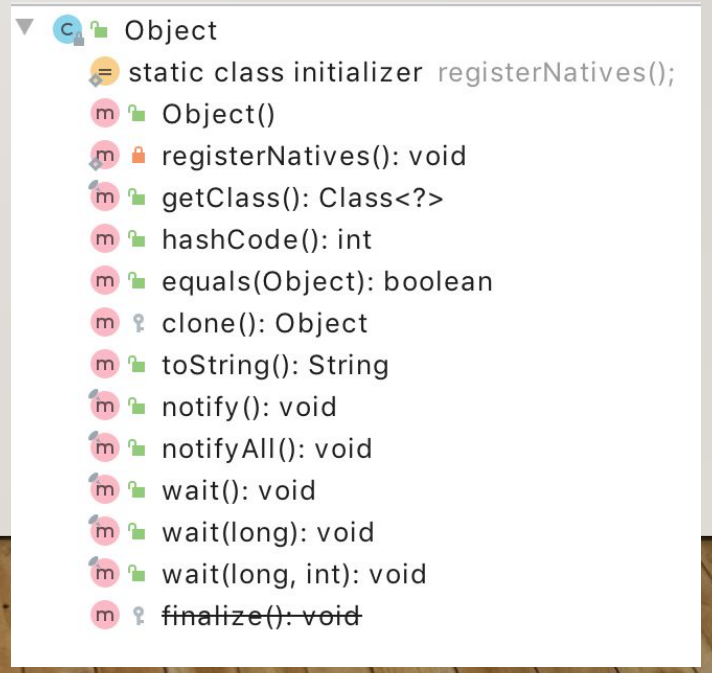
ЗАНЯТИЕ 18

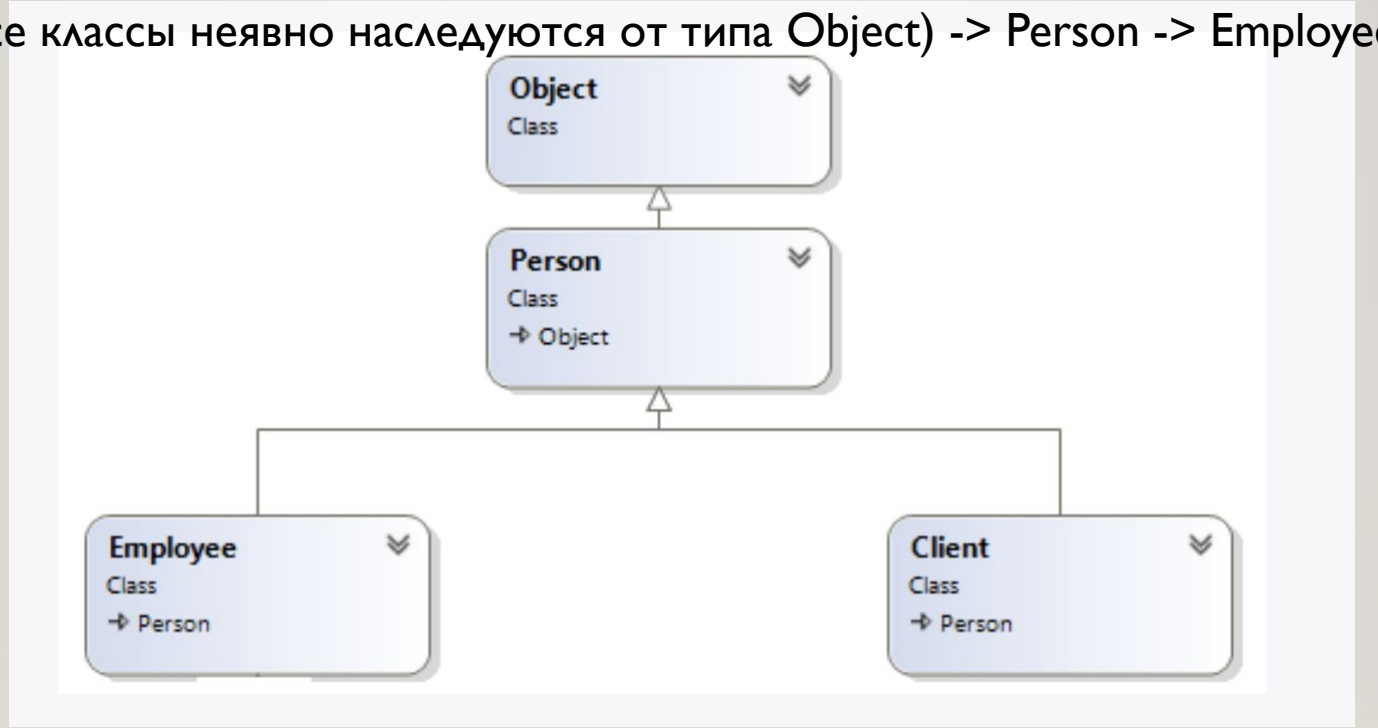
КЛАСС OBJECT

- Хотя мы можем создать обычный класс, который не является наследником, но фактически все классы наследуются от класса **Object**. Все остальные классы, даже те, которые мы добавляем в свой проект, являются неявно производными от класса Object. Поэтому все типы и классы могут реализовать те методы, которые определены в классе Object.
- Point1/Person



КЛАСС OBJECT

В этой иерархии классов можно проследить следующую цепь наследования:
Object (все классы неявно наследуются от типа Object) -> Person -> Employee|Client.



ОПЕРАТОР INSTANCEOF

- Оператор **instanceof** позволяет выяснить, является ли переданный в качестве параметра объект объектом определенного класса. Результат – true/false
- Point1/InstanceOperator.java

- https://javarush.ru/groups/posts/2018-kak-rabotaet-operator-instanceof

АБСТРАКТНЫЕ КЛАССЫ

- Кроме обычных классов в Java есть **абстрактные классы**. Абстрактный класс похож на обычный класс. В абстрактном классе также можно определить поля и методы, но в то же время нельзя создать объект или экземпляр абстрактного класса. Абстрактные классы призваны предоставлять базовый функционал для классов-наследников. А производные классы уже реализуют этот функционал.
- При определении абстрактных классов используется ключевое слово **abstract**
- point_2/Human.java

АБСТРАКТНЫЕ МЕТОДЫ

- Кроме обычных методов абстрактный класс может содержать **абстрактные методы**. Такие методы определяются с помощью ключевого слова abstract и не имеют никакой реализации
- Point_2/Person.java, Employee, Client
- Производный класс обязан переопределить и реализовать все абстрактные методы, которые имеются в базовом абстрактном классе. Также следует учитывать, что если класс имеет хотя бы один абстрактный метод, то данный класс должен быть определен как абстрактный.

ЗАЧЕМ НУЖНЫ ТАКИЕ КЛАССЫ?

- Иногда нам не нужно явно создавать объект производного класса, так как понятие о нем как таково и нет. Например, геометрическая фигура
- В реальности не существует геометрической фигуры как таковой. Есть круг, прямоугольник, квадрат, но просто фигуры нет. Однако же и круг, и прямоугольник имеют что-то общее и являются фигурами
- Point_2/Figure.java,Rectangle.java

ИНТЕРФЕЙСЫ HTTPS://METANIT.COM/JAVA/TUTORIAL/3.7.PHP

- Механизм наследования очень удобен, но он имеет свои ограничения. В частности мы можем наследовать только от одного класса, в отличие, например, от языка C++, где имеется множественное наследование.
- В языке Java подобную проблему частично позволяют решить интерфейсы. Интерфейсы определяют некоторый функционал, не имеющий конкретной реализации, который затем реализуют классы, применяющие эти интерфейсы. И один класс может применить множество интерфейсов.
- Чтобы определить интерфейс, используется ключевое слово **interface**.
- Интерфейс – описание функционала, который надо будет реализовать в классах, которые его имплементируют (implements)
- point_3.sub_1

ИНТЕРФЕЙСЫ

- Также в интерфейсах можно создавать поля, они по умолчанию являются public static final (константами), но лучше таких интерфейсов избегать.

- В Java 8 появились в интерфейсах дефолтные методы для обеспечения обратной совместимости (чтобы добавить метод в интерфейс, не задев все имплементации), а также статические, которые можно вызывать без создания объекта
- Point_3/sub_2
- Множественное наследование интерфейсов
- Point_3/sub_3

ABSTRACT CLASS VS INTERFACE

- **Интерфейс описывает только поведение. У него нет состояния. А у абстрактного класса состояние есть: он описывает и то, и другое.**
- **Абстрактный класс связывает между собой и объединяет классы, имеющие очень близкую связь. В то же время, один и тот же интерфейс могут реализовать классы, у которых вообще нет ничего общего.**
- **Классы могут реализовывать сколько угодно интерфейсов, но наследоваться можно только от одного класса.**
- **АБСТРАКТНЫЙ КЛАСС РАСШИРЯЕТСЯ, А ИНТЕРФЕЙС РЕАЛИЗУЕТСЯ**

ABSTRACT CLASS VS INTERFACE

Abstract class	Interface
An abstract classes can have non-abstract method as well as abstract method.	An Interface can contain only deceleration of a methods
An Abstract classes are used when we want to share common functionality in parent child relationship.	An Interfaces are used to define contract, enforce standardization, decoupling and dynamic polymorphism.
We can declare variables	In interface we cannot declare variables
An Abstract classes are inherited.	An Interfaces are implemented.
An Abstract classes can contain constructors	An Interface cannot contain constructors

ПРАКТИКА

- Task 1 (на интерфейс)
  Есть интерфейс Computer,у которого есть методы включить, выключить и резет. Создайте класс, который бы имплементировал
  этот интерфейс
- Task 2 (на абстрактный класс)
  Есть два типа домов - городская многоэтажка и загородный дом.
  У каждого типа дома есть методы "получить кол-во этажей", "включить отопление", "получить кол-во жильцов".
  Есть каталог домов (например класс, внутри которого какой-то массив).
  Создайте каталог и поместите туда несколько домов.