Модификаторы доступа, наследование.

1. Инкапсуляция
2. Наследование полей и методов
3. Наследование конструкторов

**Инкапсуляция**

Объектно-ориетированное программирование.

Java – язык объектно-ориентированного программирования, т.е. программа состоит из различных объектов, которые взаимодействуют между собой через вызов команд и обращения к их полям.

Поэтому объекты, которые мы создаем, должны быть прописаны грамотно: так, чтобы вероятность их некорректной работы стремилась к нулю

Инкапсуляция:

public class Person {  
 public String name;  
 public int age;  
   
 public boolean isTooYoung(){  
 if (age < 18) {  
 return true;  
 } else {  
 return false;  
 }  
 }  
}

Использование инкапсуляции: создадим метод, задачей которого будет менять возраст.

Чтобы защитить нашу программу от ошибки, установим условие, при котором она будет выполняться, используя метод setAge. Если условие не соблюдяается, то программа просто выйдет, ничего не изменив, - early exit.

Также создадим специальный метод, с помощью которого мы просим человека дать нам свой возраст – getAge

public class Person {  
 public String name;  
 public int age;  
 public void setAge(int age){  
 if (age <0 || age > 200) {  
 return;  
 }  
 this.age = age;  
 }  
  
 public int getAge() {  
 return age;  
 }  
}

Использование инкапсуляции:

Чтобы метод работал, создадим команду, которая будет просить установить возраст объекту.

Программа выведет на экран только корректное значение, подходящее заданным условиям

**Модификаторы Доступа**

* **Public:** Публичный, общедоступный класс или член класса. Поля и методы, объявленные с модификатором public видны другим классам из текущего пакета и из внешних пакетов.
* **Private:** закрытый класс или член класса. Противоположность модификатору public. Доступен только из коды в том же классе
* **Protected:** такой класс или член класса доступен из любого места в текущем классе, пакете или в классах-наследниках, даже если они находятся в других пакетах
* **Default:** модификатор по умолчанию. Когда мы не пишем модификатор доступа, он по умолчанию имеет значение default. Такие поля или методы видны всем классам в текущем пакете

Один из ключевых аспектов объектно-ориентированного программирования – это наследование.

С помощью наследования можно расширить функционал уже имеющихся классов за счет добавления нового функционала или изменения старого. Так, один класс может наследовать характеристики другого – его методы и переменные.

**Наследование полей**

* Класс, являющийся основой, называют базовым, суперклассом, родительским.
* Класс, который создают, называют потомком, наследником или производным классом

Сама Java тоже использует механизм наследования. Например, все классы языка наследуют класс **Object**

Наследование методов

Производный класс имеет доступ ко всем методам и полям базового класса (даже если базовый класс находится в другом пакете) кроме тех, которые определены с модификатором private. При этом производный класс также может добавлять свои поля и методы.

Добавление полей и методов в производный класс никак не влияет на базовый

Также в производном классе можно модифицировать поля и методы базового класса. Перед модифицированным методом указывается аннотация @override. Эта аннотация не обязательна, но она помогает избежать ошибок, убеждая Java, что мы точно меняет версию уже существующего метода, а не создаем новый.

При переопределении метода он должен иметь уровни доступа не меньше, чем в базовом классе.

Например, если в базовом классе метод иметь модификатор public, то и в производном классе метод должен иметь модификатор public

ПРАВИЛА НАСЛЕДОВАНИЯ:

* ПРАВИЛО 1. Наследоваться можно только от одного класса.
* ПРАВИЛО 2. Наследуется все, кроме приватных переменных и методов.
* ПРАВИЛО 3. Переделать метод класса-родителя пишем над ним @override
* ПРАВИЛО 4. Вызываем методы родителя через ключевое слово super.
* ПРАВИЛО 5. Если хотите запретить наследование, поставьте перед ним модификатор final