**Конструкторы классов в Java**

**Конструктор класса - это специальный метод. Использование конструкторов определяется их особенностями:**

* **Конструктор вызывается при создании нового объект, непосредственно перед завершением оператора new;**
* **Конструктор вызывается автоматически;**
* **Конструктор инициализирует состояние объекта. Причем даже если он не определен пользователем. В этом случае используется конструктор по умолчанию;**
* **Имя конструктора совпадает с именем класса;**
* **Конструктор не имеет возвращаемого значения;**

**В целом конструкторы работают как методы и по синтаксису схожи с ними. Из особенностей перечисленных выше выходит, что конструкторы используются для инициализации объектов, причем автоматической инициализации.**

**У конструкторов есть ряд отличий от методов, которые обусловлены их природой:**

* **Конструктор не может быть выиртуальным;**
* Конструктор не может иметь директив final и static;
* Поскольку конструктор не возвращает значения, то он не должен содержать действий, которые требуют возврата значений. Лучше ограничится только инициализацией;

Очень важным является тот факт, что в иерархии классов конструкторы **вызываются автоматически в порядке наследования**. Начиная с родительского класса и заканчивая подклассами.

**Getters and Setters**

**Одним из важных принципов ООП является инкапсуляция. Инкапсуляция - это механизм, связывающий код и данные, которыми он манипулирует, таким образом, чтобы изолировать эти компоненты друг от друга и защитить их от воздействий. Инкапсуляцию можно считать защитной оболочкой, которая предохраняет код и данные от произвольного доступа со стороны другого кода.**

**Данные нужно отделить от кода, чтобы предотвратить неверную модификацию этих данных. Соответственно для правильного доступа к данным необходимо иметь соответствующие методы. Эту и играют функции-геттеры и функции-сеттеры.**

**В реализации данных методов заключен принцип инкапсуляции. Мы не видим, как данные обрабатываются (известен лишь интерфейс, через который, и только через который вы можете взаимодействовать с объектом), а значит можно быть уверенным, в корректности обработки данных, чего нельзя сказать при работе с данными напрямую.**

**Кроме корректности мы получаем целостность, относительную простоту, прозрачный интерфейс для работы с данными.**