**Дженерики — что это?**

[Java Syntax](https://javarush.ru/quests/QUEST_JAVA_SYNTAX)

[Уровень 7](https://javarush.ru/quests/lectures?quest=QUEST_JAVA_SYNTAX&level=7), Лекция 7

— И ещё одна крутая тема.

— Одни сюрпризы. Прямо не день, а день рождения.

— Сегодня я расскажу тебе, что такое Generics. “Дженерики” – это типы с параметром. В Java классы-контейнеры позволяют указывать тип их внутренних объектов.

— Когда мы объявляем generic-переменную, то мы указываем не один тип, а два: тип переменной и тип данных, которые она у себя хранит.

Хороший пример этого – ArrayList. Когда мы создаём новый объект/переменную типа ArrayList, нам удобно указать, значения какого типа будут храниться внутри этого списка.

|  |  |
| --- | --- |
| **Код** | **Пояснение** |
| ArrayList<String> list = new ArrayList<String>(); | Мы создали переменную list типа ArrayList. Занесли в нее объект типа ArrayList. В таком списке можно хранить только переменные типа String. |
| ArrayList list = new ArrayList(); | Мы создали переменную list типа ArrayList. Занесли в нее объект типа ArrayList.В таком листе можно хранить переменные любого типа. |
| ArrayList<Integer> list = new ArrayList<Integer>(); | Мы создали переменную list типа ArrayList. Занесли в нее объект типа ArrayList. В таком листе можно хранить только переменные типа Integer и int. |

— Звучит очень интересно. Особенно про любой тип.

— Это только кажется, что это хорошо. На самом деле, если в одном методе в ArrayList кладутся строки, а в другом мы работаем с его содержимым и ожидаем, что там будут только числа, программа упадет (закроется с ошибкой).

— Ясно.

— Пока что мы не будем создавать **свои** классы с типами-параметрами, но будем использовать чужие.

— А в качестве типа-параметра можно поставить любой класс, даже тот, что напишу я?

— Да, любой тип, кроме примитивных типов. Все классы-параметры должны быть унаследованы от класса Object.

— То есть, я не могу написать ArrayList<int>?

— В общем-то, да, не можешь. Но для примитивных типов разработчики языка Java написали их непримитивные аналоги — классы, унаследованные от **Object**. Вот как это будет выглядеть:

|  |  |  |
| --- | --- | --- |
| **Примитивный тип** | **Класс** | **Список** |
| int | Integer | ArrayList<Integer> |
| double | Double | ArrayList<Double> |
| boolean | Boolean | ArrayList<Boolean> |
| char | Character | ArrayList<Character> |
| byte | Byte | ArrayList<Byte> |

— Примитивные типы и их классы-аналоги (классы-обёртки) можно спокойно присваивать друг другу:

|  |
| --- |
| **Примеры** |
| int a = 5;  Integer b = a;  int c = b; |
| Character c = 'c'; //литерал 'c' имеет тип char  char d = c; |
| Byte b = (byte) 77; //литерал 77 имеет тип int |
| Boolean isOk = true; //литерал true имеет тип boolean |
| Double d = 1.0d; //литерал 1.0d имеет тип double |

— Отлично. Тогда я думаю, я буду почаще использовать ArrayList.