Generic (обобщения)

Generic (обобщения) - описание класса, метода или атрибута без использования конкретного типа данных.
В более широком смысле, *Generic (обобщение)* - синтаксическая конструкция языка, реализующая концепцию обобщенного программирования (generic programming).
Обобщенное программирование (generic programming) – описание данных и алгоритмов в программе, которое можно применить к различным типам данных, не меняя при этом само это описание.

Пример 1:
Класс LinkedList хранит список объектов, при этом операции добавления, поиска и удаления объекта из листа ни как не зависят от того, какой объект хранится в листе. С другой стороны, отказаться от типизации листа совсем было бы неудобно, т.к. *тип данных* определяет поведение объекта, и если не известен *тип*, не известно и поведение объекта.

Пример 2:
Алгоритм сортировки, например Bubble Sort абсолютно не зависит от того, какие элементы необходимо сортировать. В любом случае нам надо перебирать элементы, переставлять их местами, сравнивать.При этом очевидно, что только операция сравнения зависит от того, что мы сравниваем.

В Java generic-класс как бы «настраивает себя» на определенный тип данных, который он получает в *параметрах типа* задаваемых в <>

```
List<String> stringList = new ArrayList<>();
List<Integer> integerList = new ArrayList<>();
List<Person> personList = new ArrayList<>();
List<List<Integer>> integerList = new ArrayList<>();
```

В вышеприведенных объявлениях, мы всякий раз создаем ArrayList, но каждый раз он "настраивается" под хранение объектов конкретного типа, соответственно, String, Integer, Person и листов из Integer

Внимание! Generic в Java не работают с примитивными типами данных, вместо примитивов используются соответствующие классы-обертки (Integer,Double, Boolean и т.д).

Создание generic класса

```
public class TwoValueBox<T1,T2> {
    T1 value1;
    T2 value2;

    public TwoValueBox(T1 value1, T2 value2) {
        this.value1 = value1;
        this.value2 = value2;
    }

    public T1 getValue1() {
        return value1;
    }

    public T2 getValue2() {
        return value2;
    }
}
```

Приведенный код описывает класс, в обьекте которого можно сохранить два любых значения.Обратите внимани на заголовок класса `class TwoValueBox<T1,T2>` Два идентификатор T1 и T2 между угловыми скобками (< >) — это *параметры-типа (type parameter)*. T1 и T2 - всего лишь имена параметров-типа, вместо них можно использовать любые другие. Обычно используют заглавные буквы (E, K, V, ...). Наличие <T1,T2> говорит о том, что класс является generic-классом, и при создание объекта будет необходимо указат конкретные значения типов для парраметров T1 и T2. Например так:

```
TwoValueBox<String,Integer> = new TwoValueBox<>("строка",10);
TwoValueBox<Integer,Integer> = new TwoValueBox<>(-10,10);
TwoValueBox<String,String> = new TwoValueBox<>("строка1","строка1");
```

После определения конкретного типа, еще это называют *воплощением типа (instantiating the type)* (иногда можно встретить и другие названия: *реализация типа, инстанцирование типа*) класс "настроится" на использование данных заданного типа и везде, вместо символов T1 и T2 будут подставлены конкретные значения типа данных.

Параметры-типы можно использовать в объявлениях переменных экземпляров, аргументов методов и возвращаемых типов методов.

Ограниченные (bounded) типы

В примерах, рассмотренных выше, параметры типов можно было заменить любыми ссылочными типами. Но что если нужно ограничить набор этих перечень типов, например только числавми значениями?

```
public class NumbersBox<T extends Number>
```

Класс объявленный таким образом является generic-классом, но объекты типа T являются Number или подклассами класса Number. Таким образом, данный класс не позволит работать с не числовыми данными. Такая запись называется ограничение типа сверу