Параметризованные методы

Параметризованный (generic) метод определяет базовый набор операций, которые будут применяться к разным типам данных, получаемых методом в качестве параметра, и может быть записан, например, в виде:

```
<T extends Тип> returnType methodName(T arg) {}
<T> returnType methodName(T arg) {}
```

Описание типа должно находиться перед возвращаемым типом. Запись первого вида означает, что в метод можно передавать объекты, типы которых являются подклассами класса, указанного после **extends**. Второй способ объявления метода никаких ограничений на передаваемый тип не ставит.

Generic-методы могут находиться как в параметризованных классах, так и в обычных. Параметр метода может не иметь никакого отношения к параметру своего класса. Метасимволы применимы и к generic-методам.

/ пример # 13 : параметризованный метод: GenericMethod.java */*

```
public class GenericMethod {
     public static <T extends Number> byte asByte(T num) {
             long n = num.longValue();
             if (n >= -128 && n <= 127) return (byte)n;
             else return 0;
     }
     public static void main(String [] args) {
          System.out.println(asByte(7));
          System.out.println(asByte(new Float("7.f")));
          // System.out.println(asByte(new Character('7'))); // ошибка компиляции
     }
}
```

Объекты типа **Integer** (**int** будет в него упакован) и **Float** являются подклассами абстрактного класса **Number**, поэтому компиляция проходит без затруднений. Класс **Character** не обладает вышеуказанным свойством, и его объект не может передаваться в метод **asByte(T num)**.

Методы с переменным числом параметров

Возможность передачи в метод нефиксированного числа параметров позволяет отказаться от предварительного создания массива объектов для его последующей передачи в метод.

/ пример # 14: определение количества параметров метода: DemoVarargs.java */*

```
package chapt03;

public class DemoVarargs {
      public static int getArgCount(Integer... args) {
             if (args.length == 0)
                   System.out.print("No arg=");
             for (int i : args)
                   System.out.print("arg:" + i + "  ");
             return args.length;
      }
```