МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧЕРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»

Факультет информационных технологий и программирования

**Лабораторная работа № 3**

Выполнил студент группы № M3307  
Бойцов Виталий

САНКТ-ПЕТЕРБУРГ  
2019

# Практика к Лекции 3

# Упражнение 1-1 Использование конструкторов.

Цель упражнения: Изучить предназначение конструкторов.

Описание упражнения: В этом упражнении вы добавите в классы проекта StockListProject необходимые конструкторы.

1. Добавьте в класс GenericItem три конструктора:
   * 1. public GenericItem(String name, float price, Category category)
     2. public GenericItem(String name, float price, GenericItem analog)
     3. public GenericItem()
2. Добавьте в класс FoodItem три конструктора:
   * 1. public FoodItem(String name, float price, FoodItem analog, Date date, short expires)
     2. public FoodItem(String name, float price, short expires)
     3. public FoodItem(String name)
3. Частные конструкторы класса FoodItem (№ 2 и 3) должны обращаться к общему конструктору   
   (№ 1), передавая ему часть параметров в виде значений по умолчанию.
4. Добавьте в класс GenericItem статическое поле static int currentID, хранящее максимальный назначенный ID товара в текущей сессии.
5. Добавьте в конструкторы класса GenericItem строку, автоматически инициализирующую поле ID товара очередным свободным номером: this.ID = GenericItem.currentID++;
6. public class GenericItem {  
    protected static int *currentID* = 1;  
    protected int ID;  
    protected String name;  
    protected float price;  
    protected Category type = Category.*GENERAL*;  
    protected GenericItem analog;  
     
    public GenericItem(String name, float price, Category type) {  
    this.ID = GenericItem.*currentID*++;  
    this.name = name;  
    this.price = price;  
    this.type = type;  
    }  
     
    public GenericItem(String name, float price, GenericItem analog) {  
    this.name = name;  
    this.price = price;  
    this.analog = analog;  
    }  
     
    public GenericItem() {}

# Упражнение 2-1. Обобщенное программирование

Цель упражнения: Изучить возможности, предоставляемые настраиваемыми типами.

* Описание упражнения: В этом упражнении вы воспользуетесь возможностями generics для создания настраиваемых типов.

1. Требуется создать класс, в котором есть метод sum для получения суммы числового массива. Точный тип элементов массива не определен, известно только, что они являются наследниками числового типа Number.
   1. Создать класс U0901WorkArray. В заголовке класса должно быть указано ограничение по типу Number - <T extends Number>
   2. На уровне экземпляра класса должна быть объявлена переменная arrNums – массив с типом, указанным в ограничении:  
      T[] arrNums;
   3. В классе требуется объявить конструктор с параметром numP – массивом, с типом соответствующим ограничению. В этом конструкторе переменной arrNums присваивается ссылка входного параметра:  
      arrNums=numP;
   4. Создаем метод sum без входных параметров, но с возвращаемым типом – double. (Чтобы хватило на любого наследника типа Number).
   5. В методе sum объявляется переменная doubleWork типа double, в которую в цикле for будут инкрементироваться значения массива arrNums.
   6. Значение переменной doubleWork возвращается из метода sum.
   7. Создать класс U0901Main c методом main для проверки работоспособности класса U0901WorkArray.
   8. В методе main этого класса объявить 4 переменные:
      1. Массив Integer-значений intArr и заполнить его несколькими значениями:  
         Integer intArr[]={10,20,15}
      2. Массив Float-значений floatArr и заполнить его произвольными значениями в цикле for или while.
      3. Экземпляр класса U0901WorkArray с именем insWorkArrayInt и инициировать ее конструктором, в котором в качестве параметра передается массив intArr.
      4. Экземпляр класса U0901WorkArray с именем insWorkArrayFloat и инициировать ее конструктором, в котором в качестве параметра передается массив floatArr.
   9. Для переменных-экземпляров класса U0901WorkArray вызвать метод sum и вывести на экран полученное значение.
   10. При желании, можно проверить работоспособность ограничения – создать массив String‑значений и передать его в качестве параметра в конструктор экземпляра класса U0901WorkArray. Должна быть ошибка времени компиляции.
2. public class U0901Main {  
    public static void main(String[] args) {  
    Integer intArr[]={10,20,15};  
    Float floatArr[] = new Float[4];  
    int i=0;  
    while (i<4){  
    floatArr[i] = (float)i+10\*3/8;  
    i++;  
    }  
    U0901WorkArray insWorkArrayInt = new U0901WorkArray(intArr);  
    U0901WorkArray insWorkArrayFloat = new U0901WorkArray(floatArr);  
    System.*out*.println(insWorkArrayInt.sum());  
    System.*out*.println(insWorkArrayFloat.sum());  
     
    //Упражнение 3-1  
    String line = "Конфеты ’Маска’;45;120";  
    String[] item\_fld = line.split(";");  
    FoodItem foodItem =  
    new FoodItem(item\_fld[0], Float.*valueOf*(item\_fld[1]), Short.*valueOf*(item\_fld[2]));  
    //foodItem.printAll();  
    // доп  
    String line2 = "name [Конфеты Маска] price [45] expires [120]";  
    String[] item\_fld2 = line.split("\\s+");  
    for (String x : item\_fld2)  
    System.*out*.println(x);  
     
    }  
     
   }
3. package com.company;  
     
   public class U0901WorkArray<T extends Number> {  
    T[] arrNums;  
     
    public U0901WorkArray(T[] numP) {  
    this.arrNums = numP;  
    }  
     
    double sum () {  
    double doubleWork=0;  
    for (Number n: arrNums)  
    doubleWork += n.doubleValue();  
    return doubleWork;  
    }  
     
    private static <T extends Number> double sum (T[] arr) {  
    double sum=0;  
    for (Number n : arr)  
    sum += n.doubleValue();  
    return sum;  
    }  
     
   }