МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

Санкт-Петербургский национальный исследовательский университет информационных технологий механики и оптики

Мегафакультет трансляционных информационных технологий

Факультет информационных технологий и программирования

Лабораторная работа № 3

По дисциплине «Технологии программирования»

Выполнил студент группы №M3209
Бабурин Тимур

Проверил
Собенников Виктор Леонидович

САНКТ-ПЕТЕРБУРГ
2021

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++;`

Артефакты выполнения упражнения:

```java
public class GenericItem {
    public static int currentID;
    public int ID;
    public String name;
    public float price;
    public GenericItem analog;
    public Category itemCategory = Category.GENERAL;

    public GenericItem(String name, float price, Category category) {
        this.name = name;
        this.price = price;
        this.itemCategory = category;
        this.ID = GenericItem.currentID++;
    }
    public GenericItem(String name, float price, GenericItem analog) {
        this.name = name;
        this.price = price;
        this.analog = analog;
        this.ID = GenericItem.currentID++;
    }
    public GenericItem() { this.ID = GenericItem.currentID++; }
```

```java
import java.util.Date;

public class FoodItem extends GenericItem {
    Date dateOfIncome; // дата производства
    short expires; // срок годности

    public FoodItem(String name, float price, FoodItem analog, Date date, short expires) {
        this.name = name;
        this.price = price;
        this.analog = analog;
        this.dateOfIncome = date;
        this.expires = expires;
    }

    FoodItem(String name, float price, short expires) { this(name, price, analog: null, date: null, expires); }

    public FoodItem(String name) {
        this(name, price: 0, analog: null, date: null, (short) 0);
    }

    public FoodItem() { this( name: null, price: 0, analog: null, date: null, (short) 0); }
```

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.

6) 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` с 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.

Артефакты выполнения упражнения:

```
class u0901WorkArray<T extends Number> {
    T[] arrnums;

    public u0901WorkArray(T[] numP) { this.arrnums = numP; }

    double sum() {
        double doubleWork = 0.0D;

        for(int i = 0; i < this.arrnums.length; ++i) {
            doubleWork += Double.parseDouble(String.valueOf(this.arrnums[i]));
        }

        return doubleWork;
    }
}
```

```
public static void main(String[] args) {
    Integer intArr[] = {10, 25, 15};

    Float[] floatArr = new Float[10];
    for (int i = 0; i < floatArr.length; i++) {
        floatArr[i] = (float)Math.round(Math.random() * 10);
    }

    u0901WorkArray insWorkArrayInt = new u0901WorkArray(intArr);
    u0901WorkArray insWorkArrayFloat = new u0901WorkArray(floatArr);

    System.out.println("Sum of int:" + insWorkArrayInt.sum());
    System.out.println("Sum of float:" + insWorkArrayFloat.sum());
```

```
Sum of int:50.0
Sum of float:45.0
```

Упражнение 3-1. Работа со строками

Цель упражнения: Изучить методы класса `String`.

Описание упражнения: В этом упражнении вы воспользуетесь возможностями класса `String` для разбора строки текста. Строка будет представлять собой структурированное описание товара, на основе которого требуется создать экземпляр класса `FoodItem`. Полученный алгоритм будет в дальнейшем использован для массовой загрузки описаний товаров из текстового файла.

7) Пусть имеется текстовая строка `"Конфеты 'Маска';45;120"`, хранящая информацию о пищевом товаре в формате `<name>;<price>;<expires>`. Объявите в методе `main` класса `Main` переменную `line` типа `String`. Присвойте ей значение `"Конфеты 'Маска';45;120"`.
8) Объявите массив текстовых строк `item_fld`.
9) Разбейте строку `line` на поля (без разделителей) и заполните значениями полей массив `item_fld` (используйте метод `split()` класса `String`).
10) Создайте новый объект класса `FoodItem` на основании элементов массива `item_fld[]`. Вызовите у этого объекта метод `printAll`, чтобы убедиться в его корректном создании.

Артефакты выполнения упражнения:

```
String line = "Конфеты 'Маска';45;120";
String item_fld[] = line.split( regex: ";");
System.out.println(Arrays.toString(item_fld));
FoodItem itemFromString = new FoodItem(item_fld[0], Float.parseFloat(item_fld[1]), Short.parseShort(item_fld[2]));
itemFromString.printAll();
```

```
[Конфеты 'Маска', 45, 120]
ID: 0 , Name: Конфеты 'Маска' , price: 45.00, category: GENERAL       , dateOfIncome: null.null.null, expires: 120
```