

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 № 2

По дисциплине «Технологии программирования»

Выполнил студент группы №М32091

Бабурин Тимур

Проверил

Собенников Виктор Леонидович

САНКТ-ПЕТЕРБУРГ
2021

УПРАЖНЕНИЕ 1 -1. СОЗДАНИЕ СОБСТВЕННЫХ КЛАССОВ

Цель упражнения: Научиться создавать собственные классы, описывать их структуру и использовать в программе.

Описание упражнения: В этом упражнении вам предлагается реализовать класс `GenericItem`, описывающий отдельный товар из интернет-магазина. Каждый товар характеризуется:

- 1) Уникальным числовым идентификатором
- 2) Наименованием
- 3) Ценой

Реализуйте в новом проекте класс `GenericItem` и создайте несколько его экземпляров.

- 1) Создайте в Eclipse новый проект `StockListProject`
- 2) Создайте в проекте `StockListProject` новый класс `GenericItem` со

следующими полями экземпляра:

```
public int ID; // ID товара
public String name; // Наименование товара
public float price; //Цена товара
```

- 2) Добавьте в класс `GenericItem` метод `printAll()`, выводящий на экран значения всех полей экземпляра класса `GenericItem`:

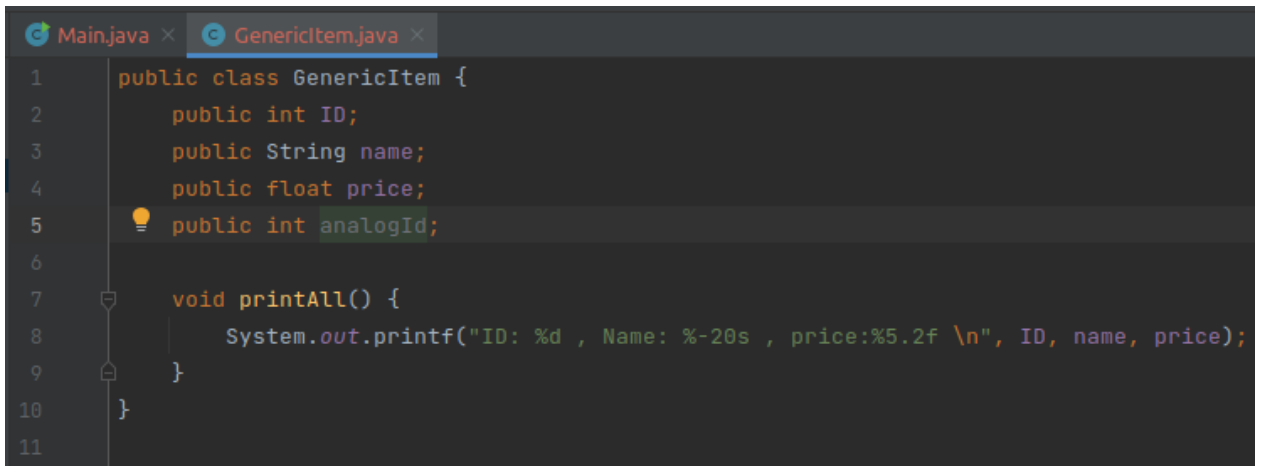
```
class GenericItem {
    public int ID;
    public String name;
    public float price;
    void printAll(){
        System.out.printf("ID: %d , Name: %-20s , price:%5.2f\n", ID, name, price);
    }
}
```

- 3) Создайте стартовый класс по имени `Main`. Добавьте в этот класс метод `main`. В методе `main` создайте три экземпляра класса `GenericItem` и присвойте их полям различные значения. Распечатайте значения полей для всех экземпляров методом `printAll()`;
- 4) (Опционально) Предположим, что для каждого товара необходимо хранить информацию о том, какой товар является его аналогом. Подумайте, какое поле

необходимо добавить в класс `GenericItem` для хранения такой информации.

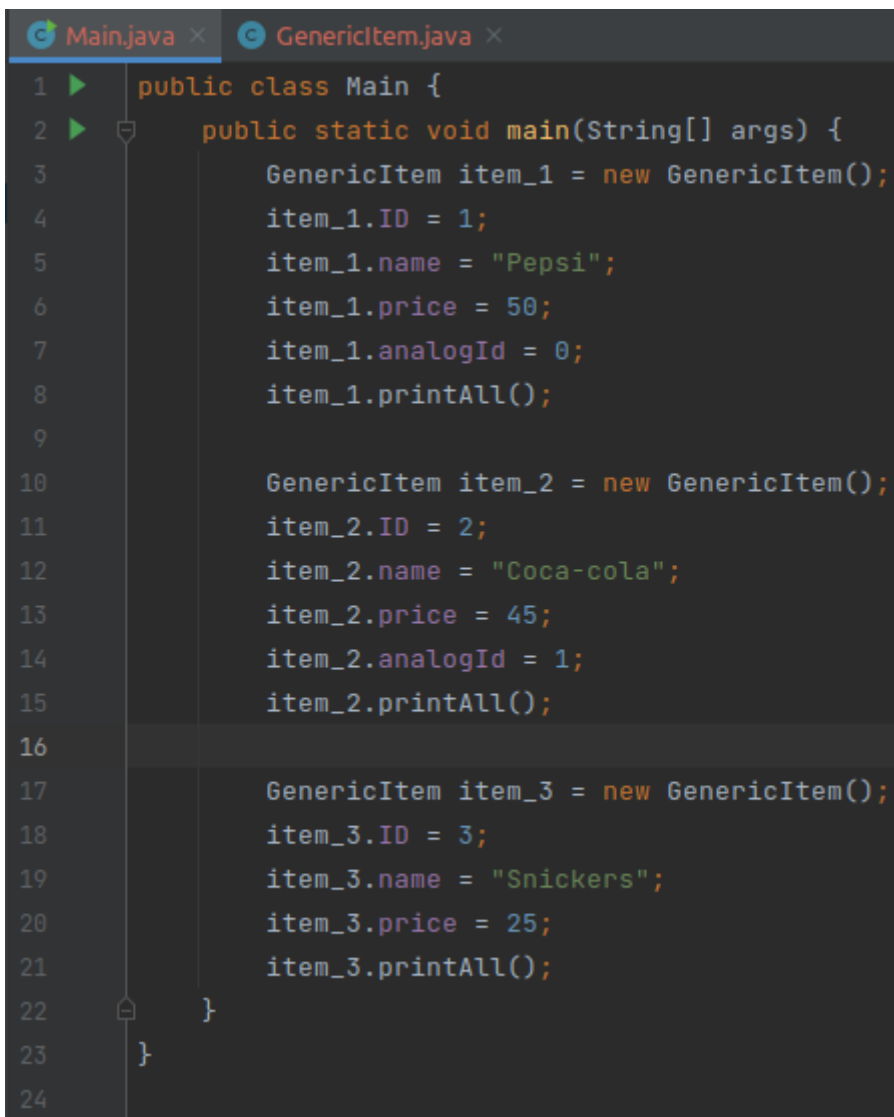
Внесите в класс `GenericItem` соответствующие изменения.

Класс `GenericItem`:



```
1 public class GenericItem {
2     public int ID;
3     public String name;
4     public float price;
5     public int analogId;
6
7     void printAll() {
8         System.out.printf("ID: %d , Name: %-20s , price:%5.2f \n", ID, name, price);
9     }
10 }
11
```

Класс `Main`:



```
1 public class Main {
2     public static void main(String[] args) {
3         GenericItem item_1 = new GenericItem();
4         item_1.ID = 1;
5         item_1.name = "Pepsi";
6         item_1.price = 50;
7         item_1.analogId = 0;
8         item_1.printAll();
9
10        GenericItem item_2 = new GenericItem();
11        item_2.ID = 2;
12        item_2.name = "Coca-cola";
13        item_2.price = 45;
14        item_2.analogId = 1;
15        item_2.printAll();
16
17        GenericItem item_3 = new GenericItem();
18        item_3.ID = 3;
19        item_3.name = "Snickers";
20        item_3.price = 25;
21        item_3.printAll();
22    }
23 }
24
```

Для хранения информации об аналоге можно использовать поле типа `int` в котором будет храниться `id` аналога, так как у товаров уникальный `id`.

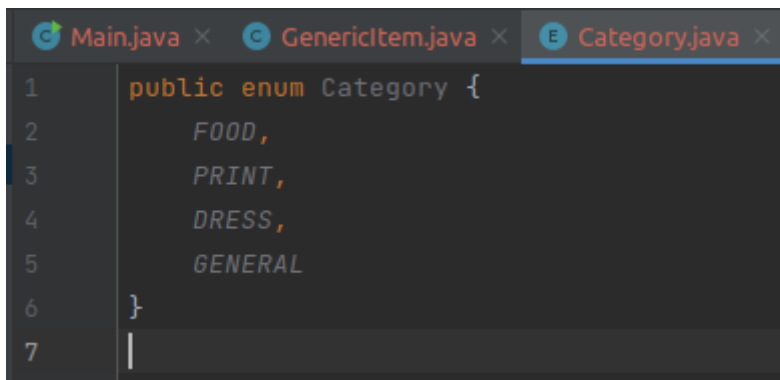
Упражнение 1-2. Создание перечислений.

Цель упражнения: Научиться работать с перечислениями.

Описание упражнения: Необходимо добавить в класс `GenericItem` поле перечислимого типа, характеризующее категорию товара (пищевой, одежда, печатная продукция и т.д.)

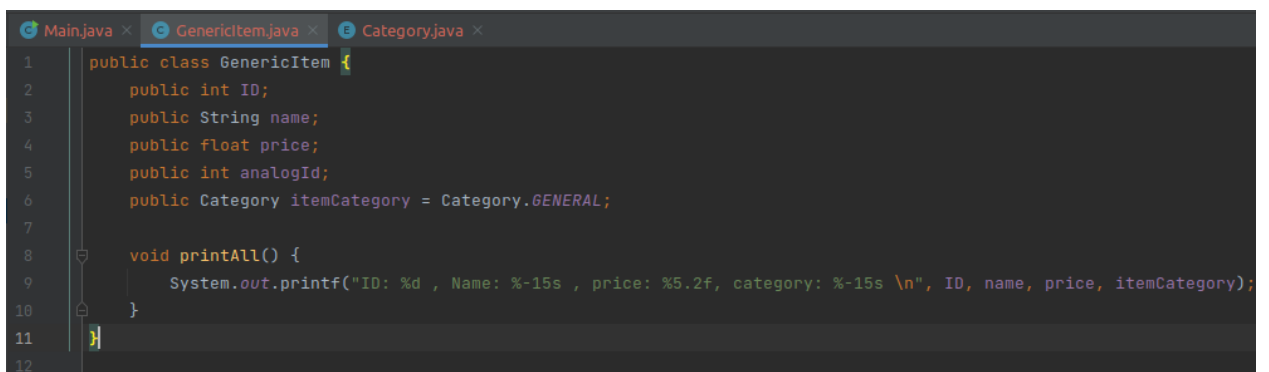
- 1) Создайте в проекте `StockListProject` новое перечисление `Category` со следующими значениями: `FOOD` (пищевой), `PRINT` (печатная продукция), `DRESS` (одежда), `GENERAL` (иная категория)

```
public enum Category { FOOD, PRINT, DRESS, GENERAL }
```



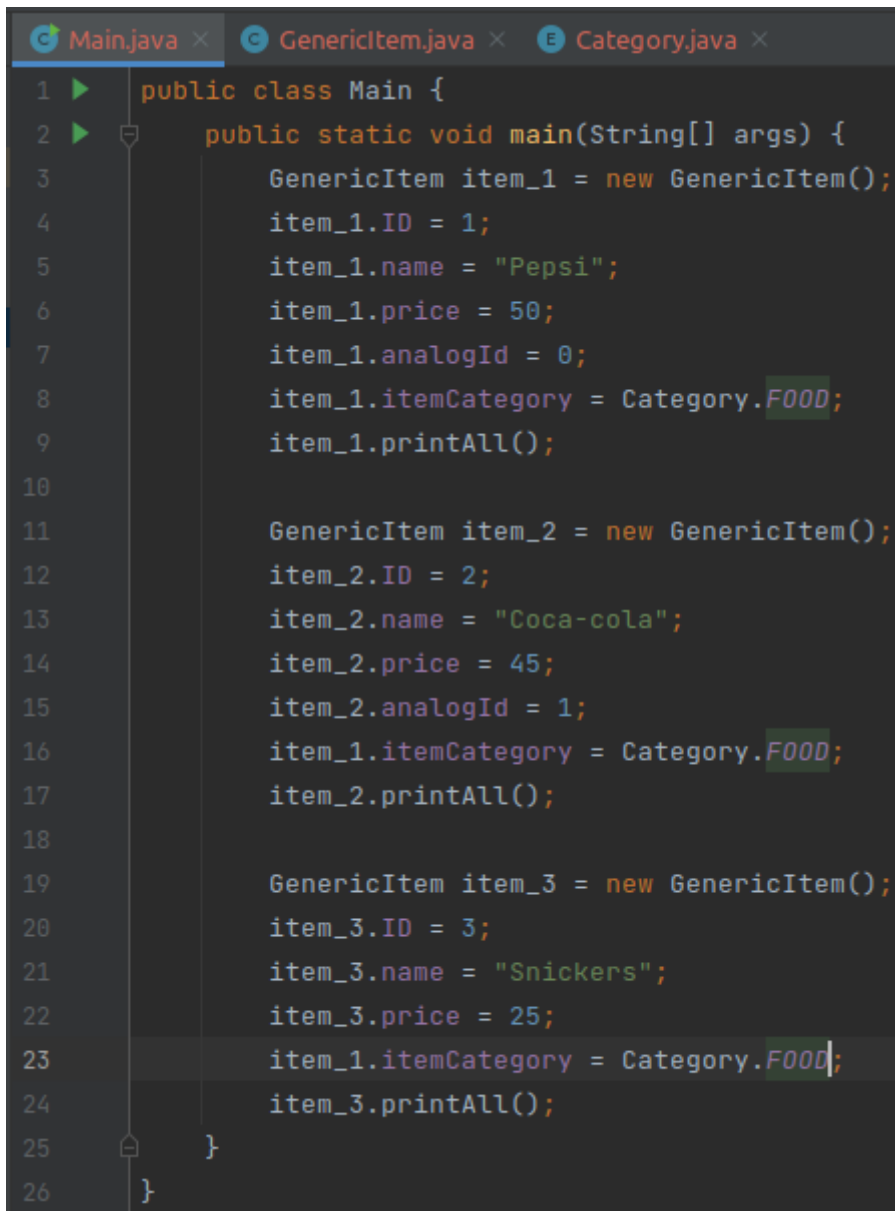
```
1 public enum Category {
2     FOOD,
3     PRINT,
4     DRESS,
5     GENERAL
6 }
7
```

- 5) Добавьте в класс `GenericItem`, поле типа `Category` со значением по умолчанию `GENERAL`. Внесите изменения в метод `printAll`, позволяющие печатать категорию товара.



```
1 public class GenericItem {
2     public int ID;
3     public String name;
4     public float price;
5     public int analogId;
6     public Category itemCategory = Category.GENERAL;
7
8     void printAll() {
9         System.out.printf("ID: %d , Name: %-15s , price: %5.2f, category: %-15s \n", ID, name, price, itemCategory);
10    }
11
12 }
```

Класс Main:



```
1  public class Main {
2      public static void main(String[] args) {
3          GenericItem item_1 = new GenericItem();
4          item_1.ID = 1;
5          item_1.name = "Pepsi";
6          item_1.price = 50;
7          item_1.analogId = 0;
8          item_1.itemCategory = Category.FOOD;
9          item_1.printAll();
10
11         GenericItem item_2 = new GenericItem();
12         item_2.ID = 2;
13         item_2.name = "Coca-cola";
14         item_2.price = 45;
15         item_2.analogId = 1;
16         item_1.itemCategory = Category.FOOD;
17         item_2.printAll();
18
19         GenericItem item_3 = new GenericItem();
20         item_3.ID = 3;
21         item_3.name = "Snickers";
22         item_3.price = 25;
23         item_1.itemCategory = Category.FOOD;
24         item_3.printAll();
25     }
26 }
```

УПРАЖНЕНИЕ 2-1. ПРИМЕНЕНИЕ НАСЛЕДОВАНИЯ

Цель упражнения: Научиться использовать механизмы наследования в Java.

Описание упражнения: В этом упражнении вы примените механизмы наследования языка Java для упрощения структуры программы и уменьшения объема кода.

- 1) Унаследуйте от класса `GenericItem` (см. упражнения 6-1, 6-2) классы

`FoodItem`, и `TechnicalItem` со следующими характеристиками:

- 1) Класс `FoodItem` имеет дополнительные поля:

- 1) `Date dateOfIncome;` // дата производства
- 2) `short expires;` // срок годности

- 2) Класс `TechnicalItem` имеет дополнительные поля

- 1) `short warrantyTime;` // гарантийный срок (суток)

- 6) Перекройте метод `printAll` в обоих наследниках так, чтобы он выводил на экран помимо общих индивидуальные характеристики объекта.
- 7) В классе `Main` создайте экземпляры классов `FoodItem` и `TechnicalItem`, поместите их в один массив. Переберите в цикле элементы массива и выведите на экран информацию об этих элементах с помощью метода `printAll`.
- 8) Скомпилируйте и выполните проект.

`FoodItem`:

```
import java.util.Date;

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

```
@Override
void printAll() {
    System.out.printf("ID: %d , Name: %-15s , price: %5.2f, category: %-15s, dateOfIncome: %ty.%S$tm.%S$td, expires: %d \n", ID, name, price, itemCategory, dateOfIncome, expires);
}
```

`TechnicalItem`:

```
public class TechnicalItem extends GenericItem{
    short warrantyTime; // гарантийный срок (суток)
```

```
@Override
void printAll() {
    System.out.printf("ID: %d , Name: %-15s , price: %5.2f, category: %-15s, warrantyTime: %d \n", ID, name, price, itemCategory, warrantyTime);
}
```

Один массив:

```
ArrayList<GenericItem> items = new ArrayList<>();  
items.add(item_4);  
items.add(item_5);  
for (GenericItem item : items) {  
    item.printAll();  
}
```

УПРАЖНЕНИЕ 2-2(ОПЦИОНАЛЬНО). ИСПОЛЬЗОВАНИЕ МЕТОДОВ КЛАССА OBJECT.

Цель упражнения: Изучить полезные методы, предоставляемые классом `Object`.

Описание упражнения: В этом упражнении вы воспользуетесь функциями класса `Object` для сравнения и копирования объекта.

- 1) Добавьте в классы `GenericItem`, `FoodItem` и `TechnicalItem` реализацию метода

```
public boolean equals(Object o).
```

- 9) Создайте два экземпляра класса `FoodItem`. Сравните их с помощью метода `equals`. Выведите на экран результат сравнения.

- 10) Добавьте в классы `GenericItem`, `FoodItem` и `TechnicalItem` реализацию метода

```
public Object clone().
```

 Клонировать один из ранее созданных экземпляров класса `FoodItem`. Сравните с помощью метода `equals` оригинал и его клон.

- 11) Добавьте в классы `GenericItem`, `FoodItem` и `TechnicalItem` реализацию метода

```
public String toString( ).
```

`Equals:`

`GenericItem:`

```
@Override
public boolean equals(Object o) {
    // ссылаются ли оба объекта на один и тот же экземпляр
    if (this == o)
        return true;

    // имеет ли "o" значение экземпляра, а затем сравнивает каждое отдельное поле, необходимое для определения того, равны ли оба объекта
    else if (o instanceof GenericItem)
    {
        GenericItem p = (GenericItem) o;
        return (this.ID == p.ID)
            && (this.itemCategory == p.itemCategory)
            && (this.name.equals(p.name))
            && (this.analog == p.analog)
            && (this.price == p.price);
    }
    return false;
}
```


FoodItem:

```
@Override
public boolean equals(Object o) {
    if (this == o)
        return true;

    else if (o instanceof GenericItem)
    {
        FoodItem p = (FoodItem) o;
        return (this.ID == p.ID)
            && (this.itemCategory == p.itemCategory)
            && (this.name.equals(p.name))
            && (this.analog == p.analog)
            && (this.price == p.price)
            && (this.expires == p.expires)
            && (this.dateOfIncome.equals(p.dateOfIncome));
    }
    return false;
}
```

TechnicalItem:

```
@Override
public boolean equals(Object o) {
    if (this == o)
        return true;

    else if (o instanceof GenericItem)
    {
        TechnicalItem p = (TechnicalItem) o;
        return (this.ID == p.ID)
            && (this.itemCategory == p.itemCategory)
            && (this.name.equals(p.name))
            && (this.analog == p.analog)
            && (this.price == p.price)
            && (this.warrantyTime == p.warrantyTime);
    }
    return false;
}
```

Проверка:

```
if(item_5.equals(item_5.clone())) {  
    System.out.printf("They are equal!\n");  
    System.out.printf(item_5.toString() + "\n");  
    System.out.printf(item_5.clone().toString() + "\n");  
}
```

Clone:

GenericItem:

```
@Override  
public GenericItem clone() {  
    GenericItem p = new GenericItem();  
    p.ID = this.ID;  
    p.analog = this.analog;  
    p.name = this.name;  
    p.price = this.price;  
    p.itemCategory = this.itemCategory;  
    return p;  
}
```

TechnicalItem:

```
@Override  
public TechnicalItem clone() {  
    TechnicalItem p = new TechnicalItem();  
    p.ID = this.ID;  
    p.analog = this.analog;  
    p.name = this.name;  
    p.price = this.price;  
    p.itemCategory = this.itemCategory;  
    p.warrantyTime = this.warrantyTime;  
    return p;  
}
```

FoodItem:

```
@Override
public FoodItem clone() {
    FoodItem p = new FoodItem();
    p.ID = this.ID;
    p.analog = this.analog;
    p.name = this.name;
    p.price = this.price;
    p.itemCategory = this.itemCategory;
    p.expires = this.expires;
    p.dateOfIncome = this.dateOfIncome;
    return p;
}
```

Проверка:

```
if(item_5.equals(item_5.clone())) {
    System.out.printf("They are equal!\n");
    System.out.printf(item_5.toString() + "\n");
    System.out.printf(item_5.clone_analog().toString() + "\n");
}
```

ToString:

```
@Override
public String toString() {
    return "GenericItem{" +
        "ID=" + ID +
        ", name='" + name + '\'' +
        ", price=" + price +
        ", analog=" + analog +
        ", itemCategory=" + itemCategory +
        '}';
}
```


УПРАЖНЕНИЕ 2-3(ОПЦИОНАЛЬНО). РАСШИРЕННОЕ КЛОНИРОВАНИЕ.

- 1) Измените реализацию метода `clone` в классе `GenericItem` так, чтобы при клонировании товара клонировался его аналог.

CloneAnalog:

```
public GenericItem clone_analog() {  
    return analog.clone();  
}
```