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 № 2**

Выполнил студент группы № M3307  
Бойцов Виталий

САНКТ-ПЕТЕРБУРГ  
2019

# 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
3. 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);

}  
 }

1. 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();
2. (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.
3. public class GenericItem {  
    private int ID;  
    private String name;  
    private float price;  
    private Category category = Category.*GENERAL*;  
    public GenericItem analog;  
     
    private static int *currentID* = 0;  
     
     
    public GenericItem(String name, float price, Category category) {  
    this.ID = GenericItem.*currentID*++;  
    this.name = name;  
    this.price = price;  
    this.category = category;  
    }  
     
    public GenericItem(String name, float price, GenericItem analog) {  
    this.ID = GenericItem.*currentID*++;  
    this.name = name;  
    this.price = price;  
    this.analog = analog;  
    }  
     
    public GenericItem() {  
    this.ID = GenericItem.*currentID*++;  
    }  
     
    public int getID() {  
    return ID;  
    }  
     
    public void setID(int ID) {  
    this.ID = ID;  
    }  
     
    public String getName() {  
    return name;  
    }  
     
    public void setName(String name) {  
    this.name = name;  
    }  
     
    public float getPrice() {  
    return price;  
    }  
     
    public void setPrice(float price) {  
    this.price = price;  
    }  
     
    public Category getCategory() {  
    return category;  
    }  
     
    public void setCategory(Category category) {  
    this.category = category;  
    }  
     
    public static int getCurrentID() {  
    return *currentID*;  
    }  
     
    public static void setCurrentID(int currentID)  
     
    {  
    GenericItem.*currentID* = currentID;  
    }  
     
     
    @Override  
    public boolean equals(Object object) {  
    if (this == object) return true;  
    if (!(object instanceof GenericItem)) return false;  
    if (!super.equals(object)) return false;  
    GenericItem that = (GenericItem) object;  
    return ID == that.ID &&  
    Float.*compare*(that.price, price) == 0 &&  
    name.equals(that.name) &&  
    category.equals(that.category);  
    }  
      
     
    @Override  
    public String toString() {  
    return "GenericItem{" +  
    "ID=" + ID +  
    ", name='" + name + '\'' +  
    ", price=" + price +  
    ", category=" + category +  
    '}';  
    }  
     
     
    public Object clone() {  
    try {  
    GenericItem clone = (GenericItem) super.clone();  
    clone.ID = this.ID;  
    clone.name = this.name;  
    clone.price = this.price;  
    clone.category = this.category;  
    clone.analog = this.analog;  
    return clone;  
    } catch (CloneNotSupportedException e) {  
    throw new RuntimeException(e);  
    }  
    }  
     
    public void printAll() {  
    System.*out*.printf("ID: %d , Name: %-20s , price:%.2f , category:%-10s \n", ID, name, price, category);  
    }  
   }

# 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 }
2. Добавьте в класс GenericItem, поле типа Category со значением по умолчанию GENERAL. Внесите изменения в метод printAll, позволяющие печатать категорию товара.
3. public enum Category { *FOOD*, *PRINT*, *DRESS*, *GENERAL* }

# 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 TechicalItem 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)
2. Перекройте метод printAll в обоих наследниках так, чтобы он выводил на экран помимо общих индивидуальные характеристики объекта.
3. 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.
4. Скомпилируйте и выполните проект.
5. import java.util.Date;  
   import java.util.Objects;  
     
   public class FoodItem extends GenericItem {  
    private Date dateOfIncome;  
    private short expires;  
     
    public FoodItem() {  
    super();  
    }  
     
    public void setDateOfIncome(Date dateOfIncome) {  
    this.dateOfIncome = dateOfIncome;  
    }  
     
    public Date getDateOfIncome() {  
    return this.dateOfIncome;  
    }  
     
    public short getExpires() {  
    return expires;  
    }  
     
    public void setExpires(short expires) {  
    this.expires = expires;  
    }  
     
    public FoodItem(String name, float price, GenericItem analog, Date date, short expires) {  
    setName("name");  
    setPrice(52);  
    this.analog = analog;  
    this.dateOfIncome = date;  
    this.expires = expires;  
    }  
     
    public FoodItem(String name, float price, short expires) {  
    this(name, price,null, new Date(), expires);  
    }  
     
    public FoodItem(String name) {  
    this(name, 0,null, new Date(), (short) 0);  
    }  
     
    @Override  
    public boolean equals(Object object) {  
    if (this == object) return true;  
    if (!(object instanceof FoodItem)) return false;  
    if (!super.equals(object)) return false;  
    FoodItem foodItem = (FoodItem) object;  
    return expires == foodItem.expires &&  
    dateOfIncome.equals(foodItem.dateOfIncome);  
    }  
     
    @Override  
    public int hashCode() {  
    return Objects.*hash*(super.hashCode(), dateOfIncome, expires);  
    }  
     
    @Override  
    public java.lang.String toString() {  
    return "FoodItem{" +  
    "dateOfIncome=" + dateOfIncome +  
    ", expires=" + expires +  
    '}';  
    }  
     
    @Override  
    public Object clone() {  
    FoodItem clone = new FoodItem();  
    clone.dateOfIncome = this.dateOfIncome;  
    clone.expires = this.expires;  
    return clone;  
    }  
     
    public void printAll() {  
    super.printAll();  
    System.*out*.println("Data:" + dateOfIncome);  
    System.*out*.println("Expires:" + expires);  
     
    }  
   }

package com.company;  
  
import com.company.GenericItem;  
  
import java.util.Objects;  
  
public class TechnicalItem extends GenericItem {  
 private short warrantyTime;  
  
 public short getWarrantyTime() {  
 return warrantyTime;  
 }  
  
 public void setWarrantyTime(short warrantyTime) {  
 this.warrantyTime = warrantyTime;  
 }  
  
 @Override  
 public boolean equals(Object object) {  
 if (this == object) return true;  
 if (!(object instanceof TechnicalItem)) return false;  
 if (!super.equals(object)) return false;  
 TechnicalItem that = (TechnicalItem) object;  
 return warrantyTime == that.warrantyTime;  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(super.hashCode(), warrantyTime);  
 }  
  
 @Override  
 public java.lang.String toString() {  
 return "TechnicalItem{" +  
 "warrantyTime=" + warrantyTime +  
 '}';  
 }  
  
 @Override  
 public Object clone() {  
 TechnicalItem clone = (TechnicalItem) super.clone();  
 clone.warrantyTime = this.warrantyTime;  
 return super.clone();  
 }  
  
 public void printAll() {  
 super.printAll();  
 System.*out*.println("WarrantyTime:" + warrantyTime);  
  
 }  
}