**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ЛАБОРАТОРНАЯ РАБОТА №3

по дисциплине: **«Визуальные средства разработки программных**

**приложений»**

на тему: **Использование шаблонов проектирования**

Выполнил: студент гр. ИТП-31

Дашкевич Д.А.

Принял: ассистент

Белявский Е. В.

Гомель 2020

**Цель**: научиться использовать шаблоны проектирования.

**Ход работы**

**Вариант 2**

# **Задание:**

Создать приложение, которое должно реализовывать как минимум два шаблона проектирования, за исключением следующих: Одиночка, Абстрактная фабрика, Фабричный метод.

**Используемые шаблоны проектирования:**

Строитель – это порождающий шаблон проектирования предоставляет способ создания составного объекта. Отделяет конструирование сложного объекта от его представления так, что в результате одного и того же процесса конструирования могут получаться разные представления.

Неизменяемый интерфейс – это шаблон для проектирования неизменяемого объекта. Неизменяемый интерфейс включает в себя определение типа, который не предоставляет никаких методов, которые могут изменить состояние. Объекты, на которые ссылается этот тип, выглядят как неизменяемые.

**Результат работы программы:**

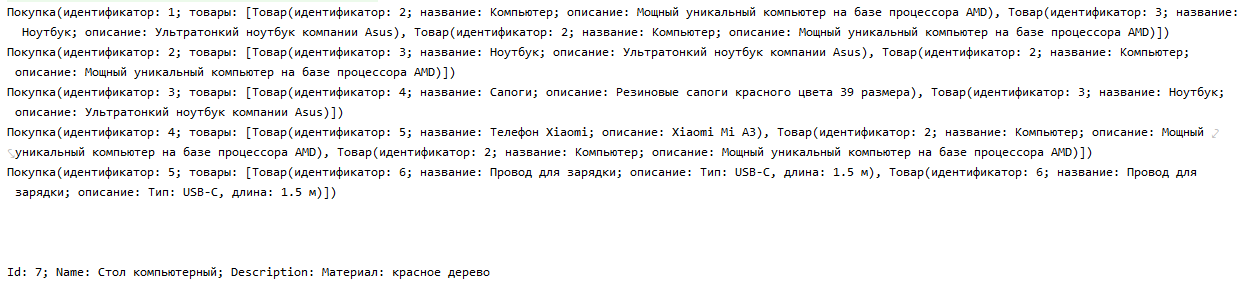
****

Рисунок 1 – Вывод в консоль результата работы программы

**Вывод:** в результате выполнения лабораторной работы были изучены шаблоны проектирования в *Java* и разработана программа, которая использует их.

**Приложение А**

**Исходный код программы**

**Runner.java:**

**public class** Runner {  
 **public static void** main(String[] args) {  
  
 Payment.Product product0 = **new** Payment.Product(**"NULL"**, **"NULL"**);  
 Payment.Product product1 = **new** Payment.Product(**"Компьютер"**, **"Мощный уникальный компьютер на базе процессора AMD"**);  
 Payment.Product product2 = **new** Payment.Product(**"Ноутбук"**, **"Ультратонкий ноутбук компании Asus"**);  
 Payment.Product product3 = **new** Payment.Product(**"Сапоги"**, **"Резиновые сапоги красного цвета 39 размера"**);  
 Payment.Product product4 = **new** Payment.Product(**"Телефон Xiaomi"**, **"Xiaomi Mi A3"**);  
 Payment.Product product5 = **new** Payment.Product(**"Провод для зарядки"**, **"Тип: USB-C, длина: 1.5 м"**);  
  
 Payment payment1 = **new** Payment.Builder().add(product1).add(product2).add(product1).build();  
 Payment payment2 = **new** Payment.Builder().add(product2).add(product1).add(product0).remove(product0).build();  
 Payment payment3 = **new** Payment.Builder().add(product3).add(product2).add(product2).remove(product2).build();  
 Payment payment4 = **new** Payment.Builder().add(product4).add(product1).add(product1).build();  
 Payment payment5 = **new** Payment.Builder().add(product5).add(product5).build();  
  
 String payments = String.*join*(**"\n"**, payment1.toString(), payment2.toString(), payment3.toString(),  
 payment4.toString(), payment5.toString());  
  
 System.***out***.println(payments);  
  
 System.***out***.println(**"\n\n"**);  
 Payment.ImmutableProduct product6 = **new** Payment.Product(**"Стол компьютерный"**, **"Материал: красное дерево"**);  
 System.***out***.println(String.*format*(**"Id: %d; Name: %s; Description: %s"**,  
 product6.getId(), product6.getName(), product6.getDescription()));  
 }  
}

**Payment.java:**

**public class** Payment {  
  
 **private final int id**;  
 **private final** List<Product> **products**;  
  
 **private** Payment(**int** id, List<Product> products) {  
 **this**.**id** = id;  
 **this**.**products** = products;  
 }  
  
 **public int** getId() {  
 **return id**;  
 }  
  
 @Override  
 **public boolean** equals(Object o) {  
 **if** (**this** == o) **return true**;  
 **if** (o == **null** || getClass() != o.getClass()) **return false**;  
 Payment payment = (Payment) o;  
 **return id** == payment.**id** && **products**.equals(payment.**products**);  
 }  
  
 @Override  
 **public int** hashCode() {  
 **return** Objects.*hash*(**id**);  
 }  
  
 @Override  
 **public** String toString() {  
 **return** String.*format*(**"Покупка(идентификатор: %d; товары: %s)"**, **id**, **products**);  
 }  
  
 **public static class** Builder {  
 **private static int** *paymentCounter* = 1;  
  
 **private** List<Product> **productList** = **new** ArrayList<>();  
  
 **public** Payment.Builder add(Product product) {  
 **if** (product == **null**) {  
 **throw new** IllegalArgumentException(**"Product == null"**);  
 }  
 **productList**.add(product);  
 **return this**;  
 }  
  
 **public** Payment.Builder remove(Product product) {  
 **if** (product == **null**) {  
 **throw new** IllegalArgumentException(**"Product == null"**);  
 }  
 **productList**.remove(product);  
 **return this**;  
 }  
  
 **public** Payment build() {  
 **if** (**productList**.size() == 0) {  
 **throw new** IllegalStateException(**"productList.size() == 0"**);  
 }  
 **return new** Payment(*paymentCounter*++, **productList**);  
 }  
 }  
  
  
 **public interface** ImmutableProduct {  
 **int** getId();  
 String getName();  
 String getDescription();  
 }  
  
  
 **public static class** Product **implements** ImmutableProduct {  
  
 **private static int** *productCounter* = 1;  
  
 **private final int id** = *productCounter*++;  
 **private final** String **name**;  
 **private final** String **description**;  
  
 **public** Product(String name, String description) {  
 **this**.**name** = name;  
 **this**.**description** = description;  
 }  
  
 **public int** getId() {  
 **return id**;  
 }  
  
 **public** String getName() {  
 **return name**;  
 }  
  
 **public** String getDescription() {  
 **return description**;  
 }  
  
 @Override  
 **public boolean** equals(Object o) {  
 **if** (**this** == o) **return true**;  
 **if** (o == **null** || getClass() != o.getClass()) **return false**;  
 Product product = (Product) o;  
 **return id** == product.**id** &&  
 Objects.*equals*(**name**, product.**name**) &&  
 Objects.*equals*(**description**, product.**description**);  
 }  
  
 @Override  
 **public int** hashCode() {  
 **return** Objects.*hash*(**id**);  
 }  
  
 @Override  
 **public** String toString() {  
 **return** String.*format*(**"Товар(идентификатор: %d; название: %s; описание: %s)"**, **id**, **name**, **description**);  
 }  
 }  
}