Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Кафедра «Информатика»
кафедра

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 1

Java Core, наследование
Тема

Преподаватель _____ А.С. Черниговский
Подпись, дата Инициалы, Фамилия

Студент КИ19-17/1Б, №031939174 _____ А. К. Никитин
Номер группы, зачетной книжки Подпись, дата Инициалы, Фамилия

Красноярск 2021

1 **Цель**

Ознакомиться с механизмом наследования в языке Java. Повторить основные языковые конструкции языка Java.

2 Задачи

Для каждого варианта имеется набор из четырех сущностей. Необходимо выстроить иерархию наследования. В каждом классе (базовом и производных) должно быть минимум одно числовое и одно текстовое поле. При вводе числовых параметров **обязательна** проверка на число и на диапазон (даже если число может быть любое, проверку необходимо реализовать).

Для всех классов должны быть реализованы конструкторы (по умолчанию, с параметрами), методы equals(), hashCode(), toString();

Реализовать консольное Java-приложение, которое имеет простейшее пользовательское меню, состоящее как минимум из следующих пунктов:

- Добавить новый элемент. (Элементы должны добавляться в коллекцию элементов типа базового класса. Необходимо предусмотреть возможность добавления любого объекта производного класса в данную коллекцию.)
- Удалить элемент по индексу.
- Вывод всех элементов в консоль.
- Сравнение двух элементов на равенство (по индексам).
- Завершение работы приложения.

3 Описание варианта

Цепочка наследования «Продукты, фрукты, молочная продукция, мясо».

4 Исходные тексты программ

На листингах представлен код классов, реализующих задание.

Листинг 1 – Класс Product

```
public abstract class Product {
    private final String NAME;
    private double price;

    public Product(String productName, double price) {
        this.NAME = productName;
        this.price = price;
    }

    public abstract String toString();

    public 3oolean equals(Object o) {
        if (this == o) return true;
        if (o == null) return false;

        if (getClass() != o.getClass())
            return false;

        Product that = (Product) o;
        return Double.compare(that.getPrice(), this.getPrice()) == 0 &&
                that.getNAME().equals(this.getNAME());
    }

    public abstract int hashCode();

    public void setPrice(int price) {
        this.price = price;
    }

    public String getNAME() {
        return NAME;
    }

    public double getPrice() {
        return price;
    }
}
```

Листинг 2 – Класс Meat

```
public class Meat extends Product implements Weighable {
    private final double WEIGHT; // В килограммах
    public Meat(String productName, double price, double weight) {
        super(productName, price);
        this.WEIGHT = weight;
    }

    public double weigh() {
        return Math.round(this.WEIGHT * this.getPrice());
    }

    public double getWEIGHT() {
        return WEIGHT;
    }

    @Override
    public boolean equals(Object o) {
        if (!super.equals(o)) return false;
        Meat that = (Meat) o;
        return Double.compare(that.WEIGHT, this.WEIGHT) == 0;
    }

    @Override
    public int hashCode() {
        return Objects.hash(this.WEIGHT, this.getPrice(), this.getNAME());
    }

    @Override
    public String toString() {
        return "Название продукта: " + this.getNAME() + ";\n" +
                "Цена за кг: " + this.getPrice() + " рублей;\n" +
                "Вес: " + this.WEIGHT + " кг;\n" +
                "Итоговая цена: " + this.weigh() + " рублей.";
    }
}
```

Листинг 3 – Класс MilkProduct

```
public class MilkProduct extends Product {
    private final double FAT_CONTENT;
    public MilkProduct(String productName, double price, double fatContent) {
        super(productName, price);
```

```
        this.FAT_CONTENT = fatContent;
    }

    public double getFAT_CONTENT() {
        return FAT_CONTENT;
    }

    @Override
    public boolean equals(Object o) {
        if (!super.equals(o)) return false;
        MilkProduct that = (MilkProduct) o;
        return Double.compare(that.FAT_CONTENT, this.FAT_CONTENT) == 0;
    }

    @Override
    public int hashCode() {
        return Objects.hash(this.FAT_CONTENT, this.getPrice(), this.getNAME());
    }

    @Override
    public String toString() {
        return "Название продукта: " + this.getNAME() + ";\n" +
                "Цена: " + this.getPrice() + " рублей;\n" +
                "Жирность: " + this.FAT_CONTENT + "%.";
    }
}
```

Листинг 4 – Класс Fruit

```
public class Fruit extends Product implements Weighable {
    private final double WEIGHT; // В килограммах
    public Fruit(String productName, double price, double weight) {
        super(productName, price);
        this.WEIGHT = weight;
    }

    public double weigh() {
        return Math.round(this.WEIGHT * this.getPrice());
    }

    public double getWEIGHT() {
        return WEIGHT;
    }
```

```
    @Override
    public boolean equals(Object o) {
        if (!super.equals(o)) return false;
        Fruit that = (Fruit) o;
        return Double.compare(that.WEIGHT, this.WEIGHT) == 0;
    }

    @Override
    public int hashCode() {
        return Objects.hash(this.WEIGHT, this.getPrice(), this.getNAME());
    }

    @Override
    public String toString() {
        return "Название продукта: " + this.getNAME() + ";\n" +
                "Цена за кг: " + this.getPrice() + " рублей;\n" +
                "Вес: " + this.WEIGHT + " кг;\n" +
                "Итоговая цена: " + this.weigh() + " рублей.";
    }
}
```

Листинг 5 – Класс ShoppingCart

```
public class ShoppingCart {
    private final ArrayList<Product> products = new ArrayList<>();

    public void putProduct(Product product) {
        this.products.add(product);
    }

    public void removeProduct(Product product) {

        this.products.remove(product);
    }

    public double getTotalPrice() {
        double totalPrice = 0.;
        for (Product product: this.products) {
            if (product instanceof Weighable)
                totalPrice += ((Weighable) product).weigh();
            else
                totalPrice += product.getPrice();
        }
        return Math.round(totalPrice);
```

```
    }

    public boolean isInCart(Product custom_product) {
        for (Product array_product: this.products) {
            return custom_product.equals(array_product);
        }
        return false;
    }

    public ArrayList<Product> getProducts() {
        return products;
    }
}
```

Листинг 6 – Интерфейс Weighable

```
public interface Weighable {
    double weigh();
}
```

Листинг 7 – Класс Main

```
public class Main {

    public static int inputNat() {
        Scanner in = new Scanner(System.in);

        int x;
        while (true) {
            try {
                x = in.nextInt();
                in.nextLine();

                if (x < 0)
                    throw new java.util.InputMismatchException("Value is below
zero");
                break;

            } catch (java.util.InputMismatchException e) {
                System.out.println("Input the correct value");
                in.nextLine();
            }
        }
        return x;
    }
```

```
    public static double inputDouble(double bottomBound, double upperBound) {
        Scanner in = new Scanner(System.in);

        double x;
        while (true) {
            try {
                x = in.nextDouble();
                in.nextLine();

                if (x < bottomBound || x > upperBound)
                    throw new java.util.InputMismatchException("Incorrect
value!");
                break;

            } catch (java.util.InputMismatchException e) {
                System.out.println("Input the correct value!");
                in.nextLine();
            }
        }
        return x;
    }

    public static String inputString() {
        Scanner in = new Scanner(System.in);
        return in.nextLine();
    }

    public static Product chooseProduct() {
        while (true) {
            System.out.println("Введите, какой продукт вас интересует:");
            System.out.println("1. Молочный продукт;");
            System.out.println("2. Мясной продукт;");
            System.out.println("3. Фрукты;");
            int productChoice = inputNat();
            if (productChoice > 3)
                continue;

            System.out.println("Введите название продукта:");
            String productName = inputString();
            switch (Products.values()[productChoice - 1]) {
                case MILK -> {
```

```
                    System.out.println("Введите цену продукта:");
                    double price = inputDouble(0, Double.POSITIVE_INFINITY);

                    System.out.println("Введите жирность продукта (в
процентах):");
                    double fatContent = inputDouble(Double.NEGATIVE_INFINITY,
99.9);

                    return new MilkProduct(productName, price, fatContent);
                }

                case MEAT -> {
                    System.out.println("Введите цену продукта за кг:");
                    double price = inputDouble(0, Double.POSITIVE_INFINITY);

                    System.out.println("Введите вес продукта в кг:");
                    double weight = inputDouble(0, Double.POSITIVE_INFINITY);

                    return new Meat(productName, price, weight);
                }

                case FRUIT -> {
                    System.out.println("Введите цену продукта за кг:");
                    double price = inputDouble(0, Double.POSITIVE_INFINITY);

                    System.out.println("Введите вес продукта в кг:");
                    double weight = inputDouble(0, Double.POSITIVE_INFINITY);

                    return new Fruit(productName, price, weight);
                }
            }
        }
    }

    public static int inputIndex(Collection<Product> array) {
        int index;
        while (true) {
            index = inputNat();
            if (index < array.size() && index >= 0)
                break;
            System.out.println("Введите корректный индекс!");
        }
```

```
        return index;
    }

    public static void main(String[] args) {
        ShoppingCart shoppingCart = new ShoppingCart();
        while (true) {
            System.out.println("Введите команду:");
            System.out.println("1. Добавить новый элемент;");
            System.out.println("2. Удалить элемент по индексу;");
            System.out.println("3. Вывод всех элементов в консоль;");
            System.out.println("4. Сравнение двух элементов на равенство;");
            System.out.println("5. Завершение работы приложения.");

            int choice = inputNat();
            switch (UserMenu.values()[choice - 1]) {
                case ADD -> {
                    Product product = chooseProduct();
                    shoppingCart.putProduct(product);
                    System.out.println("Элемент добавлен");
                }

                case REMOVE -> {
                    Product product = chooseProduct();
                    shoppingCart.removeProduct(product);
                    System.out.println("Элемент удален");
                }

                case OUTPUT -> {
                    if (shoppingCart.getProducts().size() == 0) {
                        System.out.println("Корзина пуста.");
                        break;
                    }

                    System.out.println("-----------------------------------");
                    for (Product prod : shoppingCart.getProducts()) {
                        System.out.println(prod);
                        System.out.println("-----------------------------------
-");
                    }
                    System.out.println("-----------------------------------");
                    System.out.println("Итоговая цена: " +
shoppingCart.getTotalPrice());
```

```
                }

                case EQUALS -> {
                    ArrayList<Product> products = shoppingCart.getProducts();

                    System.out.println("Введите первый индекс:");
                    int index1 = inputIndex(products);
                    System.out.println("Введите второй индекс:");
                    int index2 = inputIndex(products);

                    if (products.get(index1).equals(products.get(index2)))
                        System.out.println("Продукты совпадают.");
                    else System.out.println("Продукты не совпадают.");
                }

                case EXIT -> {
                    System.exit(1);
                }

                default -> {
                    System.out.println("Введите значение от 1 до 5!");
                }
            }
        }
    }
}
```