**Ссылка на код в Гит:**

<https://github.com/ekbond/attestation.git> папка attestation01

**Постановка задачи:**

Необходимо реализовать приложение, принимающее список пользователей, продуктов и обрабатывающее покупку пользователя.

Подробное описание функционала приложения

1. Создать классы Покупатель (Person) и Продукт (Product). Характеристики Покупателя: имя, сумма денег и пакет с продуктами (массив объектов типа Продукт). Имя не может быть пустой строкой и не может быть короче 3 символов. Деньги не могут быть отрицательным числом.

Если Покупатель может позволить себе Продукт, то Продукт добавляется в пакет. Если у Покупателя недостаточно денег, то добавление не происходит. Характеристики Продукта: название и стоимость. Название продукта не может быть пустой строкой, оно должно быть. Стоимость продукта не может быть отрицательным числом.

2. Поля в классах должны быть private, доступ к полям осуществляется через геттеры и сеттеры или конструктор класса.

3. В классах переопределены методы toString(), equals(), hashcode().

4. Создать в классе App метод main и проверить работу приложения. Данные Покупателей и Продукты вводятся с клавиатуры, для считывания данных потребуется использовать класс Scanner и его метод nextLine(). Продукты в цикле выбираются покупателями по очереди и, пока не введено слово END, наполняется пакет.

5. Обработать следующие ситуации:

а. Если покупатель не может позволить себе продукт, то напечатайте соответствующее сообщение ("[Имя человека] не может позволить себе [Название продукта]").

б. Если ничего не куплено, выведите имя человека, за которым следует "Ничего не куплено".

в. В случае неверного ввода - сообщение: "Деньги не могут быть отрицательными", пустого имени - сообщение: "Имя не может быть пустым" или длина имени менее 3 символов – сообщение: "Имя не может быть короче 3 символов".

Программа реализуется в отдельной ветке git attestation/attestation01. При сохранении состояния программы (коммиты) пишется сообщение с описанием хода работы по задаче.  
В корне папки с программой должен быть файл .gitignore.  
Программа локально коммитится и публикуется в репозиторий GitHub на проверку.

Код:

Класс Person

import java.util.ArrayList;

import java.util.List;

import java.util.Objects;

public class Person {

    // Поля

    // (Условие: Поля сделать private)

    private String buyer;

    private int amount\_money;

    private List<Product> shopping\_bag;

    // Конструктор класса Покупатель

    public Person(String buyer, int amount\_money) {

        if (!setNameBuyer(buyer) || !setAmountMoney(amount\_money)) {

            System.exit(0);

        }

        this.shopping\_bag = new ArrayList<>();

    }

    // Свойства (геттеры и сеттеры)

    public String getNameBuyer() {

        return buyer;

    }

    public int getAmount\_money() {

        return amount\_money;

    }

    public boolean setNameBuyer(String buyer) {

        if (buyer == null || buyer.trim().isEmpty()) {

            System.out.println("Имя не может быть пустым");

            return false;

        }

        if (buyer.length() < 3) {

            System.out.println("Имя не может быть короче 3 символов");

            return false;

        }

        this.buyer = buyer;

        return true;

    }

    public boolean setAmountMoney(int amount\_money) {

        if (amount\_money < 0) {

            System.out.println("Деньги не могут быть отрицательными");

            return false;

        }

        this.amount\_money = amount\_money;

        return true;

    }

    // Метод для покупки товара, product - товар для покупки

    public boolean MoneyForBuying(Product product) {

        if (product.getPriceProduct() <= amount\_money) {    // Сравниваем цену продукта из класса Продкуты

            amount\_money -= product.getPriceProduct();      // Вычисляем (вычитаем) цену продукта из суммарных денег

            shopping\_bag.add(product);                      // Добавляем продукт в корзину

            return true;

        }

        return false;

    }

    // Переопределение методов

    @Override

    public String toString() {

        if (shopping\_bag.isEmpty()) {

            return buyer + " - Ничего не куплено";

        }

        return buyer + " - " + shopping\_bag.toString().replace("[", "").replace("]", "");

    }

    @Override

    public boolean equals(Object o) {

        if (this == o) {

            return true;

        }

        if (o == null || !(o instanceof Person)) {

            return false;

        }

        Person person = (Person) o; // Привести Object к типу Person

        // Осуществляется сравнение всех полей, которые были указаны в начале кода

        return this.buyer.equals(person.buyer) && this.amount\_money == person.amount\_money && this.shopping\_bag.equals(person.shopping\_bag);

    }

    @Override

    public int hashCode() {

    return Objects.hash(buyer, amount\_money, shopping\_bag);

    }

}

Класс Product

import java.util.ArrayList;

import java.util.Objects;

public class Product {

    // Поля

    // (Условие: Поля сделать private)

    private String name\_product;

    private int price\_product;

    // Конструктор класса Продукты

    public Product(String name\_product, int price\_product) {

        if (!setNameProduct(name\_product) || !setPriceProduct(price\_product)) {

            System.exit(0);

        }

    }

    // Свойства (геттеры и сеттеры)

    public String getNameProduct() {

        return name\_product;

    }

    public int getPriceProduct() {

        return price\_product;

    }

    public boolean setNameProduct(String name\_product) {

        if (name\_product == null || name\_product.trim().isEmpty()) {

            System.out.println("Название продукта не может быть пустым. Запустите программу снова и введите корретные данные.");

            return false;

        }

        this.name\_product = name\_product;

        return true;

    }

    public boolean setPriceProduct(int price\_product) {

        if (price\_product < 0) {

            System.out.println("Стоимость продукта не может быть отрицательной. Запустите программу снова и введите корретные данные.");

            return false;

        }

        this.price\_product = price\_product;

        return true;

    }

    // Переопределение методов

    @Override

    public String toString() {

        return name\_product;

    }

    @Override

    public boolean equals(Object o) {

        if (this == o) {

            return true;

        }

        if (o == null || !(o instanceof Product)) {

            return false;

        }

        Product product = (Product) o; // Привести Object к типу Product

        // Осуществляется сравнение всех полей, которые были указаны в начале кода

        return this.name\_product.equals(product.name\_product) && this.price\_product == product.price\_product;

    }

    @Override

    public int hashCode() {

    return Objects.hash(name\_product, price\_product);

    }

}

Класс App

import java.util.ArrayList;

import java.util.Scanner;

public class App {

    public static void main(String[] args) {

        // Информация о Покупателе и продуктах будет вводится с клавиатуры

        Scanner scanner = new Scanner(System.in, "cp866");

        // Списки для хранения покупателей и их денег, продуктов и их цена

        ArrayList<Person> people\_money = new ArrayList<>();

        ArrayList<Product> product\_price = new ArrayList<>();

        System.out.println("\nПриложение по обработке списка покупателей и продуктов");

        // Вводится информация о покупателях с клавиатуры

        System.out.println("\nВведите информацию о покупателях(-е): " );

        System.out.println("Формат: Имя = Сумма. Вводить следующие данные через ;");

        String input\_info  = scanner.nextLine();            // Запомнить данные, которые введены с клавиатуры

        String[] paths = input\_info.split(";");         // Цикл указан без явного использования индекса

        for (String i : paths) {

            try {

                String[] words = i.trim().split("=");     // Парсинг входящий строки по символу =

                String name\_person = words[0].trim();              // Убрать пробелы у имени покупателя

                int money\_person = Integer.parseInt(words[1].trim());   // Преобразовать второе значение списка (деньги) в число

                people\_money.add(new Person(name\_person, money\_person));   // Добавить в список значение [ (Имя, деньги), (Имя, деньги), ...]

            } catch (Exception ex) {

                System.exit(0);

            }

        }

        // Вводится информация о продуктах с клавиатуры

        System.out.println("\nВведите информацию о продуктах: ");

        System.out.println("Формат: Название = Цена.  Вводить следующие данные через ;");

        String input\_product\_list = scanner.nextLine();

        String[] paths\_prod = input\_product\_list.split(";");

        for (String k : paths\_prod) {

            try{

                String[] prod = k.trim().split("=");

                String name = prod[0].trim();

                int prod\_price = Integer.parseInt(prod[1].trim());

                product\_price.add(new Product(name, prod\_price));

            } catch (Exception ex) {

                System.exit(0);

            }

        }

        // Тестирование промежуточного результата

        //System.out.println(people\_money);

        //System.out.println(product\_price);

        // Вводится информация о покупках с клавиатуры и обработка

        System.out.println("\nВведите покупки каждого покупателя: " );

        System.out.println("Формат: Имя - Товар (для завершения введите END)");

        while (true) {

            String input\_person\_product = scanner.nextLine();

            if (input\_person\_product.equalsIgnoreCase("END"))

            break;

            String[] PerPro = input\_person\_product.split("-");

            String personName = PerPro[0].trim();

            String productName = PerPro[1].trim();

            // Перебор по совпадениям для имен

            Person person = null;

            for (Person p : people\_money) {

                if (p.getNameBuyer().equals(personName)) {

                    person = p;

                    break;

                }

            }

            // Перебор по совпадениям дли продуктов

            Product product = null;

            for (Product pr : product\_price) {

                if (pr.getNameProduct().equals(productName)) {

                    product = pr;

                    break;

                }

            }

            // Пытаемся купить продукт

            if (person.MoneyForBuying(product)) {

                System.out.println(person.getNameBuyer() + " купил " + product.getNameProduct());

            } else {

                System.out.println(person.getNameBuyer() + " не может позволить себе " + product.getNameProduct());

            }

        }

        System.out.println("\nИтоговые покупки:");

        for (Person person : people\_money) {

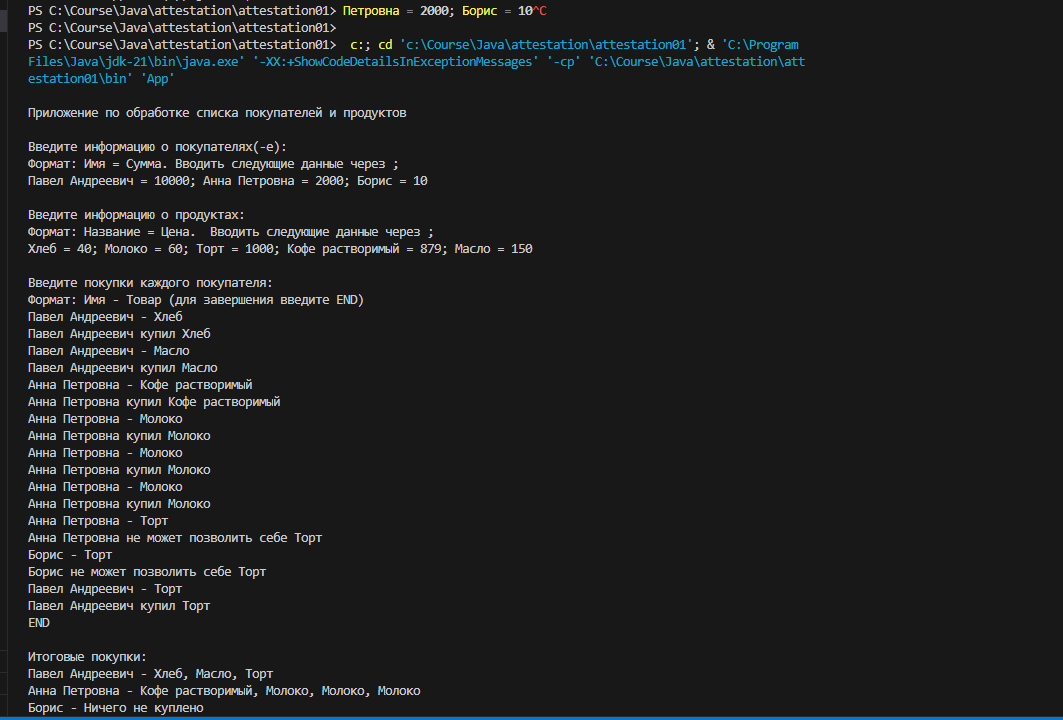
            System.out.println(person);

        }

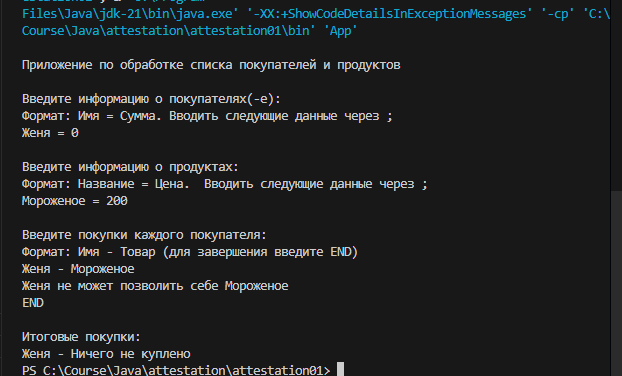
    }

}

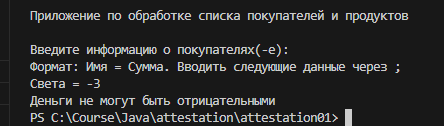
1й пример тестирования:



2й пример тестирования:



3й пример тестирования:



4й пример тестирования:

