Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційні систем та технологій

Лабораторна робота № 13

із дисципліни «Основи програмування»

Тема: «Exceptions»

Виконали:
Студентки групи ІА-24
Ганжа Х. М.
Кійко А. О.
Мелешко Ю. С.
Перевірив: Колеснік Валерій Миколайович

Код:

```
import java.util.*;
public class Main {
    public static void main(String[] args) throws ProductNotFoundException {
        Product product1 = new Product("bread", 14);
        Product product2 = new Product("butter", 40);
        Product product3 = new Product("juice", 55);
        Product product4 = new Product("milk", 30);
        Product product5 = new Product("cheese", 70);

        HashSet<Product> productsFromManufacture = new HashSet<Product>();
        HashSet<Product> first = new HashSet<Product>();
        HashSet<Product> second = new HashSet<Product>();
        HashSet<Product> third = new HashSet<Product>();

        productsFromManufacture.addAll(Set.of(product1, product2, product3, product4,
product5));

        first.addAll(Set.of(product1, product3));
        second.addAll(Set.of(product2, product4));
        third.addAll(Set.of(product5, product1));

        Shop shop = new Shop("Ціни від виробника: ", productsFromManufacture);
        System.out.println(shop);


        Shop shop1 = new Shop("jj", first, 18);
        Shop shop2 = new Shop("ABC", second, 23);
        Shop shop3 = new Shop("Cool", third, 19);


        HashSet<Shop> ALLSHOPS = new HashSet<Shop>();
        ALLSHOPS.addAll(Set.of(shop1, shop2, shop3));

        AllShop allShop = new AllShop(ALLSHOPS);
        System.out.println(allShop);

        String nameProduct = "apple";
        try {
            int minpriceProduct = task1(nameProduct, allShop);
            System.out.println("minpriceProduct: " + nameProduct + ": " +
minpriceProduct);
        } catch (ProductNotFoundException e) {
            System.out.println(e.getMessage() + nameProduct);
        } catch (IllegalArgumentException e) {
            System.out.println(e.getMessage());
        } catch (NullPointerException e) {
            System.out.println(e.getMessage());
        } finally {
            System.out.println("Done 1 task");
            System.out.println();
        }
        try {
            int minpriceProduct = task1(nameProduct, allShop);
            HashSet<Shop> shopListMinPriceOfProduct = task2(minpriceProduct, allShop,
nameProduct);
            System.out.println("Можна купити по мінімальній ціні у: " +
shopListMinPriceOfProduct);
        } catch (ProductNotFoundException e) {
            System.out.println(e.getMessage() + nameProduct);
        } finally {
            System.out.println("Done 2 task");
        }
        System.out.println();
```

```
        task3(productsFromManufacture, allShop);
        System.out.println();

        task4(allShop);
        System.out.println();

        String my_fav_store = shop2.getName();
        PropositionOfWeek(allShop, my_fav_store);



    }


    public static int task1(String product, AllShop allShop) throws
ProductNotFoundException {
        if (product == null ||product.isEmpty()) {
            throw new IllegalArgumentException("Неправильна назва продукту");
        }
        if (allShop == null) {
            throw new NullPointerException("'allShop' є null");
        }
        int minprice = Integer.MAX_VALUE;
        boolean isProductFound = false;
        for (Shop shop : allShop.getAllshops()) {
            for (Product p : shop.getHashSet()) {
                if (p.getProduct().equals(product)) {
                    isProductFound = true;
                    if (p.getPrice() < minprice) {
                        minprice = p.getPrice();
                    }
                }
            }
            if (!isProductFound) {
                throw new ProductNotFoundException("Продукт не знайдено в жодному
магазині: ");
            }
        }
        return minprice;
    }



    public static HashSet<Shop> task2(int minpriceProduct, AllShop allShop, String
productMin) throws ProductNotFoundException {
        HashSet<Shop> shopMinPrice = new HashSet<Shop>();
        boolean isProductWithMinPrice = false;
        for (Iterator i = allShop.getAllshops().iterator(); i.hasNext(); ) {
            Object o = i.next();
            if (o instanceof Shop) {
                Shop shop = (Shop) o;
                for (Iterator p = shop.getHashSet().iterator(); p.hasNext(); ) {

                    Object object = p.next();
                    if (object instanceof Product) {
                        Product product = (Product) object;
                        if (product.getProduct().equals(productMin)) {
                            if (product.getPrice() == minpriceProduct) {
                                isProductWithMinPrice = true;
                            }
                        }
                    }
                }
                if (isProductWithMinPrice) {
                    shopMinPrice.add(shop);
                    isProductWithMinPrice = false;
                }
            }
```

```
        }
        return shopMinPrice;
    }

    public static void task3(HashSet<Product> productsFromManufacture, AllShop allShop) {
        HashSet<Shop> shopMinPrice = new HashSet<Shop>();
        boolean isProductWithLessPriceManuf;
        for (Iterator<Shop> i = allShop.getAllshops().iterator(); i.hasNext(); ) {
            Shop shop = i.next();
            isProductWithLessPriceManuf = false;
            int countProduct = 0;
            for (Iterator<Product> p = shop.getHashSet().iterator(); p.hasNext(); ) {
                Product product = p.next();
                for (Iterator<Product> pm = productsFromManufacture.iterator();
pm.hasNext(); ) {
                    Product productM = pm.next();

                    if (product.getProduct().equals(productM.getProduct())) {
                        if (product.getPrice() < productM.getPrice()) {
                            isProductWithLessPriceManuf = true;
                            countProduct++;

                        }
                    }
                }
            }

            if (countProduct == shop.getHashSet().size()) {
                shopMinPrice.add(shop);
                System.out.println("У " + shop.getName() + " можна купити товари по
цінам, дешевшим ніж рекомендована ціна виробника");
                System.out.println(shop);
            } else {
                System.out.println("У " + shop.getName() + " не можна купити товари по
цінам, дешевшим ніж рекомендована ціна виробника");
            }

        }
    }

    public static void task4(AllShop allShop) {
        for (Shop shop : allShop.getAllshops()) {
            System.out.println(shop.getTime_closed());
        }
    }

    public static void PropositionOfWeek(AllShop allShop, String my_fav_store) {
        HashMap<String, String> hashMap = new HashMap<>();
        for (Shop shop : allShop.getAllshops()) {
            for (Product product : shop.getAllProducts()) {
                hashMap.put(shop.getName(), product.getProduct());
            }
        }
        System.out.println("PROPOSITION OF WEEK " +
(hashMap.get(my_fav_store).toUpperCase()  + "  in store  "  +
my_fav_store.toUpperCase()));

    }
}
public class AllShop {
    private HashSet<Shop> Allshops = new HashSet<Shop>();

    public AllShop(HashSet<Shop> allshops) {
        this.Allshops = allshops;
    }

    public AllShop() {
    }
```

```
    public HashSet<Shop> getAllshops() {

        return Allshops;
    }

    public void setAllshops(HashSet<Shop> allshops) {
        this.Allshops = allshops;
    }

    public void addShop(Collection<Shop> shopsHashSet) {
        this.Allshops.addAll(shopsHashSet);
    }

    @Override
    public String toString() {
        return "Allshops: " + Allshops;
    }
}
public class Product {

    private int price;
    private String product;


    public Product(String product, int price) {
        this.product = product;
        this.price = price;

    }

    public Product() {
    }


    public int getPrice() {
        return price;
    }

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

    public String getProduct() {
        return product;
    }

    public void setProduct(String product) {
        this.product = product;
    }

    @Override
    public String toString() {
        return product + " price: "+ price;
    }

}
public class ProductNotFoundException extends Exception {
    public ProductNotFoundException(String message) {
        super(message);
    }
}
public class Shop {

    private String name;
    private HashSet<Product> allProd = new HashSet<Product>();
    private int time_closed;
```

```
    public Shop(String name, HashSet<Product> hashSet, int clock) {
        this.name = name;
        this.allProd = hashSet;
        this.time_closed = clock;
        HashMap hashMap = new HashMap<>();
        hashMap.put(name, clock);
    }

    public Shop() {
    }

    public Shop(String name, HashSet<Product> hashSet) {
        this.name = name;
        this.allProd = hashSet;
    }

    public void addProduct(Collection<Product> productCollection) {
        this.allProd.addAll(productCollection);
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public HashSet<Product> getHashSet() {
        return allProd;
    }

    public void setHashSet(HashSet<Product> hashSet) {
        this.allProd = hashSet;
    }

    @Override
    public String toString() {
        return name + " " + allProd.toString();
    }

    public String getTime_closed() {
        return "The magazine  " + name + " is closed in " + time_closed;
    }

    public HashSet<Product> getAllProducts() {

        return allProd;
    }
}
```

Результат:

```
Продукт не знайдено в жодному магазині: apple
Done 1 task

Продукт не знайдено в жодному магазині: apple
Done 2 task
```

Висновок: на цій лабораторній роботі ми проаналізували предметну область з л/р 12, створили свій власний тип checked exception, додали throws-декларацію цього exception до відповідних методів, додали блок try-catch-finally для демонстрації виклику цього

метода, додали та продемонстрували код для «викидання» кількох стандартних RuntimeException (IllegalArgumentException, NullPointerException).