**Міністерство освіти і науки України**

**Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"**

**Факультет інформатики та обчислювальної техніки**

**Звіт**

з лабораторної роботи № 2 з дисципліни

«Основи розробки програмного

забезпечення на платформі Java»

„**Дослідження механізму обробки виключень в JAVA**”

**Виконала**

*Колбун Ірина Олексіївна, ІП-92*

**Варіант 26**

Київ 2021

Зміст

[Мета роботи та завдання 3](#_Toc66965726)

[Код програми 3](#_Toc66965727)

[Результати виконання програми 6](#_Toc66965728)

[Висновок 7](#_Toc66965729)

# Мета роботи та завдання

**Мета роботи:** Ознайомитись з поняттям виключення та механізмом обробки виключень в Java. Дослідити створення і використання власних типів виключень.

**Завдання:** доповнити застосунок, який був створений за варіантом лабораторної роботи № 1, наступним:

1. описом власних типів виключень;
2. застосуванням їх при обробці помилок введення даних, використовуючи механізм обробки виключень.

# Код програми

Файл “ItemNotFountException.java”

public class ItemNotFoundException extends RuntimeException {  
 ItemNotFoundException() {  
 super("Не знайдено жодного виробу");  
 }  
  
 ItemNotFoundException(String message) {  
 super(message);  
 }  
}

Файл “PlumbingController.java”

import java.util.Arrays;  
import java.util.InputMismatchException;  
  
class PlumbingController { // Controller  
 private PlumbingView view;  
 private PlumbingStorage storage;  
  
 public PlumbingController(){  
 this.storage = new PlumbingStorage();  
 storage.getModelsFromDatabase();  
 this.view = new PlumbingView();  
 }  
  
 public void run()  
 {  
 int option;  
 do {  
 do {  
 option = view.menuPrompt();  
 } while(option < 0 || option > 3);  
  
 switch(option) {  
 case 1:  
 System.*out*.print(storage);  
 break;  
 case 2:  
 String manufacturer = view.manufacturerPrompt();  
 PlumbingStorage search;  
 try {  
 search = *getByManufacturer*(manufacturer, storage.getModels());  
 System.*out*.print(search);  
 } catch (ItemNotFoundException ex) {  
 view.printException(ex);  
 }  
 break;  
 case 3:  
 String kind = view.kindPrompt();  
 double price;  
 try {  
 price = view.maxPricePrompt();  
 } catch (InputMismatchException ex) {  
 view.printInvalidInput();  
 break;  
 }  
 try {  
 search = *getByKindCostLessThan*(kind, price, storage.getModels());  
 System.*out*.print(search);  
 } catch (ItemNotFoundException ex) {  
 view.printException(ex);  
 }  
 break;  
 default:  
 break;  
 }  
 } while (option != 0);  
 }  
  
 public static PlumbingStorage getByManufacturer(String manufacturer, Plumbing[] plumbings) {  
 PlumbingStorage filtered\_plumbings = new PlumbingStorage();  
 for (Plumbing plumbing : plumbings) {  
 if (plumbing.getManufacturer().equalsIgnoreCase(manufacturer))  
 {  
 filtered\_plumbings.add(plumbing);  
 }  
 }  
 if (filtered\_plumbings.len() == 0)  
 throw new ItemNotFoundException();  
 return filtered\_plumbings;  
 }  
  
 public static PlumbingStorage getByKindCostLessThan(String kind, double max\_price, Plumbing[] plumbings) {  
 PlumbingStorage filtered\_plumbings = new PlumbingStorage();  
 for (Plumbing plumbing : plumbings) {  
 if (plumbing.getKind().equalsIgnoreCase(kind) && plumbing.getPrice() < max\_price)  
 {  
 filtered\_plumbings.add(plumbing);  
 }  
 }  
 if (filtered\_plumbings.len() == 0)  
 throw new ItemNotFoundException();  
 return filtered\_plumbings;  
 }  
  
}

plumbings[6] = new Plumbing(6, "pipe", "FloPlast", "FLOPLAST OVERFLOW PIPE WHITE 21.5MM X 3M", 30, 2.59, "68593");  
 plumbings[7] = new Plumbing(7, "pipe", "FloPlast", "FLOPLAST SOLVENT WELD PIPES WHITE 50MM X 3M", 100, 10.68, "88188");  
 plumbings[8] = new Plumbing(8, "toilet", "AuthenticPlumbing", "TOILET-TO-GO CLOSE-COUPLED TOILET", 0, 300.99, "0A312");  
 plumbings[9] = new Plumbing(9, "toilet", "CASSELLIE", "MONTEGO COMFORT HEIGHT CLOSE-COUPLED TOILET DUAL-FLUSH", 0, 129.99, "892GV");  
 return plumbings;  
 }  
  
}

Файл “PlumbingView.java”

void printException(RuntimeException ex) {  
 System.*out*.println(ex.getMessage());  
 }  
  
 void printInvalidInput() {  
 System.*out*.printf("\"%s\" - невірний формат вводу!%n", in.nextLine());  
 }  
  
}

# Результати виконання програми

Нижче представлені результати виконання програми у консолі.

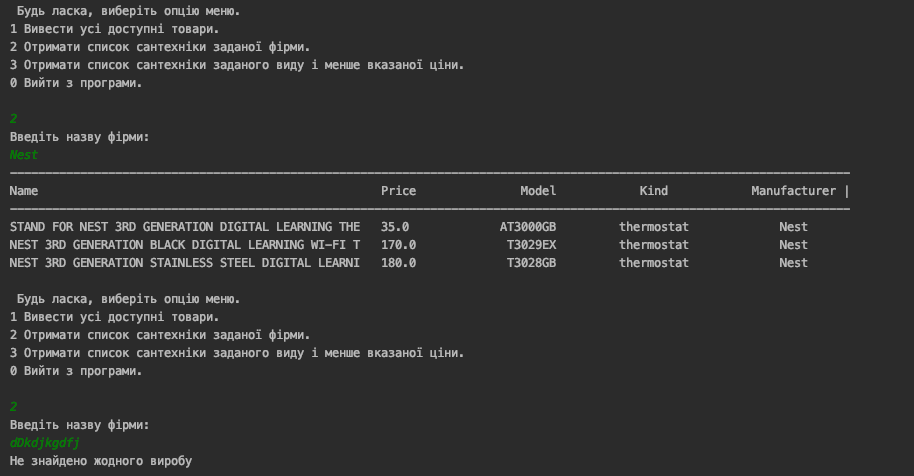


рис. 1 Введення назви існуючої фірми та неіснуючої,

виключення оброблюється

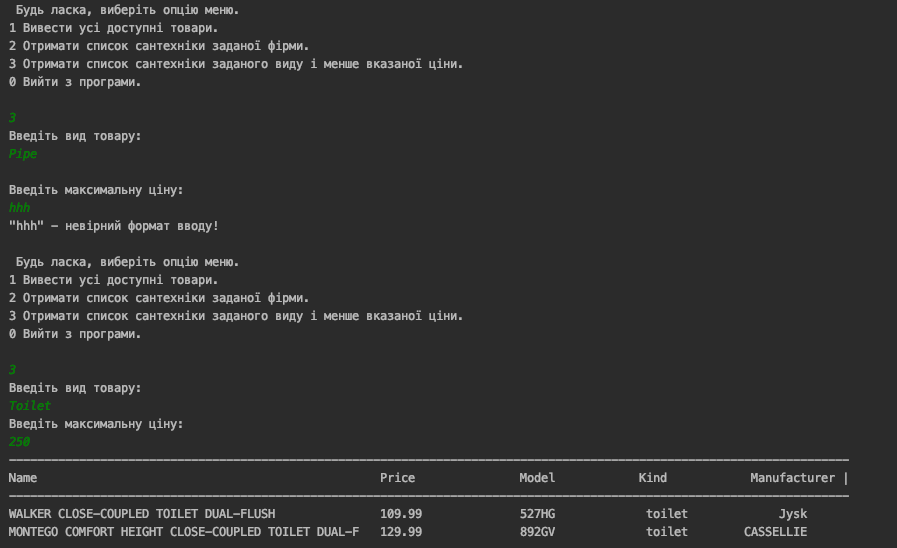


рис. 2 Введення ціни числом та не числом,

відповідна обробка виключення

# Висновок

Під час виконання даної лабораторної роботи я дослідила ієрархію виключень у Java. Описала власний клас виключень, що наслідував від класу RuntimeException.

Я модифікувала вже існуючу програму, щоб додати в неї виключення. Для цього необхідно було додати конструкції try catch у Controller та виведення повідомлення про помилку у View.