## Міністерство освіти і науки України

# Національний університет “Львівська політехніка”

Кафедра інформаційних систем та мереж



Прикладне програмування

Звіт

До лабораторної роботи № 3

«Java Exceptions, Stream API та багатопотоковість»

Виконав:

ст. гр. ІТ–22

Гук П. М.

Прийняла:

Завущак І. І.

«\_\_\_» \_\_\_\_\_\_ 2020 р.

∑ = \_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

Львів – 2020

**Мета**​: ознайомитися із Java Exceptions, Stream API та багатопотоковістю.

Хід роботи

**Завдання No 1.**

Створити власний checked exception у якому повинні бути 1-2 поля для

інформації про виключну ситуацію. Скопіювати завдання No 1 з

лабораторної роботи No 1 (Калькулятор) і обробити ділення на 0

використовуючи створений exception.

**Код файлу Calculator.java**

package task1;

import java.util.Locale;

import java.util.Scanner;

enum Operation {

SUM {

public double action(double arg0, double arg1) { return arg0 + arg1; }

},

SUBTRACT {

public double action(double arg0, double arg1) { return arg0 - arg1; }

},

MULTIPLY {

public double action(double arg0, double arg1) { return arg0 \* arg1; }

},

DIVIDE {

public double action(double arg0, double arg1) throws ZeroDivisionException {

if (arg1 != 0.0) return arg0 / arg1;

throw new ZeroDivisionException("You can't divide by zero");

}

};

public abstract double action(double arg0, double arg1) throws ZeroDivisionException;

}

public class Calculator {

double result;

public static void main(String[] args) {

double result = new Calculator().init().getResult();

System.out.printf("Result: %s", result);

}

private double readNumber(String question, String errorMessage) {

Scanner in = new Scanner(System.in).useLocale(Locale.US);

System.out.print(question);

try {

return in.nextDouble();

} catch (Exception ex) {

System.out.println(errorMessage);

return this.readNumber(question, errorMessage);

}

}

private Operation readOperation(String question, String errorMessage) {

Scanner in = new Scanner(System.in).useLocale(Locale.US);

System.out.print(question);

String operation = in.nextLine();

switch (operation) {

case "+": return Operation.SUM;

case "-": return Operation.SUBTRACT;

case "\*": return Operation.MULTIPLY;

case "/": return Operation.DIVIDE;

default:

System.out.println(errorMessage);

return this.readOperation(question, errorMessage);

}

}

public Calculator init() {

try {

final String numberQuestion = "Enter your number: ";

final String numberErrorMessage = "Invalid number. Try again...";

final String operationQuestion = "Enter your operation. Available is \"+\" \"-\" \"\*\" \"/\": ";

final String operationErrorMessage = "Invalid value. Try one of these: \"+\" \"-\" \"\*\" \"/\"";

final double firstValue = this.readNumber(numberQuestion, numberErrorMessage);

final Operation operation = this.readOperation(operationQuestion, operationErrorMessage);

final double secondValue = this.readNumber(numberQuestion, numberErrorMessage);

this.result = operation.action(firstValue, secondValue);

} catch(ZeroDivisionException ex) {

System.out.println(ex.getMessage());

System.exit(0);

}

return this;

}

public double getResult() {

return this.result;

}

}

**Код файлу ZeroDivisionException.java**

package task1;

public class ZeroDivisionException extends Exception {

public ZeroDivisionException(String errorMessage) {

super(errorMessage);

}

}

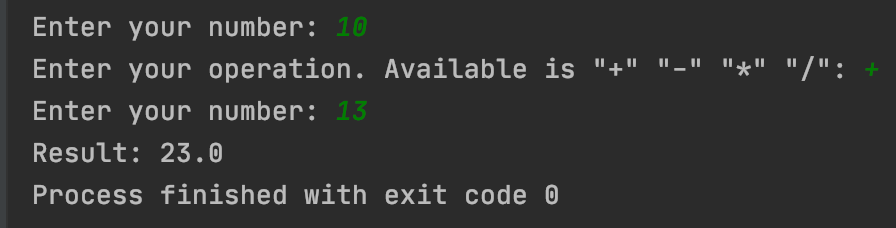
****

Рис.1 Результат виконання роботи програми

**Завдання No 2.**

Створити unchecked exception, одним із полів якого повинен

бути enum із значеннями кодів (200, 400, 500 - стандартні HTTP коди).

Створити клас Address та клас User (клас Address є полем класу User, клас

User повинен мати щонайменше 5 полів).

Створити метод validate(...), який буде перевіряти усі поля класу User на

коректність заповнення.

У разі некоректного (невалідного) значення потрібно створити виключну

ситуацію.

У звіті потрібно вказати які саме правила були використані для валідації.

Наприклад:

● Поле firstName - непорожнє, не повинне містити чисел та пробілів

*Мета цього завдання симулювати один із етапів реєстрації користувача,*

*а саме - перевірку усіх заповнених полів.*

Для валідації **Користувача** були використані такі правила:

● Поле firstName - непорожнє, не повинне містити чисел та пробілів

● Поле lastName - непорожнє, не повинне містити чисел та пробілів

● Поле email – відповідає регулярному виразу ^[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,6}$

● Поле age – число, більше рівне 0

● Поле firstName - непорожнє, не повинне містити чисел та пробілів

● Поле firstName - непорожнє, не повинне містити чисел та пробілів

Для валідації **Адреси** були використані такі правила:

● Поле city - непорожнє, не повинне містити чисел та пробілів

● Поле street – опціональне, не повинне містити чисел та пробілів

**Код файлу HttpException.java**

package task2;

enum HttpStatus {

OK(200), BAD\_REQUEST(400), INTERNAL\_SERVER\_ERROR(500);

private final int status;

HttpStatus(int status) {

this.status = status;

}

public int getStatusCode() {

return this.status;

}

}

public class HttpException extends RuntimeException {

private final HttpStatus status;

public HttpException(HttpStatus status, String errorMessage) {

super(errorMessage);

this.status = status;

}

public int getStatusCode() {

return this.status.getStatusCode();

}

}

**Код файлу Address.java**

package task2;

import java.util.Objects;

public class Address {

final String city;

final String street;

public Address(String city, String street) {

this.city = city;

this.street = street;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

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

Address address = (Address) o;

return Objects.equals(city, address.city) &&

Objects.equals(street, address.street);

}

@Override

public int hashCode() {

return Objects.hash(city, street);

}

public void validate() throws HttpException {

if (!this.city.matches("[a-zA-Z]+")) {

throw new HttpException(HttpStatus.BAD\_REQUEST, "Invalid city");

}

if (!this.street.matches("[a-zA-Z]\*")) {

throw new HttpException(HttpStatus.BAD\_REQUEST, "Invalid street");

}

}

}

**Код файлу User.java**

package task2;

import java.util.Comparator;

public class User {

final String firstName;

final String lastName;

final String email;

final int age;

final Address address;

public String getEmail() {

return email;

}

public User(String firstName, String lastName, String email, int age, Address address) {

this.firstName = firstName;

this.lastName = lastName;

this.email = email;

this.age = age;

this.address = address;

}

public static void main(String[] args) {

new User(

"Pavlo",

"",

"pavlo.huk@gmail.com",

19,

new Address("Pustomyty", "Luhova")

).validate();

}

public int getAge() {

return age;

}

public void validate() throws HttpException {

if (!this.firstName.matches("[a-zA-Z]+")) {

throw new HttpException(HttpStatus.BAD\_REQUEST, "Invalid firstName");

}

if (!this.lastName.matches("[a-zA-Z]\*")) {

throw new HttpException(HttpStatus.BAD\_REQUEST, "Invalid lastName");

}

if (!this.email.matches("^[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\\.[A-Za-z]{2,6}$")) {

throw new HttpException(HttpStatus.BAD\_REQUEST, "Invalid email");

}

if (this.age < 0) {

throw new HttpException(HttpStatus.BAD\_REQUEST, "Invalid age");

}

this.address.validate();

}

}

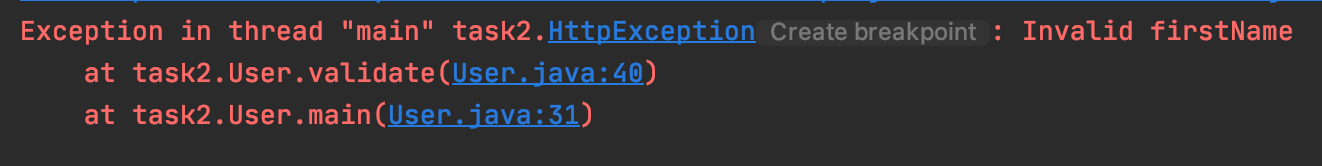


Рис.2 Результат виконання роботи програми

**Завдання No 3.**

Використовуючи Stream API та класи User i Address із завдання No2

продемонструвати використання методів map(), filter(), distinct(), peek(),

limit(), sorted(), findFirst(), collect(), forEach() та reduce().

**Код файлу Main.java**

package task3;

import task2.Address;

import task2.User;

import java.util.Comparator;

import java.util.List;

import java.util.function.Supplier;

import java.util.stream.Collectors;

import java.util.stream.Stream;

public class Main {

public static void main(String[] args) {

Supplier<Stream<User>> users = () -> Stream.of(

new User("Pavlo", "Huk", "pavlo.huk@gmail.com", 19, new Address("Pustomyty", "Luhova")),

new User("Anton", "Skorochkin", "anton@gmail.com", 25, new Address("Lviv", "Shevchenka"))

);

Supplier<Stream<Address>> addresses = () -> Stream.of(

new Address("Pustomyty", "Luhova"),

new Address("Pustomyty", "Luhova")

);

System.out.println("Map emails:");

users.get().map(user -> user.getEmail()).forEach(System.out::println);

System.out.println("Filter emails:");

users.get().filter(user -> !user.getEmail().startsWith("pavlo")).forEach(user -> System.out.println(user.getEmail()));

System.out.println("Distinct count: " + addresses.get().distinct().count());

System.out.println("Peek emails:");

users.get().peek(user -> System.out.println("Peeking email: " + user.getEmail())).forEach(user -> System.out.println(user.getEmail()));

System.out.println("Limit emails:");

users.get().limit(1).forEach(user -> System.out.println(user.getEmail()));

System.out.println("Sort by age:");

users.get().sorted(new UserComparator()).forEach(user -> System.out.println(user.getAge()));

System.out.println("Find first email: " + users.get().findFirst().get().getEmail());

System.out.println("Collect emails:");

List<User> listUsers = users.get().collect(Collectors.toList());

for(User user : listUsers){

System.out.println(user.getEmail());

}

System.out.println("Reduce users: "+ users.get().reduce("Emails:", (x, y) -> x + " " + y.getEmail(), (x, y) -> x + y));

}

}

class UserComparator implements Comparator<User> {

public int compare(User a, User b){

return a.getAge() > b.getAge() ? 1 : -1;

}

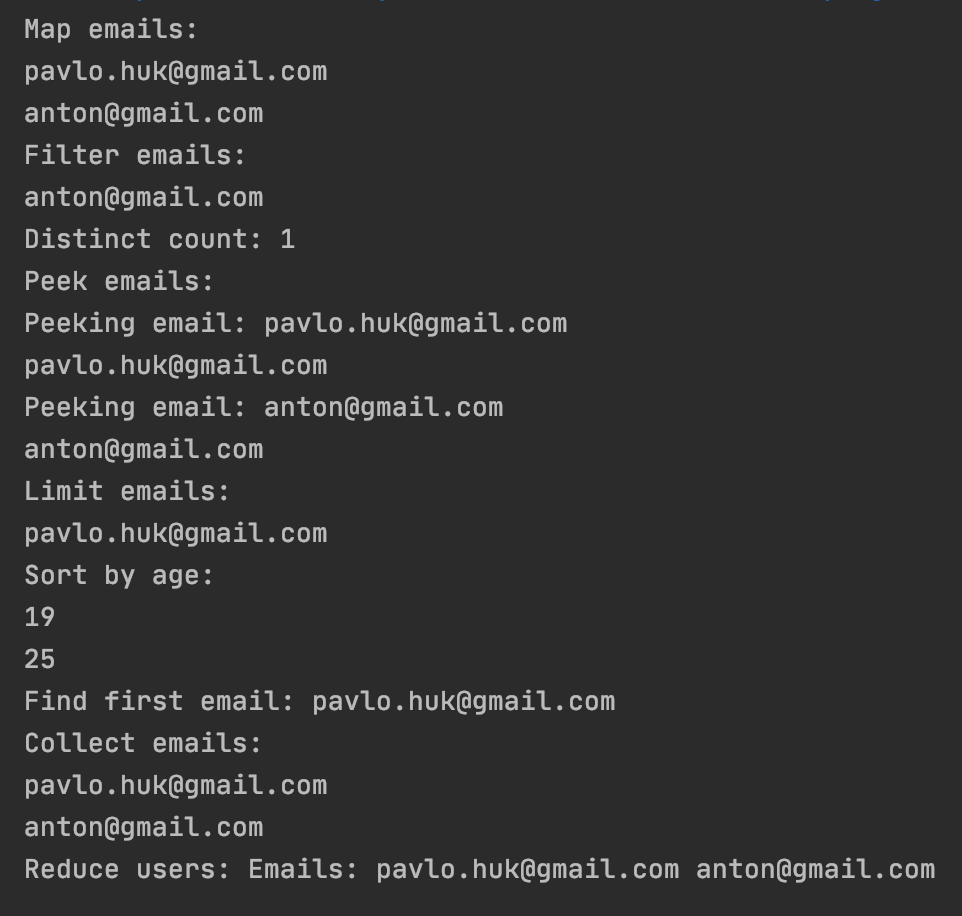
} 

Рис.3 Результат виконання роботи програми

**Завдання № 4.**

Написати програму для обчислення простих калькуляцій використовуючи

багатопотоковість в Java.

**Код файлу Main.java**

package task4;

public class Main {

public static void main(String[] args) {

System.out.printf("%s started... \n", Thread.currentThread().getName());

int[][] matrix = new int[][]{

{ 1, 2, 3, 4, 5 },

{ 6, 7, 8, 9, 10 },

{ 11, 12, 13, 14, 15 },

{ 16, 17, 18, 19, 20 },

{ 21, 22, 23, 24, 25 },

};

Manager manager = new Manager(matrix.length);

for (int i = 0; i < matrix.length; i++) {

new Helper(manager, i, matrix[i]).start();

}

}

**Код файлу Helper.java**

package task4;

public class Helper extends Thread {

final int[] array;

final int index;

final Manager manager;

Helper(Manager manager, int index, int[] array) {

super("Row: " + index);

this.array = array;

this.index = index;

this.manager = manager;

}

public void run() {

System.out.printf("%s started... \n", Thread.currentThread().getName());

try {

Thread.sleep(2000);

} catch (InterruptedException e) {

e.printStackTrace();

}

int result = 0;

for (int number: array) {

result += number;

}

System.out.printf("%s finished... \n", Thread.currentThread().getName());

manager.addResult(result, this.index);

}

}

**Код файлу Manager.java**

package task4;

public class Manager {

int elementsNumber;

final int[] results;

Manager(int elementsNumber) {

this.elementsNumber = elementsNumber;

this.results = new int[elementsNumber];

}

public synchronized void addResult(int result, int index) {

this.results[index] = result;

this.elementsNumber--;

while (elementsNumber != 0) {

try {

wait();

}

catch (InterruptedException e) {

}

}

for (int i = 0; i < results.length; i++) {

System.out.println("--------------");

System.out.println("Row " + i + " sum: " + results[i]);

}

System.out.println("--------------");

}

}

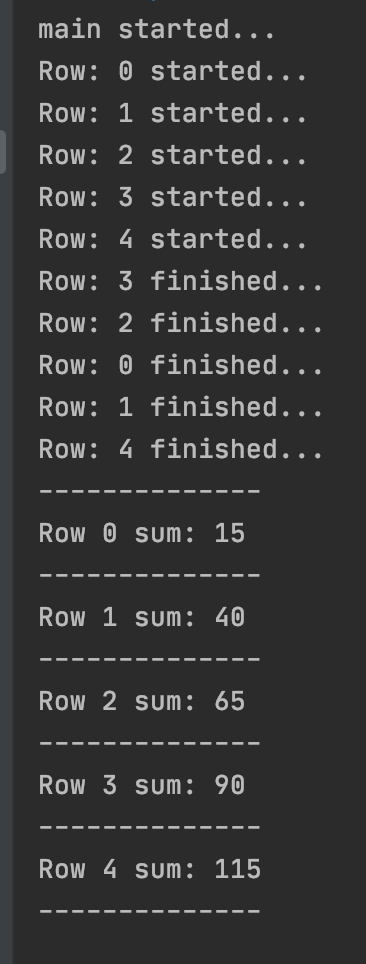
} 

Рис.4 Результат виконання роботи програми

**Висновок:** на даній лабораторній роботі я ознайомився із Java Exceptions, Stream API та

багатопотоковістю..