Міністерство освіти і науки України

Національний університет «Львівська політехніка»



# Звіт

Про виконання

Лабораторної роботи № 3

з дисципліни: «Прикладне програмування»

на тему:

«Java Exceptions, Stream API та багатопотоковість»

**Виконав:**

cтудент групи ІТ-22

Степанчук. С. Т.

**Прийняв:**

асистент Тихонов В.А.

Львів

2021

## Мета роботи:

Ознайомитись із Java Exceptions, Stream API та багатопоточністю.

## Завдання 1

### Умова

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.*

### Код

// Address.java

package Task1;

public class Address {

String address;

public Address(String address) {

this.address = address;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

}

// HttpCode.java

package Task1;

public enum HttpCode {

OK\_200,

BAD\_REQUEST\_400,

GENERIC\_ERROR\_RESPONSE\_500

}

// HttpException.java

package Task1;

public class HttpException extends RuntimeException

{

HttpCode httpCode;

public HttpException(final HttpCode httpCode) {

this(httpCode, "");

}

public HttpException(final HttpCode httpCode, final String message) {

this(httpCode, message, "");

}

public HttpException(final HttpCode httpCode, final String message, final String addiotionalInfo) {

super("HTTP ERROR: " + httpCode.toString() +

(message.length() > 0 ? "\n\tMessage: " + message : "" ) +

(addiotionalInfo.length() > 0 ? "\n\tAddiotional info: " + addiotionalInfo : ""));

this.httpCode = httpCode;

}

public HttpCode getHttpCode() {

return httpCode;

}

}

// User.java

package Task1;

public class User {

String firstName;

String lastName;

Address address;

int age;

String phoneNumber;

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

this.firstName = firstName;

this.lastName = lastName;

this.address = address;

this.age = age;

this.phoneNumber = phoneNumber;

validate();

}

public void validate() {

if (firstName.matches("(.\*[\\d\\s]+.\*)+"))

throw new HttpException(HttpCode.BAD\_REQUEST\_400, "Incorrect firstname!", firstName);

if (lastName.matches("(.\*[\\d\\s]+.\*)+"))

throw new HttpException(HttpCode.BAD\_REQUEST\_400, "Incorrect lastname!", firstName);

if (age <= 0)

throw new HttpException(HttpCode.BAD\_REQUEST\_400, "Incorrect age!", String.valueOf(age));

/\*

\* (123) 456 7899

\* (123).456.7899

\* (123)-456-7899

\* 123-456-7899

\* 123 456 7899

\* 1234567899

\* \*/

if (!phoneNumber.matches("\\(?([0-9]{3})\\)?([ .-]?)([0-9]{3})\\2([0-9]{4})"))

throw new HttpException(HttpCode.BAD\_REQUEST\_400, "Incorrect phonenumber!", phoneNumber);

}

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

validate();

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

validate();

}

public Address getAddress() {

return address;

}

public void setAddress(Address address) {

this.address = address;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

validate();

}

public String getPhoneNumber() {

return phoneNumber;

}

public void setPhoneNumber(String phoneNumber) {

this.phoneNumber = phoneNumber;

validate();

}

@Override

public String toString() {

return lastName + " " + firstName + "\n" +

"\tAge: " + age + "\n" +

"\tAddress: " + address + "\n" +

"\tpPhone number: " + phoneNumber;

}

}

// Task1.java

package Task1;

public class Task1 {

public static void main(String[] args) {

final Address address = new Address("5.248.90.57");

try {

final User user1 = new User("Ser gii", "Stupachnuk", address,32, "0970875944");

}

catch (final HttpException er){

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

}

}

}

Правила валідації:

* Поле firstName - непорожнє, не повинне містити чисел та пробілів;
* Поле lastName - непорожнє, не повинне містити чисел та пробілів;
* Поле age – непорожнє, не мусить бути менше 0;
* Поле phoneNumber – повинно бути у такому форматі
  + \* (123) 456 7899
  + (123).456.7899
  + (123)-456-7899
  + 123-456-7899
  + 123 456 7899
  + 1234567899

### Результат виконання на рисунку 1

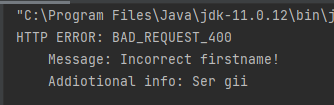


Рис. 1

## Завдання 2

### Умова

Створити клас який можна використовувати у конструкції try-with-resources.

### Код

// Car.java

package Task2;

import java.io.Closeable;

import java.io.IOException;

public class Car implements Closeable{

String name;

public Car(final String name) {

this.name = name;

validate();

}

public void validate() {

if (name.matches("(.\*[\\d\\s]+.\*)+"))

throw new RuntimeException("Exception in name");

}

public String getName() {

return name;

}

public void setName(final String name) {

this.name = name;

}

@Override

public void close() throws IOException {

}

}

// Task2.java

package Task2;

public class Task2 {

public static void main(String[] args) {

try(final Car car = new Car("RenoLogan3")) {

System.out.println(car.getName());

}

catch (final Exception e){

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

}

}

}

### Результат виконання на рисунку 2

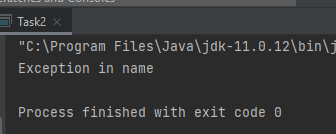


Рис. 2

## Завдання 3

### Умова

Використати ієрархію і дані з лабораторної №2 (завдання 3, 4). Виконати завдання використовуючи Stream API:

● Створити клас StudentSimple з основними даними про студента. Написати метод для перетворення List вхідних даних (List<Student>) у List<StudentSimple>;

● Написати метод для видалення усіх студентів з середнім балом < 3;

● Написати метод для виведення усіх оцінок з Англійської Мови без повторень;

● Написати метод для сортування списку студентів за прізвищем, іменем та по-батькові;

● Написати метод для знаходження студента з найвищим середнім балом;

● Написати метод який буде повертати прізвища усіх студентів через дефіс.

### Код

// Subject.java

package Task3;

public class Subject

{

private String name;

public Subject(String name)

{

this.name = name;

}

public String getName() {

return name;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (!(o instanceof Subject) && !(o instanceof String)) return false;

return name.equals(o instanceof String ? o : ((Subject)o).name);

}

@Override

public int hashCode() {

return name.hashCode();

}

@Override

public String toString() {

return name;

}

}

// Mark.java

package Task3;

public class Mark

{

private int value;

private Subject subject;

public Mark(int value, Subject subject)

{

this.value = value;

this.subject = subject;

}

@Override

public boolean equals(Object o)

{

if (this == o) return true;

if (!(o instanceof Mark)) return false;

Mark mark = (Mark) o;

return subject != null ? subject.equals(mark.subject) : mark.subject == null;

}

@Override

public int hashCode()

{

return subject != null ? subject.hashCode() : 0;

}

@Override

public String toString() {

return subject + ": " + value;

}

public int getValue() {

return value;

}

public String getSubjectName()

{

return subject.getName();

}

}

// Student.java

package Task3;

import java.util.Comparator;

import java.util.HashSet;

import java.util.List;

import java.util.Set;

import java.util.stream.Collectors;

public class Student

{

private Set<Mark> marks;

private String id;

private String name,

surname,

lastname;

private char classChar;

private int numberOfClass;

public Student(String id, String name, String surname, String lastname, char classChar, int numberOfClass, Set<Mark> marks)

{

this.id = id;

this.name = name;

this.surname = surname;

this.lastname = lastname;

this.classChar = classChar;

this.numberOfClass = numberOfClass;

this.marks = marks;

}

public Student(String id, String name, String surname, String lastname, char classChar, int numberOfClass)

{

this(id, name, surname, lastname, classChar, numberOfClass, new HashSet<Mark>());

}

public String getName() {

return name;

}

public String getSurname() {

return surname;

}

public String getLastname() {

return lastname;

}

public char getClassChar() {

return classChar;

}

public int getNumberOfClass() {

return numberOfClass;

}

public String getId() {

return id;

}

@Override

public String toString() {

return name + " " + surname + " " + lastname + "\n\t" +

marks.stream().map(Mark::toString).collect(Collectors.joining("\n\t")) +

"\n\tAvarage: " + getAvarageMark();

}

public void addMark(Mark mark)

{

marks.add(mark);

}

public double getAvarageMark()

{

return marks.stream().mapToInt(mark -> mark.getValue()).average().getAsDouble();

}

public Mark getMarkBySubject(Subject subject)

{

return marks.stream().filter(mark -> subject.equals(mark.getSubjectName())).findFirst().get();

}

public static void sortByPIB(List<Student> students)

{

students.sort(Comparator.comparing(val -> val.getName() + " " + val.getSurname() + " " + val.getLastname(), String::compareTo));

}

}

// StudentList.java

package Task3;

import java.util.\*;

import java.util.stream.Collectors;

public class StudentList

{

private Set<Subject> subjects;

private Map<String, Student> students;

public StudentList(Set<Subject> subjects, Student[] students)

{

this.subjects = subjects;

this.students = new HashMap<String, Student>();

for (var student : students)

this.students.put(student.getId(), student);

}

public List<Student> getStudents()

{

return students.values().stream().collect(Collectors.toList());

}

public void addSubject(Subject subject)

{

subjects.add(subject);

}

public void addSubjectWithMarks(Subject subject, Map<String, Integer> marks)

{

addSubject(subject);

for (var mark : marks.entrySet())

students.get(mark.getKey()).addMark(new Mark(mark.getValue(), subject));

}

public Student getStudentById(String id)

{

return students.get(id);

}

@Override

public String toString() {

return students.values().stream().map(Student::toString).collect(Collectors.joining("\n"));

}

public Student getBestStudent()

{

var arr = new ArrayList<Student>(students.values());

arr.sort(Comparator.comparingDouble(Student::getAvarageMark).reversed());

return arr.get(0);

}

public Map<String, Integer> getStudentsMarksBySubject(Subject subject)

{

var resMap = new HashMap<String, Integer>();

for (var student: students.entrySet())

resMap.put(student.getKey(), student.getValue().getMarkBySubject(subject).getValue());

return resMap;

}

public String studentsMarksBySubjectToString(Map<String, Integer> marks)

{

return marks.entrySet().stream().map(entry -> {

var student = students.get(entry.getKey());

return student.getName() + " " + student.getSurname() + " " + student.getLastname() + ": " + entry.getValue();

}).collect(Collectors.joining("\n"));

}

public void removeBadStudents()

{

var std = getStudents();

for (var student: std)

if (student.getAvarageMark() < 3)

students.remove(student.getId());

}

public String getAllSurnames()

{

return students.values().stream().map(Student::getSurname).collect(Collectors.joining("-"));

}

}

// StudentSimple.java

package Task3;

import java.util.List;

import java.util.stream.Collectors;

public class StudentSimple {

private String id;

private String name;

private String surname;

private String lastname;

public StudentSimple() {

}

public StudentSimple(final Student student) {

this.name = student.getName();

this.surname = student.getSurname();

this.lastname = student.getLastname();

this.id = student.getId();

}

public String getName() {

return name;

}

public void setName(final String name) {

this.name = name;

}

public String getSurName() {

return surname;

}

public void setSurname(final String surName) {

this.surname = surname;

}

public String getId() {

return id;

}

public void setId(final String id) { this.id = id; }

@Override

public String toString() {

return name + " " + surname + " " + lastname;

}

public static List<StudentSimple> fromStudent(final List<Student> students) {

return students.stream()

.map(StudentSimple::new)

.collect(Collectors.toList());

}

}

// Task3.java

package Task3;

import java.util.\*;

public class Task3

{

public static void main(final String[] args) {

final var sb\_phythics = new Subject("Фізика");

final var sb\_english = new Subject("Англійська мова");

final var sb\_math = new Subject("Математика");

final Set<Subject> subjects = new HashSet<Subject>(Arrays.asList(

sb\_phythics,

sb\_english,

sb\_math

));

var studentList = new StudentList(subjects, new Student[] {

new Student("AAAAAA", "Serhii", "Stepanchuk", "Tarasovych", 'A', 11, new HashSet<Mark>(Arrays.asList(

new Mark(4, sb\_phythics),

new Mark(4, sb\_english),

new Mark(4, sb\_math)

))),

new Student("BBBBBB","Maria", "Gryziak", "Lybomirivna", 'B', 10, new HashSet<Mark>(Arrays.asList(

new Mark(1, sb\_phythics),

new Mark(1, sb\_english),

new Mark(1, sb\_math)

))),

new Student("CCCCCC","Andrii", "Sobor", "Vasylivych", 'B', 11, new HashSet<Mark>(Arrays.asList(

new Mark(5, sb\_phythics),

new Mark(5, sb\_english),

new Mark(5, sb\_math)

)))

});

System.out.println(studentList);

final var studentSimples = StudentSimple.fromStudent(studentList.getStudents());

System.out.println("-----------REMOVED BAD STUDENTS-----------");

studentList.removeBadStudents();

System.out.println(studentList);

System.out.println("-----------ALL ENGLISH MARKS-----------");

System.out.println(studentList.studentsMarksBySubjectToString(studentList.getStudentsMarksBySubject(sb\_english)));

System.out.println("-----------NOT SORTED-----------");

var students = studentList.getStudents();

students.forEach(System.out::println);

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

Student.sortByPIB(students);

students.forEach(System.out::println);

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

System.out.println(studentList.getBestStudent());

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

System.out.println(studentList.getAllSurnames());

}

}

### Результат виконання на рисунку 3 та 4

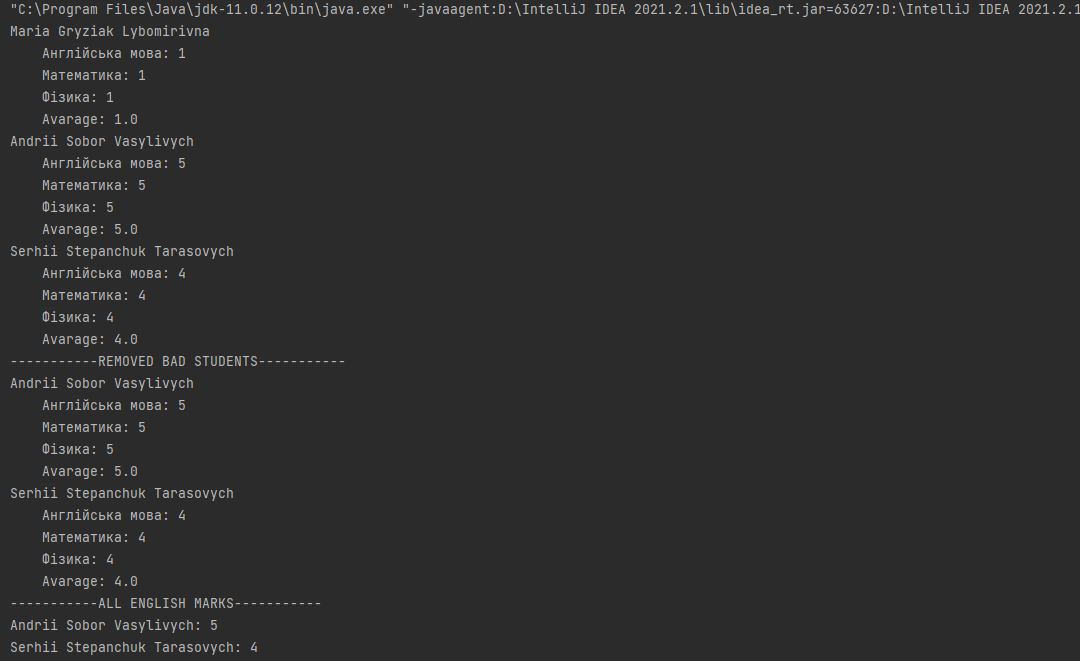


Рис. 3



Рис 4

## Завдання 4

### Умова

Написати програму для обчислення простих калькуляцій використовуючи багатопотоковість в Java. Виконати завдання використовуючи низькорівневу і високорівневу багатопотоковість

### Код

// Calculations.java

public class Calculations {

public static void increment(final int first, final int second) {

System.out.println(first+second + " increment completed");

}

public static void decrement(final int first, final int second) {

System.out.println(first-second + " decrement completed");

}

public static void multi(final int first, final int second) {

System.out.println(first\*second + " multiplication completed");

}

public static void division(final double first, final double second) {

System.out.println(first/second + " division completed");

}

}

// LowLevel.java

public class LowLevel {

public static void main(String[] args) {

final Thread1 thread1 = new Thread1();

final Thread2 thread2 = new Thread2();

final Thread3 thread3 = new Thread3();

final Thread4 thread4 = new Thread4();

thread1.start();

thread2.start();

thread3.start();

thread4.start();

}

public static class Thread1 extends Thread{

@Override

public void run() {

Calculations.increment(72,9);

}

}

public static class Thread2 extends Thread{

@Override

public void run() {

Calculations.decrement(100,43);

}

}

public static class Thread3 extends Thread{

@Override

public void run() {

Calculations.multi(10,43);

}

}

public static class Thread4 extends Thread{

@Override

public void run() {

Calculations.division(25.5,5);

}

}

}

// HighLevel.java

public class HighLevel {

public static void main(String[] args) {

final ExecutorService executor = Executors.newFixedThreadPool(4);

executor.submit(() -> Calculations.increment(16,9));

executor.submit(() -> Calculations.decrement(50,40));

executor.submit(() -> Calculations.multi(4,5));

executor.submit(() -> Calculations.division(36.6,6));

executor.shutdown();

}

}

### Результат виконання програми LowLevel.java

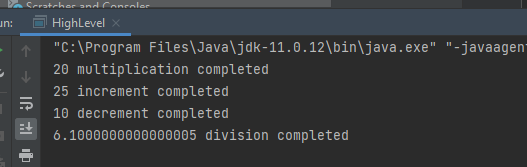


Рис.5

### Результат виконання програми 5 та 6

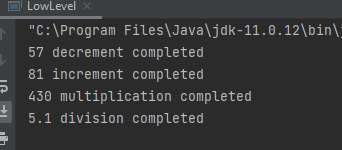


Рис. 6

## Посилання на git-репозиторій з проектом:

<https://github.com/SergiyStepanchuk/APP_Lab_3>

## Висновок

На даній лабораторній роботі, я ознайомився із Java Exceptions, Stream API та багатопоточністю.