Частное учреждение образования

«Колледж бизнеса и права»

Лабораторная работа № 7

Учащийся Е. А. Чужавко

группа: Т-992

2021

**Задание 1.**

Для своего варианта задания лабораторной работы N5 оптимизировать программу, включив в нее обработку исключительных ситуаций: ввода данных, операции деления и др.

Результат с выполнением программы показан на рисунке 1.

package z2;

import java.util.ArrayList;

import java.util.List;

public class Student {

public String name;

public String group;

public int course;

int[] marks;

public Student(String name, String group, int course)

{

try {

if(group.length()<1 | group.length()>5)

throw new GroupException("Длина группы не может быть меньше 1 или больше 4");

else

this.group = group;

}

catch(GroupException ex) {

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

}

try

{

if (course <= 0 | course > 5)

throw new CourseException("Курс не может быть меньше 1 или больше 4");

else

this.course = course;

}

catch (CourseException ex) {

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

}

try {

if (name.length() < 1 | name.length() > 32)

throw new NameException("Длина имени не может быть меньше 1 или больше 31");

else

this.name = name;

}

catch (NameException ex) {

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

}

this.marks = new int[10];

for (int i = 0; i < 10; i++)

{

this.marks[i] = (int)(Math.random()\*6);

}

}

public void upCourse() {

course++;

}

public static void checkSrBall(ArrayList<Student> students) {

int id=0;

ArrayList<Integer> idtodelete=new ArrayList<Integer>();

for(Student std : students) {

int balli=0;

int k=0;

for(int s : std.marks ) {

balli+=s;

k++;

}

if((double)balli/k<3) {

idtodelete.add(id);

}

else {

if((double)balli/k>=3) {

students.get(id).course+=1;

}

}

id++;

}

int qqq=0;

for(Integer idd : idtodelete) {

students.remove(students.get(idd-qqq));

qqq++;

}

}

public static void printStudents(List<Student> students, int coursee) {

for(Student std : students) {

if(std.course==coursee) {

System.out.println(std);

}

}

}

public static void printStudents(List<Student> students) {

for(Student std : students) {

System.out.println("Name = "+std.name +" Course = "+std.course);

}

}

@Override

public String toString() {

String smarks="";

for(int m : marks) {

smarks+=Integer.toString(m)+" ";

}

return "Student [name=" + name + ", group=" + group + ", course=" + course + ", оценки:" + smarks + "]";

}

}

class NameException extends Exception

{

public NameException(String message)

{

super(message);

}

}

class CourseException extends Exception

{

public CourseException(String message)

{

super(message);

}

}

class GroupException extends Exception

{

public GroupException(String message) {

super(message);

}

}

package z2;

import java.util.ArrayList;

public class test {

public static void main(String[] args) {

Student student1 = new Student("Vladislav", "К111", 6);

Student student2 = new Student("Gena","K011",4);

Student student3 = new Student("Petya","K111",3);

ArrayList<Student>students=new ArrayList<Student>();

students.add(student2);

students.add(student1);

students.add(student3);

System.out.println("Вывод всех студентов и их номер курса...");

Student.printStudents(students);

System.out.println("Вывод студентов 3 курса ");

Student.printStudents(students, 3);

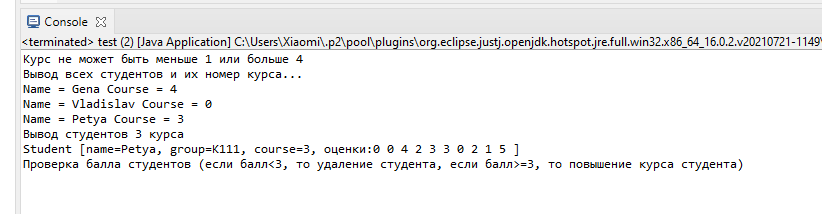
System.out.println("Проверка балла студентов (если балл<3, то удаление студента, если балл>=3, то повышение курса студента)");

Student.checkSrBall(students);

Student.printStudents(students);

}

}



*Рисунок 1.*

**Задание 2.**

Классы

Свойства и методы

Фирма

название (get, set)

Отдел

название (get, set)

количество сотрудников (get, set)

Сотрудник

фио (get, set)

должность (get, set)

оклад (get, set)

рассчитать зарплату()

Штатный сотрудник

премия (get, set)

рассчитать зарплату()

Сотрудник по контракту

рассчитать зарплату()

а) Обработать все Exception с помощью блока try...catch(Exception ...) в методе «рассчитать зарплату» классов Штатный сотрудник и Сотрудник по контракту. При возникновении Exception выводить на экран сообщение об ошибке.

Описать собственный класс Exception PremiyaException. В методе «рассчитать зарплату» класса Штатный сотрудник выбрасывать собственное Exception типа PremiyaException при отрицательном значении свойства Премия. В этом же методе обработать PremiyaException в блоке catch. При возникновении Exception выводить сообщение об ошибке на экран. В основной программе проверить работу блока обработки Exception метода «рассчитать зарплату».

б) Описать собственный класс Exception OkladException. В конструкторе реализовать проверку значения оклада, при отрицательном значении выбрасывать собственное Exception типа OkladException. Обработать Exception OkladException с помощью блока try...catch, в блоке обработки Exception вывести на экран сообщение «Невозможно создать сотрудника – указан отрицательный оклад: <оклад> » и повторно создать Exception.

В основной программе (main) обработать вызов конструктора класса. Сотрудник и проверить работу обработчика Exception.

Результат работы программы показан на рисунке 2.

package z3;

public class ContractEmployee extends Employee {

ContractEmployee (String fullName, dolgnost \_dolgnost, double oklad) {

super(fullName, \_dolgnost, oklad);

try {

if (oklad < 0) {

throw new OkladExeption("Оклад не может быть отрицательным, оклад: " + oklad);

}

}

catch (OkladExeption ex) {

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

return;

}

}

@Override

public double Rate()

{

double rating = 0;

switch (\_dolgnost)

{

case emp:

rating = 1;

break;

case zam:

rating = 50;

break;

case dir:

rating = 100;

break;

}

double salary = oklad + oklad \* (rating\*10)/100;

if (salary <= 0)

{

throw new ArithmeticException("Зарплата не может быть меньше либо равна 0");

}

return salary;

}

}

package z3;

public class Department {

String name;

int countEmployee;

Employee emp;

ShtatEmployee shtatEmp;

ContractEmployee contractEmp;

Department (String name, int countEmployee, Employee emp, ShtatEmployee shtatEmp, ContractEmployee contractEmp) {

this.name = name;

this.countEmployee = countEmployee;

this.emp = emp;

this.contractEmp = contractEmp;

this.shtatEmp = shtatEmp;

}

public String getName(){

return name;

}

public void setName(String a){

name = a;

}

public int getCountEmployee(){

return countEmployee;

}

public void setCountEmployee(int a){

countEmployee = a;

}

}

package z3;

enum dolgnost

{

emp,

zam,

dir

}

public class Employee {

String fullName;

dolgnost \_dolgnost;

double oklad;

Employee (String fullName, dolgnost \_dolgnost, double oklad) {

this.fullName = fullName;

this.\_dolgnost = \_dolgnost;

this.oklad = oklad;

try {

if (oklad < 0) {

throw new OkladExeption("Оклад не может быть отрицательным, оклад: " + oklad);

}

}

catch (OkladExeption ex) {

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

return;

}

}

public String getFullName(){

return fullName;

}

public void setFullName(String a){

fullName = a;

}

public String getDolgnost(){

return fullName;

}

public void setDolgnost(String a){

fullName = a;

}

public String getOklad(){

return fullName;

}

public void setOklad(String a){

fullName = a;

}

public double Rate() throws PremiyaExeption {

double rating = 0;

switch (\_dolgnost) {

case emp:

rating = 1;

break;

case zam:

rating = 50;

break;

case dir:

rating = 100;

break;

}

double salary = oklad + oklad \* (rating\*10)/100;

if (salary <= 0) {

throw new ArithmeticException("Зарплата не может быть меньше либо равна 0");

}

return salary;

}

}

class PremiyaExeption extends Exception {

public PremiyaExeption(String message){

super(message);

}

}

class OkladExeption extends Exception {

public OkladExeption(String message){

super(message);

}

}

package z3;

public class Firm

{

String name;

Department dep;

Firm (String name, Department dep) {

this.name = name;

this.dep = dep;

}

public String getName(){

return name;

}

public void setName(String a){

name=a;

}

}

package z3;

public class ShtatEmployee extends Employee {

double premiya;

ShtatEmployee (String fullName, dolgnost \_dolgnost, double oklad, double premiya) {

super(fullName, \_dolgnost, oklad);

this.premiya = premiya;

try {

if (oklad < 0) {

throw new OkladExeption("Оклад не может быть отрицательным, оклад: " + oklad);

}

}

catch (OkladExeption ex) {

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

return;

}

}

@Override

public double Rate() throws PremiyaExeption {

double rating = 0;

switch (\_dolgnost) {

case emp:

rating = 1;

break;

case zam:

rating = 50;

break;

case dir:

rating = 100;

break;

}

double salary = oklad + oklad \* (rating\*10)/100;

if (salary <= 0) {

throw new ArithmeticException("Зарплата не может быть меньше либо равна 0");

}

if (premiya < 0) {

throw new PremiyaExeption("Премия не может быть отрицательным значением");

}

return salary + premiya;

}

}

package z3;

public class Test {

public static void main(String[] args) {

Employee emp = new Employee("Александр", dolgnost.emp, -500);

ShtatEmployee shtatEmp = new ShtatEmployee("Дмитрий", dolgnost.zam, 1000, 1000);

ContractEmployee contractEmployee = new ContractEmployee("Егор", dolgnost.dir, 1500);

Department dep = new Department("Департамент №1", 3, emp, shtatEmp, contractEmployee);

Firm firm = new Firm("Фирма №1", dep);

try {

System.out.println(firm.dep.emp.Rate());

System.out.println(firm.dep.shtatEmp.Rate());

System.out.println(firm.dep.contractEmp.Rate());

}

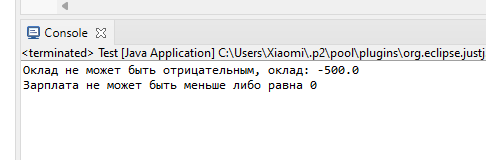
catch (Exception ex) {

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

}

}

}



*Рисунок 2.*