|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| *Федеральное государственное бюджетное образовательное учреждение высшего образования* ***«МИРЭА – Российский технологический университет»***  **РТУ МИРЭА** |

**Институт информационных технологий (ИТ)**

**Кафедра инструментального и прикладного программного обеспечения (ИиППО)**

**Дисциплина «Программирование на языке Джава»**

**ОТЧЕТ**

**ПО ПРАКТИЧЕСКИМ ЗАНЯТИЯМ №9, №10**

Выполнил студент группы ИНБО-02-20 Колмаков Е.Ю.

Принял Степанов П.В.

Практические работы выполнены «\_\_\_»\_\_\_\_\_\_\_2021г.

«\_\_\_\_\_\_\_\_\_\_\_\_\_\_» «\_\_\_»\_\_\_\_\_\_\_2021г.

Отметка о выполнении

**Москва 2021 г.**

**Практическое занятие №9**

**Цель работы**

Научиться создавать собственные исключения.

**Задание**

Клиент совершает покупку онлайн. При оформлении заказа у пользователя запрашивается фио и номер ИНН. В программе проверяется, действителен ли номер ИНН для такого клиента. Исключение будет выдано в том случае, если введен недействительный ИНН.

Предлагается модернизировать задачу из предыдущей лабораторной работы (см. методические указания по выполнению лабораторных работ №1-8) – задача сортировки студентов по среднему баллу. Необходимо разработать пользовательский интерфейс для задачи поиска и сортировки (использовать массив интерфейсных ссылок- пример в лекции 5). Дополнить ее поиском студента по фио – в случае отсутствия такого студента необходимо выдавать собственное исключение.

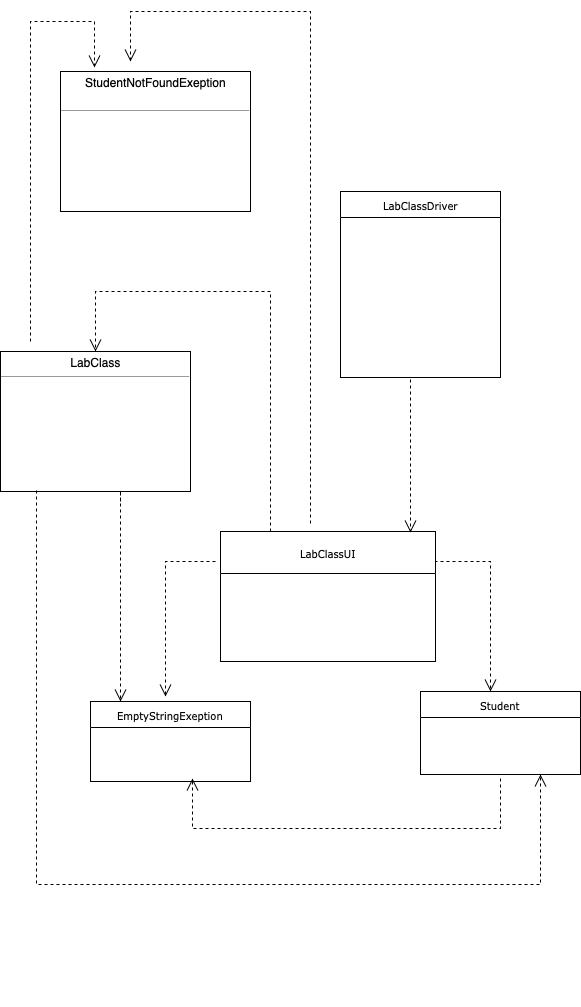
****

Рис. 1 – UML – диаграмма

**Упражнение №1**

**Класс TestLab9ex1**

package Lab9.ex1;  
import Lab9.ex1.InnFormatException;  
  
import java.util.Scanner;  
  
  
public class TestLab9ex1 {  
 public static void main(String[] str) throws InnFormatException {  
 Scanner sc = new Scanner(System.*in*);  
 while (true) {  
 long inn = Long.*parseLong*(sc.nextLine());  
 int i = (int) Math.*log10*(inn) + 1;  
 if (i != 10 && i != 12) {  
 throw new InnFormatException("Неправильный формат ИНН!!!");  
 }  
  
  
 }  
 }  
}

**Класс InnFormatException**

package Lab9.ex1;  
  
public class InnFormatException extends Exception {  
 public InnFormatException(String message) {  
 super(message);  
 }  
}

**Упражнение №2**

**Класс Starter**

package Lab9.ex2;  
  
import java.util.ArrayList;  
  
public class Starter {  
 public static void main(String[] args) {  
 ArrayList<Student> s = new ArrayList<>();  
 s.add(new Student("Сидоров1", 67, 12, 18));  
 s.add(new Student("Сидоров2", 57, 12, 18));  
 s.add(new Student("Сидоров3", 93, 12, 19));  
 s.add(new Student("Сидоров4", 24, 2, 18));  
 s.add(new Student("Сидоров5", 66, 12, 18));  
  
 new LabClassUI(s);  
 }  
  
}

**Класс EmptyStringExeption**

package Lab9.ex2;  
  
public class EmptyStringException extends IllegalArgumentException {  
 public EmptyStringException() {  
 super("Передана пустая строка!");  
 }  
}

**Интерфейс GroupComparator**

package Lab9.ex2;  
  
import java.util.Comparator;  
  
public class GroupComparator implements Comparator<Student> {  
 @Override  
 public int compare(Student o1, Student o2) {  
 return o1.getGroupNum() - o2.getGroupNum();  
 }  
}

**Класс LabClassUI**

package Lab9.ex2;  
  
import javax.swing.\*;  
import javax.swing.event.MouseInputListener;  
import javax.swing.table.DefaultTableModel;  
import javax.swing.table.JTableHeader;  
import java.awt.\*;  
import java.awt.event.MouseEvent;  
import java.util.ArrayList;  
import java.util.Comparator;  
  
public class LabClassUI extends JFrame {  
  
 private ArrayList<Student> students;  
 private JTable studTable;  
  
 public LabClassUI(ArrayList<Student> students){  
 // base  
 super("Студенты");  
 setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);  
 setSize(700, 500);  
 setLocationRelativeTo(null);  
  
 this.students = students;  
  
 // Buttons  
 JPanel pN = new JPanel(new FlowLayout());  
 JButton addStudentBtn = new JButton("Добавить студента");  
 JButton remStudentBtn = new JButton("Удалить студента");  
 JButton findStudentBtn = new JButton("Найти студента");  
  
 pN.add(addStudentBtn);  
 pN.add(remStudentBtn);  
 pN.add(findStudentBtn);  
  
 addStudentBtn.addActionListener(e -> addBtnClicked());  
  
 remStudentBtn.addActionListener(e-> remBtnClicked());  
  
 findStudentBtn.addActionListener(e->{  
 try{  
 findBtnClicked();  
 }catch (StudentNotFoundException ex){  
 JOptionPane.*showMessageDialog*(this, ex.getMessage());  
 }  
 });  
  
 // JTable  
 Object[] headers = new String[] {"ФИО", "Возраст", "Группа", "Средний балл"};  
 Object [][] startStudents = new String[students.size()][4];  
 for(int i = 0; i < students.size(); i++){  
 startStudents[i][0] = students.get(i).getFio();  
 startStudents[i][1] = ((Integer)(students.get(i).getAge())).toString();  
 startStudents[i][2] = ((Integer)(students.get(i).getGroupNum())).toString();  
 startStudents[i][3] = ((Integer)(students.get(i).getAvgPoint())).toString();  
 }  
 studTable = new JTable( new DefaultTableModel(startStudents, headers)){  
 @Override  
 public boolean isCellEditable(int x, int y){  
 return false;  
 }  
 };  
  
 JTableHeader header = studTable.getTableHeader();  
 header.setReorderingAllowed(false);  
 header.setResizingAllowed(false);  
  
 header.addMouseListener(new MouseInputListener() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 int count = e.getClickCount();  
 if (count == 2){  
 if(header.getHeaderRect(0).contains(e.getX(), e.getY()))  
 sortStudents(Student::compareTo);  
 if(header.getHeaderRect(1).contains(e.getX(), e.getY()))  
 sortStudents(Student.*AGE\_COMP*);  
 if(header.getHeaderRect(2).contains(e.getX(), e.getY()))  
 sortStudents(Student.*GROUP\_COMP*);  
 if(header.getHeaderRect(3).contains(e.getX(), e.getY()))  
 sortStudents(Student.*AVG\_COMP*);  
 }  
 }  
  
 @Override  
 public void mousePressed(MouseEvent e) {  
  
 }  
  
 @Override  
 public void mouseReleased(MouseEvent e) {  
  
 }  
  
 @Override  
 public void mouseEntered(MouseEvent e) {  
  
 }  
  
 @Override  
 public void mouseExited(MouseEvent e) {  
  
 }  
  
 @Override  
 public void mouseDragged(MouseEvent e) {  
  
 }  
  
 @Override  
 public void mouseMoved(MouseEvent e) {  
  
 }  
 });  
  
 // Adding comps  
 getContentPane().add(new JScrollPane(studTable), BorderLayout.*CENTER*);  
  
 getContentPane().add(pN, BorderLayout.*NORTH*);  
  
 setVisible(true);  
 }  
  
 private void sortStudents(Comparator<Student> comp){  
 Student temp;  
 for (int i = 1; i < students.size(); i++) {  
 Student current = students.get(i);  
 int j = i-1;  
 for(; j >= 0 && comp.compare(current, students.get(j)) < 0; j--) {  
 students.set(j+1, students.get(j));  
 }  
 students.set(j+1, current);  
 }  
 DefaultTableModel dtm = (DefaultTableModel) studTable.getModel();  
  
 for (int i = 0; i < students.size(); i++) {  
 dtm.removeRow(i);  
 Student st = students.get(i);  
 dtm.insertRow(i, new Object[]{st.getFio(), st.getAge(), st.getGroupNum(), st.getAvgPoint()});  
 }  
 }  
  
 private void findBtnClicked() throws StudentNotFoundException {  
 String s = JOptionPane.*showInputDialog*("Введите искомые ФИО");  
 for(int i = 0; i < students.size(); i++){  
 if(students.get(i).getFio().equals(s)){  
 JOptionPane.*showMessageDialog*(this, "Найден студент: "+  
 students.get(i).toString());  
 return;  
 }  
 }  
 throw new StudentNotFoundException(s);  
 }  
  
 private void remBtnClicked() {  
 int c = studTable.getSelectedRowCount();  
 if(c != 1) {  
 JOptionPane.*showMessageDialog*(this, "Выберете ровно одну строку!");  
 return;  
 }  
 c = studTable.getSelectedRow();  
 students.remove(c);  
 DefaultTableModel dtm = (DefaultTableModel) studTable.getModel();  
  
 dtm.removeRow(c);  
  
 }  
  
 private void addBtnClicked() throws IllegalArgumentException {  
 String fio = JOptionPane.*showInputDialog*("Пожалуйста, введите фио!");  
 if(fio.equals("")) throw new EmptyStringException();  
 String age = JOptionPane.*showInputDialog*("Пожалуйста, введите возраст!");  
 String group = JOptionPane.*showInputDialog*("Пожалуйста, введите номер группы!");  
 String avg = JOptionPane.*showInputDialog*("Пожалуйста, введите средний балл!");  
  
 int avgI =0;  
 int groupI = 0;  
 int ageI =0;  
 try {  
 avgI = Integer.*parseInt*(avg);  
 groupI = Integer.*parseInt*(group);  
 ageI = Integer.*parseInt*(age);  
 }catch (NumberFormatException e){  
 throw new NumberFormatException("Невозможно прочитать строку "+e.getMessage().substring(17) + "!!! Введите число!");  
 }  
  
 students.add(new Student(fio, avgI, groupI, ageI));  
  
 DefaultTableModel dtm = (DefaultTableModel) studTable.getModel();  
  
 dtm.addRow(new Object[]{fio, age, group, avg});  
 }  
}

**Класс NotFoundStudentExeption**

package Lab9.ex2;  
  
public class StudentNotFoundException extends Exception {  
 public StudentNotFoundException(String name) {  
 super("Студент с ФИО "+ name + " не найден!");  
 }  
}

**Класс AgeComparator**

package Lab9.ex2;  
  
import java.util.Comparator;  
  
public class AgeComparator implements Comparator<Student> {  
 @Override  
 public int compare(Student o1, Student o2) {  
 return o1.getAge() - o2.getAge();  
 }  
}

**Класс AvgComparator**

package Lab9.ex2;  
  
import java.util.Comparator;  
  
public class AvgComparator implements Comparator<Student> {  
 @Override  
 public int compare(Student o1, Student o2) {  
 return o1.getAvgPoint() - o2.getAvgPoint();  
 }  
}

**Класс Student**

package Lab9.ex2;  
  
public class Student implements Comparable<Student>{  
 private String fio;  
 private int avgPoint, groupNum, age;  
  
 public static final AgeComparator *AGE\_COMP* = new AgeComparator();  
 public static final AvgComparator *AVG\_COMP* = new AvgComparator();  
 public static final GroupComparator *GROUP\_COMP* = new GroupComparator();  
  
 public Student(String fio, int avgPoint, int groupNum, int age) {  
 this.fio = fio;  
 this.avgPoint = avgPoint;  
 this.groupNum = groupNum;  
 this.age = age;  
 }  
  
 public String getFio() {  
 return fio;  
 }  
  
 public void setFio(String fio) {  
 if(fio.equals(""))  
 throw new EmptyStringException();  
 this.fio = fio;  
 }  
  
 public int getAvgPoint() {  
 return avgPoint;  
 }  
  
 public void setAvgPoint(int avgPoint) {  
 this.avgPoint = avgPoint;  
 }  
  
 public int getGroupNum() {  
 return groupNum;  
 }  
  
 public void setGroupNum(int groupNum) {  
 this.groupNum = groupNum;  
 }  
  
 public int getAge() {  
 return age;  
 }  
  
 public void setAge(int age) {  
 this.age = age;  
 }  
  
 @Override  
 public String toString() {  
 return "Student{" +  
 "fio='" + fio + '\'' +  
 ", avgPoint=" + avgPoint +  
 ", groupNum=" + groupNum +  
 ", age=" + age +  
 '}';  
 }  
  
 @Override  
 public int compareTo(Student o) {  
 return this.fio.compareTo(o.fio);  
 }  
}

**Тестирование**

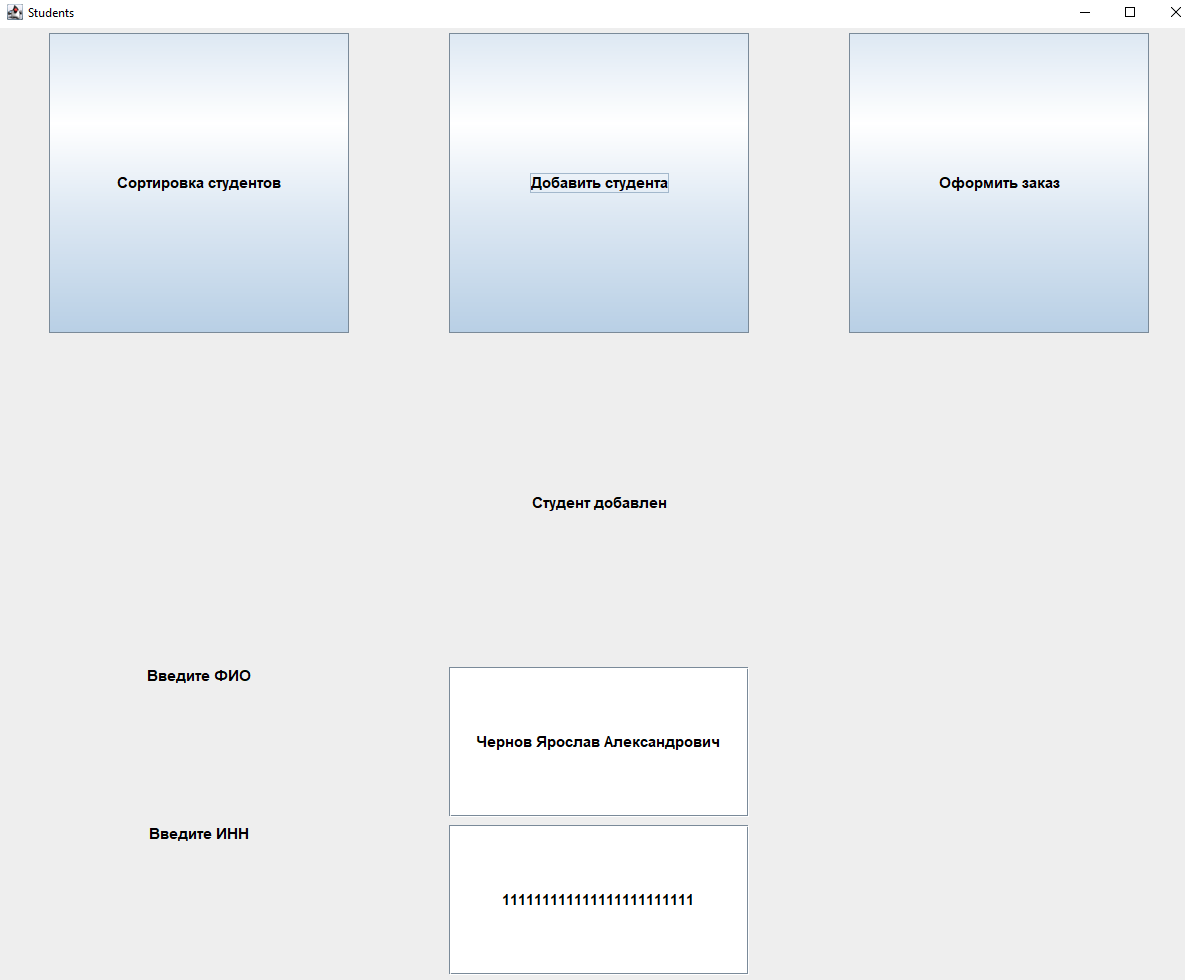


Рис. 2 – Добавление студента в базу

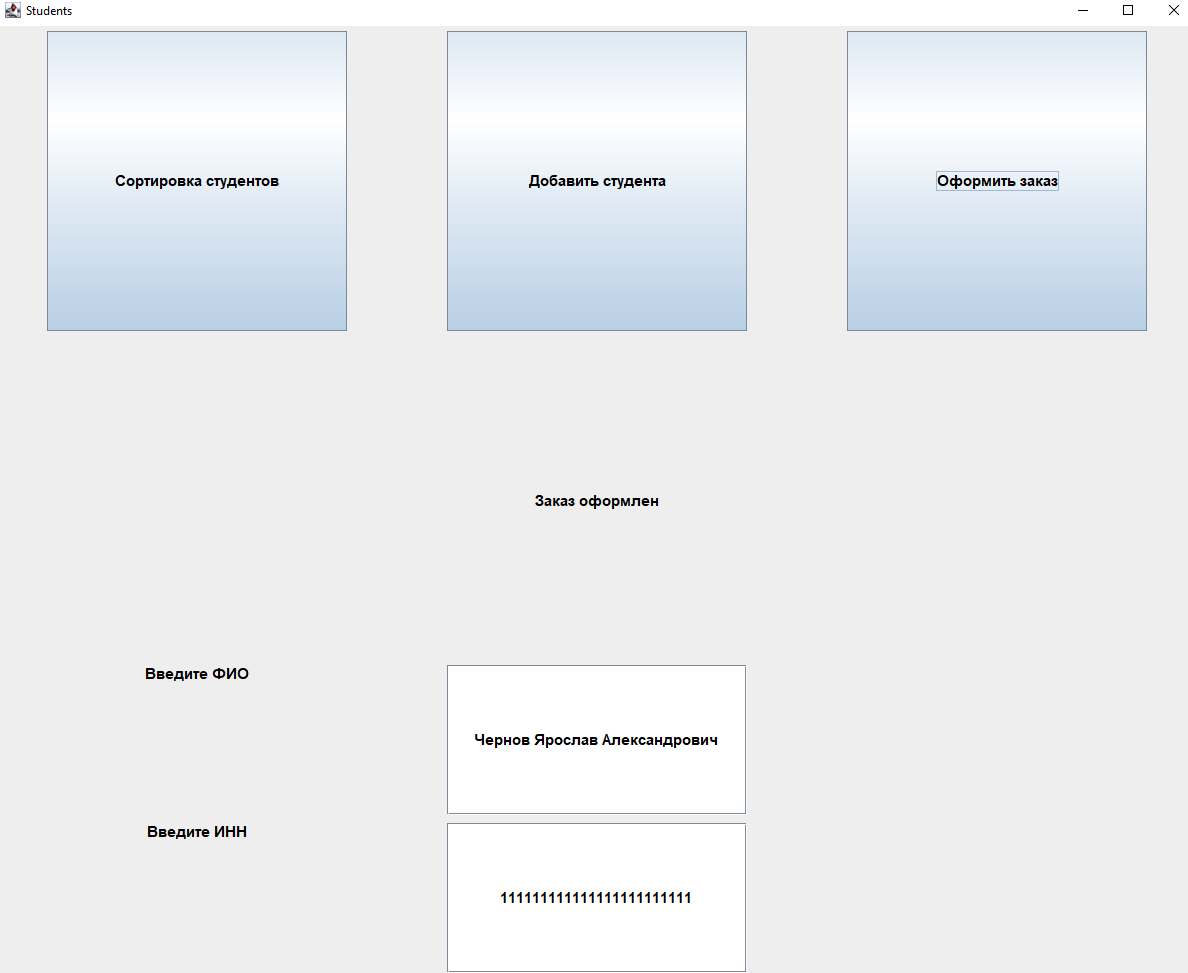


Рис.3 – Оформление заказа от добавленного студента

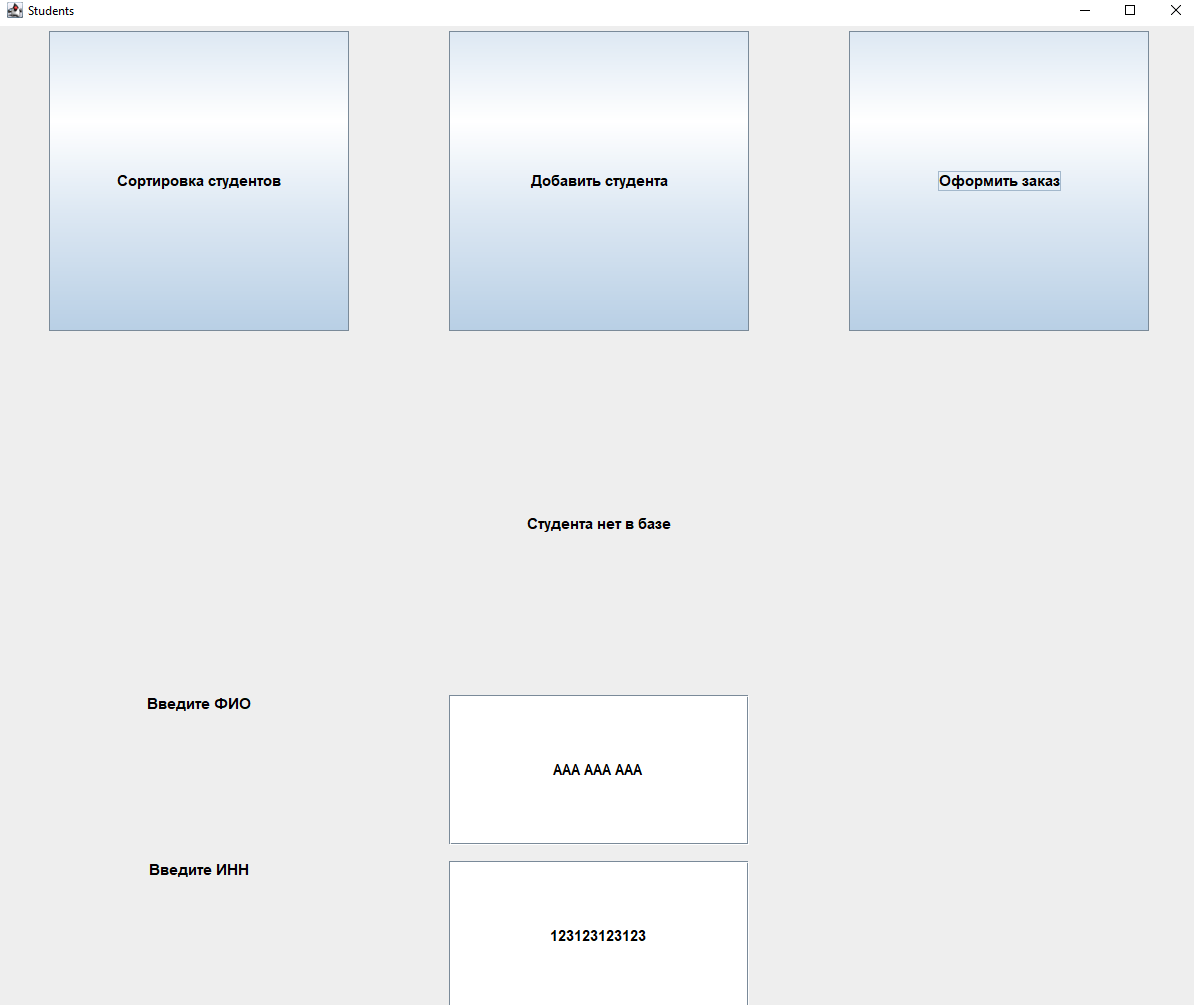


Рис.4 – Оформление заказа от неизвестного студента

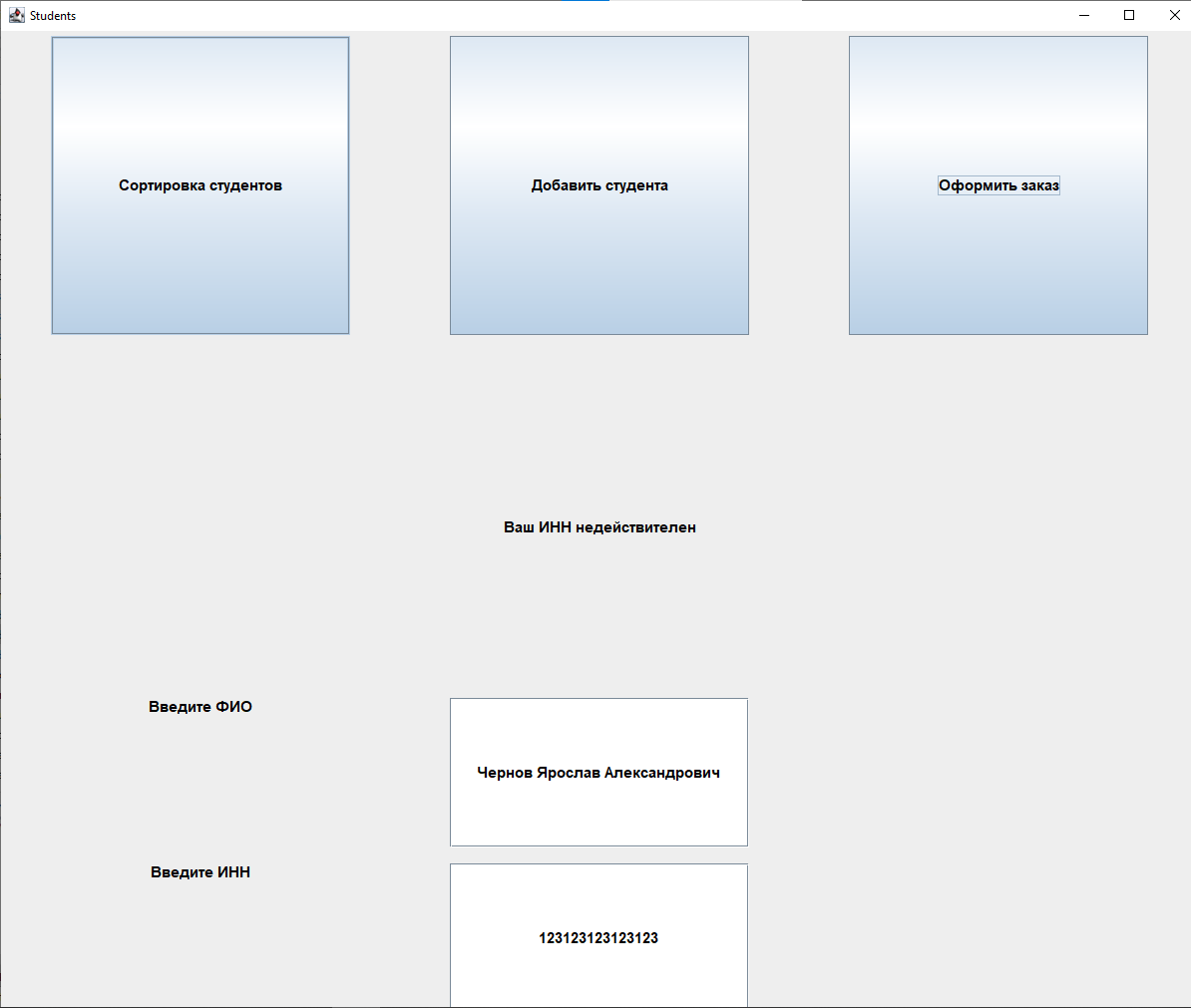


Рис.5 – Оформление заказа с неправильным ИНН

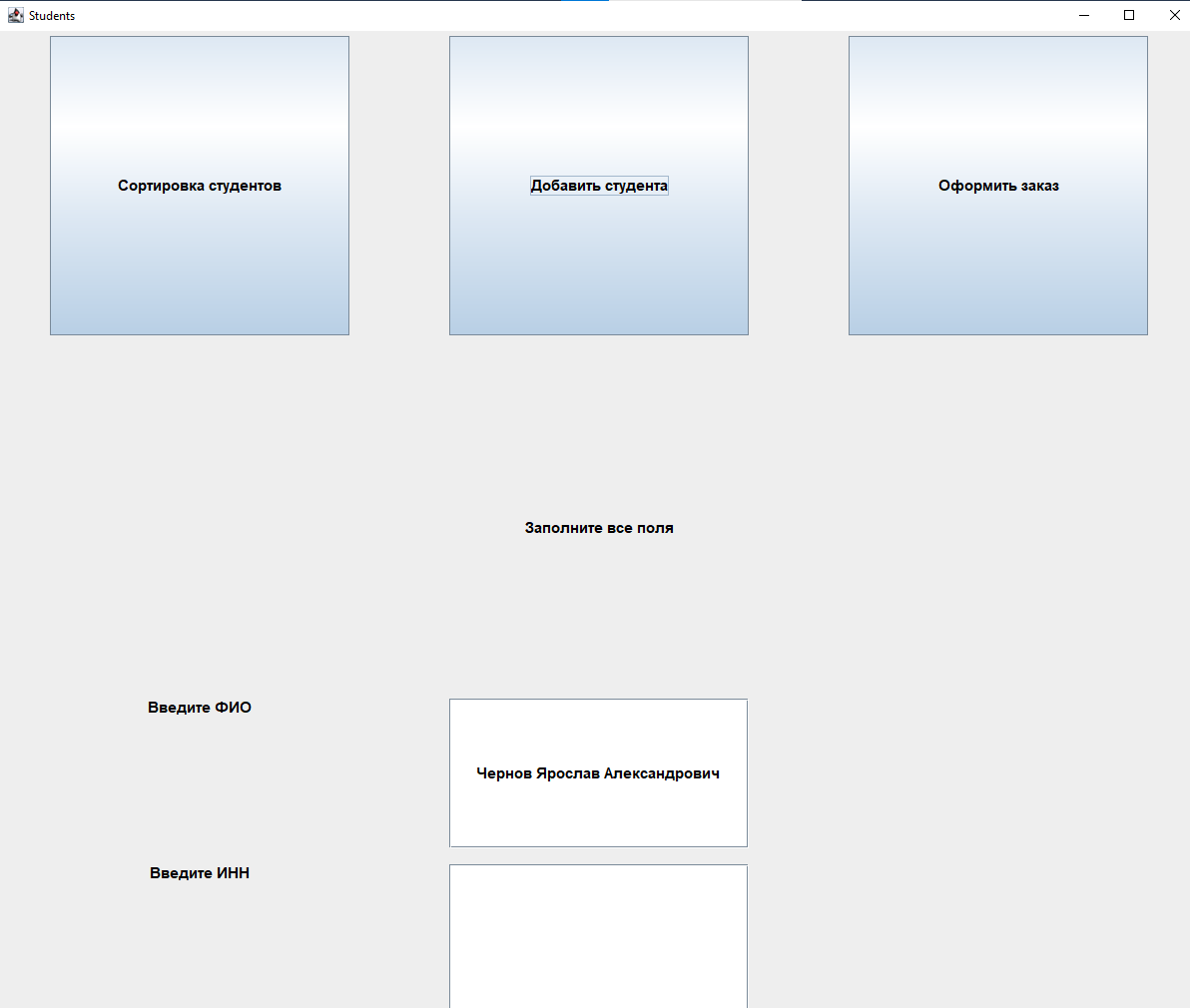


Рис. 6 – Неполностью заполненные поля

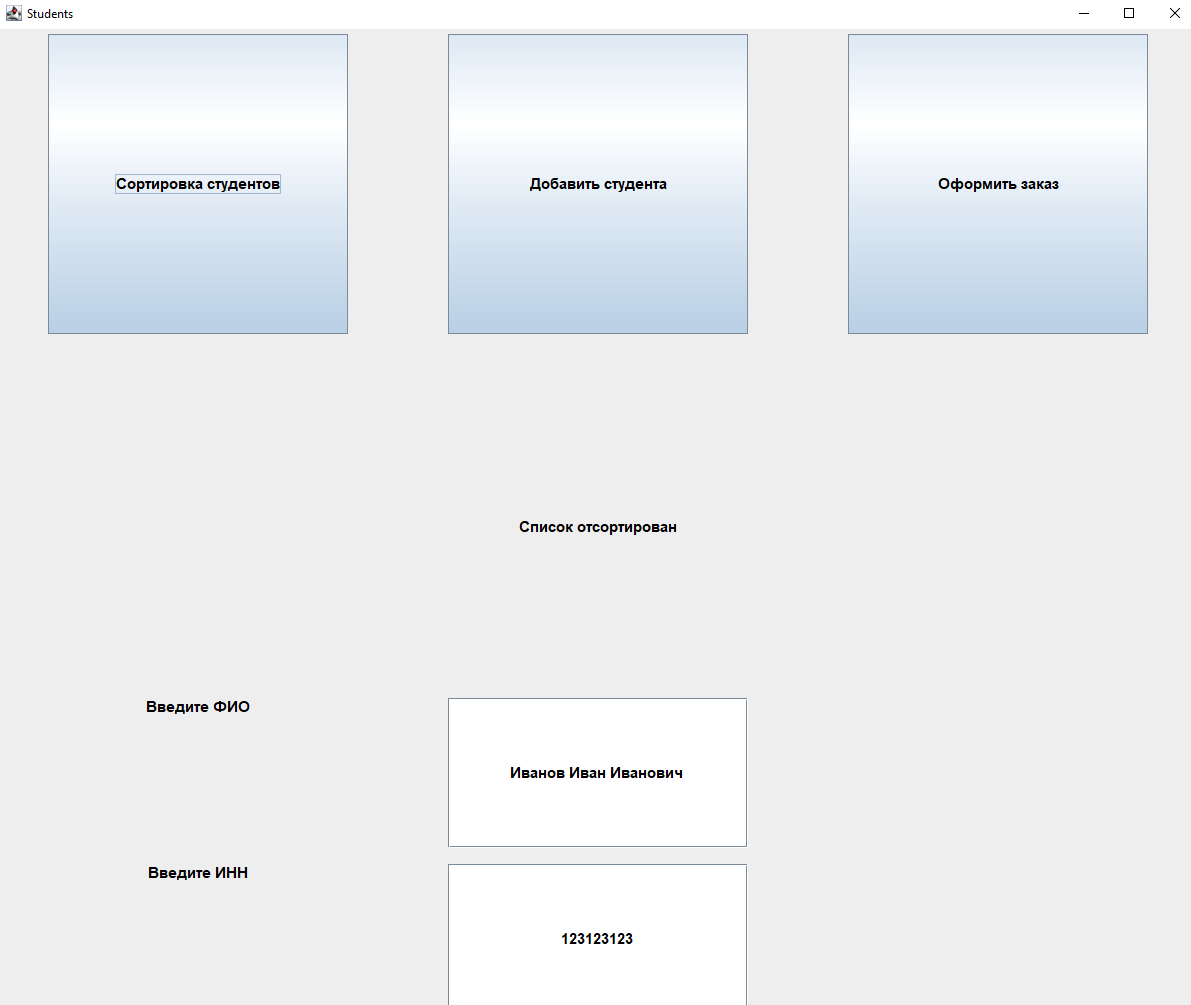


Рис.7 – Сортировка списка студентов

**Практическое занятие №10**

**Цель работы**

Цель данной практической работы – научиться применять порождающие паттерны при разработке программ на Java. В данной практической работе рекомендуется использовать следующие паттерны: Абстрактная фабрика и фабричный метод.

**Задание 2**

Реализовать класс Абстрактная фабрика для различных типов стульев: Викторианский стул, Многофункциональный стул, Магический стул, а также интерфейс Стул, от которого наследуются все классы стульев, и класс Клиент, который использует интерфейс стул в своем методе Sit (Chair chair).

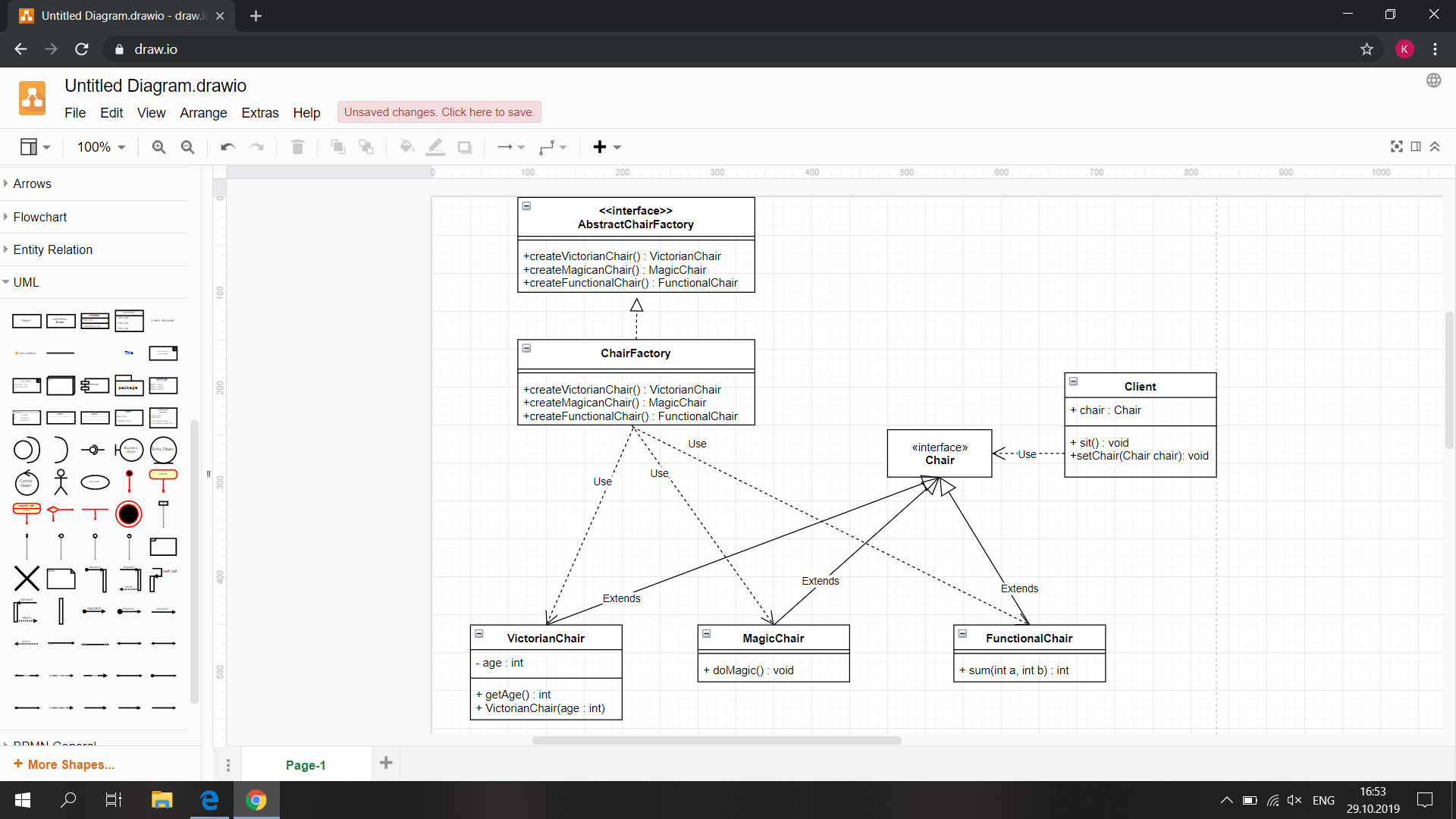


Рис.2 – UML-диаграмма

**Код программы:**

**Интерфейс AbstractChairFactory**

package Lab10;  
  
public interface AbstractChairFactory {  
 public VictorianChair createVictorianChair();  
  
 public MagicChair createMagicanChair();  
  
 public FunctionalChair createFunctionalChair();  
}

**ChairFactory**

package Lab10;  
  
public class ChairFactory implements AbstractChairFactory{  
 @Override  
 public VictorianChair createVictorianChair() {  
 return new VictorianChair();  
 }  
  
 @Override  
 public MagicChair createMagicanChair() {  
 return new MagicChair();  
 }  
  
 @Override  
 public FunctionalChair createFunctionalChair() {  
 return new FunctionalChair();  
 }  
}

**Класс Client**

package Lab10;  
  
public class Client {  
 public Chair chair;  
 public void sit() {  
 System.*out*.println("Sitting");  
 }  
  
 public Client() {  
 }  
  
 public void setChair(Chair chair) {  
 this.chair = chair;  
 }  
}

**Класс Main**

package Lab10;  
  
public class Main {  
 public static void main(String[] args) {  
 Client firstClient = new Client();  
 ChairFactory factory = new ChairFactory();  
 firstClient.setChair(factory.createMagicanChair());  
 ((MagicChair) firstClient.chair).doMagic();  
 firstClient.setChair(factory.createFunctionalChair());  
 int summa = ((FunctionalChair) firstClient.chair).sum(15, 3);  
 System.*out*.println("Summa: " + summa);  
 firstClient.setChair(factory.createVictorianChair());  
 ((VictorianChair) firstClient.chair).setAge(67);  
 int age = ((VictorianChair) firstClient.chair).getAge();  
 System.*out*.println("Age: " + age);  
 }  
}

**Интерфейс Chair**

package Lab10;  
  
public interface Chair {  
}

**Класс FunctionalChair**

package Lab10;  
  
  
public class FunctionalChair implements Chair {  
 public int sum(int a, int b){  
 return a+b;  
 }  
  
 public FunctionalChair() {  
 }  
}

**Класс MagicChair**

package Lab10;  
  
public class MagicChair implements Chair {  
 public void doMagic(){  
 System.*out*.println("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! M A G I C !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!");  
 }  
  
 public MagicChair() {  
 }  
}

**Класс VictorianChair**

package Lab10;  
  
public class VictorianChair implements Chair {  
 private int age;  
  
 public int getAge() {  
 return age;  
 }  
  
 public VictorianChair(int age) {  
 this.age = age;  
 }  
  
 public VictorianChair() {  
 }  
  
 public void setAge(int age) {  
 this.age = age;  
 }  
}

**Тестирование**

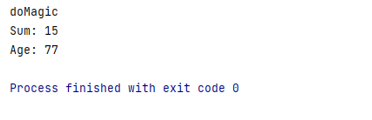


Рис.3 - Тестирование

**Выводы**

По ходу работы я научился создавать собственные исключения и реализовывать шаблоны.

Git: https://github.com/efim1111/KolmakovJavaINBO-02-20