Министерство образования и науки РФ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Тверской государственный университет»
Факультет прикладной математики и кибернетики
Направление «Прикладная математика и информатика»

ОТЧЕТ ОБ УЧЕБНОЙ ПРАКТИКЕ

Автор:
Наливайко Ярослава Олеговна
1 курс
14 группа

Научный руководитель:
Дадеркин Дмитрий Ольгердович

Задание по учебно-вычислительной практике

Вариант 22

1. Написать программу, которая выводит оценку за контрольную работу, которая выполняется путем выбора номеров ответов из представленной совокупности вариантов возможных ответов.

Имеется несколько видов контрольных работ. Информация, необходимая для оценки каждой контрольной работы:

- Наименование предмета.
- Число вопросов в контрольной работе — последовательность цифр от 1 до 5. Количество цифр равно числу вопросов. Цифры представляют собой номера правильных ответов.

Сведения об ответах студентов содержатся в файле:

- Личный номер.
- Фамилия (она запрашивается просто так и нигде не учитывается, поэтому под одним личным номером могут быть несколько разных фамилий).
- Наименование предмета.
- Ответы, выбранные студентом.

Программа должна выводить количество правильных ответов для каждого студента, оценку по каждому предмету (в пятибалльной системе):

Отл. 5 — если от 0 до 10% неправильных ответов.

Хор. 4 — если от 11 до 20%.

Уд. 3 — если от 21 до 30%.

- Предусмотреть поиск информации по фамилии студента.
- Просмотр личной карточки.
- Составить ведомость, содержащую информацию информацию обо всех неуспевающих студентах (средний балл ниже 3).

2. Создать класс «матрица n,m», где n,m – количество строк и столбцов в матрице. Каждый элемент, кроме элементов в крайних строках и столбцах, должен быть связан с соседями (звено 4-связного списка). Каждый элемент равен 0 или 1. У каждого элемента, кроме расположенных в крайних строках и столбцах, имеется 8 соседей. Заменить значение элемента на 0, если большинство соседних элементов имеет значение 0, иначе — на 1. Предусмотреть, чтобы изменение элемента не повлияло на изменение значения соседних элементов при вычислении значении следующего элемента («параллельное усреднение»).

Задание №2. 4-связный список

1. Словесное описание алгоритма

1. Задание элементов матрицы.
2. Каждый элемент матрицы через функции связывается со всеми своими соседями. Такая функция возвращает значение соседа (0 или 1), если он есть, или число -1, если элемент расположен в крайних столбцах и строках и этого соседа нет.
3. Чтобы не помешать вычислениям, создается новый массив, в который будет записываться результат работы. Для каждого элемента списка вычисляется два числа — количество нулей и количество единиц среди соседей элемента. Поочередно вызывается восемь функций, возвращающих значения соседей: если оно равно 0, то к переменной, отражающей количество нулей, прибавляется единица, если значение равно 1, то единица прибавляется к переменной, отражающей количество единиц, если же оно равно -1, то соседа нет и оба значения остаются неизменными.
4. После вычисления количества нулей и единиц среди соседей элемента эти два числа сравниваются. Если нулей строго больше чем единиц, в массив будет записано значение 0, иначе — 1.
5. Значения элементов списка заменяются значениями массива.

Примечание: Задание выполнено на языке программирования Java.

2.1 Описание классов: переменные и методы

а) MainClass – главный класс, который вызывается при запуске программы. Его главная задача — создать объект класса Spisok и задать его значения.
У этого класса есть только два поля данных: объект класса Spisok и объект класса Scanner, с помощью которого получаются значения, введенные человеком после запуска программы.
Главный метод main предназначен для задания значений, создания объекта класса Spisok и вызова метода этого объекта, заменяющего элементы списка согласно задаче.
б) Spisok – класс, воплощающий в себе 4-связный список.
У этого класса три поля данных: целочисленные количества строк и столбцов матрицы и сама матрица в виде двумерного массива.
Метод arrayMalen предназначен для вывода на экран значений матрицы.
Метод getOben возвращает значение расположенного сверху соседа элемента.
Метод getRechts возвращает значение расположенного справа соседа элемента.
Метод getUnten возвращает значение расположенного снизу соседа элемента.
Метод getLinks возвращает значение расположенного слева соседа элемента.
Метод getLinksOben возвращает значение расположенного слева сверху соседа элемента.
Метод getLinksUnten возвращает значение расположенного слева снизу соседа элемента.
Метод getRechtsOben возвращает значение расположенного справа сверху соседа элемента.
Метод getRechtsUnten возвращает значение расположенного справа снизу соседа элемента.
Метод getNeuElement вызывает перечисленные выше функции получения значений соседей элемента, подсчитывает количество нулей и единиц и возвращает число, которым должен быть заменен элемент списка.
Метод change создает массив, в который, чтобы не помешать вычислениям, записывает значения, полученные из метода getNeuElement, а затем копирует эти значения в основную матрицу.

2.2 Типы доступа

а) Метод main обязан быть типа public static.
Все поля данных объявлены private, чтобы их нельзя было изменить извне.
б) Все методы, вызываемые только внутри класса (такие, как методы, возвращающие значения соседей элемента, и getNeuElement) имеют тип private.
Методы, вызываемые в главном классе main главного класса MainClass (change и arrayMalen) обязаны быть типа public.
Все поля данных объявлены private, чтобы их нельзя было изменить извне.

2.3 Объявление методов

а) В классе MainClass есть только главный метод main:
public static void main(String [] args) {}
б) В классе Spisok 11 методов:
private int getOben(int i, int j) {}
private int getRechts(int i, int j) {}
private int getUnten(int i, int j) {}
private int getLinks(int i, int j) {}
private int getRechtsOben(int i, int j) {}
private int getLinksOben(int i, int j) {}
private int getRechtsUnten(int i, int j) {}
private int getLinksUnten(int i, int j) {}
private int getNeuElement(int i, int j) {}
public void change() {}
public void arrayMalen() {}

3.1 Реализация методов

а) Класс MainClass:

```
public static void main(String[] args){
	MainClass object = new MainClass();		//создается объект главного класса
	int m,n;					//две промежуточные переменные
	n = inp.nextInt();				//считывается количество строк
	m = inp.nextInt();				//считывается количество столбцов
	int [][] array = new int[n][m];			//создается промежуточный массив
	for (int i = 0; i < n; i++)
		for (int j = 0; j < m; j++)
			array[i][j] = inp.nextInt();	//массив заполняется значениями
	spisok = new Spisok(n, m, array);		//создается объект типа список
	spisok.change();				//матрица изменяется по задаче
	spisok.arrayMalen();				//матрица выводится на экран
}
```

б) Класс Spisok:

```
private int getOben(int i, int j){
	if (i > 0)				//если элемент не из верхней строки
		return array[i-1][j];		//вернуть значение соседа сверху
	return -1;				//=>элемент с верхней строки, вернуть -1
}
private int getUnten(int i, int j){
	if (i < n-1)				//если элемент не из нижней строки
		return array[i+1][j];		//вернуть значение соседа снизу
	return -1;				//=>элемент с нижней строки, вернуть -1
}
private int getRechts(int i, int j){
	if (j < m-1)				//если элемент не из правого столбца
		return array[i][j+1];		//вернуть значение соседа справа
	return -1;				//=>элемент из правого столбца,вернуть -1
}


private int getLinks(int i, int j){
	if (j > 0)				//если элемент не из левого столбца
		return array[i][j-1];		//вернуть значение соседа слева
	return -1;				//=>элемент из левого столбца, вернуть -1
}
```

```
private int getRechtsUnten(int i, int j){     //если элемент не из нижней строки и
      if (i < n-1 && j < m-1)                 //не из правого столбца
            return array[i+1][j+1];           //вернуть значение соседа справа снизу
      return -1;                              //=>элемент из нижней строки и/или из
}                                             //правого столбца, вернуть -1
private int getLinksOben(int i, int j){       //если элемент не из левого столбца и
      if (i > 0 && j > 0)                     //не из верхней строки
            return array[i-1][j-1];           //вернуть значение соседа слева сверху
      return -1;                              //элемент из левого столбца и/или из
}                                             //верхней строки, вернуть -1
private int getRechtsOben(int i, int j){      //если элемент не из верхней строки и
      if (i > 0 && j < m-1)                   //не из правого столбца
            return array[i-1][j+1];           //вернуть значение соседа справа сверху
      return -1;                              //=>элемент из верхней строки и/или из
}                                             //правого столбца, вернуть -1
private int getLinksUnten(int i, int j){      //если элемент не из нижней строки и
      if (i < n-1 && j > 0)                   //не из левого столбца
            return array[i+1][j-1];           //вернуть значение соседа снизу слева
      return -1;                              //=>элемент из нижней строки и/или из
}                                             //левого столбца, вернуть -1
private int getNeuElement(int i, int j){
      int zahl0 = 0, zahl1 = 0;               //количество нулей, количество единиц
      if (getRechtsOben(i,j) == 0) zahl0++;
      if (getRechtsOben(i,j) == 1) zahl1++;
      if (getOben(i,j) == 0) zahl0++;
      if (getOben(i,j) == 1) zahl1++;
      if (getLinksOben(i,j) == 0) zahl0++;
      if (getLinksOben(i,j) == 1) zahl1++;
      if (getLinks(i,j) == 0) zahl0++;
      if (getLinks(i,j) == 1) zahl1++;
      if (getRechts(i,j) == 0) zahl0++;
      if (getRechts(i,j) == 1) zahl1++;
      if (getLinksUnten(i,j) == 0) zahl0++;
      if (getLinksUnten(i,j) == 1) zahl1++;
      if (getUnten(i,j) == 0) zahl0++;
      if (getUnten(i,j) == 1) zahl1++;
      if (getRechtsUnten(i,j) == 0) zahl0++;
      if (getRechtsUnten(i,j) == 1) zahl1++;
      if (zahl0 > zahl1)
            return 0;
      return 1;
}

public void change() {
      int[][] array2 = new int[n][m];                    //промежуточный массив
      for (int i = 0; i < n; i++)
            for (int j = 0; j < m; j++)
                  array2[i][j] = getNeuElement(i, j);    //записываются значения
      for (int i = 0; i < n; i++)
            for (int j = 0; j < m; j++)                  //из промежуточного массива
                  array[i][j] = array2[i][j];            //копировать в матрицу
}

public void arrayMalen(){
      for (int i = 0; i < n; i++){
            for (int j = 0; j < m; j++)
                  System.out.print(array[i][j] + " ");   //вывод матрицы на экран
            System.out.println();
      }
}
```

3.2 Конструкторы и деструкторы

Только в классе Spisok явно задан конструктор. Деструктор не задан явно ни в одном классе.

```
public Spisok(int n, int m, int[][]array) {
	this.n = n;                              //одновременно с созданием объекта
	this.m = m;                              //необходимо задать и матрицу
	this.array = array;
}
```

4.1 Использование классов. Экземпляры классов

В главном методе main класса MainClass создается объект класса MainClass, для которого затем создается объект класса Spisok. С этим объектом производятся все необходимые вычисления.

4.2 Размещение экземпляра класса в куче

В программах, написанных на языке программирования Java все локальные переменные примитивного типа хранятся в стеке, а объекты классов и массивы примитивных типов хранятся в куче.

4.3 Вызов конструктора с параметрами

Конструктор с параметрами вызывается один раз: при создании объекта типа Spisok (spisok = new Spisok(n,m,array);), при этом ему передаются размеры матрицы и двумерный массив ее элементов.

4.4 Вызов методов для экземпляров классов

В главном методе main класса MainClass используется два метода для объекта spisok класса Spisok:

spisok.change(); //изменяет матрицу так, как задано в задаче
spisok.arrayMalen(); //выводит матрицу на экран

Задание №1. Контрольные работы

1. Словесное описание алгоритма

1. При запуске программы из файлов считывается информация о студентах и о контрольных работах и сохраняется в соответствующих массивах.

2. Открывается окно главного меню и пользователю предоставляется возможность совершить одно из предложенных действий: «Задать новую контрольную работу», «Задать нового студента», «Открыть информацию о студентах», «Составить ведомость» и «Выход».

а) «Задать новую контрольную работу»

Пользователь заносит данные в два поля ввода: «Наименование предмета» и «Ответы». При нажатии кнопки «Подтвердить» новая контрольная работа добавляется в массив и обновленный массив записывается в файл, при нажатии кнопки «Отмена» пользователь возвращается в главное меню.

б) «Задать нового студента»

Пользователь заносит данные в четыре поля ввода: «Личный номер», «Фамилия студента», «Название предмета» и «Ответы на вопросы». Из этих данных создается новый студент и его первая контрольная работа. При нажатии кнопки «Подтвердить» он добавляется в массив и обновленный массив студентов записывается в файл, при нажатии кнопки «Отмена» пользователь возвращается в главное меню.

в) «Открыть информацию о студентах»

Пользователь имеет возможность выбрать по фамилии из списка одного из уже существующих студентов и просмотреть информацию о нем: личный номер, фамилию, список сданных контрольных работ, количество правильных ответов и балл для каждой из них. Затем пользователь может задать для студента новую контрольную работу, которая добавится в массив решенных им и запишется в файл.

г) «Составить ведомость»

Для каждого студента список решенных контрольных работ сравнивается со списком существующих контрольных работ и высчитывается средний балл. Если этот балл меньше 3 (неуд.), то студент появляется на экране, как один из неуспевающих.

д) «Выход»

Закрыть программу.

Примечание: Задание выполнена на языке программирования Java.

2.1 Описание классов: переменные и методы

а) MainClass – главный класс, который вызывается при запуске программы. Его главная задача — создать окно, создать объект класса Platten и обрабатывать получаемые от него значения.

В классе MainClass три поля данных: объект класса Platten, панель класса JPanel, строка mainName, представляющее, какое окно открыто в данный момент.

Главный метод main используется для создание объекта класса MainClass и создания окна.

Метод vorbereiten нужен для того, чтобы такие объекты интерфейса, как JButton и JComboBox, могли функционировать.

Метод actionPerformed используется для обработки действий, совершаемых в окне. Он пересылает их в класс Platten, откуда получает строку. Если эта строка не пуста и не эквивалентна строке mainName, то переходим в другое окно.

б) TextReader – класс, отвечающий за чтение из файла и запись в в него.

Его единственное поле данных — это строка road, указывающая путь к файлам.

Метод writeStudentenText записывает переданный в качестве аргумента массив студентов в файл Studenten.txt. Он используется всякий раз, когда задается новый студент или новая решенная контрольная работа.

Метод writeArbeitenText записывает переданный в качестве аргумента массив контрольных работ в файл KontrolArbeiten.txt. Он используется всякий раз, когда задается новая контрольная работа.

Метод readStudentenText записывает в переданный в качестве аргумента массив студентов всех студентов, чьи данные хранятся в файле Studenten.txt. Используется только при запуске программы.

Метод readArbeitenText записывает в переданный в качестве аргумента массив контрольных работ все контрольные работы, данные которых хранятся в файле KontrolArbeiten.txt . Используется только при запуске программы.

в) Класс KontrolArbeit используется для двух целей: в нем может храниться как и правильные ответы на контрольные, так и ответы студентов.

У этого класса два поля данных: строка fachName, хранящая наименование предмета, и динамический массив fragen, хранящий в целых числах ответы на вопросы.

Метод getName возвращает строку — наименование предмета fachName.

Метод fragenSize возвращается целое число — количество вопросов в контрольной работе, то есть длину массива fragen.

Метод getFrage возвращает целое число — i-ый элемент массива fragen, где i – это переданное в качестве аргумента целое число.

г) Класс Student представляет собой одного студента.

У него три поля данных: строки nummer и name, хранящие личный номер и фамилию студента соответственно, и динамический массив arbeiten, хранящий все решенные студентом контрольные работы.

Метод getNummer возвращает строку — личный номер студента nummer.

Метод getName возвращает строку — фамилию студента name.
Метод getSize возвращает целое число — количество решенных студентом контрольных работ, то есть длину массива arbeiten.
Метод getArbeitSize возвращает целое число — количество вопросов в i-ой контрольной работе студента, где i– это переданное в качестве аргумента число.
Метод getArbeit возвращает объект класса KontrolArbeit – i-ую контрольную работу, решенную студентом, где i– это переданное в качестве аргумента число.
Метод putArbeit добавляет в динамический массив arbeiten новую контрольную работу, созданную с помощью переданных в качестве аргументов строки — наименования работы и динамического массива целых чисел — ответов на вопросы.
д) Класс Platten – самый большой класс, он отвечает за интерфейс и основные вычисление. В нем есть пять полей данных:
1. Объект класса TextReader reader.
2. Динамический массив студентов studenten .
3. Динамический массив контрольных работ arbeiten.
4. Целое число aktuellerStudent, отображающее, с каким именно студентом в данный момент работает программа.
5. Map<String,ArrayList> inf, использующийся для хранения и использование различных объектов, относящихся к интерфейсу.
Методы класса Platten:
Метод create получает в качестве аргументов строку и панель. В зависимости от того, какое слово хранится в строке, на панель добавляются объекты, так создается интерфейс.
Метод QuelleBearbeiten получает в качестве аргументов строку и объект Quelle – действие, произошедшее в окне. Сначала с помощью строки ищется название окна, открытого в данный момент, затем с помощью массива inf определяется, какая именно кнопка была нажата. Этот метод осуществляет переходы между окнами и самые главные действия: своевременный вызов других методов, создающих студентов и контрольные работы, и методов, записывающих массивы в файлы.
Метод toList используется для преобразования введенных пользователем данных в массив заданных ответов для контрольной работы.
Метод getOtvet получает в качестве аргументов две контрольных работы и сравнивает, сколько ответов из них совпадает. Используется для выяснения того, сколько правильных ответов выбрал студент: как аргументы передаются работа студента и оригинальная работа с правильными ответами.
Метод getBall поначалу использует тот же алгоритм, что и getOtvet: считает, сколько ответов правильно. Затем высчитывается сколько процентов из ста соответствует каждому вопросу и наконец высчитывается балл по сто балльной системе. Из нее получается балл по пятибалльной системе.

2.2 Типы доступа

а) MainClass:
Все поля данных объявлены private, чтобы их нельзя было изменить извне.
Метод main обязан быть типа public static.
Метод vorbereiten объявлен типа private, так как используется только в классе MainClass.
Метод actionPerformed обязан быть типа public.
б) TextReader:
Все методы записи и чтения должны быть типа public, чтобы их можно было использовать вне класса.
в) KontrolArbeit:
Все методы должны быть public, чтобы их можно было использовать вне класса.
г) Student:
Все методы должны быть public, чтобы их можно было использовать вне класса.
д) Platten:
Методы QuelleBearbeiten и create должны быть типа public, так как используются вне класса.
Методы getBall, getOtvet и toList имеют тип private, так как используются только внутри класса.

2.3 Объявление методов

а) MainClass:
```
public static void main(String [] args) {}
private void vorbereiten(String name) {}
public void actionPerformed(ActionEvent Ereignis) {}
```
б) TextReader:
```
public void writeStudentenText(ArrayList<Student> studenten) {}
public void writeArbeitenText(ArrayList<KontrolArbeit> arbeiten) {}
public void readStudentenText(ArrayList<Student> studenten) {}
public void readArbeitenText(ArrayList<KontrolArbeit> arbeiten) {}
```
в) KontrolArbeit:
```
public String getName() {}
public int fragenSize() {}
public int getFrage(int i) {}
```
г) Student:
```
public String getNummer() {}
public String getName() {}
public int getSize() {}
public int getArbeitSize(int i) {}
public KontrolArbeit getArbeit(int i) {}
public void putArbeit(String name, ArrayList<Integer> fragen) {}
```

д) Platten:
public String QuelleBearbeiten(String name, Object Quelle) {}
public JPanel create(String Text, Jpanel Platte) {}
private int getBall(KontrolArbeit arbeit, KontrolArbeit antworten) {}
private int getOtvet(KontrolArbeit arbeit, KontrolArbeit antworten) {}
private ArrayList<Integer> toList (JtextArea textArea) {}

3.1 Реализация методов

а) MainClass:

```
public static void main(String[] args) {
	MainClass Frame = new MainClass();			//создание объекта класса
	Frame.setSize(800,800);
	Frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	Frame.setVisible(true);
}
private void vorbereiten(String name){
	int zahl = Platte.inf.get(name+"Button").size();	//число кнопок на панели
	JButton button;
	for (int i = 0; i < zahl; i++){
		button = (JButton) Platte.inf.get(name+"Button").get(i);
		button.addActionListener(this);			//добавить слушателя
	}
	zahl = Platte.inf.get(name+"Box").size();		//число списков на панели
	JComboBox box;
	for (int i = 0; i < zahl; i++){
		box = (JComboBox) Platte.inf.get(name+"Box").get(i);
		box.addActionListener(this);			//добавить слушателя
	}
}
public void actionPerformed(ActionEvent Ereignis) {
	Object Quelle = Ereignis.getSource();			//получить действие из окна
	String Text = Platte.QuelleBearbeiten(mainName, Quelle);   //получить указание
	if (!Text.equals("") && !Text.equals(mainName)){	//если название другое
		getContentPane().removeAll();
		mainPlatte = Platte.create(Text, mainPlatte);
		setContentPane(mainPlatte);
		mainName = Text;
		vorbereiten(Text);				//перейти в другое окно
	}
}
```

б) TextReader

```
public void writeStudentenText(ArrayList<Student> studenten){
	try {
		BufferedWriter Datei = new BufferedWriter(new FileWriter(road +
"Studenten.txt"));				//открыть файл
		for (int i = 0; i < studenten.size(); i++){
			Datei.write(studenten.get(i).getNummer() + "\r\n");//записать номер
			Datei.write(studenten.get(i).getName() + "\r\n");//записать фамилию
			for (int j = 0; j < studenten.get(i).getSize(); j++){
				Datei.write(studenten.get(i).getArbeit(j).getName() +
"\r\n");	//записать наименование работы
				for (int a = 0; a < studenten.get(i).getArbeitSize(j); a++)
				Datei.write(studenten.get(i).getArbeit(j).getFrage(a)
+"\r\n");
				Datei.write("-\r\n");
				if (j != studenten.get(i).getSize()-1)
				Datei.write("+\r\n");
			}
```

```
                Datei.write("*\r\n");
            }
            Datei.close();
        }
        catch(IOException x){
            JOptionPane.showMessageDialog(null, "Kann Daten nicht speichern!");
        }
}
```

Функция сохраняет данные следующим образом:

Личный номер1

Фамилия1

Наименование предмета1

Ответ1

…

Ответ n

-

+ //значит, что дальше есть еще контрольные

Наименование предмета2

Ответ m

…

Ответ r

-

* //конец данных об одном студенте

```
public void writeArbeitenText(ArrayList<KontrolArbeit> arbeiten){
        try {
            BufferedWriter Datei = new BufferedWriter(new FileWriter(road +
"KontrolArbeiten.txt"));
            for (int i = 0; i < arbeiten.size(); i++)
            {
                Datei.write(arbeiten.get(i).getName() + "\r\n");
                for (int j = 0; j < arbeiten.get(i).fragenSize(); j++)
                    Datei.write(arbeiten.get(i).getFrage(j) + "\r\n");
                Datei.write("*\r\n");
            }
            Datei.close();
        }
        catch(IOException x){
            JOptionPane.showMessageDialog(null, "Kann Daten nicht speichern!");
        }
}
```

Функция сохраняет данные следующим образом:

Наименование предмета1

Ответ1

…

Ответ n

*

Методы readStudentenText и readArbeitenText можно найти в приложении. Они считывают данные из файлов согласно предоставленным выше примерам.

в) KontrolArbeit

```
public String getName()                 //возвращает наименование предмета
{
	return fachName;
}
public int fragenSize()                 //возвращает количество вопросов
{
	return fragen.size();
}

public int getFrage(int i)              //возвращает вопрос
{
	return fragen.get(i);
}
```

г) Student:

```
public String getNummer()        //возвращает личный номер студента
{
	return nummer;
}

public String getName()          //возвращает фамилию студента
{
	return name;
}

public int getSize()             //возвращает количество работ студента
{
	return arbeiten.size();
}

public int getArbeitSize(int i)  //возвращает количество вопросов в работе студента
{
	return arbeiten.get(i).fragenSize();
}

public void putArbeit(String name, ArrayList<Integer> fragen)//добавляет новую работу
{
	arbeiten.add(new KontrolArbeit(name,fragen));
}

public KontrolArbeit getArbeit(int i)         //возвращает работу
{
	return arbeiten.get(i);
}
```

д) Platten:

Методы create и QuelleBearbeiten можно найти в приложении.

```
private int getBall(KontrolArbeit arbeit, KontrolArbeit antworten)
{
	if (arbeit.fragenSize() != antworten.fragenSize())
		return -1;                //если количество вопросов не равно, вернуть -1
	boolean[] ball = new boolean[arbeit.fragenSize()];
	for (int i = 0; i < arbeit.fragenSize(); i++)
		if (arbeit.getFrage(i) == antworten.getFrage(i))
			ball[i] = true;           //верный ответ
		else ball[i] = false;             //неверный ответ
	float teil = (float) (100 / arbeit.fragenSize()), summe = 0;
	for (int i = 0; i < arbeit.fragenSize(); i++)
		if (ball[i])
```

```
                summe += teil;
        if (summe >= 90.0)
                return 5;
        if (summe >= 80.0)
                return 4;
        if (summe >= 70.0)
                return 3;
        return 2;
}

private int getOtvet(KontrolArbeit arbeit, KontrolArbeit antworten)
{
        if (arbeit.fragenSize() != antworten.fragenSize())
                return -1;
        boolean[] ball = new boolean[arbeit.fragenSize()];
        for (int i = 0; i < arbeit.fragenSize(); i++)
                if (arbeit.getFrage(i) == antworten.getFrage(i))
                        ball[i] = true;
                else ball[i] = false;
        int summe = 0;
        for (int i = 0; i < arbeit.fragenSize(); i++)
                if (ball[i])
                        summe++;            //если ответ верный, плюс один к количеству
        return summe;
}

private ArrayList<Integer> toList(JTextArea textArea) {
          String[] al = textArea.getText().split(" ");       //разделить по пробелу
          ArrayList<Integer> arrays = new ArrayList<Integer>();
          for (int a = 0; a < al.length; a++)
                arrays.add(Integer.parseInt(al[a]));         //добавить как целое
          return arrays;
}
```

3.2 Явное задание конструктора с параметрами

а)
```
public MainClass(){
        mainName = "ChoiceWindow";              //задается первое окно
        Platte = new Platten();
        mainPlatte = Platte.create(mainName, mainPlatte);
        setContentPane(mainPlatte);
        vorbereiten(mainName);
}
```

б)
```
public TextReader(){
        File myPath = new File(road);
        myPath.mkdir();
        myPath.mkdirs();                //если не обнаружена папка, то она создается
        try {
                File file = new File(road + "Studenten.txt");
               file.createNewFile();      //если не обнаружен файл, то он создается
        }
        catch (IOException e) {
               e.printStackTrace();
        }
        try {
                File file = new File(road + "KontrolArbeiten.txt");
                file.createNewFile();     //если не обнаружен файл, то он создается
        }
        catch (IOException e) {
               e.printStackTrace();
        }
```

```
}
```
в)
```
public KontrolArbeit(String name, ArrayList<Integer> antworten){
	fachName = name;			//поля данных задаются сразу же
	fragen = antworten;
}
```
г)
```
public Student(String nummer, String name, ArrayList<KontrolArbeit> arbeiten){
	this.nummer = nummer;
	this.name = name;
	this.arbeiten = arbeiten;		//поля данных задаются сразу же
}
```
д)
```
public Platten(){
	reader.readArbeitenText(arbeiten);	//при создании читаются данные из файлов
	reader.readStudentenText(studenten);
}
```

4.1 Использование классов. Экземпляры классов

В главном методе main класса MainClass создается объект класса MainClass, в конструкторе этого же класса создается объект класса Platten, отвечающего за интерфейс и основные вычисления. В классе Platten создается объект класса TextReader, массив объектов класса Student и массив объекта класса KontrolArbeit. В объекте класса Student также создается массив класса KontrolArbeit.

В методе putArbeit класса Student создается объект класса KontrolArbeit, который добавляется в массив.

В методе readArbeitenText класса readText создается объект класса KontrolArbeit, который добавляется в массив.

В методе readStudentenText класса readText создается объект класса Student и массив объектов класса KontrolArbeit, использующиеся для чтения из файла.

В методе QuelleBearbeiten для создания нового студента создается массив объектов класса KontrolArbeit и объект этого же класса, который добавляется в этот массив.

В методе create при подсчете балла за контрольную работу создается два объекта класса KontrolArbeit для упрощения кода программы. Так же делается при составлении ведомости.

4.2 Размещение экземпляра класса в куче

В программах, написанных на языке программирования Java все локальные переменные примитивного типа хранятся в стеке, а объекты классов и массивы примитивных типов хранятся в куче.

4.3 Вызов конструктора с параметрами

В классах MainClass и KontrolArbeit не вызывается ни одного конструктора с параметрами.

В классе Student конструктор с параметрами используется один раз — в методе putArbeit.

В методе readArbeitenText класса readText используется конструктор с параметрами класса KontrolArbeit.

В методе readStudentenText класса readText используется конструктор с параметрами класса Student.
В методе QuelleBearbeiten для создания нового студента создается объект класса KontrolArbeit, аргументы которого считываются из полей ввода.

4.4 Вызов методов для экземпляров классов

В класса MainClass используется два метода для объекта Platte класса Platten:
String Text = Platte.QuelleBearbeiten(mainName,Quelle);
В строке сохраняется указание от метода о том, что делать дальше.
MainPlatte = Platte.create(Text, mainPlatte);
Устанавливается новое окно.
В классе Student в методе getArbeitSize для того, чтобы узнать, сколько в работе вопросов используется:
return arbeit.get(i).fragenSize();
В классе KontrolArbeit нет экземпляров классов.
В классе readText для созданных экземпляров классов не вызываются конструкторы.
В классе Platten для работы с файлами используются все четыре метода класса TextReader объекта reader.
В классе Platten для вычисления баллов за контрольную, для добавления контрольных работ, для создания списка студентов используются такие методы классов KontrolArbeit и Student:
studenten.get(aktuellerStudent).putArbeit(texts.get(0).getText(),array);
Leute[i] = studenten.get(i).getName();
И так далее.

Список использованной литературы

1. Java für Kids.
2. Java lernen mit BlueJ.
3. Java von Kopf bis Fuß.