Узбекский комитет связи и информатизации

Ташкентский Университет Информационный Технологий

Курсовая работа

По предмету:

«Объектно – ориентированный языки программирования»

Выполнил: Джалилов Г. Г.

Студент группы 224-10 ИТр.

Ташкент - 2013

«Утверждаю»

Заведующий кафедрой

_____

«__»_____201_г.

Задание

На курсовую работу

По дисциплине: Объектно-ориентированные языки программирования

Студент: Джалилов Г.Г. группа 224-10 ИТр

Защита курсовой работы: _____

Тема: Классы (тема №2). Номер варианта 2.5

Задание: Определить иерархию объектов (в соответствии с вариантом). Реализовать путем наследования методы для автоматизации поиска указанной информации и указанных операций.

Вариант: **2.5.** ОБЩЕЖИТИЕ (название, КОРПУСА,КОМНАТЫ, СТУДЕНТЫ, ФАКУЛЬТЕТЫ)

Все данные относительно комнаты

Содержание работы. Объем __ листов А4.

Подпись преподавателя: _____

Подпись студента: _____

Дата выдачи задания:

Аннотация

Данная курсовая работа по предмету: «Объектно-ориентированные языки программирования» на тему: «Отношения между классами (тема №2). Номер варианта 2.5». В данной курсовой работе рассматриваются иерархия объектов (в соответствии с вариантом). Реализованные путем наследования методы для автоматизации поиска указанной информации и указанных операций – Вывод данных относительно комнаты. Программа разработана в среде программирования Java, в среде Eclipse.

Объем работы ____ листов

Содержание

Цель

Создание иерархии объектов.

Постановка задачи

Определить иерархию объектов (в соответствии с вариантом). Реализовать путем наследования методы для автоматизации поиска указанной информации и указанных операций.

Вариант задания

Вариант 2.5 : ОБЩЕЖИТИЕ (название, КОРПУСА,КОМНАТЫ, СТУДЕНТЫ, ФАКУЛЬТЕТЫ).

Реализация задания

Теоретическая часть

Насле́дование — механизм объектно-ориентированного программирования (наряду с инкапсуляцией, полиморфизмом и абстракцией), позволяющий описать новый класс на основе уже существующего (родительского), при этом свойства и функциональность родительского класса заимствуются новым классом.

Другими словами, класс-наследник реализует спецификацию уже существующего класса (базовый класс). Это позволяет обращаться с объектами класса-наследника точно так же, как с объектами базового класса...

Класс, от которого произошло наследование, называется базовым или родительским (англ. base class). Классы, которые произошли от базового, называются потомками, наследниками или производными классами (англ. derived class).

В некоторых языках используются абстрактные классы. Абстрактный класс — это класс, содержащий хотя бы один абстрактный метод, он описан в программе, имеет поля, методы и не может использоваться для непосредственного создания объекта. То есть от абстрактного класса можно только наследовать. Объекты создаются только на основе производных классов, наследованных от абстрактного. Например, абстрактным классом может быть базовый класс «сотрудник вуза», от которого наследуются классы «аспирант», «профессор» и т. д. Так как производные классы имеют общие поля и функции (например, поле «год рождения»), то эти члены класса могут быть описаны в базовом классе. В программе создаются объекты на основе классов «аспирант», «профессор», но нет смысла создавать объект на основе класса «сотрудник вуза».

Множественное наследование

При множественном наследовании у класса может быть более одного предка. В этом случае класс наследует методы всех предков. Достоинства такого подхода в большей гибкости. Множественное наследование реализовано в C++. Из других языков, предоставляющих эту возможность, можно отметить Python и Эйфель. Множественное наследование поддерживается в языке UML.

Множественное наследование — потенциальный источник ошибок, которые могут возникнуть из-за наличия одинаковых имен методов в предках. В языках, которые позиционируются как наследники C++ (Java, C# и др.), от множественного наследования было решено отказаться в пользу интерфейсов. Практически всегда можно обойтись без использования данного механизма. Однако, если такая необходимость все-таки возникла, то, для разрешения конфликтов использования наследованных методов с одинаковыми именами, возможно, например, применить операцию расширения видимости — «::» — для вызова конкретного метода конкретного родителя.

Попытка решения проблемы наличия одинаковых имен методов в предках была предпринята в языке Эйфель, в котором при описании нового класса необходимо явно указывать импортируемые члены каждого из наследуемых классов и их именование в дочернем классе.

Большинство современных объектно-ориентированных языков программирования (C#, Java, Delphi и др.) поддерживают возможность одновременно наследоваться от класса-предка и реализовать методы нескольких интерфейсов одним и тем же классом. Этот механизм позволяет во многом заменить множественное наследование — методы интерфейсов необходимо переопределять явно, что исключает ошибки при наследовании функциональности одинаковых методов различных классов-предков.

Наследование дает возможность объявить производный класс, который наследует свойства, данные, методы и события всех своих предшественников в иерархии классов, а также может объявлять новые характеристики и перегружать некоторые из наследуемых функций.

В JAVA непосредственный родитель класса называется его суперклассом.

Обобщенный синтаксис объявления производного класса:

class <имя класса> extends <имя родительского класса> {…}

Наследование конструкторов

Поскольку конструкторы не наследуются, при создании производного класса наследуемые им данные-члены должны инициализироваться конструктором базового класса. Конструктор базового класса вызывается автоматически и выполняется до конструктора производного класса. Параметры конструктора базового класса указываются в определении конструктора производного класса с помощью ключевого слова **super.**

Виртуальные методы

К механизму виртуальных функций обращаются в тех случаях, когда в каждом производном классе требуется свой вариант некоторой

компонентной функции. Классы, включающие такие функции, называются **полиморфными** и играют особую роль в ООП.

Виртуальные функции предоставляют механизм **позднего** (**отложенного**) или **динамического связывания.**

При позднем связывании адреса определяются динамически во время выполнения программы, а не статически во время компиляции, как в традиционных компилируемых языках, в которых применяется *раннее связывание*.

Виртуальность наследуется. После того как функция определена как виртуальная, ее повторное определение в производном классе (с тем же самым прототипом) создает в этом классе новую виртуальную функцию.

В JAVA все методы являются виртуальными.

Все методы и переменные объектов могут быть замещены по умолчанию. Если же вы хотите объявить, что подклассы не имеют права замещать какие-либо переменные и методы вашего класса, вам нужно объявить их как final (в Delphi / C++ не писать слово **virtual**).

final int FILE_NEW = 1;

По общепринятому соглашению при выборе имен переменных типа final — используются только символы верхнего регистра (т.е. используются как аналог препроцерных констант C++). Использование final-методов порой приводит к выигрышу в скорости выполнения кода — поскольку они не могут быть замещены, транслятору ничто не мешает заменять их вызовы *встроенным* (in-line) кодом (байт-код копируется непосредственно в код вызывающего метода).

Реализация. Описание процедур

```
package javaapplication12;
import javax.swing.*;
```

Класс комната

```
class comnata {
      public float площадь;
      comnata (float площадь){
            this.площадь=площадь;
      }
}
```

Класс корпуса, где одним из полей является объект класса комната

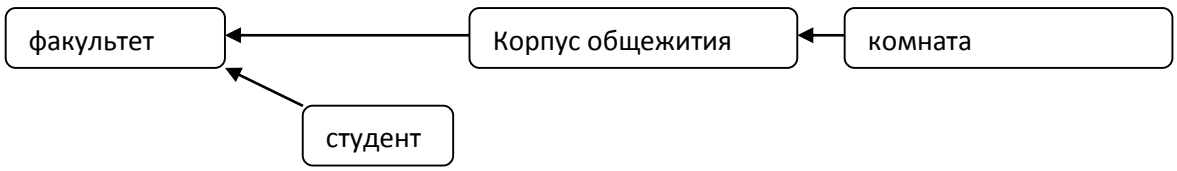
```
class korpusa{
      public String название;
      public comnata comnati;
      korpusa (String название, float площадь){
            this.название=название;
            comnati = new comnata(площадь);
      }
      public void print (){
            System.out.println("Название корпуса: "+название);
            System.out.println("Площадь комнат: "+comnati.площадь);
      }
}
```

Класс студент

```
class student{
      public String ФИО;
      public int возраст;
      student (String ФИО,int возраст){
            this.ФИО=ФИО;
            this.возраст=возраст;
      }
}
```

Класс Факультет. Во первых наследует класс корпуса. Как я думаю, наверно каждый факультет имеет свое собственное общежитие. Кроме того в классе факультеты в качестве поля имеется объект класса студент.

Имеем следующую иерархию




```
class fakultet extends korpusa{
      public String декан;
      public student студент;
      public String название_факультета;
```

```
      fakultet (String декан,String название_факультета,String ФИО,int
возраст,String название, float площадь){
            super(название, площадь);
            this.декан=декан;
            this.название_факультета=название_факультета;
            студент = new student (ФИО,возраст);
      }
```

Выводятся все данные относительно комнаты,

```
      public void show(){
            print();
            System.out.println("декан: " + декан);
            System.out.println("название_факультета: " + название_факультета);
            System.out.println("студента возраст: " +студент.возраст);
            System.out.println("студента ФИО: "+студент.ФИО);
            System.out.println("декан: " + декан);
      }
}
```

Запрос, который реализует поиск в определенном корпусе общежития, определенного факультета количество студентов

```
class zapros{
      public void find_name(fakultet mas_fak[]){
            System.out.println("Запрос по названию");
            String название_найти = JOptionPane.showInputDialog("ВВЕДИТЕ
название Корпуса:");
            String название_факультета = JOptionPane.showInputDialog("ВВЕДИТЕ
название факультета:");
            int количество=0;
```

Циклическая прокрутка массива, с проверкой по заданному условию

```
            for (int i=0; i<mas_fak.length; i++){
                  if (mas_fak[i].название.equals(название_найти)){
                        if
(mas_fak[i].название_факультета.equals(название_факультета)){
                              количество++;
                        }
                  }
            }
            System.out.println("Количество студентов в корпусе" +
                        название_найти+" и в факультете" +
                                    ""+ название_факультета+ "="+количество
                        );
      }

}

public class my_own_class {
      /**
       * @param args
       */
      public static void main(String[] args) {
            // TODO Auto-generated method stub
```

Запрашиваем у пользователя количество факультетов для записи

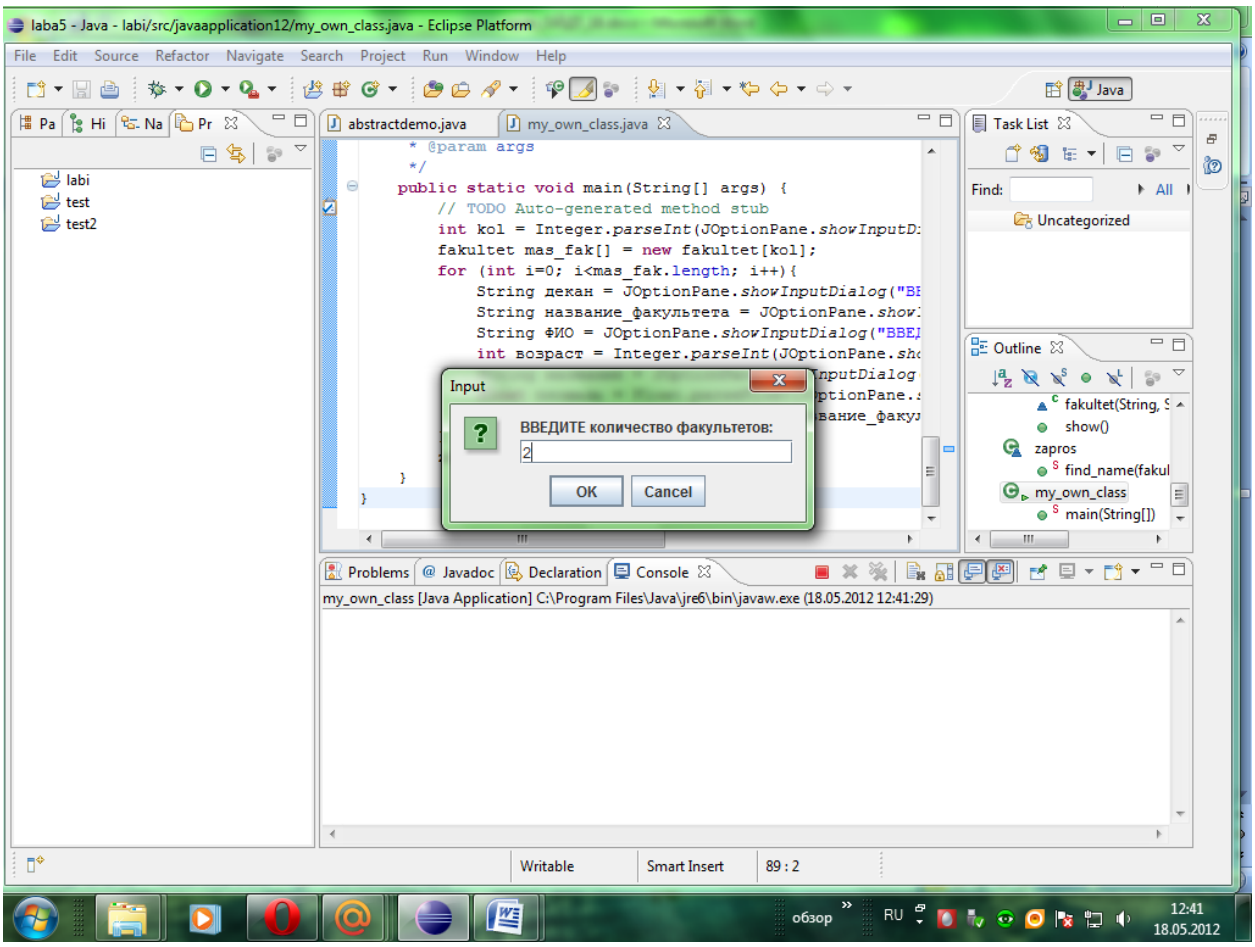
```
            int kol = Integer.parseInt(JOptionPane.showInputDialog("ВВЕДИТЕ
количество факультетов:"));
```

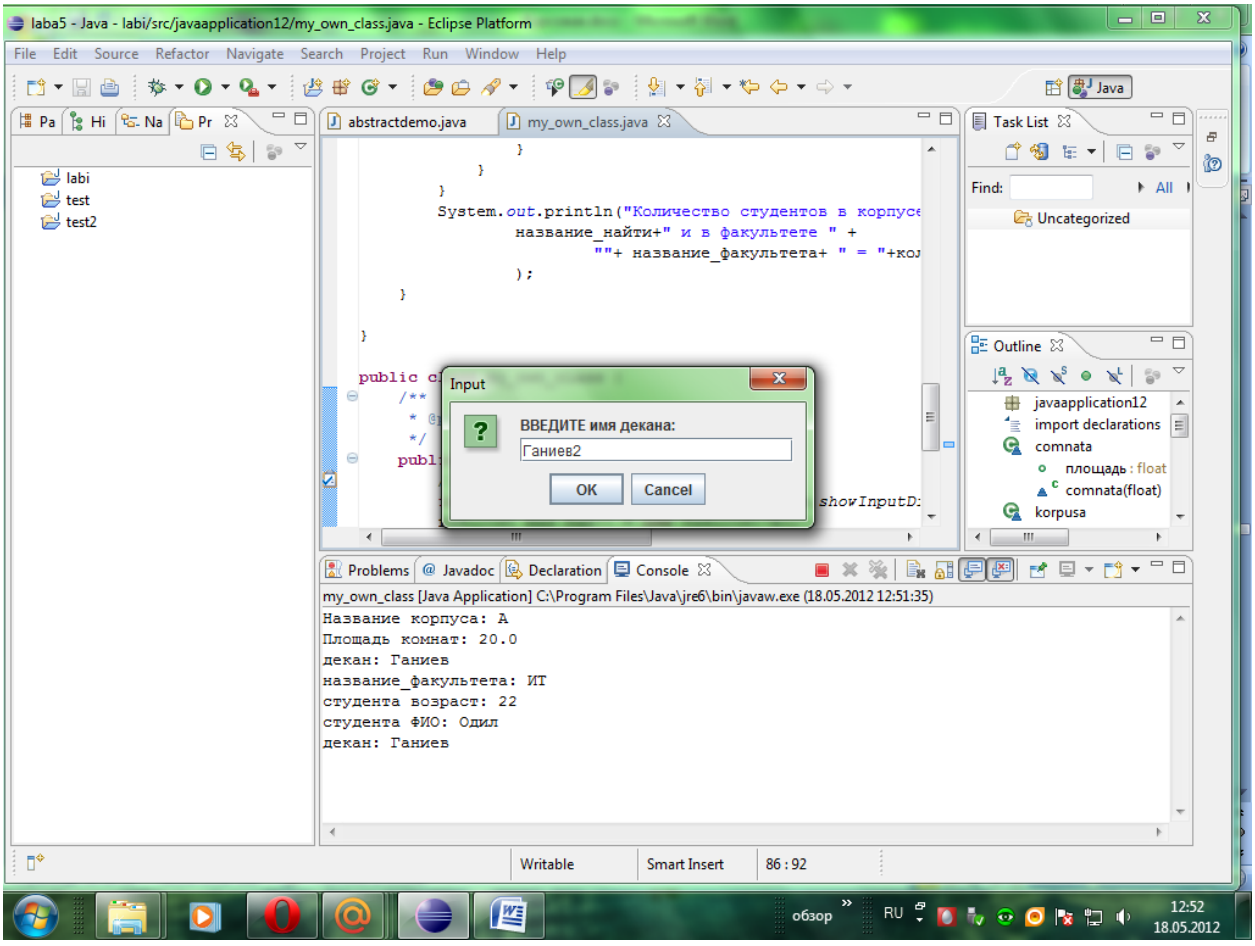
Создаем массив факультетов

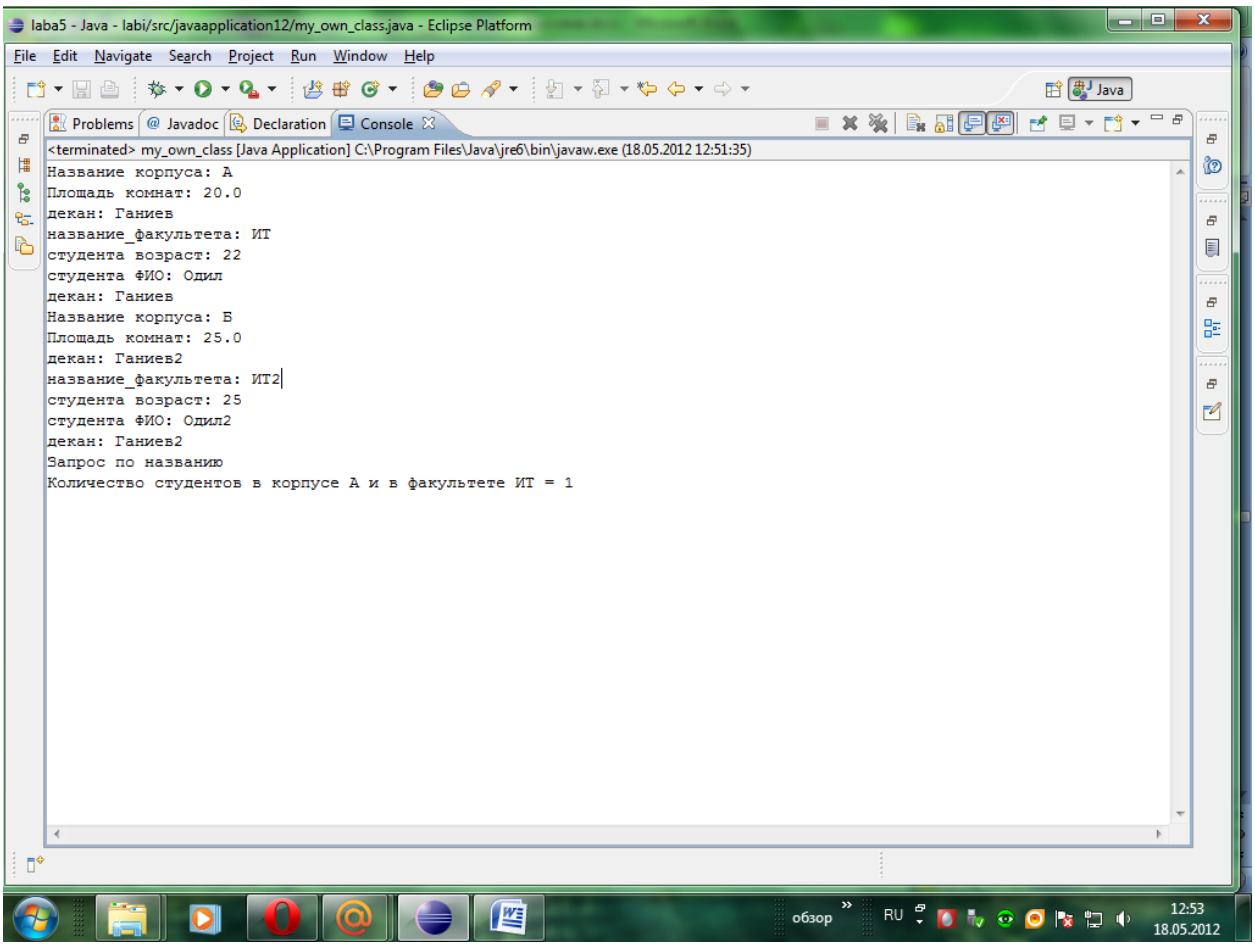
```
            fakultet mas_fak[] = new fakultet[kol];
```

и циклически забиваем массив, в конце цикла вызываем конструктор класса факультеты.

```
            for (int i=0; i<mas_fak.length; i++){
                  String декан = JOptionPane.showInputDialog("ВВЕДИТЕ имя
декана:");
                  String название_факультета =
JOptionPane.showInputDialog("ВВЕДИТЕ название факультета:");
                  String ФИО = JOptionPane.showInputDialog("ВВЕДИТЕ ФИО
студента:");
                  int возраст =
Integer.parseInt(JOptionPane.showInputDialog("ВВЕДИТЕ возраст студента:"));
                  String название = JOptionPane.showInputDialog("ВВЕДИТЕ
название корпуса :");
                  float площадь =
Float.parseFloat(JOptionPane.showInputDialog("ВВЕДИТЕ площадь комнаты:"));
                  mas_fak[i] = new
fakultet(декан,название_факультета,ФИО,возраст,название,площадь);
                  mas_fak[i].show();
            }
            zapros.find_name(mas_fak);
      }
}
```


laba5 - Java - labi/src/javaapplication12/my_own_class.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
labi
test
test2
abstractdemo.java
my_own_class.java
* @param args
*/
public static void main(String[] args) {
// TODO Auto-generated method stub
int kol = Integer.parseInt(JOptionPane.showInputD
fakultet mas_fak[] = new fakultet[kol];
for (int i=0; i<mas_fak.length; i++){
String декан = JOptionPane.showInputDialog("В
String название_факультета = JOptionPane.show
String ФИО = JOptionPane.showInputDialog("ВВЕД
int возраст = Integer.parseInt(JOptionPane.sh
Input
ВВЕДИТЕ количество факультетов:
2
OK
Cancel
Task List
Find:
All
Uncategorized
Outline
fakultet(String, S
show()
zapros
find_name(fakul
my_own_class
main(String[])
Problems
Javadoc
Declaration
Console
my_own_class [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (18.05.2012 12:41:29)
Writable
Smart Insert
89 : 2
обзор
RU
12:41
18.05.2012


laba5 - Java - labi/src/javaapplication12/my_own_class.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
labi
test
test2
abstractdemo.java
my_own_class.java
System.out.println("Количество студентов в корпусе
название_найти+" и в факультете " +
""+ название_факультета+ " = "+ко
);
Input
ВВЕДИТЕ имя декана:
Ганиев2
OK
Cancel
Task List
Find:
All
Uncategorized
Outline
javaapplication12
import declarations
comnata
площадь : float
comnata(float)
korpusa
Problems
Javadoc
Declaration
Console
my_own_class [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (18.05.2012 12:51:35)
Название корпуса: A
Площадь комнат: 20.0
декан: Ганиев
название_факультета: ИТ
студента возраст: 22
студента ФИО: Одил
декан: Ганиев
Writable
Smart Insert
86 : 92
обзор
RU
12:52
18.05.2012


laba5 - Java - labi/src/javaapplication12/my_own_class.java - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Java
Problems Javadoc Declaration Console
<terminated> my_own_class [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (18.05.2012 12:51:35)
Название корпуса: А
Площадь комнат: 20.0
декан: Ганиев
название_факультета: ИТ
студента возраст: 22
студента ФИО: Одил
декан: Ганиев
Название корпуса: Б
Площадь комнат: 25.0
декан: Ганиев2
название_факультета: ИТ2
студента возраст: 25
студента ФИО: Одил2
декан: Ганиев2
Запрос по названию
Количество студентов в корпусе А и в факультете ИТ = 1
обзор
RU
12:53
18.05.2012

Полный исходный код

```
package javaapplication12;
import javax.swing.*;

class comnata {
	public float площадь;
	comnata (float площадь){
		this.площадь=площадь;
	}
}
class korpusa{
	public String название;
	public comnata comnati;
	korpusa (String название, float площадь){
		this.название=название;
		comnati = new comnata(площадь);
	}
	public void print (){
		System.out.println("Название корпуса: "+название);
		System.out.println("Площадь комнат: "+comnati.площадь);
	}
}

class student{
	public String ФИО;
	public int возраст;
	student (String ФИО,int возраст){
		this.ФИО=ФИО;
		this.возраст=возраст;
	}
}
class fakultet extends korpusa{
	public String декан;
	public student студент;
	public String название_факультета;
	fakultet (String декан,String название_факультета,String ФИО,int
возраст,String название, float площадь){
		super(название, площадь);
		this.декан=декан;
		this.название_факультета=название_факультета;
		студент = new student (ФИО,возраст);
	}
	public void show(){
		print();
		System.out.println("декан: " + декан);
		System.out.println("название_факультета: " + название_факультета);
		System.out.println("студента возраст: " +студент.возраст);
		System.out.println("студента ФИО: "+студент.ФИО);
		System.out.println("декан: " + декан);
	}
}

class zapros{
	public static void find_name(fakultet mas_fak[]){
		System.out.println("Запрос по названию");
		String название_найти = JOptionPane.showInputDialog("ВВЕДИТЕ
название Корпуса:");
		String название_факультета = JOptionPane.showInputDialog("ВВЕДИТЕ
название факультета:");
		int количество=0;
```

```
            for (int i=0; i<mas_fak.length; i++){
                  if (mas_fak[i].название.equals(название_найти)){
                        if
(mas_fak[i].название_факультета.equals(название_факультета)){
                              количество++;
                        }
                  }
            }
            System.out.println("Количество студентов в корпусе " +
                        название_найти+" и в факультете " +
                              ""+ название_факультета+ " = "+количество
                        );
      }

}

public class my_own_class {
      /**
       * @param args
       */
      public static void main(String[] args) {
            // TODO Auto-generated method stub
            int kol = Integer.parseInt(JOptionPane.showInputDialog("ВВЕДИТЕ
количество факультетов:"));
            fakultet mas_fak[] = new fakultet[kol];
            for (int i=0; i<mas_fak.length; i++){
                  String декан = JOptionPane.showInputDialog("ВВЕДИТЕ имя
декана:");
                  String название_факультета =
JOptionPane.showInputDialog("ВВЕДИТЕ название факультета:");
                  String ФИО = JOptionPane.showInputDialog("ВВЕДИТЕ ФИО
студента:");
                  int возраст =
Integer.parseInt(JOptionPane.showInputDialog("ВВЕДИТЕ возраст студента:"));
                  String название = JOptionPane.showInputDialog("ВВЕДИТЕ
название корпуса :");
                  float площадь =
Float.parseFloat(JOptionPane.showInputDialog("ВВЕДИТЕ площадь комнаты:"));
                  mas_fak[i] = new
fakultet(декан,название_факультета,ФИО,возраст,название,площадь);
                  mas_fak[i].show();
            }
            zapros.find_name(mas_fak);
      }
}
```

Заключение

Данная курсовая работа по предмету: «Объектно-ориентированные языки программирования» на тему: «Отношения между классами (тема №2). Номер варианта 2.5». В данной курсовой работе были рассмотрены иерархия объектов. Реализация путем наследования методов для автоматизации поиска указанной информации, также реализован автоматический вывод всех данных относительно комнаты. Программа была реализована Java, в среде Eclipse.

Список используемой литературы

1. Библиотека профессионала Java 2 Том 1 Кей С. Хорстманн • Гари Корнелл
2. П.Ноутон, Г.Шилдт - Java2. Наиболее полное руководство