Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Институт информационных технологий

Факультет компьютерных технологий

Кафедра информационных систем и технологий

**Контрольная работа**

по дисциплине

«Конструирования программного обеспечения»

|  |  |
| --- | --- |
|  | Выполнил студент группы 181073  Наркевич Михаил Викторович |
|  |  |

Минск 2022

**Контрольная работа**

**по дисциплине «Конструирования программного обеспечения» (КПО)**

**Основная цель** выполнения контрольной работы состоит в практической реализации теоретических вопросов курса при изучении дисциплины «Конструирование программного обеспечения» (в первую очередь при самостоятельной работе студента над материалом дисциплины).

**Часть 1: «Проектирование классов»**

В каждом варианте предлагается предметная область. Необходимо спроектировать:

* Не менее 5 классов
* Не менее 3 полей для каждого класса.
* Не менее 3 операций (методов) для каждого класса.
* Имена классов – существительные.

Вариант 30: Туристическое оборудование

Решение:

namespace EducationHigherProfessionalEducation

{

class Program

{

static void Main(string[] args)

{

Group group = new Group();

group.setName("181000");

group.updateStudyYear(2);

group.updateAmountStudents(22);

Console.WriteLine("Is the name correct? - {0}", group.getName() == "181000");

Console.WriteLine("Is the year of study correct? - {0}", group.getStudyYear() == 2);

Console.WriteLine("Is the amount students correct? - {0}", group.getAmountStudents() == 22);

Department department = new Department();

department.setName("test");

department.setHeadDepartmentName("Inna");

department.setFacultyName("linguistics");

department.holdSeminar(group);

Audience audience = department.getNewAudience();

audience.setTeacherName("Ivan");

audience.setName("404");

audience.changeGroup(group);

audience.giveLecture();

Exam exam = new Exam();

exam.setTeacherName(audience.getTeacherName());

exam.startExam();

Console.WriteLine("Exam grade is 10? - {0}", exam.turnAnswers(new List<bool>() {true, true, false, true}) == 10);

exam.finishExam();

University university = new University();

university.setRectorName("Ivan");

university.setName("Factorian University");

university.deduct(group, 1);

Console.WriteLine("amount students = 21? - {0}", group.getAmountStudents() == 21);

university.enroll(group, 2);

Console.WriteLine("amount students = 23? - {0}", group.getAmountStudents() == 23);

}

}

class University

{

private string rectorName;

private string name;

private Audience audience;

public string getRectorName()

{

return rectorName;

}

public string getname()

{

return name;

}

public Audience getAudience()

{

return audience;

}

public void deduct(Group group, int amount)

{

if (rectorName == null || rectorName == "")

{

Console.WriteLine("error: rector needed");

return;

}

if (group == null)

{

Console.WriteLine("error: group is invalid");

return;

}

if (amount < 0)

{

Console.WriteLine("error: amount < 0");

return;

}

if (group.getAmountStudents() < amount)

{

Console.WriteLine("error: amount > amount students");

return;

}

group.updateAmountStudents(group.getAmountStudents() - amount);

}

public void enroll(Group group, int amount)

{

if (rectorName == null || rectorName == "")

{

Console.WriteLine("error: rector needed");

return;

}

if (group == null)

{

Console.WriteLine("error: group is invalid");

return;

}

if (amount < 0)

{

Console.WriteLine("error: amount < 0");

return;

}

group.updateAmountStudents(group.getAmountStudents() + amount);

}

public void setName(string name)

{

if (name == null || name == "")

{

Console.WriteLine("error: name is incorrect");

return;

}

this.name = name;

}

public void setRectorName(string rectorName)

{

if (rectorName == null || rectorName == "")

{

Console.WriteLine("error: rector name is incorrect");

return;

}

this.rectorName = rectorName;

}

}

class Department

{

private string name;

private string headDepartmentName;

private string facultyName;

public string getName()

{

return name;

}

public string getHeadDepartmentName()

{

return headDepartmentName;

}

public string getFacultyName()

{

return facultyName;

}

public void setName(string name)

{

if (name == null || name == "")

{

Console.WriteLine("error: name is incorrect");

return;

}

this.name = name;

}

public void setHeadDepartmentName(string headDepartmentName)

{

if (headDepartmentName == null || headDepartmentName == "")

{

Console.WriteLine("error: head department name is incorrect");

return;

}

this.headDepartmentName = headDepartmentName;

}

public void setFacultyName(string facultyName)

{

if (facultyName == null || facultyName == "")

{

Console.WriteLine("error: faculty name is incorrect");

return;

}

this.facultyName = facultyName;

}

public Audience getNewAudience()

{

if (headDepartmentName == null || name == "")

{

Console.WriteLine("error: head department needed");

}

return new Audience();

}

public void holdSeminar(Group group)

{

if (group == null || headDepartmentName == null || headDepartmentName == "")

{

Console.WriteLine("error: data is incorrect");

return;

}

Console.WriteLine("Seminar was hosted by department head ({0}) for group ({1})", headDepartmentName, group.getName());

}

}

class Exam

{

private List<bool> answers = new List<bool>() {true, true, false, true};

private bool isFinished = true;

private string teacherName;

public List<bool> getAnswers()

{

return answers;

}

public bool getIsFinished()

{

return isFinished;

}

public string getTeacherName()

{

return teacherName;

}

public void updateAnswers(List<bool> newAnswers)

{

if (newAnswers == null || newAnswers.Count == 0)

{

Console.WriteLine("error: new answers is incorrect");

}

answers = newAnswers;

}

public void setTeacherName(string name)

{

if (name == null || name == "")

{

Console.WriteLine("error: teacher name is incorrect");

return;

}

this.teacherName = name;

}

public void startExam()

{

if (teacherName == null)

{

Console.WriteLine("error: no teacher to start the exam");

}

isFinished = false;

}

public void finishExam()

{

isFinished = true;

}

public int turnAnswers(List<bool> answers)

{

int countAnswer = this.answers.Count();

if (answers.Count() != countAnswer)

{

Console.WriteLine("error: the number of responses does not match");

return -1;

}

int numberCorrectAnswers = 0;

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

{

if (this.answers[i] == answers[i])

{

numberCorrectAnswers++;

}

}

if (numberCorrectAnswers > 0 && numberCorrectAnswers != countAnswer)

{

return 8;

}

else if (numberCorrectAnswers == countAnswer)

{

return 10;

}

else

{

return 4;

}

}

}

class Audience

{

private string name;

private Group group;

private string teacherName;

public string getName()

{

return name;

}

public Group getGroup()

{

return group;

}

public string getTeacherName()

{

return teacherName;

}

public void setName(string name)

{

if (name == null || name == "")

{

Console.WriteLine("error: name is incorrect");

return;

}

this.name = name;

}

public void setTeacherName(string name)

{

if (name == null || name == "")

{

Console.WriteLine("error: teacher name is incorrect");

return;

}

this.teacherName = name;

}

public void changeGroup(Group group)

{

if (group.getAmountStudents() == 0 || group.getName() == "" || group.getName() == null)

{

Console.WriteLine("error: Group is incorrect");

return;

}

this.group = group;

}

public void giveLecture()

{

if(group == null

|| name == null || name == ""

|| teacherName == null || teacherName == "")

{

Console.WriteLine("error: data is incorrect");

return;

}

Console.WriteLine("The lecture was held by the teacher ({0}) for the group ({1})", teacherName, group.getName());

}

}

class Group

{

private string name;

private int amountStudents = 0;

private int studyYear = 0;

public string getName()

{

return name;

}

public int getAmountStudents()

{

return amountStudents;

}

public int getStudyYear()

{

return studyYear;

}

public void setName(string name)

{

if (name == null || name == "")

{

Console.WriteLine("error: name is incorrect");

return;

}

this.name = name;

}

public void updateAmountStudents(int amount)

{

if (amount < 1)

{

Console.WriteLine("error: amount < 0");

return;

}

amountStudents = amount;

}

public void updateStudyYear(int studyYear)

{

if (studyYear < 1 || studyYear > 4)

{

Console.WriteLine("error: study year < 0 or study year > 4");

return;

}

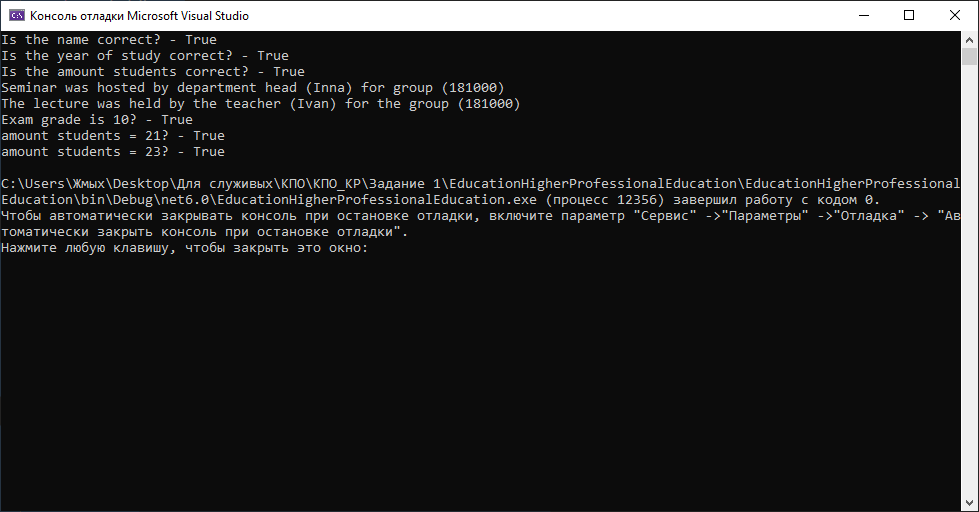
this.studyYear = studyYear;

}

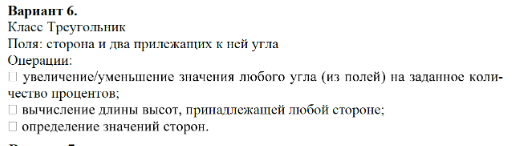
}

}

Скриншот:



**Часть 2: «Реализация классов» ЕЩЕ НЕ ГОТОВО**

Организовать проверку существования необходимой фигуры.

В каждом классе необходимо дополнительно реализовать:

* + закрытые поля;
  + к каждому полю свойства;
  + 3 конструктора;
  + метод ToString().

Решение:

Triangle ABC = new Triangle();

int upOrDown = 1;

Console.Write("Введите сторону треугольника \nВвод: ");

ABC.A = Convert.ToDouble(Console.ReadLine());

Console.Write("Введите два угола придежащих к стороне \nВвод: ");

ABC.ANGLB = Convert.ToDouble(Console.ReadLine());

Console.Write("Ввод: ");

ABC.ANGLC = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("\nУвеличение или уменьшение угола:\n1)Увеличение угла Бета\n2)Увеличение угла Гамма\n3)Уменьшение угла Бета\n4)Уменьшение угла Гамма\n0)Выход");

while (upOrDown != 0)

{

Console.Write("\nВвод: ");

upOrDown = Convert.ToInt32(Console.ReadLine());

if (upOrDown == 0)

{

continue;

}

Console.Write("Ведите колличество процентов: ");

int valueUpOrDown = Convert.ToInt32(Console.ReadLine());

switch (upOrDown)

{

case 1:

Console.WriteLine("Угол Бета = {0}", ABC.upAnglB(valueUpOrDown));

break;

case 2:

Console.WriteLine("Угол Гамма = {0}", ABC.upAnglC(valueUpOrDown));

break;

case 3:

Console.WriteLine("Угол Бета = {0}", ABC.downAnglB(valueUpOrDown));

break;

case 4:

Console.WriteLine("Угол Гамма= {0}", ABC.downAnglC(valueUpOrDown));

break;

default:

Console.WriteLine("default");

break;

}

}

double alpha = ABC.Alpha(ABC.ANGLB, ABC.ANGLC);

double sideB = ABC.sideB(ABC.A, ABC.ANGLB, alpha);

double sideC = ABC.sideC(ABC.A, ABC.ANGLC, alpha);

Console.WriteLine("\nЗначение высоты сторон:\nДлинна высоты А = {0}", Convert.ToInt32(ABC.height(ABC.A, sideB, sideC, ABC.A)));

Console.WriteLine("Длинна высоты B = {0}", Convert.ToInt32(ABC.height(ABC.A, sideB, sideC, sideB)));

Console.WriteLine("Длинна высоты С = {0}", Convert.ToInt32(ABC.height(ABC.A, sideB, sideC, sideC)));

Console.WriteLine("\nЗначение сторон:\nА = {0}", Convert.ToInt32(ABC.A));

Console.WriteLine("B = {0}", Convert.ToInt32(sideB));

Console.WriteLine("C = {0}", Convert.ToInt32(sideC));

class Triangle

{

private double a, anglB, anglC;

public double A

{

get { return a; }

set { a = value; }

}

public double ANGLB

{

get { return anglB; }

set { anglB = value; }

}

public double ANGLC

{

get { return anglC; }

set { anglC = value; }

}

public Triangle(Triangle Triangle)

{

a = Triangle.a;

anglB = Triangle.anglB;

anglC = Triangle.anglC;

}

public Triangle(double a, double anglB, double anglC)

{

this.a = a;

this.anglB = anglB;

this.anglC = anglC;

}

public Triangle()

{

a = 0;

anglB = 0;

anglC = 0;

}

public override string ToString()

{

return "\r\n" + base.ToString() + "\r\n" + "1 сторона = " + a + "\r\n" + "1 угол = " + anglB + "\r\n" + "2 угол = " + anglC;

}

public double upAnglB(double p)

{

return this.anglB = (anglB + (anglB \* (p / 100)));

}

public double upAnglC(double p)

{

return this.anglC = (anglC + (anglC \* (p / 100)));

}

public double downAnglB(double p)

{

return this.anglB = (anglC - (anglC \* (p / 100)));

}

public double downAnglC(double p)

{

return this.anglC = (anglC - (anglC \* (p / 100)));

}

public double valueC(double a, double b, double anglAB)

{

return Math.Sqrt((a \* a) + (b \* b) - ((2 \* a \* b) \* Math.Cos(Math.PI \* anglAB / 180.0)));

}

public double Alpha(double Beta, double Gamma)

{

double x = 180;

return (x - (Beta + Gamma));

}

public double sideB(double a, double beta, double alpha)

{

return (a \* Math.Sin(Math.PI \* beta / 180.0))/ Math.Sin(Math.PI \* alpha / 180.0);

}

public double sideC(double a, double gamma, double alpha)

{

return (a \* Math.Sin(Math.PI \* gamma / 180.0)) / Math.Sin(Math.PI \* alpha / 180.0);

}

//public double bisector(double b, double c, double angl)

//{

// return ((2 \* b \* c) / (b + c)) \* Math.Cos((Math.PI \* angl / 180.0) / 2);

//}

//public double lengthFirst(double a, double b, double c)

//{

// return (a \* c) / (a + b);

//}

public double height(double a, double b, double c, double lineFind)

{

double p = ((a + b + c) / 2);

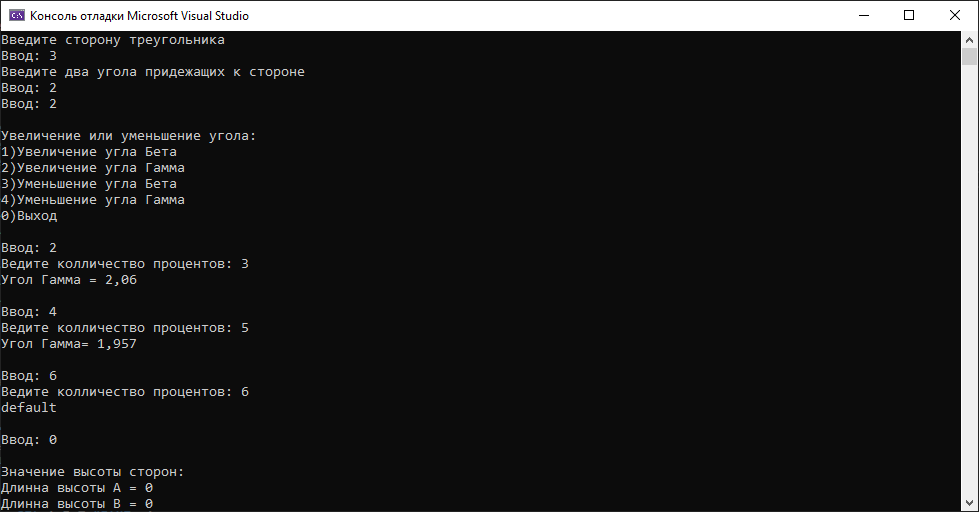
double h = (2 \* Math.Sqrt(p \* (p - a) \* (p - b) \* (p - c)))/ lineFind;

return h;

}

}

Скриншот:



Вывод: проектирование классов с соблюдением принципов ООП является основной задачей программиста для создания понятного, удобного в расширении и гибкого кода.