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»

по теме «Бинарные деревья»

Вариант 19

|  |  |
| --- | --- |
|  | Выполнил студент группы 181073  Корсак Алексей Андреевич |
|  | Проверил ассистент кафедры ИСиТ Потоцкий Дмитрий Сергеевич |

Минск 2022

**Контрольная работа**

**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).

**Порядок выполнения работы**

1.     Изучить теоретические сведения.

2.     Ознакомиться с материалами литературных источников.

3       Разработать алгоритм программы.

4.     Написать, отладить и выполнить программу.

Контрольная работа состоит из двух частей. Обе части обязательны для выполнения. Номер варианта соответствует Вашему номеру по списку.

**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.

**Код программы:**

**namespace Metallurgy**

**{**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**Factory factory = new Factory();**

**factory.getStorage().loadWood(101);**

**factory.getStorage().loadMaterial(factory.getStorage().getCopperOre(),200);**

**factory.getStorage().loadMaterial(factory.getStorage().getTinOre(), 101);**

**factory.getCharcoalKilns().loadWood(factory.getStorage().cutWood(101));**

**factory.getCharcoalKilns().makeCoal();**

**factory.getIndustrialOven().loadCoal(factory.getStorage().cutCoal(303));**

**factory.getIndustrialOven().smeltCopperOre(200);**

**factory.getIndustrialOven().smeltTinOre(101);**

**factory.switchActive();**

**factory.makeBronzeIngot(100);**

**factory.makeDetail(100, MaterialType.BRONZE);**

**factory.makeDetail(1, MaterialType.TIN);**

**Console.WriteLine("Bronze Detail - {0}", factory.getStorage().getBronzeDetail().getAmount());**

**Console.WriteLine("Tin Detail - {0}", factory.getStorage().getTinDetail().getAmount());**

**}**

**}**

**class Factory**

**{**

**private CharcoalKilns charcoalKilns;**

**private IndustrialOven industrialOven;**

**private Storage storage;**

**private bool isActive;**

**public Factory()**

**{**

**this.storage = new Storage();**

**charcoalKilns = new CharcoalKilns(storage);**

**industrialOven = new IndustrialOven(storage);**

**}**

**public CharcoalKilns getCharcoalKilns()**

**{**

**return charcoalKilns;**

**}**

**public IndustrialOven getIndustrialOven()**

**{**

**return industrialOven;**

**}**

**public Storage getStorage()**

**{**

**return storage;**

**}**

**// 2 меди и 1 олово = 1 бронза**

**public void makeBronzeIngot(int amount)**

**{**

**if (amount < 0)**

**{**

**Console.WriteLine("Error: amount < 0");**

**return;**

**}**

**if (!isActive)**

**{**

**Console.WriteLine("need to turn on the plant");**

**return;**

**}**

**if (storage.getCopperIngot().getAmount() < amount \* 2)**

**{**

**Console.WriteLine("not enough copper");**

**return;**

**}**

**if (storage.getTinIngot().getAmount() < amount)**

**{**

**Console.WriteLine("not enough tin");**

**return;**

**}**

**storage.cutMaterial(storage.getCopperIngot(), amount \* 2);**

**storage.cutMaterial(storage.getTinIngot(), amount);**

**storage.loadMaterial(storage.getBronzeIngot(), amount);**

**}**

**public void makeDetail(int amount, MaterialType materialType)**

**{**

**Material ingot = getIgnotByType(materialType);**

**Material detail = getDetailByType(materialType);**

**if (amount < 1)**

**{**

**Console.WriteLine("Error: amount < 1");**

**return;**

**}**

**if (!isActive)**

**{**

**Console.WriteLine("need to turn on the plant");**

**return;**

**}**

**if (ingot.getAmount() < amount)**

**{**

**Console.WriteLine("not enough resources");**

**return;**

**}**

**storage.cutMaterial(ingot, amount);**

**storage.loadMaterial(detail, amount);**

**}**

**public void switchActive()**

**{**

**if (isActive)**

**{**

**isActive = false;**

**}**

**else**

**{**

**isActive = true;**

**}**

**}**

**private Material getIgnotByType(MaterialType materialType)**

**{**

**if (materialType == MaterialType.COPPER)**

**{**

**return storage.getCopperIngot();**

**}**

**if (materialType == MaterialType.TIN)**

**{**

**return storage.getTinIngot();**

**}**

**else**

**{**

**return storage.getBronzeIngot();**

**}**

**}**

**private Material getDetailByType(MaterialType materialType)**

**{**

**if (materialType == MaterialType.COPPER)**

**{**

**return storage.getCopperDetail();**

**}**

**if (materialType == MaterialType.TIN)**

**{**

**return storage.getTinDetail();**

**}**

**else**

**{**

**return storage.getBronzeDetail();**

**}**

**}**

**}**

**class CharcoalKilns**

**{**

**private int amountWood = 0;**

**// количество угля за единицу дерева**

**private int efficiency = 3;**

**private Storage storage;**

**public CharcoalKilns(Storage storage)**

**{**

**this.storage = storage;**

**}**

**public void makeCoal()**

**{**

**if (amountWood < 1)**

**{**

**Console.WriteLine("Error: amount < 1");**

**return;**

**}**

**storage.loadCoal(amountWood \* efficiency);**

**amountWood = 0;**

**}**

**public void loadWood(int amount)**

**{**

**if (amount < 0)**

**{**

**Console.WriteLine("Error: amount < 0");**

**return;**

**}**

**amountWood += amount;**

**}**

**public void changeEfficiency(int newValue)**

**{**

**if (newValue < 0)**

**{**

**Console.WriteLine("Error: newValue < 0");**

**return;**

**}**

**efficiency = newValue;**

**}**

**}**

**class IndustrialOven**

**{**

**private int amountCoal = 0;**

**// количество угля необходимое для переплавки руды**

**private int efficiency = 1;**

**private Storage storage;**

**public IndustrialOven(Storage storage)**

**{**

**this.storage = storage;**

**}**

**public void smeltCopperOre(int amount)**

**{**

**if (amount < 0)**

**{**

**Console.WriteLine("Error: amount < 0");**

**}**

**if (amountCoal < amount \* efficiency)**

**{**

**Console.WriteLine("Error: coal shortage");**

**}**

**if (storage.getCopperOre().getAmount() < amount)**

**{**

**Console.WriteLine("Error: there is not enough ore in the warehouse");**

**}**

**amountCoal -= amount \* efficiency;**

**storage.cutMaterial(storage.getCopperOre(), amount);**

**storage.loadMaterial(storage.getCopperIngot(), amount);**

**Console.WriteLine("");**

**}**

**public void smeltTinOre(int amount)**

**{**

**if (amount < 0)**

**{**

**Console.WriteLine("Error: amount < 0");**

**return;**

**}**

**if (amountCoal < amount \* efficiency)**

**{**

**Console.WriteLine("Error: coal shortage");**

**return;**

**}**

**if (storage.getTinOre().getAmount() < amount)**

**{**

**Console.WriteLine("Error: there is not enough ore in the warehouse");**

**return;**

**}**

**amountCoal -= amount \* efficiency;**

**storage.cutMaterial(storage.getTinOre(), amount);**

**storage.loadMaterial(storage.getTinIngot(), amount);**

**Console.WriteLine("");**

**}**

**public void loadCoal(int amount)**

**{**

**if (amount < 0)**

**{**

**Console.WriteLine("Error: amount < 0");**

**return;**

**}**

**amountCoal += amount;**

**}**

**public void changeEfficiency(int newValue)**

**{**

**if (newValue < 0)**

**{**

**Console.WriteLine("Error: newValue < 0");**

**return;**

**}**

**efficiency = newValue;**

**}**

**}**

**class Storage**

**{**

**private Material copperOre = new Material(MaterialType.COPPER, State.ORE);**

**private Material tinOre = new Material(MaterialType.TIN, State.ORE);**

**private Material copperIngot = new Material(MaterialType.COPPER, State.INGOT);**

**private Material tinIngot = new Material(MaterialType.TIN, State.INGOT);**

**private Material bronzeIngot = new Material(MaterialType.BRONZE, State.INGOT);**

**private Material copperDetail = new Material(MaterialType.COPPER, State.DETAIL);**

**private Material tinDetail = new Material(MaterialType.TIN, State.DETAIL);**

**private Material bronzeDetail = new Material(MaterialType.BRONZE, State.DETAIL);**

**private int amountCoal = 0;**

**private int amountWood = 0;**

**public Storage() {}**

**public Material getCopperOre()**

**{**

**return copperOre;**

**}**

**public Material getTinOre()**

**{**

**return tinOre;**

**}**

**public Material getCopperIngot()**

**{**

**return copperIngot;**

**}**

**public Material getTinIngot()**

**{**

**return tinIngot;**

**}**

**public Material getBronzeIngot()**

**{**

**return bronzeIngot;**

**}**

**public Material getCopperDetail()**

**{**

**return bronzeIngot;**

**}**

**public Material getTinDetail()**

**{**

**return tinDetail;**

**}**

**public Material getBronzeDetail()**

**{**

**return bronzeDetail;**

**}**

**public int getAmountCoal()**

**{**

**return amountCoal;**

**}**

**public int getAmountWood()**

**{**

**return amountWood;**

**}**

**public void loadWood(int amount)**

**{**

**if (amount < 0)**

**{**

**return;**

**}**

**amountWood += amount;**

**}**

**public int cutWood(int requiredAmount)**

**{**

**if (requiredAmount < 0 || amountWood < requiredAmount)**

**{**

**return 0;**

**}**

**amountWood -= requiredAmount;**

**return requiredAmount;**

**}**

**public void loadCoal(int amount)**

**{**

**if (amount < 0)**

**{**

**return;**

**}**

**amountCoal += amount;**

**}**

**public int cutCoal(int requiredAmount)**

**{**

**if (requiredAmount < 0 || amountCoal < requiredAmount)**

**{**

**return 0;**

**}**

**amountCoal -= requiredAmount;**

**return requiredAmount;**

**}**

**public void loadMaterial(Material material, int amount)**

**{**

**if (material == null || amount < 0)**

**{**

**return;**

**}**

**if (material.getType == copperOre.getType && material.getState == copperOre.getState)**

**{**

**copperOre.setAmount(copperOre.getAmount() + amount);**

**}**

**else if (material.getType == tinOre.getType && material.getState == tinOre.getState)**

**{**

**tinOre.setAmount(tinOre.getAmount() + amount);**

**}**

**else if (material.getType == copperIngot.getType && material.getState == copperIngot.getState)**

**{**

**copperIngot.setAmount(copperIngot.getAmount() + amount);**

**}**

**else if (material.getType == tinIngot.getType && material.getState == tinIngot.getState)**

**{**

**tinIngot.setAmount(tinIngot.getAmount() + amount);**

**}**

**else if (material.getType == bronzeIngot.getType && material.getState == bronzeIngot.getState)**

**{**

**bronzeIngot.setAmount(bronzeIngot.getAmount() + amount);**

**}**

**else if (material.getType == copperDetail.getType && material.getState == copperDetail.getState)**

**{**

**copperDetail.setAmount(copperDetail.getAmount() + amount);**

**}**

**else if (material.getType == tinDetail.getType && material.getState == tinDetail.getState)**

**{**

**tinDetail.setAmount(tinDetail.getAmount() + amount);**

**}**

**else if (material.getType == bronzeDetail.getType && material.getState == bronzeDetail.getState)**

**{**

**bronzeDetail.setAmount(bronzeDetail.getAmount() + amount);**

**}**

**}**

**public int cutMaterial(Material material, int requiredAmount)**

**{**

**if (material == null || requiredAmount < 0)**

**{**

**return 0;**

**}**

**if (material.getType == copperOre.getType && material.getState == copperOre.getState)**

**{**

**if (copperOre.getAmount() < requiredAmount)**

**{**

**return 0;**

**}**

**copperOre.setAmount(copperOre.getAmount() - requiredAmount);**

**return requiredAmount;**

**}**

**else if (material.getType == tinOre.getType && material.getState == tinOre.getState)**

**{**

**if (tinOre.getAmount() < requiredAmount)**

**{**

**return 0;**

**}**

**tinOre.setAmount(tinOre.getAmount() - requiredAmount);**

**return requiredAmount;**

**}**

**else if (material.getType == copperIngot.getType && material.getState == copperIngot.getState)**

**{**

**if (copperIngot.getAmount() < requiredAmount)**

**{**

**return 0;**

**}**

**copperIngot.setAmount(copperIngot.getAmount() - requiredAmount);**

**return requiredAmount;**

**}**

**else if (material.getType == tinIngot.getType && material.getState == tinIngot.getState)**

**{**

**if (tinIngot.getAmount() < requiredAmount)**

**{**

**return 0;**

**}**

**tinIngot.setAmount(tinIngot.getAmount() - requiredAmount);**

**return requiredAmount;**

**}**

**else if (material.getType == bronzeIngot.getType && material.getState == bronzeIngot.getState)**

**{**

**if (bronzeIngot.getAmount() < requiredAmount)**

**{**

**return 0;**

**}**

**bronzeIngot.setAmount(bronzeIngot.getAmount() - requiredAmount);**

**return requiredAmount;**

**}**

**else if (material.getType == copperDetail.getType && material.getState == copperDetail.getState)**

**{**

**if (copperDetail.getAmount() < requiredAmount)**

**{**

**return 0;**

**}**

**copperDetail.setAmount(copperDetail.getAmount() - requiredAmount);**

**return requiredAmount;**

**}**

**else if (material.getType == tinDetail.getType && material.getState == tinDetail.getState)**

**{**

**if (tinDetail.getAmount() < requiredAmount)**

**{**

**return 0;**

**}**

**tinDetail.setAmount(tinDetail.getAmount() - requiredAmount);**

**return requiredAmount;**

**}**

**else if (material.getType == bronzeDetail.getType && material.getState == bronzeDetail.getState)**

**{**

**if (bronzeDetail.getAmount() < requiredAmount)**

**{**

**return 0;**

**}**

**bronzeDetail.setAmount(bronzeDetail.getAmount() - requiredAmount);**

**return requiredAmount;**

**}**

**return 0;**

**}**

**}**

**class Material**

**{**

**private MaterialType type;**

**public State state;**

**private int amount = 0;**

**public Material(MaterialType type, State state)**

**{**

**this.type = type;**

**this.state = state;**

**}**

**public int getAmount()**

**{**

**return amount;**

**}**

**public void setAmount(int amount)**

**{**

**this.amount = amount;**

**}**

**public MaterialType getType()**

**{**

**return type;**

**}**

**public State getState()**

**{**

**return state;**

**}**

**}**

**enum MaterialType**

**{**

**COPPER,**

**TIN,**

**BRONZE**

**}**

**enum State**

**{**

**ORE,**

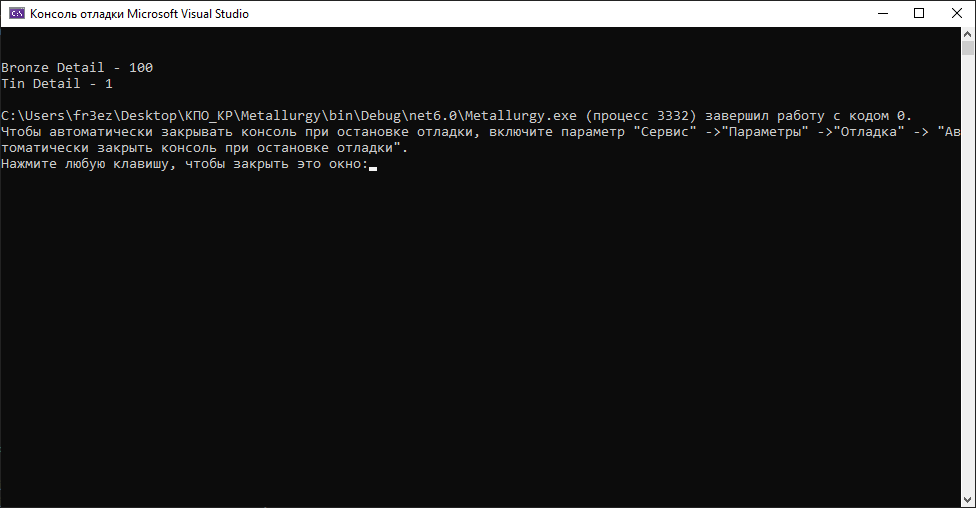
**INGOT,**

**DETAIL**

**}**

**}**

**Результат выполнения программы:**



**Часть 2: «Реализация классов»**

Организовать проверку существования необходимой фигуры. Начиная с 21 студента в списке необходимо определить свой вариант следующим образом:

ваш вариант соответствует **сумме цифр N, где N – ваш номер по списку**.

В каждом классе необходимо дополнительно реализовать:

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

**Код программы:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace КруговоеКольцо

{

class Sector

{

public override string ToString() // Переопределение стандартного метода ToString()

{

return "\r\n" + base.ToString() + "\r\n" + "Центральный угол сектора равен: " + \_centugol + "\r\n" +

"Радиус сектора равен: " + \_radius + "\r\n" +

"Площадь сектора:" + (3.14 \* Math.Pow(\_radius, 2) \* (\_centugol / 360)) + "\r\n" +

"Длинна дуги сектора: " + (3.14 \* \_radius \* \_centugol) / 180 + " (рад)" + "\r\n" +

"Длинна окружности: " + (2 \* 3.14 \* \_radius ) + "\r\n" +

"Диаметр окружности: " + (\_radius \* 2) + "\r\n";

}

public Sector() //Конструктор класса (При создании без параметров)

{

\_radius = 6;

\_centugol = 5;

}

public Sector(double radius, double centugol) //Конструктор класса (При создании с параметрами)

{

\_radius = radius;

\_centugol = centugol;

}

public Sector(Sector figure) //Конструктор класса для копирования класса

{

\_radius = figure.\_radius;

\_centugol = figure.\_centugol;

}

private double \_radius; //Поле

private double \_centugol; //Поле

public double Radius //Свойства Поля

{

get

{

return \_radius;

}

set

{

if (value > 0)

{

\_radius = value;

}

else

{

\_radius = 0;

}

return;

}

}

public double Centugol //Свойства Поля

{

get

{

return \_centugol;

}

set

{

if (value > 0)

{

\_centugol = value;

}

else

{

\_centugol = 0;

}

return;

}

}

}

class Program

{

static void Main(string[] args)

{

double vl;

Console.Write("Введите значение центрального угла сектора: ");

double centugol = Convert.ToInt32(Console.ReadLine());

Console.WriteLine();

Console.Write("Введите значение радиуса сектора: ");

double radius = Convert.ToInt32(Console.ReadLine());

if (centugol > 0 || radius > 0)

{

if (radius > 0 && radius > 0)

{

Sector sector = new Sector(radius,centugol);// Создания класса с введенными параметрами

Sector sector1 = new Sector(sector);

Setvalue(sector, sector1);

}

if (radius > 0)

{

Sector sector = new Sector(radius, radius - 1);// Создания класса с введенными параметрами

Sector sector1 = new Sector(sector);

Setvalue(sector, sector1);

}

else

{

Sector sector = new Sector(centugol + 1, centugol);// Создания класса с введенными параметрами

Sector sector1 = new Sector(sector);

Setvalue(sector, sector1);

}

}

else

{

Sector figure = new Sector();// Создание стандартного класса

Sector figure1 = new Sector(figure);

Setvalue(figure, figure1);

}

void Setvalue(Sector sector, Sector sector1)

{

Console.WriteLine(sector1.ToString());

Console.Write("Выберите поле для увеличения (1 - центральный угол сектора, 2 - радиус сектора: ");

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

if (pole > 0 && pole < 3)

{

Console.Write("1 - Уменьшить, 2 - Увеличить: ");

int pole1 = Convert.ToInt32(Console.ReadLine());

if (pole1 > 0 && pole1 < 3)

{

Console.Write("Введите значение: ");

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

if (pole == 1 && pole1 == 1)

{

sector1.Centugol = sector1.Centugol / vl;

}

else if (pole == 1 && pole1 == 2)

{

sector1.Centugol = sector1.Centugol \* vl;

}

else if (pole == 2 && pole1 == 1)

{

sector1.Radius = sector1.Radius / vl;

}

else if (pole == 2 && pole1 == 2)

{

sector1.Radius = sector1.Radius \* vl;

}

}

}

Console.WriteLine(sector1.ToString());

Console.ReadLine();

}

}

}

}

**Результат выполнения программы:**

