**Занятие № 10**

**Дата выполнения работы:** 29.04.2023

# **Тема работы:** «Полиморфизм»

**Ход работы**

**Задание 1**

Создать абстрактный класс Body. На его основе реализовать классы

Parallelepiped (прямоугольный параллелепипед), Cone (конус) и Ball (шар) с

возможностью вычисления площади поверхности и объема.

Создать класс Series (набор), содержащий массив/параметризованную

коллекцию объектов этих классов в динамической памяти. Предусмотреть

возможность вывода характеристик объектов списка. Написать

демонстрационную программу, в которой будут использоваться все методы

классов.

Дополнительное задание: дополнить класс методами сортировки по

некоторому критерию, вывода в файл и считывания из файла.

**Листинг программы:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace App

{

public partial class Form1 : Form

{

private IEnumerable<Body> bodiesList;

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

Series series = new Series();

series.AddBody(new Parallelepiped(2, 3, 4));

series.AddBody(new Cone(1, 2));

series.AddBody(new Ball(3));

series.PrintBodies(); // Выводит информацию о всех объектах

series.SortByVolume();

series.PrintBodies(); // Выводит информацию о всех объектах, отсортированных по объему

series.WriteToFile("D:\\Учёба\\Практика КПиЯП\\день 10\\bodies.txt");

series.ReadFromFile("D:\\Учёба\\Практика КПиЯП\\день 10\\bodies.txt");

series.PrintBodies(); // Выводит информацию о всех объектах, прочитанных из файла

}

}

// Абстрактный класс Body

public abstract class Body

{

public abstract double GetSurfaceArea();

public abstract double GetVolume();

}

// Класс прямоугольного параллелепипеда

public class Parallelepiped : Body

{

public double Length { get; }

public double Width { get; }

public double Height { get; }

public Parallelepiped(double length, double width, double height)

{

Length = length;

Width = width;

Height = height;

}

public override double GetSurfaceArea()

{

return 2 \* (Length \* Width + Length \* Height + Width \* Height);

}

public override double GetVolume()

{

return Length \* Width \* Height;

}

}

// Класс конуса

public class Cone : Body

{

public double Radius { get; }

public double Height { get; }

public Cone(double radius, double height)

{

Radius = radius;

Height = height;

}

public override double GetSurfaceArea()

{

double slantHeight = Math.Sqrt(Radius \* Radius + Height \* Height);

return Math.PI \* Radius \* slantHeight + Math.PI \* Radius \* Radius;

}

public override double GetVolume()

{

return Math.PI \* Radius \* Radius \* Height / 3;

}

}

public class Ball : Body

{

public double Radius { get; }

public Ball(double radius)

{

Radius = radius;

}

public override double GetSurfaceArea()

{

return 4 \* Math.PI \* Radius \* Radius;

}

public override double GetVolume()

{

return 4 \* Math.PI \* Radius \* Radius \* Radius / 3;

}

}

public class Series

{

private List<Body> \_bodies;

public Series()

{

\_bodies = new List<Body>();

}

public void AddBody(Body body)

{

\_bodies.Add(body);

}

public void RemoveBody(Body body)

{

\_bodies.Remove(body);

}

public void PrintBodies()

{

foreach (Body body in \_bodies)

{

Console.WriteLine("Type: {0}, Surface Area: {1}, Volume: {2}",

body.GetType().Name, body.GetSurfaceArea(), body.GetVolume());

}

}

// Дополнительное задание: сортировка по возрастанию объема

public void SortByVolume()

{

\_bodies.Sort((b1, b2) => b1.GetVolume().CompareTo(b2.GetVolume()));

}

// Дополнительное задание: вывод в файл

public void WriteToFile(string path)

{

using (StreamWriter writer = new StreamWriter(path))

{

foreach (Body body in \_bodies)

{

writer.WriteLine("{0},{1},{2}", body.GetType().Name,

body.GetSurfaceArea(), body.GetVolume());

}

}

}

// Дополнительное задание: чтение из файла

public void ReadFromFile(string path)

{

\_bodies.Clear();

using (StreamReader reader = new StreamReader(path))

{

while (!reader.EndOfStream)

{

string line = reader.ReadLine();

string[] parts = line.Split(',');

string typeName = parts[0];

double surfaceArea = double.Parse(parts[1]);

double volume = double.Parse(parts[2]);

switch (typeName)

{

case "Parallelepiped":

\_bodies.Add(new Parallelepiped(surfaceArea / 2, surfaceArea / (2 \* (parts.Length - 1)), volume / (surfaceArea / 2)));

break;

case "Cone":

\_bodies.Add(new Cone(Math.Sqrt(surfaceArea / Math.PI), volume / (Math.PI \* surfaceArea / 3)));

break;

case "Ball":

\_bodies.Add(new Ball(Math.Pow(surfaceArea / (4 \* Math.PI), 1.0 / 3)));

break;

}

}

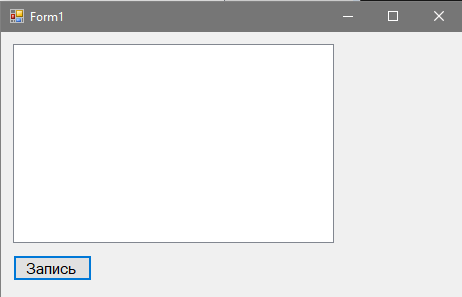
}

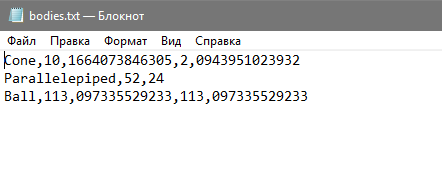
}

}

}

**Результат:**

****

****

**Вывод:** В процессе изучения этой темы я понял, что полиморфизм - это способность объектов разных классов обладать одинаковым интерфейсом, что позволяет использовать эти объекты взаимозаменяемо. Это позволяет создавать более гибкие и масштабируемые программы, что является одним из важных принципов объектно-ориентированного программирования. Применение полиморфизма упрощает код и позволяет легко добавлять новые классы, которые поддерживают интерфейс. Также использование полиморфизма позволяет создавать коллекции объектов разных классов и работать с ними, как с единым списком. В целом, полиморфизм является мощным инструментом, который может значительно улучшить качество программного кода и упростить его разработку.