Министерство науки и ВЫСШЕГО образования   
Российской Федерации

Федеральное автономное образовательное учреждение высшего образования  
«Санкт-Петербургский государственный политехнический   
университет Петра Великого»  
**Институт среднего профессионального образования**

**Лабораторная работа № 6  
Тема:** Разработка и отладка программ демонстрации полиморфизма

МДК.01.01 Разработка программных модулей

**Вариант 9**

Выполнил: студент группы 32919/8  
Крупина Е.А.

Проверил: преподаватель   
Прокофьев А.А.

Санкт-Петербург  
2024

**Задание:**

*Разработка должна вестись по следующему плану:*

1. Проверьте правильность описания базового и дочерних классов. В каждом классе **должен** быть описан **свой конструктор**. В дочерних классах **не могут** повторяться те же поля, что и в родительском классе, т.к. они наследуются.
2. Некоторые виртуальные методы могут быть переопределены в дочерних классах. Переопределённые методы пометьте словом **override**.
3. В случае необходимости опишите одно или несколько свойств для получения и (или) установки значений полей. Свойство описываются внутри того класса, где задано объявлено скрытое поле, с которым оно связано.
4. Постройте диаграмму классов в графическом редакторе MS Visio(или ему подобных). Выбираем раздел фигур **Дополнительные фигуры→ Программы и базы данных→ Программное обеспечение → Класс UML.**
5. Начните разработку главной функции. Объявите массив или список типом родительского класса.
6. **Организуйте заполнение массива** объектами с учётом, что реально он может заполняться элементами как родительского, так и дочерних классов (или только дочерних). Создание объектов конкретного класса осуществляется с помощью вызова конструктора (командой new).
7. Организуйте вывод на экран полученного массива или списка при помощи отдельного цикла, в котором будет вызываться метод Show().
8. Создайте функции для изменения списка, удаление элементов из списка, добавление элементов в список, изменения объекта списка **пользователем**.
9. Организуйте выбор вызова функций с помощью **меню.**

**Диаграмма классов:**



Рис.1 – Диаграмма классов

**Код программы:**

1. Program.cs

using Lab6;

using System;

using System.Collections.Generic;

using System.Drawing;

namespace Lab6

{

class Program

{

static List<Point> figures = new List<Point>();

static double Input(string message)

{

Console.Write(message);

string input = Console.ReadLine();

if (!double.TryParse(input, out double res))

{

Console.WriteLine("Неккоректный ввод, введите заново");

return Input(message);

}

else { return res; }

}

static double InputRadius(string message)

{

Console.Write(message);

string input = Console.ReadLine();

if (!double.TryParse(input, out double res) || res <= 0)

{

Console.WriteLine("Неккоректный ввод, введите заново");

return Input(message);

}

else { return res; }

}

static void AddTOList()

{

int m = 1;

do

{

bool ism = false;

do

{

try

{

Console.WriteLine(" Введите номер класса обьекты , который хотите добавить:");

Console.WriteLine("1 - Точка");

Console.WriteLine("2 - Окружность");

Console.WriteLine("3 - Эллипс");

Console.WriteLine("4 - Вывод списка");

Console.WriteLine("5 - Выход");

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

if (m < 1 || m > 5)

{

throw new FormatException();

}

ism = true;

}

catch (FormatException)

{

Console.WriteLine();

Console.WriteLine("Неверный ввод. Введите заново.");

Console.WriteLine();

}

} while (!ism);

switch (m)

{

case 1:

Point point = new Point(Input("Введите координату X: "), Input("Введите координату Y:"));

figures.Add(point);

break;

case 2:

Circle circle = new Circle(Input("Введите координату X: "), Input("Введите координату Y:"), InputRadius("Введите радиус окружности: "));

figures.Add(circle);

break;

case 3:

Ellipse ellipse = new Ellipse(Input("Введите координату X: "), Input("Введите координату Y:"), InputRadius("Введите первый радиус: "), InputRadius("Введите второй радиус"));

figures.Add(ellipse);

break;

case 4:

for (int i = 0; i < figures.Count; i++)

{

figures[i].Print();

Console.WriteLine();

}

break;

case 5:

break;

}

} while (m < 5);

}

static void Delete()

{

if (figures.Count == 0)

{

Console.WriteLine("На данный момент в списке нет ни одного объекта.");

return;

}

Console.WriteLine("Введите порядковый номер элемента на удаление: ");

Console.Write("Доступные номера на удаление: ");

for (int i = 0; i < figures.Count; i++)

{

Console.Write($"{i + 1} ");

}

Console.WriteLine();

int number;

bool validInput = false;

while (!validInput)

{

string input = Console.ReadLine();

if (!int.TryParse(input, out number) || number <= 0 || number > figures.Count)

{

Console.WriteLine("Некорректный ввод, введите заново");

}

else

{

validInput = true;

figures.RemoveAt(number - 1);

Console.WriteLine($"Элемент с номером {number} был успешно удален.");

}

}

}

static void Edit()

{

if (figures.Count == 0)

{

Console.WriteLine("На данный момент в списке нет ни одного обьекта.");

return;

}

Console.WriteLine("Введите порядковый номер элемента для корректировки: ");

Console.Write("Доступные номера:");

for (int i = 0; i < figures.Count; i++) Console.Write($"{i + 1} ");

Console.WriteLine();

string input = Console.ReadLine();

bool edit = false;

int number;

do

{

if (!int.TryParse(input, out number) || number <= 0 || number > figures.Count)

{

Console.WriteLine("Некорректный ввод, введите заново");

input = Console.ReadLine(); // Запрашиваем ввод снова

}

else

{

edit = true;

number--; // Приводим к индексу

}

} while (!edit);

figures[number].Print();

string inputChoice;

if (figures[number] is Point)

{

int choice = 0;

bool editChoice = false;

while (!editChoice)

{

Console.WriteLine(" Что вы хотите изменить?");

Console.WriteLine("1 - Координата X");

Console.WriteLine("2 - Координата Y");

Console.WriteLine("3 - Все поля");

Console.WriteLine("4 - Выход");

if (int.TryParse(Console.ReadLine(), out choice) && choice >= 1 && choice <= 6)

{

editChoice = true;

}

else

{

Console.WriteLine("Некорректный ввод, введите заново");

}

}

switch (choice)

{

case 1:

figures[number].X = Input("Введите новое значение координаты X: ");

break;

case 2:

figures[number].Y = Input("Введите новое значение координаты Y: ");

break;

case 3:

figures[number].X = Input("Введите новое значение координаты X: ");

figures[number].X = Input("Введите новое значение координаты Y: ");

break;

case 4:

break;

}

}

else if (figures[number] is Circle circle)

{

int choice = 0;

bool editChoice = false;

while (!editChoice)

{

Console.WriteLine(" Что вы хотите изменить?");

Console.WriteLine("1 - Координата X");

Console.WriteLine("2 - Координата Y");

Console.WriteLine("3 - Первый радиус");

Console.WriteLine("4 - Все поля");

Console.WriteLine("5 - Выход");

if (int.TryParse(Console.ReadLine(), out choice) && choice >= 1 && choice <= 6)

{

editChoice = true;

}

else

{

Console.WriteLine("Некорректный ввод, введите заново");

}

}

switch (choice)

{

case 1:

circle.X = Input("Введите новое значение координаты X: ");

break;

case 2:

circle.Y = Input("Введите новое значение координаты Y: ");

break;

case 3:

circle.Radius = InputRadius("Введите новое значение Радиуса: ");

break;

case 4:

circle.X = Input("Введите новое значение координаты X: ");

circle.Y = Input("Введите новое значение координаты Y: ");

circle.Radius = InputRadius("Введите новое значение Радиуса: ");

break;

case 5:

break;

}

}

else if (figures[number] is Ellipse ellipse)

{

int choice = 0;

bool editChoice = false;

while (!editChoice)

{

Console.WriteLine("Что вы хотите изменить?");

Console.WriteLine("1 - Координата X");

Console.WriteLine("2 - Координата Y");

Console.WriteLine("3 - Первый радиус");

Console.WriteLine("4 - Второй радиус");

Console.WriteLine("5 - Все поля");

Console.WriteLine("6 - Выход");

if (int.TryParse(Console.ReadLine(), out choice) && choice >= 1 && choice <= 6)

{

editChoice = true;

}

else

{

Console.WriteLine("Некорректный ввод, введите заново");

}

}

switch (choice)

{

case 1:

ellipse.X = Input("Введите новое значение координаты X: ");

break;

case 2:

ellipse.Y = Input("Введите новое значение координаты Y: ");

break;

case 3:

ellipse.Radius = InputRadius("Введите новое значение Первого радиуса: ");

break;

case 4:

ellipse.SecondRadius = InputRadius("Введите новое значение Второго радиуса: ");

break;

case 5:

ellipse.X = Input("Введите новое значение координаты X: ");

ellipse.Y = Input("Введите новое значение координаты Y: ");

ellipse.Radius = InputRadius("Введите новое значение Первого радиуса: ");

ellipse.SecondRadius = InputRadius("Введите новое значение Второго радиуса: ");

break;

case 6:

break;

}

}

}

static void Show()

{

for (int i = 0; i < figures.Count; i++)

{

Console.WriteLine($"Порядковый номер обьекта: {i + 1}");

Console.WriteLine("----------------------------------");

figures[i].Print();

Console.WriteLine();

Console.WriteLine();

}

}

static void Main()

{

Point point = new Point(1, 2);

figures.Add(point);

Circle circle = new Circle(1, 2, 3);

figures.Add(circle);

Circle ellipse = new Ellipse(1, 2, 3, 2);

figures.Add(ellipse);

Ellipse ellipse2 = new Ellipse(32, 34, 5, 6);

figures.Add(ellipse2);

bool isChoice = false;

int choice;

do

{

Console.WriteLine(" Меню");

Console.WriteLine("1 - Вывод списка");

Console.WriteLine("2 - Удаление объекта");

Console.WriteLine("3 - Добавление объекта");

Console.WriteLine("4 - Изменение объекта");

Console.WriteLine("5 - Выход");

// Проверяем ввод пользователя

while (!int.TryParse(Console.ReadLine(), out choice) || choice < 1 || choice > 5)

{

Console.WriteLine("Некорректный ввод, введите заново");

}

switch (choice)

{

case 1:

Show();

break;

case 2:

Delete();

break;

case 3:

AddTOList();

break;

case 4:

Edit();

break;

case 5:

Console.WriteLine("Выход из программы.");

break;

}

} while (choice != 5);

Console.ReadKey();

}

}

}

namespace Lab5

{

class Program

{

static void Main()

{

//Проверка функционала конструкторов и методов для вывода

Point point = new Point(1, 2);

point.Print();

//Проверка функционала свойств

point.X = 3;

point.Y = 4; Console.WriteLine($"Координата точки X: {point.X}");

Console.WriteLine();

Console.WriteLine();

Circle circle = new Circle(1, 2, 3);

circle.Print(); Console.WriteLine();

Circle ellipse = new Ellipse(1, 2, 3, 2);

ellipse.Print();

//Проверка функционала методов для вычисления площади

Console.WriteLine();

Console.WriteLine($"Площадь окружности: {circle.Square()}");

Console.WriteLine($"Площадь эллипса: {ellipse.Square()}");

Console.WriteLine();

//Проверка функционала базовых конструкторов

Point point2 = new Point();

point2.Print();

Console.WriteLine();

Ellipse ellipse2 = new Ellipse(); ellipse2.Print();

Console.WriteLine(); Circle circle2 = new Circle();

circle2.Print(); Console.WriteLine();

}

}

}

1. Point.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab5

{

class Point //Базовый класс Точка

{

protected double x { get; set; }

protected double y { get; set; }

public double X

{

get { return x; }

set { x = value; }

}

public double Y

{

get { return y; }

set { y = value; }

}

public Point()

{ //Базовый конструктор

this.X = 0; this.Y = 0;

}

public Point(double x, double y)

{ //Конструктор

this.X = x;

this.Y = y;

}

public virtual void Print()

{ // Метод вывода на экран

Console.Write($"Координаты точки: ({X}, {Y})\n");

}

}

}

1. Circle.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab5

{

class Circle : Point

{ //Порожденный класс Окружность

protected double radius { get; set; }

public double Radius

{

get { return radius; }

set { radius = value; }

}

public Circle()

{

this.Radius = 0;

}

public Circle(double x, double y, double radius) : base(x, y)

{

this.Radius = radius;

}

public override void Print()

{

Console.Write($"Окружность:\n");

Console.Write($"Координаты центра: ({X}, {Y})\n"); Console.Write($"Радиус: {Radius}\n");

}

public virtual double Square()

{

double square = 3.14 \* Math.Pow(Radius, 2); return square;

Вывод программы:

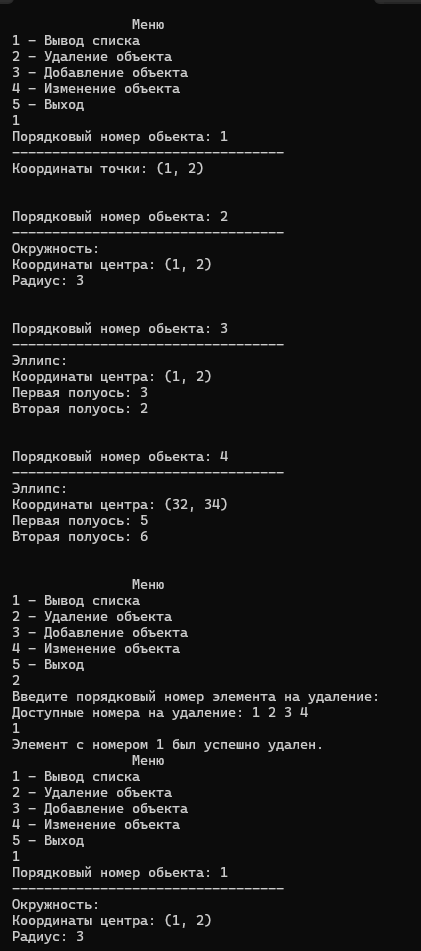


Рис.2 - Вывод

1. Ellipse.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab5

{

class Ellipse : Circle

{ //Порожденный класс Эллипс

protected double secondRadius;

public double SecondRadius

{

get { return secondRadius; }

set { secondRadius = value; }

}

public Ellipse()

{

this.secondRadius = 0;

}

public Ellipse(double x, double y, double radius, double secondRadius) : base(x, y, radius)

{

this.secondRadius = secondRadius;

}

public override void Print()

{

Console.Write($"Эллипс:\n"); Console.Write($"Координаты центра: ({X}, {Y})\n");

Console.Write($"Первая полуось: {Radius}\n"); Console.Write($"Вторая полуось: {SecondRadius}\n");

}

public override double Square()

{

double square = 3.14 \* radius \* secondRadius; return square;

}

}

}

**Результат испытаний программы:**

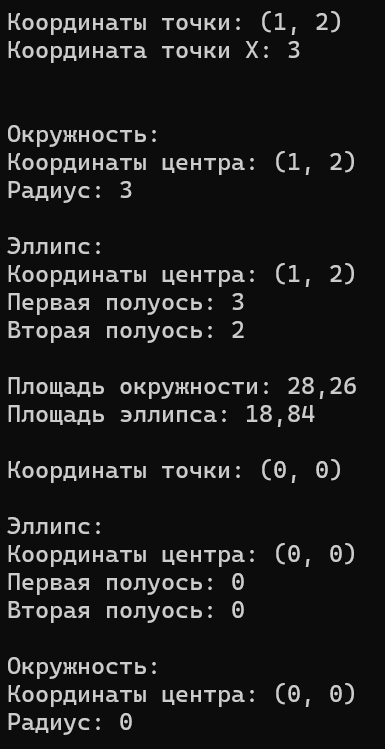


Рис.2 – Результат испытаний программы