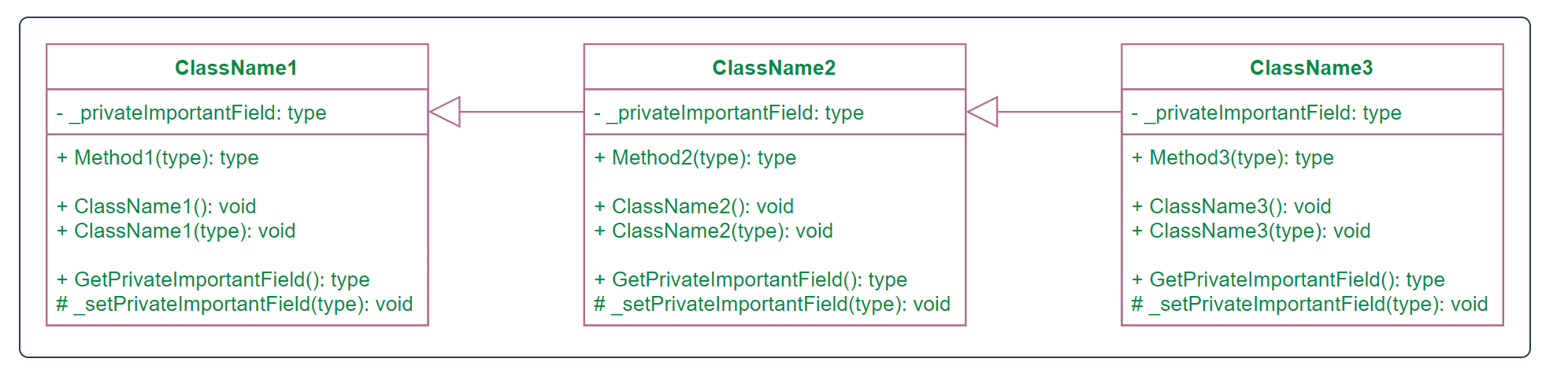
**Лабораторная работа №1**

**Классы, объекты, инкапсуляция, наследование**

Дана UML диаграмма классов. У каждого класса есть приватное поле (privateImportantField), конструктор без параметров и один перегруженный конструктор с параметром, значение которого передается в упомянутое поле.

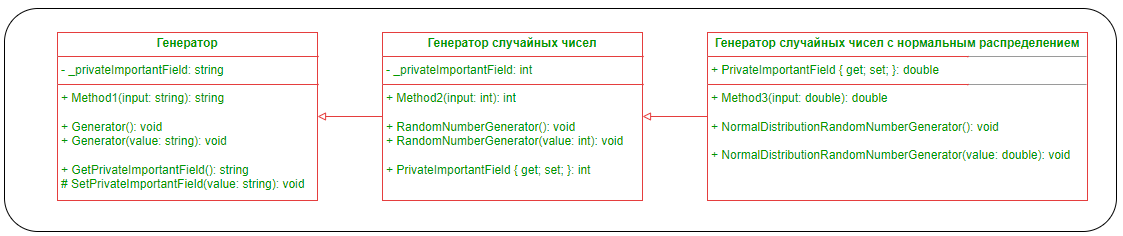


Необходимо придумать осмысленные поле и метод для каждого из классов (осмысленные и для его наследников). По желанию можно сделать классы более конкретными.

**Задание:**

* В draw.io https://app.diagrams.net/, изменить согласно варианту названия классов, полей и методов (ClassName1 – Генератор, ClassName2 – Генератор случайных чисел, ClassName3 – Генератор случайных чисел с нормальным распределением)
* Реализовать иерархию классов в коде. Геттер и сеттер для приватного поля реализовать:
  + для ClassName1 через методы (как на диаграмме);
  + для ClassName2 через свойство;
  + для ClassName3 через автосвойство.
* В основном теле программы создать экземпляры классов разными способами, продемонстрировать доступные и недоступные способы работы с ними.

Обновленная диаграмма:



Исходный код:

using System;

namespace lab1

{

public class Generator

{

private string \_privateImportantField;

// Конструктор без параметров

public Generator()

{

\_privateImportantField = "Default Generator";

}

// Конструктор с параметром

public Generator(string value)

{

\_privateImportantField = value;

}

// Метод

public string Method1(string input)

{

return $"Generator: {input}";

}

// Геттер через метод

public string GetPrivateImportantField()

{

return \_privateImportantField;

}

// Сеттер через метод

protected void SetPrivateImportantField(string value)

{

\_privateImportantField = value;

}

}

public class RandomNumberGenerator : Generator

{

private int \_privateImportantField;

// Конструктор без параметров

public RandomNumberGenerator() : base()

{

\_privateImportantField = 42; // Пример значения

}

// Конструктор с параметром

public RandomNumberGenerator(int value) : base()

{

\_privateImportantField = value;

}

// Метод

public int Method2(int input)

{

return \_privateImportantField \* input;

}

// Свойство для доступа к приватному полю

public int PrivateImportantField

{

get { return \_privateImportantField; }

set { \_privateImportantField = value; }

}

}

public class NormalDistributionRandomNumberGenerator : RandomNumberGenerator

{

// Автосвойство для приватного поля

public double PrivateImportantField { get; set; }

// Конструктор без параметров

public NormalDistributionRandomNumberGenerator() : base()

{

PrivateImportantField = 0.0;

}

// Конструктор с параметром

public NormalDistributionRandomNumberGenerator(double value) : base()

{

PrivateImportantField = value;

}

// Метод

public double Method3(double input)

{

return PrivateImportantField + input;

}

}

public class Program

{

public static void Main(string[] args)

{

// Экземпляр класса Generator

Generator generator = new Generator("Main Generator");

Console.WriteLine(generator.GetPrivateImportantField());

Console.WriteLine(generator.Method1("Hello"));

// Экземпляр класса RandomNumberGenerator

RandomNumberGenerator rng = new RandomNumberGenerator(100);

Console.WriteLine(rng.PrivateImportantField);

rng.PrivateImportantField = 200;

Console.WriteLine(rng.Method2(3));

// Экземпляр класса NormalDistributionRandomNumberGenerator

NormalDistributionRandomNumberGenerator normalRng = new NormalDistributionRandomNumberGenerator(1.5);

Console.WriteLine(normalRng.PrivateImportantField);

Console.WriteLine(normalRng.Method3(0.5));

}

}

}

**Класс Generator** имеет приватное поле \_privateImportantField, которое доступно через методы GetPrivateImportantField() и SetPrivateImportantField(). В конструкторе без параметров задается значение по умолчанию, в конструкторе с параметром – передается значение поля.

**Класс RandomNumberGenerator** наследуется от Generator, имеет свое приватное поле \_privateImportantField типа int. Для доступа к нему используется свойство PrivateImportantField.

**Класс NormalDistributionRandomNumberGenerator** наследуется от RandomNumberGenerator, имеет автосвойство PrivateImportantField типа double.

В основном теле программы создаются экземпляры каждого класса, демонстрируется доступ к полям через методы, свойства и автосвойства.

Результат:

