ДЕНЬ 8. СПЕЦИФИКАЦИИ 2

8.1. Расширить функционал заданного класса, используя наследование

Вам дан код класса, выполняющий преобразование текста

```
class TextTransformer
{
    protected string[] lines;
    public TextTransformer(string text)
    {
        lines = text.Split('\n');
    }

    virtual public string TransformWord(string word, int lineIndex, int wordIndex,
                                        int wordCount)
    {
        return System.Text.RegularExpressions.Regex.Replace(word, "[а-я]", "*");
    }

    virtual public string TransformLine(string line, int lineIndex)
    {
        string[] words = line.Split(' ');
        for (int i = 0; i < words.Length; i++)
        {
            words[i] = TransformWord(words[i], lineIndex, i, words.Length);
        }
        return string.Join(" ", words);
    }

    public string GetResult()
    {
        string[] result = new string[lines.Length];
        for (int i = 0; i < lines.Length; i++)
        {
            result[i] = TransformLine(lines[i], i);
        }
        return string.Join("\n", result);
    }
}
```

Ваша задача – поменять способ трансформации текста, создав класс, который наследуется от *TextTransformer*. В созданном подклассе для изменения поведения Вы должны будете переопределить методы *TransformWord* и (или) *TransformLine*. Все дополнительно созданные члены класса должны иметь спецификатор доступа private. Метод *GetResult* должен остаться без изменения.

Изменять класс *TextTransformer* недопустимо

В каждом варианте задания помимо использования нового преобразования для некоторых частей текста необходимо сохранить исходное (см. варианты).

Для демонстрации работы Вам необходимо разработать программу (консольную или оконную, на Ваш выбор) со следующей логикой:

1. пользовать вводит многострочный текст
2. создается экземпляр вашего подкласса и в конструктор передается исходный текст
3. с помощью вызова метода *GetResult* у экземпляра получается преобразованный текст
4. преобразованный текст выводится на экран

Пример

Новое преобразование: удалить каждое второе слово.

Исходное преобразование: каждая вторая строка

```
class DerivedTextTransformer: TextTransformer
{
    public DerivedTextTransformer(string text) : base(text)
    {
    }

    override public string TransformWord(string word, int lineIndex, int wordIndex,
                                         int wordCount)
    {
        if (lineIndex % 2 == 1)
        {
            return base.TransformWord(word, lineIndex, wordIndex, wordCount);
        }
        return wordIndex % 2 == 1 ? "" : word;
    }
}
```

Исходный текст:

```
В томленьях грусти безнадежной
В тревогах ШУМНОЙ суеты,
Звучал мне долго голос нежный
И снились МИЛЫЕ ЧЕРТЫ.
```

Преобразованный текст:

```
В  грусти
В ******** ШУМНОЙ *****
Звучал  долго  нежный
И ******* МИЛЫЕ ЧЕРТЫ.
```

Варианты

№	Новое преобразование	Старое преобразование
1	Удаление слов с цифрами	Каждое слово, которое совпадает с предыдущим словом в строке
2	Удалить знаки препинания	Каждое слово, заканчивающееся на согласную
3	Удалить все гласные	Каждое слово с дефисом
4	Перемешать в случайном порядке все слова в строке	Каждая строка, в которой присутствуют слова короче 5 символов
5	Поменять местами слова в каждой паре слов: аб вг де жз -> вг аб жз де	Каждая строка длиннее 4 слов
6	Удалить все согласные	Каждое слово длиннее 5 символов
7	Перевести в верхний регистр каждую вторую букву слов: абв гдеж -> аБв гДеЖ	Каждая строка короче 5 слов
8	Перевернуть строки (абг деж -> жед гба)	Каждая строка, начинающаяся с большой буквы
9	Удалить первое и последнее слова в строках	Каждая строка, в которой присутствуют слова длиннее 6 символов

10	Удалить все слова без гласных букв	Каждое слово, заканчивающееся на гласную
11	Удалить все слова, в которых меньше 5 согласных	Каждое слово, содержащее цифры
12	Перевести все буквы первого слова каждой строке в верхний регистр	Каждая строка, которая длиннее по количеству слов, чем предыдущая
13	Перевернуть каждое слово в строке (абг деж -> гба жед)	Каждое слово, начинающееся с большой буквы
14	Удалить все слова размером меньше, чем из 5 букв	Каждая строка, содержащая два одинаковых слова подряд
15	Удалить все слова размером больше, чем из 5 букв	Каждая строка длиннее 3 слов
16	Перевести в верхний регистр все согласные	Каждое слово длиннее 10 символов
17	Удалить все слова, в которых больше 5 согласных	Каждая строка короче 4 слов
18	Перевести первую букву каждого слова в верхний регистр	Каждое слово короче 8 символов
19	Удалить все слова без согласных букв	Каждая строка, заканчивающаяся знаком препинания
20	Перевести в верхний регистр все гласные	Каждое слово короче 5 символов

Примечание: под словами в данном задании понимаются подстроки, состоящие из любого количества любых символов (кроме пробела), разделенные одним пробелом

8.2. Создать класс, реализующий заданный интерфейс

Необходимо создать консольное приложение, позволяющее вводить числа с клавиатуры и выполнять с ними заданную операцию.

В коде программы должен быть класс, реализующий следующий интерфейс:

```
interface ICalculator
{
    void AddNumber(float number);
    float Calculate();
}
```

При запуске программы должен создаваться экземпляр класса, реализующего интерфейс *ICalculator*. Далее в цикле ввода чисел каждое новое число передается в метод *AddNumber*. Как только пользователь ввел пустую строку, ввод заканчивается, и программа должна вызывать метод *Calculate*, который выполняет заданную операцию с теми числами, которые передавались в метод *AddNumber*. После расчета полученный результат должен быть выведен в консоль.

В созданном классе, реализующем *ICalculator*, не должно быть обращения к консоли. Все члены этого класса, не указанные в интерфейсе, должны иметь модификатор доступа private.

Варианты

№	Операция	№	Операция
1	У каждого второго числа поменять знак и просуммировать все числа	11	Перемножить дробные части всех чисел

2	Просуммировать дробные части всех чисел	12	Просуммировать модули всех чисел
3	Просуммировать все числа, которые меньше, чем предыдущее число	13	Умножить числа на их порядковые номера и вычислить среднее
4	Вычислить среднее из квадратов чисел	14	Просуммировать квадраты всех чисел
5	Перемножить все отрицательные числа	15	У каждого второго числа поменять знак и посчитать среднее
6	Просуммировать все числа, которые больше, чем предыдущее число	16	Просуммировать все числа, которые больше первого из чисел
7	Просуммировать все четные числа	17	Просуммировать все нечетные числа
8	Извлечь квадратный корень из суммы квадратов чисел	18	Умножить числа на их порядковые номера и сложить результаты
9	Округлить числа и сложить результаты	19	Перемножить все положительные числа
10	Разделить все числа на первое из чисел и вычислить сумму результатов	20	Вычислить среднее значение

8.3. Прочитать метаданные из файла заданного бинарного формата

Вам необходимо разработать программу, которая считывает заданное поле метаданных (см. варианты) из указанного бинарного формата и выводит его на экран.

Обрабатываемый файл указывается пользователем при запуске программы, которая может быть, на Ваш выбор, консольной или оконной.

Возможные варианты бинарных форматов:

Формат	**Ссылки на описание**
BMP	https://ru.wikipedia.org/wiki/BMP
PNG	https://habr.com/ru/post/130472/
JPEG	https://ru.wikipedia.org/wiki/JPEG, https://habr.com/ru/post/102521/
GIF	http://home.onego.ru/~chiezo/gif.htm, https://habr.com/ru/post/274917/
WAV	http://microsin.net/programming/pc/wav-format.html, https://audiocoding.ru/articles/2008-05-22-wav-file-structure/
AVI	https://ru.wikipedia.org/wiki/Audio_Video_Interleave

Для анализа содержимого файла вы можете воспользоваться бесплатным онлайн-ресурсом https://hexed.it/

В папке https://github.com/Nordth/istu-priklad-practic-2020/tree/master/Day8/Task_ReadBinaryFiles приведены примеры тестовых файлов, метаданные которых приведены в следующей таблице:

Файл	**Поле метаданных**	**Значение**
sun.bmp	ширина	64
	высота	58
	битность растра	24
bird.png	ширина	16
	высота	16
	битовая глубина цвета	8
	наличие альфа-канала	да
lenna.jpg	ширина	220

	высота	220
	количество каналов	3
cat.gif	ширина	64
	высота	67
	количество кадров	52
	наличие глобальной палитры	да
laugh.wav	число каналов	2
	частота дискретизации	48000
	количество бит в сэмпле	16
coding.avi	ширина	480
	высота	270
	число кадров в секунду	10

Ваша программа должна обрабатывать любые файлы заданного формата. В папке файлы приведены исключительно для примера

Варианты

№	**Формат**	**Поле метаданных**
1	BMP	ширина
2	BMP	высота
3	BMP	битность растра
4	PNG	ширина
5	PNG	высота
6	PNG	битовая глубина цвета
7	PNG	наличие альфа-канала
8	JPEG	ширина
9	JPEG	высота
10	JPEG	количество каналов
11	GIF	ширина
12	GIF	высота
13	GIF	количество кадров
14	GIF	наличие глобальной палитры
15	WAV	число каналов
16	WAV	частота дискретизации
17	WAV	количество бит в сэмпле
18	AVI	ширина
19	AVI	высота
20	AVI	число кадров в секунду

8.4. Сконвертировать входной CSV-файл в бинарный файл заданного формата

Вы должны разработать программу (консольную или оконную, на Ваш выбор), которая преобразует входной CSV-файл в бинарный файл заданного формата. Бинарный файл должен состоять из трех секций:

1. Заголовок
2. Оглавление
3. Данные

Секция «Заголовок» состоит из 6 байт:

Смещение	Размер	Описание
00-01	2	Два байта 0x50 (сигнатура)
02-05	4	Типы полей вашей структуры по варианту. Первый байт задает тип первого поля, второй – второго и т.д. Возможные значения типа: • 0x49 – целый • 0x44 – вещественный • 0x42 – логический • 0x53 – строковый • 0x00 – поля нет

Секция «Оглавление» состоит из двух элементов:

- количество записей в файле (4 байта)
- список, в котором для каждой записи указано смещение, по которому находятся данные этой записи. Смещения указываются относительно начала секции «Данные» и занимают 4 байта. Таким образом, если в файле N записей, размер списка – 4N байт

В секции «Данные» последовательно располагается содержимое записей. В зависимости от типа, поля в записи занимают разный объем памяти

- *Целое поле* занимает 4 байта
- *Вещественное поле* – 8 байт
- *Логическое поле* – 1 байт (0x01 – true, 0x00 – false)
- *Строковое поле* записывается в виде двух значений: размер строки в байтах (2 байта) и сама строка (в кодировке UTF-8)

Порядок записи байт в числах – от младшего к старшему (little-endian)

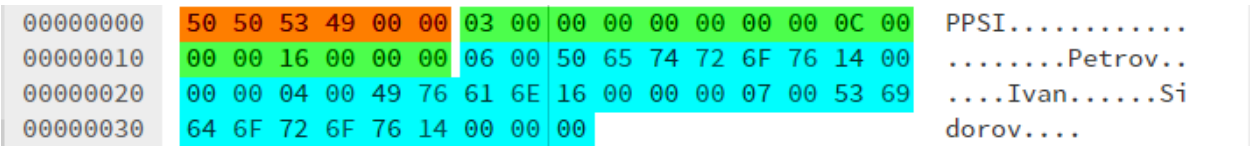
Пример

Запись: фамилия (строка), возраст (целое число)

Входной файл:

```
Petrov, 20
Ivan, 22
Sidorov, 20
```

Выходной бинарный файл (в шестнадцатеричном представлении)



Секция «Заголовок» обозначена оранжевым цветом:

Фрагмент	Описание
50 50	Сигнатура файла
53 49 00 00	Два поля в структуре: 1) строковое, 2) целочисленное

Секция «Оглавление» обозначена зеленым цветом

Фрагмент	**Описание**
03 00 00 00	Количество записей в файле (3)
00 00 00 00	Смещение первой записи относительно секции «Данные» (0)
0C 00 00 00	Смещение второй записи относительно секции «Данные» (12)
16 00 00 00	Смещение третьей записи относительно секции «Данные» (22)

Секция «Данные» обозначена голубым цветом

Фрагмент	**Описание**
06 00 50 65 74 72 6F 76	Поле «Фамилия» записи №1 (длина 6 байт, “Petrov”)
14 00 00 00	Поле «Возраст» записи №1 (20)
04 00 49 76 61 6E	Поле «Фамилия» записи №2 (длина 4 байта, “Ivan”)
16 00 00 00	Поле «Возраст» записи №2 (22)
07 00 53 69 64 6F 72 6F 76	Поле «Фамилия» записи №3 (длина 7 байт, “Sidorov”)
14 00 00 00	Поле «Возраст» записи №3 (20)

Варианты

№	**Запись**
1	Фамилия (строка), номер группы (строка), номер в группе (целое число), число выполненных заданий (целое число)
2	Тема письма (строка), адресат (строка), есть ли вложения (логический тип), число слов (целое число)
3	Фамилия (строка), число ролей (целое число), гонорар в млн. руб. (вещественное число)
4	Производитель (строка), объем выпуска (вещественное число), средняя цена (вещественное число)
5	Адрес отправления (строка), адрес доставки (строка), вес (вещественное число)
6	Название товара (строка), количество на складе (целое число), количество зарезервированных (целое число)
7	Город (строка), улица (строка), номер дома (число), номер этажа (целое число)
8	Номер заказа (строка), описание (строка), выполнен или нет (логический тип), сумма заказа (целое число)
9	Номер телефона (строка), имя оператора (строка), баланс в копейках (целое число)
10	Название материала (строка), объем (вещественное число), вес (вещественное число)
11	Фамилия (строка), год поступления (целое число), средний балл (вещественное число)
12	Название (строка), число сезонов (целое число), год выпуска первого сезона (целое число)
13	Фамилия (строка), оценка за теорию (целое число), оценка за практику (целое число)
14	Дисциплина (строка), номер курса (целое число), количество часов (целое число)
15	Адрес сайта (строка), число посетителей (целое число), число уникальных посетителей (целое число)
16	Фамилия (строка), рост (вещественное число), вес (вещественное число)

17	Компания (строка), сумма поступлений в млн. руб. (вещественное число), сумма списаний в млн. руб. (вещественное число)
18	Автомобильный номер (строка), год выпуска (целое число), пробег в км (целое число)
19	Название цеха (строка), план выпуска деталей (целое число), фактический выпуск деталей (целое число)
20	Фамилия (строка), должность (строка), оклад в руб (целое число)

8.5. Разработать программу, выполняющую команды, полученные по сети

Вам необходимо разработать программу (консольную или оконную, на Ваш выбор) для обработки списка целых чисел. Команды для обработки чисел программа должна получать по сети путем создания TCP-подключения к серверу.

Программа-сервер уже написана, ее сборка и исходный код лежат по адресу https://github.com/Nordth/istu-priklad-practic-2020/tree/master/Day8/Task_NetServer. После запуска программа-сервер программа создаст локальный сервер на порту 3005 (порт можно изменить, указав новый порт в аргументах командной строки)

В зависимости от варианта, ваша программа должна реализовывать несколько команд из следующего списка:

1. *add* <*number*> - добавить число <number> в список для обработки
2. *range* <*number1*> <*number2*> - добавить числа от <*number1*> до <*number2*> с шагом 1 в список для обработки
3. *rand* – добавить случайное число от -100 до 100 в список для обработки и вывести его на экран
4. *copy* – добавить в список для обработки копию последнего из ранее добавленных. Если в список числа еще не добавлялись, то добавить в список число 0
5. *clear* – очистить список чисел для обработки
6. *pop* – убрать из списка для обработки последнее из добавленных чисел и вывести его на экран
7. *mul* <*number*> – умножить все числа в списке для обработки на <number>
8. *neg* – поменять знак у всех чисел в списке для обработки
9. *abs* – сделать все числа в списке для обработки положительными
10. *print* – вывести на экран все числа в списке для обработки
11. *top* – вывести на экран последнее из добавленных в список для обработки чисел. Если чисел не было, то вывести «Список пуст»
12. *count* – вывести количество чисел, находящихся в списке для обработки
13. *countodd* – вывести количество нечетных чисел, находящихся в списке для обработки
14. *counteven* – вывести количество четных чисел, находящихся в списке для обработки
15. *sum* – вывести сумму чисел, находящихся в списке для обработки
16. *sumodd* – просуммировать все нечетные числа в списке для обработки
17. *sumeven* – просуммировать все четные числа в списке для обработки
18. *avg* – вывести среднее значение чисел, находящихся в списке для обработки

Для того, чтобы создать подключение к серверу выполните следующее:

```
using (TcpClient client = new TcpClient())
{
    client.Connect("127.0.0.1", 3005);
    using (NetworkStream stream = client.GetStream())
    {
        // Действия
    }
}
```

Работа с TCP-подключением похожа на работу с файлом. Общение с сервером будет происходить по бинарному протоколу, поэтому для удобства вы можете использовать классы *BinaryReader* и *BinaryWriter* с потоком *NetworkStream.* Порядок записи байт в числах – от младшего к старшему (little-endian)

После подключения вам необходимо отправить информацию, какие команды, используются в Вашем варианте. Для этого вы должны отправить 5 номеров команд в виде 5-ти однобайтовых чисел (нумерация соответствует нумерации списка выше)

После отправки перечня команд сервер отправит 4 байта – кол-во команд, который вам пришлет сервер.

После отправки числа команд сервер будет случайным образом посылать команды на исполнение. Команда на исполнение состоит из номера команды (1 байт) и, если есть, из аргументов команды (каждый аргумент занимает 4 байта)

Вам необходимо принять команду, вывести на экран ее название, аргументы и выполнить ее.

После отправки всех команд сервер разорвет соединение.

Перед завершением работы вашей программы вы должны будете вывести текущий список чисел

Пример

1. После подключения вы отправили номера команд: 1, 2, 7, 8, 15 (0x010207080F)
2. Сервер вам в ответ прислал, что число команд равно 4 (0x04000000)
3. Сервер отправил команду *add 6* (0x0106000000)
4. У вас в списке на обработку теперь одно чисдо – шесть
5. Сервер отправил команду *range 10 15* (0x010A0000000F000000)
6. У вас в списке на обработку теперь следующие числа: 6, 10, 11, 12, 13, 14, 15
7. Сервер отправил команду *sum* (0x0F)
8. Вы выводите сумму, равную 81
9. Сервер отправил команду *mul 2* (0x0702000000)
10. У вас в списке на обработку теперь числа: 12, 20, 22, 24, 26, 28, 30

Варианты

№	Команды				
1	range	rand	pop	top	sumodd
2	add	rand	clear	countodd	avg
3	rand	copy	clear	print	sum
4	range	rand	mul	top	avg
5	add	copy	abs	print	sumeven
6	rand	copy	neg	print	avg
7	range	rand	abs	top	counteven
8	add	range	pop	counteven	sumodd
9	add	copy	abs	top	sum
10	range	rand	clear	counteven	avg
11	add	range	clear	print	counteven
12	add	rand	neg	sumodd	avg
13	range	copy	abs	top	countodd
14	add	rand	neg	countodd	sum
15	add	copy	neg	countodd	sumeven
16	rand	copy	mul	print	avg
17	range	copy	mul	sumeven	avg
18	range	copy	mul	top	count
19	add	rand	pop	print	count
20	add	range	pop	sumodd	avg