**Тема:** Наследование и полиморфизм. Абстрактные классы.

**Задание 1.**

Реализовать класс **Stack** (стек) – структура данных, выполняющая функции стека фиксированного размера. Стек работает с типом object.

Константы:

* MIN\_ALLOWED\_STACK\_SIZE: int – минимально допустимый размер стека

Поля:

* data: object[] – массив данных стека, размер массива задается при создании объекта через конструктор
* capacity: int – кол-во элементов в стеке (изначально == 0, не путайте с размером массива data.Length)

Методы:

* Stack(int n) – конструктор, создающий стек заданного размера, если n < MIN\_ALLOWED\_STACK\_SIZE, то размер стека делать равным MIN\_ALLOWED\_STACK\_SIZE
* Stack() – конструктор, создающий стек со стандартным размером, равным MIN\_ALLOWED\_STACK\_SIZE
* Push(object value): bool – метод добавления элемента в стек, возвращает true если в стеке есть место и элемент добавлен, иначе false
* Pop(out bool ok): object – метод извлечения элемента из стека, возвращает элемент с верхушки стека, при это удаляя его из стека, так же через out-параметр передается true если элемент стек непустой, иначе false
* Top(out bool ok): object – аналогично Pop но только без удаления элемента из стека
* ToString(): string – перегруженный метод получения стека в виде строки, выводить заполненную часть стека, его размер и максимально допустимый размер

Далее реализовать класс **ExpandableStack** (расширяемый стек) – наследник класса **Stack,** выполняющий те же функции, что и стек, но при этом позволяющий увеличивать/уменьшать размер стек при необходимости (алгоритм соответствует СД ArrayList).

Константы:

* DEFAULT\_GROW\_COEFFICIENT: double – коэффициент расширения стека по умолчанию

Поля:

* grow\_coefficient: double – коэффициент расширения стека

Методы:

* ExpandableStack(int n) – конструктор, создающий стек заданного размера, если n < MIN\_ALLOWED\_STACK\_SIZE, то размер стека делать равным MIN\_ALLOWED\_STACK\_SIZE, grow\_coefficient инициализируется значением DEFAULT\_GROW\_COEFFICIENT
* ExpandableStack(int n, double grow\_coefficient) - конструктор, создающий стек заданного размера, если n < MIN\_ALLOWED\_STACK\_SIZE, то размер стека делать равным MIN\_ALLOWED\_STACK\_SIZE, grow\_coefficient инициализируется значением переданным значением если оно больше 1.0, иначе устанавливается значение по умолчанию
* ExpandableStack() - конструктор, создающий стек со стандартным размером, равным MIN\_ALLOWED\_STACK\_SIZE и стандартным коэффициентом расширения DEFAULT\_GROW\_COEFFICIENT
* new Push(object value): bool – метод добавления элемента в стек, возвращает true если в стеке есть место и элемент добавлен, иначе false; переопределяет метод базового класса через new, отличие поведения в том – что если в стеке недостаточно места, то его размер увеличивается прямопропорционально коэффициенту расширения. Всегда возвращает true
* Shrink(): void – метод ужатия емкости стека. Делает емкость стека равной текущем размеру стека (capcity == size)

**При необходимости можно реализовать дополнительные вспомогательные поля и методы.**

Протестировать объекты классов с помощью процедур:

* базовый класс
* наследник в ссылке наследника
* наследник в ссылке базового класса

**Задание 2.**

1. Разработать абстрактный класс «Геометрическая фигура» с полем имени фигуры. Добавить абстрактные методы получения площади и периметра. Добавить ToString(). Добавить виртуальный метод Print, выполняющий вывод имени фигуры на консоль.

2. Разработать наследников (на выбор треугольник, квадрат, прямоугольник, ромб, параллелограмм, трапеция, окружность, эллипс и т.д.). У наследников реализовать необходимые поля и методы:

* конструкторы
* свойства
* ToString()
* перегрузки абстрактных методов базового класса
* для некоторых фигур (НЕ ДЛЯ ВСЕХ) переопределить виртуальный метод Print базового класса через override

3. Разработать класс «Составная фигура», которая содержит (агрегирует) в себе несколько обычных фигур (экземпляров базового класса «Геометрическая фигура»). В данном классе реализовать необходимые методы работы с классом, в том числе:

* поиск общей площади всех фигур
* поиск общего периметра всех фигур

Данный класс так же может быть наследником Геометрической фигуры.

4. Протестировать программу, в том числе каждый класс в отдельности.

Для агрегирования фигур можно использовать:

* массивы
* List

Создать массив фигур, который содержит разные фигуры, в одном цикле посчитать для него общие площадь, периметр, сделать вывод на экран.