Задачник по ООП

Оглавление

[Раздел 1. Проектирование классов. 2](#_Toc2619484)

[Блок 1. Один класс 3](#_Toc2619485)

[Блок 2. Несколько классов 4](#_Toc2619486)

[Блок 3. Массивы 6](#_Toc2619487)

[Блок 4. Методы 8](#_Toc2619488)

[Блок 5. Конструкторы 10](#_Toc2619489)

[Блок 6. Инкапсуляция 12](#_Toc2619490)

[Блок 7. Методы и структуры 16](#_Toc2619491)

[Блок 8. Наследование 21](#_Toc2619492)

[Блок 9. Полиморфизм 25](#_Toc2619493)

[Раздел 2. Шаблонные задачи. 26](#_Toc2619494)

# Раздел 1. Проектирование классов.

Первый раздел посвящен базовым задачам проектирования классов, организации их иерархии, созданию объектов и манипуляции этими объектами.

Для успешного решения задач читатель должен быть компетентен в следующих темах:

* Концепция процедурного программирования
* С++ подобный синтаксис, в частности:
  + Объявление переменных
  + Операции (математические, логические, условные, присвоения)
  + Условные инструкции if, switch
  + Циклические инструкции for, while, do…while
* Основные алгоритмы в задачах работы с массивами и строками

Кроме того, желательно понимание ключевых структур данных (стек, очередь, списки, деревья, графы) и основных алгоритмов на них.

Ожидаемым результатом изучения данного раздела является способность читателя работать с базовыми концепциями объектно-ориентированного программирования, в частности:

* Примитивные и ссылочные типы данных
* Понятие объекта как объединения его состояния и поведения
* Класс как описание объектов: поля, методы, инициализаторы
* Понятие инкапсуляции как гарантия того, что объект всегда будет иметь легальное состояние
* Использование необрабатываемых исключений для контроля работы программы
* Использование наследования для организации иерархии родственных классов
* Использование абстрактных классов для задания корня иерархии
* Перегрузка методов для организации статического полиморфизма
* Переопределение методов для организации динамического полиморфизма
* Интерфейсы как способ разработать полиморфно гибкий код.
* Структурирование классов с использованием пакетов
* Обобщенные типы и их параметризации
* Использование анонимных классов, лямбда выражений и ссылок на методы для передачи поведения как состояния.

При изучении материала необходимо обратить внимание, что он разделен на логические блоки в соответствии с тем, какой перечень тем необходимо знать для решения данного блока задач. Необходимый перечень тем для данного блока указан пред ним. Также необходимо учитывать, что большая часть задач встречается не в одном блоке, а развивается на протяжении нескольких блоков. Так, например, в первом блоке мы видим задачу на построение точки координат, а во втором блоке ранее разработанная точка используется для описания линии. Таким образом, с одной стороны, поддерживается развитие кода от простого к сложному, а с другой стороны, для решения задач более поздних блоков, обязательно необходимо решение задач из всех предыдущих блоков. В связи с этим рекомендуется логически разносить разные задачи по проектам и пакетам, и не удалять ранее выполненные задачи.

Блок 1. Один класс: объявление классов, создание полей, метод приведения к строке, ссылочные и примитивные типы данных, использование new, обращение к полям, строки и конкатенация строк, вывод на экран.

**1.1 Дом.** Необходимо разработать сущность Дом, которая описывается количеством этажей в виде числа. У Дома можно запросить текстовую форму, которое имеет представление вида “дом с N этажами”, где N это число. Гарантировать правильное окончание фразы, в зависимости от количества этажей. Создать и вывести на экран дома с 1, 5, 23 этажами.

**1.2 Точка координат**. Необходимо разработать сущность Точка, расположенную на двумерной плоскости, которая описывается:

* Координата Х: число
* Координата Y: число
* Может возвращать текстовое представление вида “{X;Y}”

Необходимо создать три точки с разными координатами и вывести на экран их текстовое представление.

**1.3 Человек с ростом**. Необходимо разработать сущность Человек, которая описывается:

* Имя: строка
* Рост: число
* Может возвращать текстовое представление вида “Имя большой человек”, либо “Имя средний человек”, либо “Имя маленький человек”, выбор фразы должен зависеть от роста человека (правила определить самостоятельно).

Необходимо создать три человека с разным ростом и вывести их на экран.

**1.4 Комментарий.** Необходимо разработать сущность Комментарий, которая описывается:

* Текст: строка
* Рейтинг: число

Может возвращать текстовое представление вида “(рейтинг) текст”, например: “(169)  отличные  задачи!”. Необходимо создать три разных комментария и вывести их на экран.

**1.5 Человек с именем.** Необходимо разработать сущность Человек, которая описывается тремя параметрами: Фамилия, Имя, Отчество. Человек может быть приведен к строковому виду, включающему традиционное представление всех трех параметров, например “Иванов Иван Иванович”. Необходимо предусмотреть возможность того, что какой-либо из параметров может быть не задан. Создайте трех разных людей, у одного из которых заданы все три параметра, у другого нет фамилии, у третьего нет отчества и выведите их на экран.

Блок 2. Несколько классов**:** ничего нового знать не требуется.

**2.1 Прямая**. Необходимо разработать сущность Линия, расположенную на двумерной плоскости которая описывается:

* Координата начала: Точка
* Координата конца: Точка
* Может возвращать текстовое представление вида “Линия от {X1;Y1} до {X2;Y2}”

Для указания координат использовать сущность Точка, разработанную в примере 1.2. Создайте линии со следующими характеристиками:

1. Линия 1 с началом в т. {1;3} и концом в {23;8}.
2. Линия 2, горизонтальная, на высоте 10, от точки 5 до точки 25.
3. Линия 3 которая начинается всегда там же где начинается линия 1, и заканчивается всегда там же, где заканчивается линия 2.
4. Измените координаты первой и второй линий так, чтобы третья линия соответствовала требованиям задачи.
5. Измените координаты первой линии так, чтобы отвязать её от третьей линии.

**2.2 Человек с отчеством.** Необходимо разработать сущность Человек, которая описывается тремя параметрами: Фамилия, Имя, и Отец, где Отец - это тоже Человек. Человек может быть приведен к строковому виду, включающему традиционное представление: “Фамилия Имя Отчество”, где Отчество представляет собой Имя Отца с добавлением суффикса “ович”. Предусмотреть возможность отсутствия отчества.

Необходимо выполнить следующие задачи:

1. Создать людей: Чудова Ивана, Чудова Петра, Чудова Бориса
2. Сделать Ивана отцом Петра, а Петра отцом Бориса
3. Вывести на экран строковое представление всех троих людей.

При желании можно попытаться реализовать систему в более полном виде: предусмотреть разные виды суффиксов в зависимости от окончания имени, а также предусмотреть возможность задавать пол человека и менять суффикс отчества в зависимости от пола.

**2.3 Сотрудники и отделы.** Необходимо разработать сущность Сотрудник, которая описывается именем и отделом, в котором сотрудник работает, причем у каждого отдела есть название и начальник, который также является Сотрудником. Сотрудник может быть приведен к текстовой форме вида: “Имя работает в отделе Название, начальник которого Имя”

Необходимо выполнить следующие задачи:

1. Создать Сотрудников Петрова, Козлова, Сидорова работающих в отделе IT.
2. Сделать Козлова начальником IT отдела.
3. Вывести на экран текстовое представление всех трех Сотрудников (у всех троих должен оказаться один и тот же отдел и начальник).

**2.4 Комментирование комментариев.** Необходимо так модифицировать сущность Комментарий (из задачи **1.4**), что бы была возможность оставить Комментарий к Комментарию. Продумать процесс приведения данной конструкции к строковому представлению, например:

“

(169)  отличные  задачи!

---(5) совершенно согласен с этим комментарием!

------(0) что-то я ничего не понял

”

Создайте пример с вышеприведенными комментариями и выведите его на экран.

**2.5 Песни.** Необходимо разработать сущность Музыкальный Трек, которая описывается названием, автором и альбомом, к которому данный трек относится, причем альбом описывается названием и автором. Будем считать, что автор альбома - это автор всех песен, однако у него могут быть соавторы, которые указываются непосредственно в треках. Любой трек может быть приведен к строке вида: “Название, авторы: Автор1, Автор 2”, либо “Название, автор: Автор1”.

Выполнить следующие задачи:

1. Создать альбомы “Кукловод”, автор “Металлов” и “Шоссе к ООП”, автор “Асид”
2. Создать песни: “Состояние объектов”, автор “Янг”, “Шоссе к ООП” без автора, “Одноразовый программист”, автор “Бертон”.
3. Включить первые две песни в альбом “Шоссе к ООП”, а последнюю в альбом “Кукловод”.
4. Вывести на экран все три песни и проверить наличие всех авторов в песнях.

Блок 3. Массивы**:** создание одномерных и многомерных массивов, обращение к элементам.

**3.1 Человек с именами.** Основные требования к данной задаче совпадают с таковыми для задачи **2.2**, однаконеобходимо модифицировать решение таким образом, чтобы человек мог иметь множество собственных имен, например как у Эриха Марии Ремарка. С точки зрения отчествообразования используется только первое из имен. Воспользуйтесь тремя ранее созданными людьми, но замените “Ивана” на “Эрих Мария”, а “Борис” на “Жан Клод” и выведите их снова на экран.

**3.2 Студент с оценками.** Необходимо разработать сущность Студент, которая описывается:

* Имя: строка
* Оценки: массив целых чисел.
* Может возвращать текстовое представление вида “Имя: [оценка1, оценка2,…,оценкаN]”

Необходимо выполнить следующие задачи:

1. Создать студента Васю с оценками: 3,4,5.
2. Создать студента Петю и скопировать оценки Васи, присвоив содержимое поля с оценками Васи полю с оценками Пети.
3. Заменить первую оценку Пети на число 5. Вывести на экран строковое представление Васи и Пети. Объяснить результат
4. Создать студента Андрея и скопировать ему оценки Васи так, чтобы изменение оценок Васи не влияло на Андрея.

**3.3 Студент с предметами.** Необходимо разработать сущность Студент, которая описывается именем и набором оценок, причем оценки выставляются по конкретным предметам. Студент может быть приведен к текстовой форме следующего формата:

“Имя:

Предмет1: оценка1

Предмет2: оценка2

ПредметN: оценкаN”

Необходимо выполнить следующие задачи:

1. Создать студента Васю с оценками: информатика 5, математика 4, физика 5.
2. Заменить Васе физику на английский язык с оценкой 4, гарантировав, что процесс замены потребует не более чем одно присвоение
3. Создать студента Петю и скопировать ему оценки Васи.
4. Заменить оценку Пети по математике на 5. Проверить, что оценки Васи не изменились.

**3.4. Песни и ссылки.** Основные требования к данной задаче совпадают с таковыми для задачи **2.5**, однако необходимо так модифицировать решение, чтобы имея ссылку на трек, можно было бы получить доступ ко всем трекам того же альбома, к которому относится текущий трек. Выведите все треки альбома “Шоссе к ООП” имея ссылку на трек “Состояние объектов”.

**3.5 Дом с площадью.** Необходимо разработать сущность Дом, который состоит из этажей, у каждого этажа есть номер, на каждом этаже есть некоторое количество квартир, каждая квартира имеет площадь и порядковый номер. Дом может быть приведен к текстовой форме вида “Дом с площадью N”, где N это общая площадь всех квартир. Создать дом с тремя этажами, на первом этаже 1 квартира, на втором 3, на третьем 2, площадь квартир задать произвольно. Вывести на экран текстовую форму дома.

**3.6. Города.** Необходимо разработать сущность Город, которая будет представлять собой точку на карте со следующими характеристиками:

1. Название города
2. Набор путей к следующим городам, где путь представляет собой сочетание Города и стоимости поездки в него.

Реализуйте схему представленную на рисунке 1.

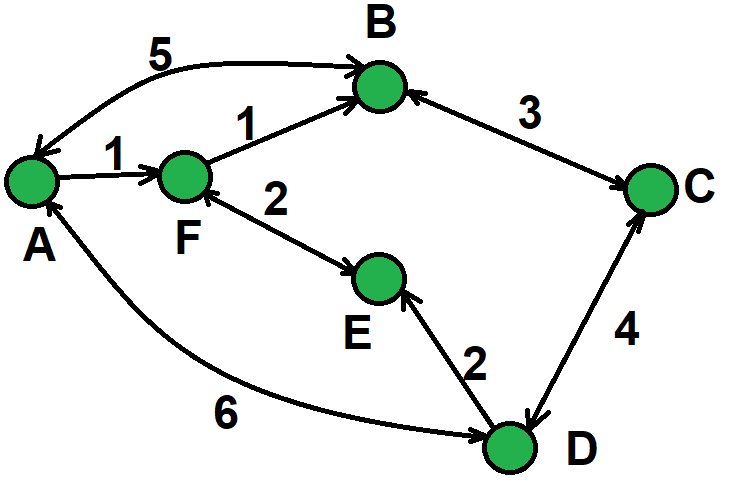


Рис. 1. Города и пути между ними.

**3.7 Публикации.** Необходимо разработать сущность Публикация, которая описывается следующими характеристиками:

* Имеет заголовок, строка
* Имеет множество тегов, каждый представляет собой одну строку
* Имеет множество Комментариев (из задачи **2.4**)
* Имеет рейтинг, целое число.

Кроме того, необходимо модифицировать Комментарии так, чтобы к одному Комментарию можно было оставлять множество других Комментариев.

Продумать приведение всего этого к строке. Создать публикацию с комментариями с тегами и рейтингом, а также комментариями к комментариям, и вывести это все на экран.

Блок 4. Методы**:** создание и вызов методов, варарги, перегрузка методов.

**4.1 Пистолет стреляет.** Необходимо разработать сущность Пистолет, которая описывается следующим образом:

1. Имеет Количество патронов
2. Может Стрелять, что приводит к выводу на экран текста “Бах!” в том случае, если количество патронов больше нуля, иначе делает “Клац!”. После каждого выстрела количество патронов уменьшается на один.

Создайте пистолет с тремя патронами и выстрелите из него пять раз.

**4.2 Кот мяукает.** Необходимо разработать сущность Кот, которая описывается следующим образом:

1. Имеет Имя
2. Может быть приведен к текстовой форме вида: “кот: Имя”
3. Может помяукать, что приводит к выводу на экран следующего текста: “Имя: мяу!”, вызвать мяуканье можно без параметров.
4. Может помяукать N раз, что приводит к выводу на экран следующего текста: “Имя: мяу-мяу-…-мяу!”, где количество “мяу” равно N.

Создайте кота по имени “Барсик”, и затем пусть он помяукает сначала один раз, а затем три раза.

**4.3 Длина линии.** Воспользуйтесь решением, полученным в задаче **2.1** по разработке сущности описывающей прямую линию. Добавьте ей возможность возвращать по запросу пользователя расстояние между точками в виде целого числа. Создайте линию из точки {1;1} в точку {10;15} и выведите на экран её длину.

**4.4 Ломаная линия.** Необходимо разработать сущность Ломаная, которая будет представлять собой ломаную линию (см. пример на рис.2). Ломаная линия представляет собой набор следующих характеристик:

* Имеет массив Точек (из задачи **1.2**), через которые линия проходит, проход осуществляется строго поочередно через все перечисленные точки.
* В любой момент к имеющимся Точкам можно добавить новые Точки (добавляется либо массив, либо просто перечень Точек, что приводит к получению нового массива, содержащего как старые, так и новые значения).
* Может вернуть общую длину Ломаной.

Необходимо выполнить следующие задачи:

1. Создать Ломаную, проходящую через точки {1;5}, {2;8}, {5;3}
2. Вывести на экран её длину
3. Добавить (к ранее созданной Ломаной) точки {5;15}, {8;10}
4. Снова вывести на экран длину Ломаной

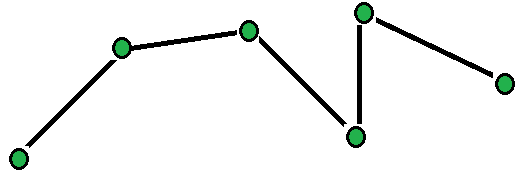


Рис. 2. Пример Ломаной

**4.5 Квадрат.** Необходимо реализовать сущность Квадрат. Квадрат имеет Точку (из задачи 1.2) отмечающую левый верхний угол и длину стороны. Необходимо реализовать Квадрату возможность вернуть его представление в виде Ломаной из задачи **4.4**.

**4.6 Поющий Человек**. Необходимо разработать сущность Человек, которая описывается:

* Имя: строка
* Текст песни: строка
* Может выводить на экран текст следующего вида: “Имя: текст песни”

Необходимо создать Человека с произвольным именем и песней и попросить Человека спеть

**4.7 Площадь этажа.** Дополните задачу 3.5 таким образом, чтобы у Дома можно было запрашивать:

* Общую площадь этажа
* Площадь квартиры по номеру
* Номера всех квартир определенного этажа

**4.8 Студент отличник.** Необходимо модифицировать сущность Студент, разработанную в задаче **3.2** таким образом, чтобы реализовать следующее требование:

* Может возвращать значение средней оценки студента
* Может возвращать информацию о том, является ли студент отличником или нет. Студент является отличником если у него только отличные оценки.

Блок 5. Конструкторы**:** создание конструкторов, перегрузка конструкторов, ключевое слово this, блоки инициализации, инлайн инициализация.

**5.1. Создаем Точку.** Дополните задачу **1.2** таким образом, чтобы для создания точки всегда требовалось бы указывать её координаты. Модифицируйте выполненные ранее задачи с учетом обновленных требований.

**5.2. Создаем Кота.** Дополните задачу **4.2** таким образом, чтобы при создании Кота можно было бы указывать или не указывать его Имя. По умолчанию всех котов зовут “Пушок”. Имя по умолчанию должно устанавливаться также в тех случаях, когда в конструктор передана пустая строка или null. Модифицируйте выполненные ранее задачи с учетом обновленных требований.

**5.3 Создаем линию.** Дополните задачу **4.3** таким образом, чтобы Линию можно было создавать двумя способами:

1. Указывая начало и конец в виде Точек из задачи **5.1**
2. Указывая предыдущую Линию и Точку конца. В этом случае, конец предыдущей линии становится началом текущей.

Необходимо выполнить следующие задачи:

1. Создать Линию1 с Точками {3;9}, {15;17}
2. Создать Линию2 с началом в конце Линии1 и концом в Точке{10;5}
3. Подвинуть конец Линии1 в координаты {1;1} таким образом, чтобы подвинулось начало Линии2.

**5.4 Создаем Человека с именами.** Дополните задачу **3.1** таким образом, чтобы создавать Человека можно было бы следующими способами:

1. Указывая только Имя. В этом случае текстовая форма не имеет ничего кроме имени.
2. Указывая только Отца. В этом случае человека будут звать “Вася”, фамилию же он получает от отца. Если вместо отца получена пустота, то фамилия будет “Васин”.
3. Указывая Отца и массив имен. В этом случае необходимо все непустые строки перекопировать в новый массив, который и будет представлять собой набор личных имен Человека, если они пустые то имя будет “Вася”, фамилию же он получает от отца. Если вместо отца получена пустота, то фамилия будет “Васин”.
4. Указывая Отца, массив имен и Фамилию. Имена переписываются так же, как и в случае 2. Если Фамилия это пустая строка, или же она не заполнена, то человек получает фамилию отца. Если вместо отца получена пустота, то фамилия будет “Васин”.

Реализация Человека должна гарантировать, что при необходимости изменить способ обработки имен или фамилии (например, если фамилия не задана, то это “Иванов”), это потребуется делать только в одном месте кода.

**5.5 Создаем Четырехугольник.** Необходимо создать класс Четырехугольник, который может представлять собой квадрат, прямоугольник, параллелограмм или трапецию. Создавать его можно следующими способами:

1. Указав Точку (задача **5.1**) левого верхнего угла и размер стороны. Получается квадрат. Размер стороны строго положительный, иначе сделать её равной 1.
2. Указав Точку (задача **5.1**) левого верхнего угла и размер двух сторон. Получается прямоугольник. Размер сторон строго положительный, иначе сделать их равным 1.
3. Указав Точку (задача **5.1**) левого верхнего угла, размер двух сторон и значение сдвига верхней стороны относительно нижней. Получается параллелограмм. Размер сторон строго положительный, иначе сделать их равным 1. Ограничений на размер сдвига нет.
4. Указав Точку (задача **5.1**) левого верхнего угла, размер трех сторон и значение сдвига верхней стороны относительно нижней. Получается трапеция. Размер сторон строго положительный, иначе сделать их равным 1. Сумма значений сдвига и длины меньшей из сторон не должна быть больше длины самой длинной стороны, иначе нужно отцентровать меньшую сторону относительно большей.

Поля объекта определите самостоятельно. Реализация должна гарантировать, что при необходимости изменить ограничения на значения каких-либо полей, это потребуется делать в наименьшем возможном количестве мест. Необходимо чтобы объекты поддерживали проверку того, является ли текущий объект квадратом, прямоугольником, параллелограммом или трапецией так, чтобы не хранить эту информацию в явном виде. Необходимо реализовать Четырехугольнику возможность вернуть его представление в виде массива связанных Линий из задачи **5.3**

**5.6 Создаем Публикации.** Необходимо модифицировать сущности Публикация и Комментарий (из задачи **3.7**) таким образом, чтобы их создание требовало указания какого-либо непустого текста, а для Публикации обязательным является указание название и как минимум одного тега. Модифицируйте выполненные ранее примеры с использованием нового способа создания объектов.

**5.7 Создаем Поющего Человека**. Необходимо доработать сущность из задачи **4.6**, гарантировав, что при создании Человека всегда будет указываться Имя, и возможно задаваться песня.

**5.8 Создаем Человека с ростом**. Необходимо доработать сущность из задачи **1.3**, гарантировав, что при создании Человека всегда будут указываться и Имя и рост.

Блок 6. Инкапсуляция**:** модификаторы public, private, геттеры и сеттеры, выбрасывание исключений через throw, класс RuntimeException и его наследники.

**6.1. Кот всегда имеет имя.** Дополните задачу **5.2** таким образом, чтобы гарантировать следующие требования:

* Имя Кота можно получить в любой момент
* Имя Кота можно изменить после создания объекта
* Гарантировать, что ни в какой момент времени нет способа сделать имя Кота пустым (т.е. null или пустая строка.

Попробуйте выполнить следующие строки кода:

1 Cat cat= new Cat(“барсик”);

2 cat.name=null;

3 Cat cat2= new Cat(null);

4 cat2.setName(null);

Первая строка кода (или её аналог) должна работать нормально, вторая строка компилироваться не должна, третья и четвертые строки должны гарантировать, что кота зовут “Пушок”.

**6.2. Сторона Квадрата.** Необходимо модифицировать решение задачи **4.5** таким образом, чтобы Квадрат имел конструктор, принимающий координату левого верхнего угла (в виде Точки из **5.1**), и размер стороны. Если сторона указана равной нулю или менее, выбрасывается ошибка с соответствующим текстом. Контроль должен осуществляться как во время создания объекта, так и позднее, в случае если будет необходимо менять размеры Квадрата. Необходимо предоставить возможность изменения размеров Квадрата и возможность узнать его размер.

**6.3 Перезарядка Пистолета.** Необходимо модифицировать сущность Пистолет из задачи **4.1**. Модификация предполагает внесение следующих дополнительных требований:

1. Имеет максимальное количество патронов. Максимальное количество устанавливается во время создания пистолета и не может быть изменено позднее. У пистолета можно узнать, какое максимальное количество он вмещает.
2. Может быть перезаряжен. Для перезарядки необходимо передать пистолету число, которое будет означать количество заряжаемых патронов. Если передано отрицательное число, необходимо выбросить ошибку, объясняющую, что отрицательного числа патронов быть не может. Если передано слишком большое число патронов – необходимо лишние вернуть.
3. Может быть разряжен. Это приводит к обнулению патронов в пистолете и возврате нужного числа пользователю.
4. Можно узнать сколько сейчас заряжено патронов.
5. Можно узнать заряжен он или разряжен.

Создайте пистолет вместимостью 7, зарядите три патрона, выстрелите из него пять раз, затем зарядите в него 8 патронов, выстрелите еще 2 раза и разрядите его, сделайте контрольный выстрел.

Если все выполнено верно, то должно быть выведено: Бах! Бах! Бах! Клац! Клац! Бах! Бах! Клац!

**6.4 Инкапсуляция линии.** Необходимо выполнить модификацию решения, полученного в задаче **5.3** таким образом, чтобы задача соответствовала следующим требованиям:

1. Должно быть возможно изменить координаты начала и конца Линии.
2. Необходимо гарантировать что при изменении координат точки начала или конца линии, должна так же изменяться соответствующая координата связанной линии (при наличии) и не мог произойти случайный разрыв.
3. У объекта можно запросить точки начала и конца.
4. Конструктор принимающий две Точки должен иметь возможность связывать линии.

Необходимо выполнить следующие задачи:

1. Создать Линию1 с Точками {3;9}, {15;17}
2. Создать Линию2 с началом в конце Линии1 и концом в Точке {10;5}
3. Попытаться получить доступ к полю, содержащему координату начала Линии1 и присвоить туда новый объект. Этот способ не должен быть доступен никаким образом.
4. Подвинуть конец Линии1 в координаты {1;1} и убедиться, что подвинулось начало Линии2.
5. Создать Линию3 с началом связанным с началом Линии 2, и с концом связанным с концом Линии 2. Подвинуть конец Линии3 в точку {23,42} и убедиться, что подвинулся конец Линии2.
6. Создать Линию4 с началом и концом совпадающим с Линией3, но не привязанной к ней. Убедиться, что изменение координат Линии4 не влияет на Линию3.

**6.5 Автомобильный номер.** Необходимо реализовать сущность представляющий собой государственный регистрационный номер российского образца на гражданский легковой автомобиль. Сущность описывается следующими характеристиками:

1. Имеет три буквы из списка русского алфавита (только те буквы, что совпадают с латиницей). При попытке сохранить что-либо другое должна быть ошибка.
2. Имеет три цифры, наличие обязательно иначе ошибка.
3. Имеет номер региона не менее двух цифр, наличие обязательно иначе ошибка.
4. При создании требуется указать все три характеристики
5. Значение любой характеристики можно узнать в любой момент.
6. Характеристики не могут меняться в течении жизни объекта.
7. Может быть приведен к строке в виде стандартного представления номера: xNNNxxRR, где x –буквы, N-цифры, а R – регион.

**6.6 Положительный круг**. Необходимо реализовать сущность Круг, которая будет использоваться для отображения кругов на мониторе. Круг описывается следующим образом:

1. Имеет Точку (описанную в задаче **5.1**) центра координат. Точка координат должна быть строго положительной, иначе должна выбрасываться ошибка. Точка может быть получена или изменена в любой момент времени
2. Имеет радиус, вещественное число, строго больше нуля, иначе должна выбрасываться ошибка. Радиус может быт получен или изменен в любой момент времени.
3. Инициализация круга требует указания обеих характеристик.
4. Может быть приведена к строке с указанием Точки и радиуса.

Обратите внимание, что менять Точку из задачи **5.1** нет необходимости. Выполните следующие задачи:

1. Создайте Точку1 с координатами {3;3}
2. Создайте круг с центром в Точке1 и радиусом 5.
3. Измените значение координат в Точке1 на координаты {-5;-5}. Проверьте, что координаты круга не изменились.
4. Получите центр координат круга и сохраните его как Точка 2.
5. Измените значение координат в Точке2 на координаты {-5;-5}. Проверьте, что координаты круга не изменились.

**6.7 Растущие Дома.** Необходимо модифицировать сущность Дом, разработанную в задаче **3.5** путем внесения следующих изменений:

1. Количество квартир и площадь квартир в доме указываются при инициализации объекта и не могут меняться в дальнейшем.
2. После того как дом был создан ему можно надстраивать дополнительные этажи с новыми квартирами, но нельзя менять этажи местами или удалять этажи, или менять квартиры на ранее имевшихся этажах.
3. У дома всегда можно получить массив всех квартир, или площадь одной конкретной квартиры. Считать, что все квартиры имеют сквозные номера (т.е. от 1 до N, где первый номер будет у первой квартиры на первом этаже)

Выполните следующие задачи:

1. Создайте дом с тремя этажами, на первом этаже 1 квартира площадью 6, на втором 3 квартиры с площадью по 2, на третьем 2 квартиры с площадью по 3.
2. Получите массив всех квартир и присвойте квартирам на третьем этаже площадь 0.
3. Добавьте дому еще один этаж, с одной квартирой площадью 4.
4. Выведите на экран текстовую форму дома, он должен иметь площадь 22.

**6.8 Правильный рост**. Необходимо модифицировать сущность Человек, разработанную в задаче **5.8** таким образом, чтобы она удовлетворяла следующим требованиям:

1. Человек неизменяем после создания
2. Всегда можно узнать рост и имя.
3. Рост и имя обязаны быть всегда. Рост должен находится в диапазоне от 10 до 300, имя должно быть не пустым, и содержать только буквы. Если эти условия не выполняются – необходимо выбросить ошибку объясняющую конкретную проблему.

**6.9 Безопасные Публикации.** Необходимо модифицировать сущности Публикация и Комментарий (из задачи **5.6**) таким образом, чтобы:

1. Название Публикации нельзя изменить после её создания, но всегда можно узнать
2. Изменение текста Публикации или Комментария доступно только до тех пор, пока данная сущность не была прокомментирована, или же не был изменен её рейтинг, иначе выбрасывается ошибка с соответствующим текстом.
3. Сущность должна предоставлять информацию о том, возможно ли её изменение.
4. Рейтинг можно увеличить или уменьшить на только на единицу за одну операцию
5. Теги не могут быть изменены после инициализации Публикации.

Возьмите ранее созданные комментарии и попытайтесь изменить их рейтинг. Добавьте новый комментарий и увеличьте его рейтинг на единицу.

**6.10 Поющего Человека знают по имени**. Необходимо доработать сущность из задачи **5.7**, гарантировав, что имя Человека нельзя изменить после его создания, но всегда можно его узнать, если при создании Человека указывается пустое имя, то Человека зовут “Вася”. Песни можно менять по желанию.

**6.11 Человека всегда имеет имя.** Необходимо доработать сущность из задачи **5.4** таким образом, чтобы имена Человека нельзя было изменить после создания, но всегда можно было их узнать. У Человека гарантированно должно быть хотя бы одно непустое имя. Отца и фамилию Человек может менять по желанию.

**6.12 Создаем Альбом**. При корректном решении задачи **3.4** вы должны были получить сущности Альбом и Трек. Доработайте эти сущности таким образом, чтобы:

1. Альбом неизменяемая сущность
2. Для создания Альбома необходимо указать авторов и список Треков
3. Каждый трек может встречаться только в одном альбоме и не должно быть способа изменить это

Блок 7. Методы и структуры**:** создание классов на основе классических структур данных. Предупреждая вопрос: готовые классы коллекций использовать нельзя.

**7.0 Неудлиняемый массив.** Необходимо разработать сущность НеудлиняемыйСписок, представляющую собой такой массив значений, элементы которого можно добавлять и удалять, однако максимально возможное количество элементов известно заранее. Сущность должна отвечать следующим требованиям:

* Имеет массив целых чисел
* Имеет текущий размер (текущее число хранящихся значений)
* Для инициализации необходимо указать максимальный размер Списка
* Можно добавить число в конец массива. Если Список уже полностью заполнен выбрасывается ошибка с соответствующим текстом.
* Можно вставить число в позицию N, причем позиция не может быть больше длины массива и не меньше 0, иначе выкинуть ошибку. Если Список уже полностью заполнен выбрасывается ошибка с соответствующим текстом.
* Можно удалить элемент Списка с номером N
* Можно получить значение из элемента N.
* Может быть приведен к строке
* Можно проверить пуст Список или нет
* Можно проверить есть еще свободное место в списке или нет.
* Можно узнать текущий размер. Текущий размер нельзя менять произвольным образом, он должен менять только при добавлении новых значений или удалении имеющихся.

**7.1 Изменяемый массив.** Необходимо сделать новую сущность Список, пользуясь функциональными требованиями задачи 7.0, но так их модифицировав, чтобы массив значений имел неограниченную длину (таким образом туда можно добавить сколько угодно значений).

**7.2 Оптимальный изменяемый массив.** Необходимо так изменить реализацию задачи **7.1,** чтобы она удовлетворяла следующим требованиям:

1. Доступ к элементу осуществляется за время O(1) (константное время).
2. Добавление нового значения в конец массива в большинстве случаев должно выполняться за константное время, исключения могут встречаться не чаще чем в каждом N+N/2 случае добавления, где N – это порядковый номер последнего добавления в конец, которое было выполнено не за константное время.
3. Гарантируйте, что как минимум первые 8 добавлений в конец (не прерванные добавлением в другие позиции) будут выполнены за константное время.
4. Все хранящиеся значения можно запросить в виде стандартного массива.
5. Значения можно: добавить в конец, добавить в позицию N, удалить из позиции N, изменить значение в позиции N, получить из позиции N. Попытка обратиться к элементу с отрицательным индексом, или индексом большим, чем количество ранее добавленных элементов должно приводить к ошибке.

**7.3 Набор кругов.** Необходимо реализовать сущность Набор, которая будет представлять собой изменяемый массив, в котором хранятся Круги из задачи **6.6**. Отличием Набора от простого массива является то, что он хранит только уникальные объекты, и в случае попытки добавления такого Круга, который уже храниться в Наборе, новый круг должен заменять старый, а старый возвращаться пользователю. Круги считать одинаковыми в том случае, если у них одинаковый радиус и одинаковая Точка центра координат. Решение должно быть расширяемо таким образом, что при необходимости применить его для хранения, к примеру, строк, потребовалось бы сменить только тип хранящихся элементов и не менять логику добавления объектов.

**7.4 Бинарное дерево.** Необходимо реализовать структуру данных, представляющую собой классическое бинарное дерево. Общее описание можно сформулировать следующим образом: необходимо разработать сущность Узел, которая описывается следующими характеристиками:

1. Хранит число
2. Хранит ссылку на родительский Узел
3. Хранит ссылки на правого и левого потомков, представляющих собой Узлы
4. Имеет метод добавить значение, который проверяет, записано ли значение в данном узле. Если нет – то записывает текущее значение и заканчивает работу. Если значение уже есть, то он проверяет, больше ли новое значение чем имеющееся или нет. Если больше, то создает правого потомка и вызывает его метод добавления передавая туда новое число, иначе создает левого потомка и отправляет значение туда.

После реализации сущности, создать её экземпляр и добавить в него числа: 3 5 4 7 1 2. Просмотреть результат.

Задание для саморазвития: выяснить, возможно ли такое бинарное дерево, которое будет гарантировать, что все находящиеся в нем элементы отсортированы.

**7.5 Маршрут.** Для выполнения данной задачи понадобится решение полученное в задаче **3.6** (Города).

Прежде всего, модифицируем сущность Город, путем внесения следующих изменений:

1. Для создания Города требуется указать его название
2. Набор путей между Городами можно изменять в любой момент добавляя или удаляя имеющиеся пути (используй сущность из задачи **7.2**), а также менять существующие.

Далее реализуем сущность Маршрут, которая будет предоставлять по запросу пользователя массив городов, через которые ему надо пройти для перемещения из одного города в другой по имеющейся карте. Сущность имеет следующие характеристики:

1. Имеет Город начала и Город конца пути.
2. Инициализация Маршрута может быть выполнена только если указана точка начала и конца пути. Если указано null значение - то ошибка.
3. Точку начала и конца можно изменить в любой момент времени, но они всегда должны быть корректны.
4. Может вернуть массив Городов, представляющий собой маршрут из начала в конец. Массив содержит все Города (в порядке очереди) через которые надо пройти что бы попасть из Города начала в Город конца, причем и начало и конец также содержаться в этом массиве. Алгоритм формирования пути в данном случае не существенен, можно выбрать вариант со случайным путем, путем проходящим через наименьшее число городов, или самым дешевым путем. Если путь найти невозможно – возвращается пустой массив.
5. Маршрут может быть приведен к строке, которая будет возвращать название всех городов маршрута.
6. Создание объекта и изменение точек начала и конца выполняется за константное время O(1).

Воспользуйтесь картой городов из задачи 3.6 и выведите маршрут из Города F в Город D

**7.6 Готовь Маршруты заранее.** Необходимо реализовать вторую версию задачи **7.5**. В этой реализации критичным является только время запроса массива городов и строкового представления Маршрута: они должны выполняться за константное время O(1).

Воспользуйтесь картой городов из задачи 3.6 и выведите маршрут из Города E в Город B

**7.7 Очередь Городов**. Необходимо дополнить решение задачи **7.6** таким образом, чтобы возвращался не массив Городов, а их Очередь. Очередь - это по сути изменяемый массив со следующими особенностями:

1. Нельзя получить доступ или изменить элемент с произвольным индексом
2. Новый элемент можно добавить только в конец
3. Получить можно только первый элемент, причем получение означает, что элемент должен уйти из Очереди. Элементы не могут пропадать из Очереди произвольно.
4. Сохраняется необходимость приводить к строке, проверять на пустоту, получать все элементы в виде массива

Воспользуйтесь картой городов из задачи 3.6 и выведите все Города пути из A в F

**7.8 Стопка Книг**. Для создания некоей информационной системы понадобилось разработать сущность Стопка, которая будет организовывать хранение Книг.

Книга представляет собой сущность со следующими характеристиками:

1. Имеет название и автора (строки, не могут быть изменены после создания, обязательны для инициализации объекта, не могут быть пустыми, узнать их можно в любой момент)
2. Имеет Список цитат (строки, могут быть добавлены или удалены в любой момент, указывать при инициализации не нужно)
3. Может сообщить, присутствует ли определенная цитата в книге.
4. Книга может быть приведена к строке вида “Автор. Название.”

Система должна позволять искать цитаты среди всех книг, для чего определяет их множество: Стопку. Основная идея Стопки в том, что пользователь кладет книги в такой последовательности, что бы сверху лежали те книги, в которых больше вероятность найти цитату, а снизу – те в которых вероятность меньше. Какой порядок правильный – определит сам пользователь, однако Стопка должна иметь следующие характеристики:

1. По сути стопка это изменяемый массив
2. Нельзя получить доступ или изменить элемент с произвольным индексом
3. Новый элемент можно добавить только в конец
4. Получить можно только последний элемент, причем получение означает, что элемент должен уйти из Стопки. Элементы не могут пропадать из Стопки в произвольном порядке.
5. Сохраняется необходимость приводить к строке, проверять на пустоту, получать все элементы в виде массива

Создайте три книги, лежащие в стопке в следующем порядке:

* Достоевский “Идиот”
* Лем “Сумма технологий”
* Бредбери ”Вино из одуванчиков”

Добавьте в каждую книгу произвольное количество любых строк текста. Выведите на экран текстовое представление книги в которой встречается цитата: “Двадцать пять – достойный пожилой возраст”

**7.9 Добавление Комментариев**. Модифицируйте решение задачи **6.9** таким образом, чтобы удовлетворять следующим требованиям:

1. Комментарии можно добавлять в любой момент времени. Для выполнения этой задачи воспользуйтесь сущностью из решения задачи 7.2
2. Комментарии нельзя удалять.
3. У любой Публикации или Комментария всегда можно получить все Комментарии к нему.

Напишите простое консольное приложение, которое позволит удобным образом просматривать комментарии, в том числе с учетом перехода по уровням вложенности, а также добавлять новые комментарии и менять рейтинг существующих Комментариев.

**7.10 Человек знает песни**. Необходимо доработать сущность из задачи **6.10**, путем внесения следующих дополнений:

1. Человек знает не одну песню, а множество
2. Песни можно добавлять и удалять в любой момент. Для выполнения этой задачи воспользуйтесь сущностью из решения задачи 7.2
3. Когда человек поет, песня для исполнения выбирается случайным образом из списка песен. Для решения данной задачи воспользуйтесь классом Random.

Создайте человека с тремя разными песнями и попросите его спеть.

**7.11 Перевод Сотрудников.** Необходимо доработать сущности Сотрудник и Отдел, полученные в результате выполнения задачи **2.3**. Необходимо что бы решение соответствовало следующим требованиям:

1. Один Сотрудник может работать только в одном Отделе.
2. Создание сотрудника требует указания непустого имени, а создание отдела – непустого названия и начальника. При нарушении требования должна быть ошибка.
3. Сотрудника можно принять на работу, перевести в другой отдел, и уволить. Принятый на работу сотрудник записывается в конкретный отдел, переведенный меняет отдел, а уволенный теряет отдел.
4. У Отдела можно запросить весь список сотрудников, начальника и название.
5. У Отдела можно сменить начальника, но нельзя сделать так, чтобы у Отдела не было Начальника.
6. Название Отдела сменить нельзя
7. В Отдел можно принять сотрудника на работу, а также уволить сотрудника.
8. Недопустима ситуация, при которой полученный у Отдела список сотрудников содержит людей не работающих в этом отделе. Так же недопустима ситуация, при которой Сотрудник, работающий в отделе не присутствует в списке, возвращаемом Отделом.

Для выполнения этой задачи можно использовать сущность из решения задачи **7.2**

**7.12. Телефонная книга**. Необходимо разработать сущность ТелефонныйСправочник, который будет использоваться для хранения контактов в телефоне. Будем считать достаточным хранение пар телефон-строка, например “89003337788 - Вася”. Сущность описывается следующими требованиями:

1. Хранит пары, состоящие из телефона и строки контакта
2. Одному контакту может соответствовать несколько разных телефонов
3. Можно добавить новую пару. Если происходит попытка добавления такой пары, которая уже была, то ничего не меняется.
4. Можно найти телефон указав строку контакта. Если такого контакта нет, возвращается null. Если контакт есть, то возвращается массив телефонов.
5. Можно получить массив всех контактов, указав часть названия (начиная с первого символа).

**7.14 Студент и изменяемые оценки.**

Необходимо доработать сущность Студент, разработанную в задаче 4.8 таким образом, чтобы выполнялись следующие требования:

* Имеет имя, строка, неизменяемое.
* Имеет набор оценок, набор хранится с помощью сущности **7.2**
* Инициализация объекта может быть выполнена следующими двумя способами:
  + С указанием имени
  + С указанием имени и набора оценок
* Студент обязательно имеет имя. Если имя пустое (имеет менее одной буквы) то пользователь долен быть принудительно уведомлен об ошибке с предъявлением текста, объясняющего ситуацию.
* Оценки можно только добавлять, оценки описываются числами 2,3,4,5. У студента не могут хранится некорректные оценки. При добавлении некорректной оценки, пользователь долен быть принудительно уведомлен об ошибке с предъявлением текста, объясняющего ситуацию.
* У студента можно получить весь список оценок, но таким образом, чтобы невозможно было студенту добавить некорректную оценку.

Блок 8. Наследование**:** наследование, extends, абстрактные классы, абстрактные методы, ключевое слово super, переопределение методов.

**8.1 Автомат**. Для выполнения данной задачи необходимо воспользоваться решение полученным в задании **6.3**. Необходимо разработать сущность Автоматический Пистолет. Автоматический пистолет. Это такой Пистолет, который может стрелять как одиночными выстрелами, так и очередями. Сущность должна отвечать следующим требованиями:

1. Имеет и умеет все тоже самое что и Пистолет
2. Имеет скорострельность (целое число, неизменяемое) которое обозначает количество выстрелов в секунду, поддерживаемое данным пистолетом.
3. Умеет стрелять N секунд, что приводит к количеству выстрелов равному N умноженное на скорострельность.
4. Инициализация может быть выполнены следующими способами:
   1. Без параметров. Скорострельность 30, вместимость 30.
   2. С указанием максимального числа патронов. Скорострельность будет равна половине обоймы
   3. С указанием максимального количества патронов в обойме и скорострельности.

Решение должно отвечать следующим требованиям:

1. Не требуется внесение изменений в Пистолет полученный после задачи 6.3
2. При появлении у Пистолета дополнительных методов – они автоматически появляются и у АвтоПистолета.
3. При изменении звука выстрела Пистолета он должен автоматически измениться и у АвтоПистолета.

**8.2 Рисовалка**. У нас появился заказчик, для которого требуется разработать векторный графический редактор. На первой итерации принято решение, что пользователю будет достаточно иметь возможность рисовать следующие примитивы:

* Линия (из задачи **6.4**)
* Квадрат
* Прямоугольник
* Круг
* Овал

Так же, в процессе разработки архитектуры принято решение, что каждый объект, который будет помещаться на поле для рисования должен:

1. Иметь координату начала в том или ином виде (в виде Точки из задачи **5.1**). Точка обязана задаваться при создании объекта и не может быть отрицательной, иначе ошибка. Считать верхний левый угол как точку {0;0}, координата X увеличивается вправо, а Y вниз.
2. Может вернуть информацию о положении по горизонтали и вертикали. Поведение неизменно для всех объектов.
3. Может быть сдвинут по горизонтали и по вертикали на N, но не уходить в отрицательную область, иначе ошибка.
4. Может вернуть длину всех линий, составляющих данный примитив. Поведение обязано присутствовать у всех, но задается каждым примитивом самостоятельно.

Разработайте все необходимые сущности, придерживаясь следующих требований:

1. Все примитивы могут менять свои размеры (применительно к конкретному типу)
2. У всех примитивов можно узнать их размеры (применительно к конкретному типу)
3. Гарантируйте отсутствие повторяющегося кода. Весь повторяющийся для нескольких сущностей код должен находится в одном месте.
4. В случае определения новых полей и методов, которые должны быть у всех объектов размещаемых на поле, они добавляются в одном месте и автоматически получаются всеми необходимым сущностями (без написания дополнительного кода)

По завершению разработки выполните следующие задачи:

1. Создайте три произвольные линии и объедините их в массив
2. Создайте два произвольных прямоугольника и один квадрат и объедините их в массив
3. Создайте два произвольных круга и один овал и объедините их в массив
4. Сдвиньте конец всех линий в массиве на 5 вниз.
5. Увеличьте длину всех прямоугольников в массиве на 10.
6. Увеличьте радиус всех кругов в массиве на 7.
7. Объедините все примитивы в массив и сдвиньте всю картинку на 23 вправо.

**8.3 Общие Маршруты**. Обратите внимание на сходство задач **7.5** и **7.6**. Реализуйте такую архитектуру, которая сохранит работоспособность сущностей из обеих задач, но позволит добиться следующих целей:

1. Будет объединять классы построения маршрутов в графе
2. Точки конца и начала должны быть объявлены только один раз, а не в каждом классе.
3. Проверка точек на корректность должна быть описана таким образом, чтобы при необходимости её изменения это нужно было бы делать только в одном месте.
4. Алгоритм построения маршрута должен быть описан таким образом, чтобы при необходимости изменить его это делалось бы только в одном месте.

Проверьте работоспособность полученной конструкции на ранее реализованных заданиях.

**8.4 Автомобильные номера**. Сущность, разработанную в задаче **6.5,** решено использовать (с необходимыми доработками) в информационной системе фотофиксации автомобилей, проезжающих под видеокамерой. Однако, для этого придется провести дополнительную разработку, так как номера могут иметь разный вид в зависимости от типа транспортного средства и страны его регистрации. К архитектуре выдвинут набор общих требований:

1. В качестве номеров может использоваться только “номерная” сущность (не строка, не список, только то, что является номером)
2. Состав информации в номере может различаться и зависит от конкретной страны, которая их утвердила
3. Все номера имеют общую характеристику: номер обязан иметь код принадлежности стране. Код состоит только из букв, не менее одной и не более трех. У любого номера можно получить его код и название страны принадлежности. Необходимо гарантировать, что разработчик не сможет создать такой номер, который нарушает эти правила.
4. Все номера неизменяемые.
5. Для номера конкретной страны должно быть возможно получить его отдельные составляющие (см. пример с номерами РФ)
6. Номера могут быть приведены к строке соответствующей их традиционному представлению
7. Гарантировать отсутствие дублирующегося кода.

В отношении номера машины зарегистрированной в России необходимо учесть следующие дополнительные вопросы:

1. Номер может быть специально для такси, в этом случае у неё не три буквы, а две, и записываются они перед цифрами.
2. Может быть приведен к текстовой форме вида “X YYY XX ZZ RUS”, где X – это буквы, Y это цифры, а ZZ – цифры кода региона.
3. У машины можно узнать, является ли она такси.

Кроме того, необходимо разработать сущность отвечающую за Легковой Номер Германии, которая описывается следующим образом:

1. Имеет от одной до трех букв (код региона)
2. Имеет еще от одной до двух букв (номерных)
3. Имеет от одной до четырех цифр
4. Каждая из выше описанных характеристик может быть получена отдельно.
5. Признак классического автомобиля: буква ‘H’ в конце номера.
6. Может быть приведен к текстовой форме вида “D XXX YY ZZZZ H”, где X – это буквы региона, Y это буквы номера, а ZZZZ – цифры номера.
7. У машины можно узнать, является она классической или нет

Создайте два номера РФ и два номера Германии. Объедините их в один массив и выведите текстовую форму объектов данного массива на экран.

**8.5 Запреты.** Запретите наследование от Легковых Номеров РФ.

**8.6 Фейерверк**. Необходимо разработать архитектуру для сущностей, представляющих собой Пиротехнику. Для всей Пиротехники общей составляющей является необходимость указать при создании срок годности и реализация метода определяющего – годна ли данная пиротехника к использованию. Считать, что при попытке использовать пиротехнику после истечения срока годности, у неё могут возникать случайные эффекты, определяемые разработчиком самостоятельно.

Подсказка: System.currentTimeMillis(); позволит получить текущее время в миллисекундах. Задавать срок годности можно в них же.

Необходимо разработать две сущности: Петарда и РимскаяСвеча. Петарда — это подрывной заряд, издающий хлопок, описанная следующим образом:

* Неизменяемая
* Имеет мощность: целое число, можно узнать, задается при инициализации
* Может хлопнуть, после чего мощность меняется до нуля. Хлопок приводит к выводу на экран текста “БУМ!”, причем размер строки может меняться в зависимости от мощности заряда, например при более мощном заряде станет “БУУУУУУМ!!!!!!!”

РимскаяСвеча — это многозарядное устройство, описанное следующим образом:

* Неизменяемая
* Имеет количество зарядов: целое число, можно узнать, задается при инициализации, количество зарядов можно запросить.
* Может стрелять, в результате чего происходит N последовательных выводов на экран текста “шшшш……бах!”, разделенных промежутком времени 200 миллисекунд, где N - это количество зарядов. После выстрела количество зарядов становится равно нулю.

Идея: попробуйте реализовать логику выстрела РимскойСвечи в виде псевдографики.

Реализуйте классы избегая дублирующегося кода. Создайте два экземпляра петарды и один РимскойСвечи, одну Петарду используйте оп истечении срока годности, а остальное нормальным образом.

**8.7 Виды птичек.** Необходимо реализовать сущность Птица. Известно что у Птицы есть тип и она умеет петь, что приводит к выводу на экран некоего текста.

Создайте следующих Птиц:

* Воробей. Поет текст “Чырык”.
* Кукушка. В результате пения происходит вывод на экран текста “ку-ку”, причем текст выводится случайное количество раз в диапазоне от 1 до 10.
* Попугай. При создании ему задается строка текста. Поет выводя эту строку на экран, причем случайно переставляя слова местами.

**8.8 Взял сверху – положи вниз**. В результате исследования вариантов использования сущности Стопка в задаче **7.8** принято решение о необходимости иметь два варианта Стопки – исходный вариант, и вариант при котором предпоследняя взятая книга отправляется вниз стопки. Для иллюстрации предположим, что у нас есть книги, которые лежат в следующем порядке (слева - направо):

A B C D E F

Сначала извлекаем книгу A – она уходит из стопки и остается последовательность

B C D E F

Если книга A нас не устроила, и мы хотим взять еще книгу, то A добавляется в конец, а B уходит из стопки, получается

C D E F A

Если книга B нас не устроила, и мы хотим взять еще книгу, то B добавляется в конец, а C уходит из стопки, получается

D E F A B

И так далее.

Выполните реализацию новой стопки таким образом, чтобы гарантировать следующие условия:

1. С каким типом данных работает обычная Стопка, с таким же работает и измененная. Например, если мы скажем, что Стопка вместо Книг работает с досками, то и изменённая работает с ними.
2. Порядок добавления элементов в стопку работает гарантированно одинаково и в обычной Стопке и в изменённой.

**8.9 Общность людей.** Проанализируйте задачи с номерами **7.10**, **6.8** и **6.11**. Как не трудно заметить, все три класса представляют собой разные варианты людей, которые, однако, объединены наличием Имени и принципов задания имен. Объедините всех людей в единую структуру, вынеся концепцию задания и обработки имен Человека в единое место. Учтите, что создание Человека не имеющего ничего кроме имени не адекватно.

Блок 9. Полиморфизм**.** Интерфейсы, абстрактные методы, множественное наследование и реализация, полиморфизм, дефолтная реализация методов.

**9.1 Мяукающие**. Напишите метод, который принимает один объект имеющий возможность мяукать и попросите его помяукать. При разработке необходимо учесть, что теоретическую возможность мяукать могут иметь не только коты, но и другие типы объектов, включая людей, решение должно поддерживать возможность такого расширения.

Дополните решение задачи **6.1** так, чтобы иметь возможность передать Кота в написанный вами метод и протестируйте работоспособность решения на одном из ранее созданных котов.

**9.2 Поющие**. Необходимо написать метод принимающий переменное число поющих объектов и вызывающий пение у каждого из них. Доработать решение задач 8.8 и 7.10 так, что бы в данный метод можно было передать любую Птицу или Поющего Человека.

**9.3 Огонь по готовности**. Необходимо написать метод принимающий переменное число стреляющих предметов и стреляющий из каждого полученного предмета (в порядке очередности, т.е. сначала один выстрел из оружия1, затем один из оружия 2, затем снова из первого (если их всего два) и т.д.) до тех пор, пока у этого предмета не кончаться патроны. Необходимо учитывать, что стрелять могут также и объекты, не укладывающиеся в ранее разработанную автоматно-пистолетную логику.

Дополните решение задач **6.3**, **8.1** и **8.6** так, чтобы иметь возможность передать Пистолет, АвтоПистолет и Фейерверк в написанный вами метод. Передайте в него под одному объекту типа Пистолет, АвтоПистолет и Фейерверк и выполните метод. Обратите внимание, что Петарда не должна подходить для этого метода.

**9.4 Цветной Фейерверк**. Необходимо изменить сущность Фейерверк из задачи **8.6** таким образом, чтобы звуки во время стрельбы издавались бы не самим Фейерверком, а зависели от того, какой тип Заряда выстреливается в данный момент. Про Заряды неизвестно ничего, кроме того, что они могут взрываться таким образом, что ничего не принимают, не возвращают, выводят текст на экран и не относятся к какой-то конкретной архитектуре классов.

После этого разработайте Заряды трех типов:

1. Заряды, которые всегда издают звук “шшшш……бах!”
2. Заряды, котором можно указать цвет, например, для красного заряда будет “шшшш……красный бах!”
3. Заряды, котором можно указать мощность, чем выше мощность, тем больше букв ‘а’ в слове “бах”

Создавайте фейерверк, в который заряжено по два заряда каждого типа, и выстрелите им.

**9.5 Комментируемое.** Необходимо модифицировать решение, полученное в задаче **7.9** таким образом, чтобы комментирование можно было выполнять с использованием разных типов комментариев, без учета их местонахождения в структуре классов.

По завершению разработки выполните следующую задачу: замените стандартные Комментарии на HTML Комментарии, у которых текст помечен (с помощью html тегов) красным цветом в том случае если рейтинг Комментария отрицательный.

# Раздел 2. Шаблонные задачи.

**Студент и оценки 4. Command.**

В рамках создания информационной системы учебного заведения необходимо разработать сущность описывающую студента. Список требований к сущности совпадает с таковым в задаче **10.4**, однако предъявлен ряд дополнительных требований:

* Студенту можно изменить имя.
* Должна быть возможность отмены последнего изменения (как по оценкам, так и по имени). Отмену можно вызывать последовательно до тех пор, пока студент не придет к состоянию, которое имел непосредственно после инициализации.

**Студент и оценки 5. Стратегия.**

В рамках создания информационной системы учебного заведения необходимо разработать сущность описывающую студента. Список требований к сущности совпадает с таковым в задаче “**Студент и оценки 3”**, однако предъявлено дополнительное требование: на момент разработки класса студента не известно, что является корректными оценками, а что нет, в том числе не известно, будет ли это диапазон чисел, либо же неопределенный набор чисел: например 0, 5, 7. Эта информация будет известна к моменту инициализации объектов, однако процесс добавления значений все равно должен проверять корректность оценки относительно этой информации.

**Студент и оценки 6. Обобщения.**

В рамках создания информационной системы учебного заведения необходимо разработать сущность описывающую студента. Список требований к сущности совпадает с таковым в задаче “**Студент и оценки 5”**, однако предъявлено дополнительное требование: на момент описания класса Студента не известен тип данных оценок. Эта информация будет известна во время создания объектов и её необходимо поддерживать.

**Студент и оценки 7. Composite.**

В рамках создания информационной системы учебного заведения необходимо разработать сущность описывающую студента. Список требований к сущности совпадает с таковым в задаче “**Студент и оценки 3”**, однако оценки теперь представляют собой комплексную сущность. Например, в сущности студента хранятся оценки по всем предметам, а оценка одного предмета представляет собой набор оценок за контрольные работы, а оценка за контрольную работу, в свою очередь, состоит из набора оценок за отдельные вопросы в контрольной, и уже оценка за вопрос является числом. Реализация такой системы оценивания должна поддерживать гибкое и прозрачное настраивание уровня вложенности оценок и поддерживать операции: сумма всех оценок, среднее значение всех оценок, получение списка всех оценок. Для упрощения реализации, оценки можно считать неизменяемыми.

**Студент и оценки 8. Изменяй неизменяемое.**

В рамках создания информационной системы учебного заведения необходимо разработать сущность описывающую студента. Список требований к сущности совпадает с таковым в задаче “**Студент и оценки 7”**, однако в этой задаче необходимо продумать концепцию изменения оценок, записанных в студенте.

**Студент и оценки 9. Memento.**

В рамках создания информационной системы учебного заведения необходимо разработать сущность описывающую студента. Список требований к сущности совпадает с таковым в задаче “**Студент и оценки 3”**, однако предъявлен ряд дополнительных требований:

* Студенту можно изменить имя.
* У студента нельзя запросить список его оценок, но можно запросить среднее значение оценки
* У студента можно запросить “чекпоинт” – объект, имеющий метод “load”, выполнение которого приводит объект студента к состоянию, которое тот имел на момент формирования чекпоинта. Важно отметить, что ни чекпоинт, ни студент не должны раскрывать содержание списка оценок студента в любой момент времени.

**Студент и оценки 10. Builder.**

В рамках создания информационной системы учебного заведения необходимо разработать сущность описывающую студента. Список требований к сущности совпадает с таковым в задаче “**Студент и оценки 3”**, однако предъявлен ряд дополнительных требований:

* Студент имеет номер телефона в виде строки.
* Студент имеет номер телефона папы в виде строки.
* Студент имеет номер телефона мамы в виде строки.
* Студент имеет номер зачетки в виде числа.
* Студент имеет номер паспорта в виде числа.
* На все поля есть геттеры и сеттеры.
* Процесс инициализации студента может быть выполнен с указанием любой комбинации свойств студента по желанию пользователя. Например только имя и номер телефона. Или имя и номер телефона мамы и т.д.

**Студент и оценки 11. Оценки за предметы.**

В рамках создания информационной системы учебного заведения необходимо разработать сущность описывающую студента. Список требований к сущности совпадает с таковым в задаче “**Студент и оценки 3”**, однако предъявлен ряд дополнительных требований:

* Оценки выставляются за предмет
* За один предмет может быть не более одной оценки
* У студента можно получить либо просто список оценок, либо список пар “предмет-оценка”

**Студент и оценки 12. Типы оценок.**

В рамках создания информационной системы учебного заведения необходимо разработать сущность описывающую студента. Список требований к сущности совпадает с таковым в задаче “**Студент и оценки 11”**, однако предъявлен ряд дополнительных требований:

* Оценка выставляется за вид отчетности по предмету.
* Имеется несколько разных типов отчетности:
  + Экзамен (оценка от 2 до 5)
  + Зачет (статус зачтено\незачтено)
  + Курсовая работа (оценка от 2 до 5)
* Каждая из отчетностей помимо оценки имеет статус “сдано\несдано”
* По одному предмету может быть несколько видов отчетностей, но только одна оценка за один вид отчетности.
* Добавлять оценки студенту можно, изменять или удалять нельзя.
* У студента можно запросить полный перечень имеющихся на текущий момент пар “отчетность - оценка”, этот перечень может быть приведен к набору строк вида “предмет – текстоваяФормаОценки”
* Решение должно поддерживать добавление новых видов отчетности без изменения структуры сущности “студент”

**Студент и оценки 13. Стримы.**

В рамках создания информационной системы учебного заведения необходимо разработать сущность описывающую студента. Список требований к сущности совпадает с таковым в задаче “**Студент и оценки 12”**, однако предъявлено дополнительное требование: у студента должен быть метод, который возвращает список всех отчетностей по определенному критерию. Например: список всех сданных отчетностей по предмету “Информатика”, или список всех экзаменов и т.д. Метод должен поддерживать возможность изменения критерия формирования списка без изменения содержания метода.

**Студент и оценки 14. Декоратор.**

В процессе разработки информационной системы учебного заведения был разработан метод со следующей сигнатурой:

public static void checkGrades(Countable count);

В этой сигнатуре, Countable это интерфейс, имеющий метод int count(), который описывается как “метод возвращающий сумму всех оценок студента”. Необходимо передать в этот метод объект студента описанный в задаче “**Студент и оценки 3”**, при этом класс студента изменяться не может.

**Вахтер. Посетитель.**

При разработке информационной системы вуза потребовалось реализовать систему безопасности. В процессе анализа определены следующие классы:

* Человек. Любой неавторизованный пользователь.
* Студент. Наследуется от класса Человек. Имеет имя (возможно пустое) и номер студенческого билета (возможно пустой). Имеет геттеры, объект неизменяемый, поля инициализируются в конструкторе.
* Преподаватель. Наследуется от класса Человек. Имеет имя (возможно пустое), номер удостоверения (возможно пустое), должность (возможно пустое). Имеет геттеры, объект неизменяемый, поля инициализируются в конструкторе.

Кроме того, разработан класс Вахтер, выполняющий проверку прав пользователей на доступ к ресурсу. В классе должен иметься метод “checkUser” (с учетом возможных перегрузок) реализующий проверку прав доступа к ресурсу по следующей логике:

* При получении на вход Студента он проверяет заполненность имени и номера студенческого билета, и если они есть, то возвращает true, иначе false
* При получении на вход Преподавателя он проверяет заполненность имени, а также заполненность либо номера удостоверения, либо должности, при заполненности – возвращает true, иначе false.
* При получении на вход любого другого Человека возвращает false

Задача должна быть решена без использования проверок на тип объекта.

**Золушка 1. Посетитель.**

Злая мачеха рассыпала на полу объекты из трех банок:

* Банка с объектами класса Горох
* Банка с объектами класса Фасоль
* Банка с объектами класса Рис

В результате чего образовался List<Собирабельное>, включающий как рассыпанные объекты, так и мусор с пола.

Злая мачеха приказала Золушке разобрать все объекты обратно по банкам (с получением List<Горох>, List<Фасоль>, List<Рис>), и сообщить количество объектов в каждой банке. Однако, при этом, запретила Золушке любые проверки на тип объекта.

Представьте себя на месте доброй феи-крестной, и помогите Золушке вывести на консоль количество объектов в каждой банке в рамках заданных условий. Будучи феей вы можете по своему желанию создавать новые классы или менять имеющиеся.

**Золушка 2. Посетитель.**

Данная задача является непосредственным продолжением задачи “**Золушка 1**”. Вернувшаяся мачеха обнаружила, что все объекты разложены назад по банкам, после чего разозлилась, раскидала объекты назад по полу (с получением List<Собирабельное>), и заявила, что во время сбора необходимо было выводить на консоль текст “собираю горошек”, “собираю фасольку”, “собираю рис” в соответствующих случаях, и фразу “а это не беру” во всех остальных.

Оставаясь феей, реализуйте логику Собирабельных объектов так, что бы её не пришлось менять каждый раз, когда у мачехи появятся новые требования, а вся реализация была бы у Золушки.

**Голос. Адаптер.**

В процессе разработки некоей информационной системы был разработан класс Кот со следующими характеристиками:

* Имеет кличку (не пустое)
* Кличка устанавливается при инициализации кота
* Каждый кот умеет мяукать, метод ничего не возвращает, и не принимает, выводит на консоль текст состоящий из имени и слова “мяу”

Далее был реализован метод, принимающий объекты по ссылке Говорящий. Ссылка описывает только один метод без реализации: метод “говорить”, который ничего не принимает и ничего не возвращает. Передайте объект какого-либо кота по этой ссылке, не изменяя при этом класса кота.

**Караоке. Адаптер.**

В процессе разработки некоей информационной системы было разработано три сущности:

* Поющий. Сущность описывает объекты имеющие метод “петь”, который принимает строку и ничего не возвращает.
* ЧитающийРеп. Сущность описывает объекты имеющие метод “читатьРеп”, который принимает строку и ничего не возвращает.
* Караоке. Сущность описывается следующими характеристиками:
  + Имеет массив строк. Это тексты песен.
  + Имеет метод “исполнить”, принимающий Поющие объекты, и передающий в метод “петь” случайную строку из всего набора строк.

В дальнейшем появилась необходимость позволить петь в караоке также и всем ЧитающимРеп, однако, изменение данной сущности запрещено. Напишите код решающий данную задачу.

**Вес. Адаптер.**

В процессе разработки некоей информационной системы было разработано две сущности:

* Взвешенный. Сущность описывает метод public int getВес() без реализации.
* Человек. Сущность описывается:
  + Имя: строка, может быть пустой
  + Вес: вещественное число, обязателен, строго положительный, в килограммах
  + Человек реализует сущность Взвешенный

Далее, в отдельном классе, был разработан метод “сложитьВеса” принимающий объект типа List<Взвешенный>, в методе складываются веса всех объектов в списке и результат выводится на консоль. В качестве составляющей указанного списка передаются также и различные люди. К сожалению, оказалось, что метод “сложитьВеса” ожидал вес в тоннах.

Сделайте так, чтобы вес человека возвращался в тоннах, при этом класс человека необходимо оставить без изменений, наследоваться от него нельзя.

**Армия. Цепочка ответственности.**

В процессе разработки некоей информационной системы для армии возникла необходимость поддерживать возможность отдавать приказ по цепочке командования. В результате анализа была сформирована следующая модель:

* Выделена сущность Приказ, он описывается:
  + Текст: строка, неизменяемый
  + Уровень: целое число, неизменяемый
* Выделены варианты сущности Военный: Полковник, Капитан, Лейтенант, Рядовой.
  + Каждой сущности задано значение уровня ответственности: 10, 7, 4, 0 соответственно.
  + Каждый Военный имеет имя
  + Каждый Военный имеет в подчинении одного человека ниже его уровнем, кроме рядового, у которого не может быть подчиненных.
  + Каждый участник цепочки командования может “выполнитьПриказ”, для этого ему передается приказ, а результатом является информация о том, был ли выполнен приказ. При выполнении приказа описанные варианты Военных принимают решение: выполнить приказ самому или передать подчиненному. Приказ выполняется самостоятельно, если уровень приказа равен или выше уровня ответственности военного, либо отсутствуют подчиненные. В настоящий момент выполнение приказа представляет собой заглушку: конкретный исполнитель просто пишет текст на экран, что приказ такой-то выполнен тем-то.
  + Геттеры\сеттеры и конструкторы определяются по необходимости

По задумке, пользователь получает ссылку на какого-либо кто может исполнять приказ, и передаёт ему приказ. Далее этот приказ спускается по цепочке командования к тому, чей уровень соответствует приказу и он его выполняет. Система должна быть реализована таким образом, чтобы она соответствовала следующим требованиям:

* Цепочки командования могут создаваться пользователем произвольно, из тех исполнителей приказа, какие ему потребуется (с учетом соответствия правилам описанным выше)
* Дальнейшие исследования могут показать, что существуют и другие Военные, например Майор или Сержант. Встраивание этих новых сущностей не должно требовать изменения уже существующих.
* В цепочку командования могут быть встроены и гражданские специалисты, если они могут исполнять приказы, эту возможность необходимо поддерживать.
* Изменение процесса выполнения приказа у одной сущности не должно затрагивать другие. Например, если Полковник поменял свое поведение, то это не оказывает влияния на Рядового

**Студент и оценки 15: проверка знаний. Цепочка ответственности.**

В рамках создания информационной системы учебного заведения необходимо разработать концепцию проверки знаний студента. В результате анализа была сформирована следующая модель:

* Выделена сущность Студент, он описывается:
  + Имя: строка, неизменяемый
  + Оценки: список целых чисел, можно добавлять оценки, весь список оценок получить нельзя, но можно получить среднее значение оценки.
  + Знания: список строк, строки можно добавлять, можно получить весь список строк для чтения
  + Другие геттеры\сеттеры и конструкторы определяются по необходимости
* Выделена сущность Экзаменатор, он описывается:
  + Имя: строка, неизменяемый
  + Способ проверки знаний: принимает сущность у которой можно получить знания, и которой можно добавить оценку, ничего не возвращает.
  + Геттеры\сеттеры и конструкторы определяются по необходимости
  + Возможны дополнения, которые на текущий момент не определены

Основная идея приложения заключается в определении, неким образом, набора экзаменаторов, которые должны оценить студента. Этому набору в метод проверки знаний передается студент. Проверка выполняется каждым преподавателем строго по очереди, причем в случае, если какой-либо из преподавателей выставляет неудовлетворительную оценку, проверка немедленно заканчивается. Сама проверка представляет собой получение знаний студента и прогон его по какому-либо правилу, например, есть ли слово “полиморфизм” в знаниях студента. По результатам проверки студенту необходимо добавить оценку к уже имеющимся.

В рамках решения данной задачи следует создать 3-4 преподавателя с разными способами проверки знаний, создать студента с некоторыми знаниями, и прогнать его через процесс оценки, а результат вывести на экран.

Необходимо учесть некоторые требования:

* Нельзя менять сигнатуру метода проверки знаний: как принимаемый параметр, так и возвращаемый
* Необходимо инкапсулировать процесс передачи Студента от Экзаменатора к Экзаменатору.
* Необходимо держать в голове возможность применения данной системы не только для вуза, но и для других учебных заведений, в связи с чем система должна поддерживать проверку не просто преподавателями, а в широком смысле всеми, кто может проверять, а также со стороны оцениваемого: не только студента, но любого, кто пригоден для оценивания.

**Карате-Пацан. Команда.**

В процессе разработки некоей информационной системы был создана сущность Каратист, которая описывается:

* Имя: строка, неизменяемое
* Каратист умеет бить ногой. В результате вызова данного метода на экран выводится строка “Имя: бац!”
* Каратист умеет бить рукой. В результате вызова данного метода на экран выводится строка “Имя: кия!”
* Каратист умеет бить в прыжке. В результате вызова данного метода на экран выводится строка “Имя: вжух!”

Необходимо создать сущность, которая будет хранить комбинации ударов (например два раза ударить рукой и один раз ногой), и может применяться любым Каратистом. Сущность должна быть реализована так, чтобы она не требовала изменений при появлении новых видов ударов у Каратиста. Создайте комбинацию ударов: два раза рукой, один раз ногой и один раз в прыжке. Создайте Каратиста Никиту и сделайте так, чтобы он выполнил эту серию.

**Карате-Пацан 2. Команда.**

В процессе разработки некоей информационной системы был создана сущность Каратист, которая описывается:

* Имя: строка, неизменяемое
* Каратист умеет бить ногой. В результате вызова данного метода на экран выводится строка “Имя: бац!”
* Каратист умеет бить рукой. В результате вызова данного метода на экран выводится строка “Имя: кия!”
* Каратист умеет бить в прыжке. В результате вызова данного метода на экран выводится строка “Имя: вжух!”

Необходимо создать сущность, которая будет позволять сохранять действия группы Каратистов как единой команды, в которой каждый каратист наносит определенные удары в определенной последовательности. Далее должна быть необходимость уметь вызывать данную последовательность (например, сначала первый каратист бьет рукой, потом второй бьет ногой, потом снова первый бьет ногой, и в конце третий бьет в прыжке). Сущность должна быть реализована так, чтобы она не требовала изменений при появлении новых видов ударов у Каратиста.

**Построение Маршрута. Стратегия.** Необходимо улучшить решение, полученное в задаче **8.3**. Внесите такое изменение в полученную архитектуру классов, что бы в дальнейшем пользователь мог получать Маршрут построенный не по заранее известному алгоритму, а по такому который указал тот, кто использует данный класс, и при этом для таких изменений не требовалось бы вносить изменения в уже имеющиеся классы, а также не требовалось бы создание подклассов.

**Построение Маршрута 1. Стратегия.** Улучшите решение, полученное в задаче **8.1**. Внесите такое изменение в полученную архитектуру классов, что бы в дальнейшем пользователь мог получать Путь построенный по алгоритму указанному тем, кто использует данный класс, при этом для таких изменений не требовалось бы вносить изменения в уже имеющиеся классы, а также не требовалось бы создание подклассов.

**Маршрутный Интерфейс. Адаптер**. Решение, полученное в задаче **8.3**, в настоящее время используется в информационной системе логистической компании. В частности, есть несколько методов получающих экземпляр Маршрута с целью получения всех Городов, с учетом возможности смены города начала или конца маршрута.

Для выполнения задачи вам потребуется реализовать пример метода. Пусть он принимает на вход объект типа Маршрут и выводит найденный маршрут на экран (соответственно ничего не возвращает). Убедитесь, что в этот метод можно передать оба варианта ранее разработанных Маршрутов.

Итак, посмотрев в интернете лекции по комбинаторике, наш системный архитектор выяснил, что ранее разработанный способ организации карты городов – не единственное возможное решение. Оказывается, карту можно хранить в виде двумерного массива чисел (матрица смежности). Классы, которые будут реализовывать такую логику будут совершенно отличаться от имеющихся: МатричныйМаршрут будет принимать двумерный массив чисел и массив названий, для смены точек начала и конца, и иметь другой алгоритм поиска маршрута. При этом тоже возможна реализация как ранней, так и ленивой инициализации, что дает нам собственную иерархию классов.

Очевидно, нам потребуется общий интерфейс для обеих реализаций. Он будет иметь возможность изменять точки начала и конца как в виде строк, так и в виде Городов, возвращать массив Городов и строковое их представление при вызове метода приведения к строке.

Сделайте так, чтобы к объектам класса **8.3** можно было обращаться через этот интерфейс, не меняя при этом классов, разработанных в задаче **8.3**.

**Группировка картинок. Композит.** Вернитесь к задаче **8.2**. Обратите внимание на то, что все объекты можно сгруппировать и сдвинуть их местоположение. Необходимо реализовать такую сущность, которая позволит группировать все объекты, которые можно сдвинуть. Причем необходимо уметь группировать группы объектов, например:

1. Создайте круг, и три линии, и сгруппируйте их (группа1)
2. Создайте круг и квадрат и сгруппируйте их (группа2)
3. Сгруппируйте группу 2 и две новых линии (группа 3)
4. Сгруппируйте группы 3 и 1 (группа4)
5. Сдвиньте группу 4 на 12 вниз.