Лабораторная работа № **24**

«Разработка алгоритмов и программ с использованием наследования»

Объектно-ориентированное программирование (ООП) представляет из себя подход в программировании, при котором мы воспринимаем весь окружающий нас мир как совокупность конкретных объектов, поведение и свойства (характеристики) которых можно отделить от конкретных объектов и создать соответствующие абстрактные шаблонные образы – классы. Все то, что может делать класс – смогут делать и объекты типа этого класса после того, как они будут созданы конструктором данного класса. Характеристики класса (поля) будут у всех его объектов, причем в полях объектов будут записаны конкретные данные, в то время как в классе просто декларируются имена полей и их типы данных (но можно еще указать значение по умолчанию или указать константное значение). Класс – это описание будущих объектов типа данного класса, которые еще даже не созданы, но могут быть созданы сразу, как только описан соответствующий им класс.

Класс – это абстракция (шаблон, матрица) некоторой сущности предметной области. В программе создаются объекты – экземпляры класса (объекты класса).

Члены класса по умолчанию закрытые – компилятор присваивает им по умолчанию неявно в коде спецификатор доступа private, что значит, что к этим членам имеют доступ только члены данного класса и его дружественные friend-функции и friend-классы. Лучше декларировать члены класса с явным указанием в коде спецификаторов доступа перед ними: public, protected или private.

```
class ClassName
{
        спецификаторДоступа: тип членКласса;//после спецификатора доступа пишется ДВОЕТОЧИЕ
}
```

Член класса, объявленный после спецификатора доступа public доступен этому и всем другим классам, объектам и методу main() данной программы.

Член класса, объявленный после спецификатора доступа protected доступен только членам данного класса, его классам-наследникам и дружественным friend-функциям и friend-классам данного класса.

Член класса, объявленный после спецификатора доступа private доступен только членам данного класса и его дружественным friend-функциям и friend-классам.

Структуры struct и классы class идентичны с точки зрения их базовых возможностей, но по умолчанию неявно члены структуры имеют спецификатор доступа public, а члены класса – private. Структура обычно используется, если надо только хранить данные в полях, а класс используется, если объект должен не только хранить данные, но и имеет поведение (то есть методы – функции внутри класса). Структура и класс имеют конструктор без параметров по умолчанию и деструктор. В рамках ООП предпочтительнее писать классы, а не структуры.

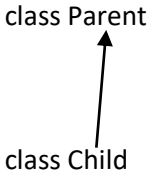
Таблица 1 – Демонстрация доступности членов класса, объявленных с определенным **спецификатором доступа**

Доступен ли член из	**public**	**protected**	**private**
Самого этого класса	**да**	**да**	**да**
Дружественных функций и дружественных классов	**да**	**да**	**да**
Классов-наследников (классов-потомков, дочерних классов)	**да**	**да**	**нет**
Извне (из функции main())	**да**	**нет**	**нет**

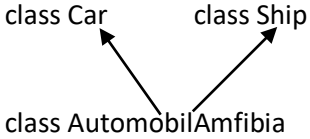
Программа, реализующая ООП должна соответствовать сразу трем основным правилам ОПП (принципам ООП, постулатам ООП). Эти принципы: **наследование**, **инкапсуляция** и Член класса, объявленный после спецификатора доступа **полиморфизм** (в литературе могут выделятся до 7 принципов ООП, которые сводятся к данным трем основным принципам ООП).

Инкапсуляция – это один из трех основных принципов ООП, предполагающий сокрытие деталей реализации класса от доступа извне данного класса (и, соответственно, извне данного объекта класса) и недопущение изменения внутреннего состояния объекта класса (значений, хранимых в его полях) без ведома самого данного объекта, а также недопущение изменений в состоянии объекта класса, которые приведут к его некорректной работе и/или уничтожению (не путать с деструктором, который является членом экземпляра класса и для того и существует, чтобы очищать оперативную память от данного объекта класса, уничтожая тем самым этот объект). Также инкапсуляция предполагает, что объект должен создаваться не просто в main'е, а внутри чего-то, внутри оболочки или хранящего контейнера (инкапсуляция – in capsule – в капсуле, внутри защищающей капсулы).

Наследование – это один из трех основных принципов ООП, предполагающий создание иерархии классов (объектов классов), в которой объекты-потомки наследуют все члены (свойства и поведение) своих классов-предков. Описывать заново явно в коде унаследованные от родителей члены у потомков не нужно. Потомки имеют унаследованные и свои «личные» свойства и поведения. В отличие от языка СиШарпа, в С++ разрешено множественное наследование: потомок может иметь множество непосредственных родителей (2 и больше).

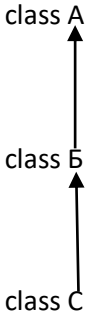


«Обычное» наследование: у класса-потомка Child есть один класс-предок Parent.



Множественное наследование: класс АвтомобильАмфибия имеет непосредственных двух предков – Автомобиль и Корабль. Запомните, что при наследовании стрелка указывает **ОТ** потомка **НА** предка (наконечник стрелки наследования указывает **НА** родительский класс).

Важнейшим принципом объектно-ориентированного программирования (ООП), реализованным в С++, является наследование. Класс, который наследуется, называется базовым классом (родительским классом, классом-предком), а наследующий класс – производным классом (дочерним классом, классом-потомком). Еще говорят, что дочерний класс наследует родительскому классу. В цепочке наследования может быть больше двух классов, то есть родительскому классу А наследует дочерний класс Б, а самому классу Б наследует класс С («внучатый класс»). В этой цепочке всего 3 класса, но можно в иерархию наследования включить больше классов, «удлинив» цепочку наследования. Заметьте, что класс Б является дочерним по отношению к классу А и при этом сам класс Б является родительским по отношению к классу С. Также класс А транзиторно, то есть через класс Б, является родительским по отношению к классу С. Класс С является дочерним по отношению к классу Б и одновременно класс С является дочерним по отношению к классу А (транзиторно через класс Б). Указателю на объект базового класса можно присвоить значение указателя на объект производного класса, но НЕ наоборот. Указателю типа родительского класса можно присвоить адрес объекта типа дочернего класса, но нельзя указателю типа дочернего класса присвоить адрес объекта типа родительского класса (это приведет к ошибке).



Иерархия наследования: класс С наследует классу Б, а класс Б наследует классу А, поэтому класс С транзиторно через класс Б наследует классу А тоже.

Есть **спецификаторы доступа** к членам класса: **public**, **protected** и **private**.

Члены класса, объявленные со спецификатором доступа **public** (открытый, публичный, общедоступный) видны другим классам и объектам, доступны для обращения к ним в коде (например, из функции main()) и, если это не константы, доступны для изменения, присвоения им новых значений. Неявно, по умолчанию, спецификатор доступа public подставляется компилятором перед каждым членом структуры struct (потому-то члены структуры struct доступны для изменения из main'а, конечно, если вы явно перед ними не напишите закрытый спецификатор private).

Члены класса, объявленные со спецификатором доступа **private** (закрытый, частный, личный) видны всем членам данного класса (данного экземпляра класса), но НЕ видны любым другим объектам, классам, недоступны для обращения и редактирования из функции main(). Спецификатор доступа private неявно по умолчанию подставляется компилятором перед каждым членом любого класса, но лучше писать этот спецификатор доступа явно в коде класса, чтобы визуально видеть его и проще было искать ошибки в коде. Спецификатор доступа private обеспечивает максимально возможную степень защищенности членов класса, бОльшую, чем спецификатор доступа protected и, конечно, public.

Члены класса, объявленные со спецификатором доступа **protected** (защищенный, полузакрытый) доступны для членов данного класса (экземпляра данного класса) и его классов-наследников, но **не**доступны для всех других объектов, классов и метода main(). Если нашему классу не наследует ни один другой класс, то спецификатор private лучше всего подходит для членов нашего класса, которые мы хотим защитить от доступа извне, но если нашему классу будет наследовать хотя бы один класс, то нужно (выгодно) для членов нашего класса, которые мы хотим закрыть для доступа извне, назначить спецификатор доступа protected, поскольку он и это позволит сделать, но еще обеспечит видимость этих членов нашего класса для его классов-наследников, то есть такие же члены окажутся в классах-наследниках нашего класса и будут там доступны для использования, но не в main'е напрямую. То есть защита от доступа извне есть и работает, а потому мы спецификатором protected достигаем сразу двух целей: и защита от доступа извне, но при этом без создания проблем для наследования.

Спецификаторы доступа используются и при записи наследования, в этом случае они **пишутся после двоеточия** перед именем родительского класса и становятся как бы фильтром, через который проходят все члены родительского класса при их переходе в дочерний класс. Если после двоеточия не указать спецификатор наследования, то по умолчанию неявно компилятор подставит спецификатор private, являющийся очень жестким фильтром, а потому обычно не применяемым при наследовании (удобнее явно написать более «мягкий» спецификатор наследования). При этом действует правило, что члену дочернего класса присваивается наиболее строгий (наиболее «закрытый») спецификатор из двух, к нему относящихся (его собственного спецификатора доступа и спецификатора наследования, написанного после двоеточия при объявлении дочернего класса). Например, если за двоеточием написан спецификатор наследования protected, то проходящий через него public член родительского класса приобретет самый строгий спецификатор из этих двух, то есть protected; если за двоеточием написан спецификатор наследования public, то проходящий через него protected член родительского класса сохранит свой спецификатор protected, потому что он более строгий (более «закрытый»), чем спецификатор наследования public.

Все члены родительского класса из разделов public и protected наследуются, а члены из раздела private – нет. Члены раздела protected являются частными (закрытыми для доступа извне) для базового и производного классов (то есть доступны только этим классам, НО не другим классам). При наследовании базового класса к его членам объявляется спецификатор доступа (public, protected и private), который отфильтрует, какие именно члены родительского класса попадут к дочернему классу и дальше по ветке наследования (если есть еще и «внучатые» классы-наследники).

Синтаксис объявления дочернего класса:

```
class имяДочернегоКласса: спецификаторДоступа имяРодительскогоКласса
{
        /*здесь пишем члены дочернего класса, которые в нем создаются, а НЕ наследуются от родительского
        класса, поскольку унаследованные от родительского класса члены появляются в дочернем классе
        автоматически, по умолчанию, при этом они появятся в нем неявно (невидимо в коде)*/
};
```

Конструктор производного класса может явно вызвать конструктор базового класса. В коде это выглядит как делегирующий конструктор (конструктор, который делегирует часть или всю свою работу по инициализации полей другому конструктору, вызывая тот конструктор после символа ДВОЕТОЧИЯ:

```
КонструкторДочернегоКласса(тип1 аргумент1, тип2 аргумент2, тип3 аргумент3) : КонструкторРодительскогоКласса (аргумент1, аргумент2)
{
        поле3 = аргумент3;
}
```

В этом коде конструктор дочернего класса вызывает конструктор родительского класса, чтобы тот проинициализировал 2 поля, которые дочерний класс унаследовал от родительского класса. Конструктор дочернего класса должен принять все аргументы, в том числе и те, которые он передаст конструктору родительского класса (в коде это аргумент1 и аргумент2). Но в дочернем классе есть «свое» поле3, которого нет в родительском классе и которое, соответственно, в принципе не может быть проинициализировано конструктором родительского класса, поскольку конструктор родительского класса не может знать о существовании этого поля (родительский класс не знает, если ли у него наследники и какие у них члены, а дочерние классы знают о существовании их родительских классов и всех **не**private членов родительских классов). Конструкторы и деструктор не наследуются, но их можно явно вызвать в коде дочерних классов, что мы тут и делаем. Итак, конструктор дочернего класса передает аргумент1 и аргумент2 в конструктор родительского класса, чтобы тот проинициализировал ими те два поля, которые унаследованы дочерним классом от родительского класса, а свое «личное, новое, неунаследованное» поле3 конструктор дочернего класса инициализирует значением сам (значением аргумента3).

Спецификатор доступа базового класса при наследовании может принимать одно из трех значений: private (по умолчанию), protected или public. Если спецификатор наследования принимает значение public, то все члены разделов public и protected базового класса становятся членами разделов public и protected производного класса. Если спецификатор доступа имеет значение private, то все члены разделов public и protected становятся членами раздела private производного класса (то есть в дочернем классе унаследованные им члены еще доступны как private, но от него их уже никто не сможет унаследовать). Если спецификатор доступа имеет значение protected, то все члены разделов public и protected становятся членами раздела protected производного класса. Помним, что члены раздела private родительского класса вообще НЕ наследуются никаким его дочерним классом при любом спецификаторе наследования.

Конструкторы, деструкторы и перегруженные операторы присваивания родительских классов не наследуются, однако при создании объектов дочерних классов конструкторы родительских классов **выполняются** в порядке наследования (то есть сначала работает родительский конструктор и потом дочерний конструктор для создания объекта дочернего класса), а деструкторы срабатывают в обратном порядке при уничтожении объекта дочернего класса (то есть сначала сработает деструктор дочернего класса и потом деструктор родительского класса). При необходимости передачи аргументов конструктору базового класса из производного класса используется синтаксис

делегирующих конструкторов (то есть когда данный конструктор вызывает сначала другой конструктор и пользуется результатами его работы, доделывая только то, что не сделал вызванный конструктор):

конструкторДочернегоКласса(тип аргумент0, тип аргумент1, тип аргумент2) : конструкторРодительскогоКласса(аргумент0, аргумент1)

{

/*тело конструктора дочернего класса, где инициализируем только те поля, которые остались НЕ проинициализированными после работы конструктора родительского класса (то есть те поля, которые объявлены (созданы) в дочернем классе и о существовании которых родительский класс (и, соответственно, родительский конструктор) НЕ знает.

Например:

поле2 = аргумент2;*/

}

Конструктор дочернего класса описывает типы и имена принимаемых аргументов, причем он должен принимать список аргументов и для работы конструктора родительского класса и собственно своего конструктора. Далее после двоеточия (:) вызывается по имени конструктор родительского класса, куда передаем нужные ему аргументы (их типы здесь писать НЕ нужно, так как этот «родительский» конструктор уже должен быть полностью описан программистом в родительском классе, а здесь он просто вызывается с фактическими входными параметрами).

Если нужно из тела дочернего класса вызвать метод родительского класса, который **переопределен** в дочернем классе с таким же именем и сигнатурой (входными параметрами), то пишем:

имяРодительскогоКласса :: имяМетода(входные параметры);

, причем мы вызываем готовый метод из родительского класса, поэтому передаем в него фактические параметры (имена переменных, а не описание «тип имяПараметра, тип2 имяПараметра2». Если написать просто имяМетода(входные параметры), то мы вызовем метод дочернего класса с таким же именем и сигнатурой, а потому для вызова «старого» переопределенного метода из родительского класса нужно явно указать имя родительского класса, поскольку в дочернем классе у него имеется метод-тезка с таким же именем. Говорят, что метод в дочернем классе закрыл одноименный ему метод из родительского класса, поскольку компилятор в локальной области видит в первую очередь «местный» элемент, а не глобальный элемент с точно таким же именем и типом.

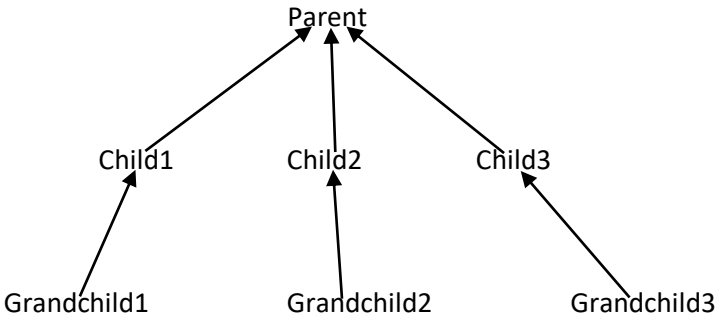
Итак, наследование – это определение производного (дочернего) класса, который может обращаться ко всем членам базового (родительского) класса, за исключением private-членов базового класса. Базовый класс называют также родительский класс, надкласс, предок, наследуемый класс, а производный класс может быть назван дочерним классом, потомком, подклассом, наследующим классом.

В производном классе надо явно определять его конструкторы, деструктор и перегруженные операторы присваивания ввиду того, что все они не наследуются от базового класса. Но всех их можно вызвать явно в коде производного класса при определении его конструкторов, деструктора и перегрузке оператора присваивания.

Перегруженные методы. Если в классе-потомке объявить метод с точно таким же именем и сигнатурой (списком входных параметров и их типов), как и у метода в базовом классе, то в дочернем классе и, соответственно, у всех его объектов по такому имени будет вызван собственный метод именно самого дочернего класса, поскольку он скрывает одноименный метод родительского класса. Такой метод называется перегруженным. Но в дочернем классе можно вызвать и собственно «скрытый» метод родительского класса, явно в коде указав пространство имен родительского класса с оператором :: (двойное двоеточие) и именем метода родительского класса, иначе (если написать только имя метода) компилятор вызовет метод дочернего класса, поскольку он закрывает одноименный метод родительского класса. Синтаксис:

РодительскийКласс :: Метод(аргумент1, аргумент2, аргумент3);

Попробуем использовать все спецификаторы доступа и наследования в иерархии наследования, чтобы увидеть, какой они окажут эффект в плане доступа к членам классов (членам экземпляров классов). Программа на следующей странице реализует нижеследующую иерархию наследования (диаграмму классов).



Иерархия наследования (диаграмма классов без детализации членов классов)

Задание 1. Наберите код программы и ответьте на вопросы в комментариях:

```cpp
#include <iostream>
#include <Windows.h>
using namespace std;

class Parent
{
public:
    int a;
    Parent(int a0, double b0, char c0)
    {
        a = a0;
        b = b0;
        c = c0;
    }
protected:
    double b;
private://спецификатор доступа private закрывает, скрывает члены своего раздела от всех других классов, даже классов-наследников
    char c;
};

class Child1 : public Parent
{
public:
    bool d;
public:
    Child1(int a1, double b1, char c1, bool d1) : Parent(a1, b1, c1)
    {
        d = d1;
        cout << Parent::a << ' ' << Parent::b << endl;//a поле private: char c; недоступно в классе Child1, поскольку оно личное, закрытое, доступно только внутри класса Parent
    }
};

class Child2 : protected Parent
{
protected:
    float e;
public:
    Child2(int a2, double b2, char c2, float e2) : Parent(a2, b2, c2)
    {
        e = e2;
        cout << Parent::a << ' ' << Parent::b << endl;//a поле private: char c; недоступно в классе Child2, поскольку оно личное, закрытое, доступно только внутри класса Parent
    }
};

class Child3 : private Parent
{
private:
    short f;
public:
    Child3(int a3, double b3, char c3, short f3) : Parent(a3, b3, c3)
    {
        f = f3;
        cout << Parent::a << ' ' << Parent::b << endl;//a поле private: char c; недоступно в классе Child3, поскольку оно личное, закрытое, доступно только внутри класса Parent
    }
};

class Grandchild1 : public Child1
{
    public:
    Grandchild1(int a4, double b4, char c4, bool d4):Child1(a4, b4, c4, d4)
    {
        cout << Child1::a << ' ' << Child1::b << ' ' << Child1::d << ' ' << Parent::a << ' ' << Parent::b << endl;//Child1::a и Child1::d имеют спецификаторы public, а Child1::b
        //имеет спецификатор protected. Почему?
    }
};
class Grandchild2 : protected Child2
{
public:
    Grandchild2(int a5, double b5, char c5, float e5) : Child2(a5, b5, c5, e5)
    {
        cout << Child2::a << ' ' << Child2::b << ' ' << Child2::e << ' ' << Parent::a << ' ' << Parent::b << endl;//все эти поя имеют тут спецификатор protected. Почему?
    }
};

class Grandchild3 : private Child3
{
public:
    Grandchild3(int a5, double b5, char c5, short f5) : Child3(a5, b5, c5, f5)
    {
        //все поля класса Child3 и Parent недоступны
    }
};

int main()
{
    Parent p(1, 2.2, 'A');
    p.a = 0;
    Child1 c(2, 3.3, 'B', 1);
    c.a = 1;
    c.d = false;
    Grandchild1 g(3, 4.4, 'C', 0);
    g.a = 2;
    g.d = true;
    Child2 c2(4, 5.5, 'D', 4.9);//не доступно ни одного члена объекта c2. Почему?
    Child3 c3(5, 6.6, 'E', 4);//не доступно ни одного члена объекта c3. Почему?
    Grandchild2 g2(6, 7.7, 'F', 6.987);//не доступно ни одного члена объекта g2. Почему?
    Grandchild3 g3(7, 8.8, 'G', 6);//не доступно ни одного члена объекта g3. Почему?
    system("pause");
    return 0;
}
```

```
Выбрать Консоль отладки Microsoft Visual Studio
2 3.3
3 4.4
3 4.4 0 3 4.4
4 5.5
5 6.6
6 7.7
6 7.7 6.987 6 7.7
7 8.8
Для продолжения нажмите любую клавишу . . .

D:\2019\Labs\x64\Debug\Lab6_8.exe (процесс 7320) завершил работу с кодом 0
.
Чтобы автоматически закрывать консоль при остановке отладки, включите пара
метр "Сервис" ->"Параметры" ->"Отладка" -> "Автоматически закрыть консоль
при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно…
```

Задания 2.

Сделать **1** программу с заданными в вашем варианте задания классами, реализовав подходящее наследование. В функции main() создать по 2 экземпляра каждого класса (динамический и нединамический) и вызвать у них все их методы.

Создать по массиву с элементами типа вашего класса и в цикле вызвать у них все их методы.

Создать массив типа родительского класса и заполнить его экземплярами как его самого, так и его дочерних и «внучатых» классов. Вызвать в цикле все методы у каждого элемента такого массива.

Номер по журналу группы	**Задание**
1	Создать базовый класс ТОЧКА с protected полями вещественного типа X и Y (это координаты точки по оси X и Y), public методом show(), который печатает значения всех полей экземпляра данного класса и конструктором, принимающим 2 параметра для инициализации полей экземпляра класса. От класса ТОЧКА унаследуйте класс ЛИНИЯ, в котором кроме полей X и Y будут еще поля X1 и Y1, так как линия имеет координаты двух своих крайних точек. Создайте в классе ЛИНИЯ конструктор с параметрами, который вызывает конструктор родительского класса, а сам доинициализирует остальные поля своего экземпляра класса. Создайте в классе ЛИНИЯ public метод show(), который печатает содержимое полей экземпляра своего класса, вызывая метод show() родительского класса, и допечатывая значения остальных полей. Создайте в классе ЛИНИЯ public метод length(), вычисляющий длину линии по формуле: L = sqrt((X1 – X2)^2 + (Y1 – Y2)^2). От класса ЛИНИЯ унаследуйте класс, подходящий по логике наследования, со своим конструктором, методом show(), прочими членами.
2	Создать базовый класс ТОЧКА с protected полями вещественного типа X и Y (это координаты точки по оси X и Y), public методом show(), который печатает значения всех полей экземпляра данного класса и конструктором, принимающим 2 параметра для инициализации полей экземпляра класса. От класса ТОЧКА унаследуйте класс ОКРУЖНОСТЬ, в котором точка с координатами X и Y – это центр окружности. В классе ОКРУЖНОСТЬ создайте protected вещественное поле радиус (длина радиуса), public метод perimetr(), вычисляющий периметр окружности по формуле: P = 2*Pi*R. Создайте в классе ОКРУЖНОСТЬ конструктор с параметрами, который вызывает конструктор родительского класса, а сам доинициализирует остальные поля своего экземпляра класса. Создайте в классе ОКРУЖНОСТЬ public метод show(), который печатает содержимое полей экземпляра своего класса, вызывая метод show() родительского класса, и допечатывая значения остальных полей. От класса ОКРУЖНОСТЬ унаследуйте класс, подходящий по логике наследования, со своим конструктором, методом show(), прочими членами.
3	Создать базовый класс ТОЧКА с protected полями вещественного типа X и Y (это координаты точки по оси X и Y), public методом show(), который печатает значения всех полей экземпляра данного класса и конструктором, принимающим 2 параметра для инициализации полей экземпляра класса. От класса ТОЧКА унаследуйте класс ОКРУЖНОСТЬ, в котором точка с координатами X и Y – это центр окружности. В классе ОКРУЖНОСТЬ создайте protected вещественное поле радиус (длина радиуса), public метод square(), вычисляющий площадь внутри окружности по формуле: S = Pi*R^2. Создайте в классе ОКРУЖНОСТЬ конструктор с параметрами, который вызывает конструктор родительского класса, а сам доинициализирует остальные поля своего экземпляра класса. Создайте в классе ОКРУЖНОСТЬ public метод show(), который печатает содержимое полей экземпляра своего класса, вызывая метод show() родительского класса, и допечатывая значения остальных полей. От класса ОКРУЖНОСТЬ унаследуйте класс, подходящий по логике наследования, со своим конструктором, методом show(), прочими членами.
4	Создать базовый класс ТОЧКА с protected полями вещественного типа X и Y (это координаты точки по оси X и Y), public методом show(), который печатает значения всех полей экземпляра данного класса и конструктором, принимающим 2 параметра для инициализации полей экземпляра класса. От класса ТОЧКА унаследуйте класс ОКРУЖНОСТЬ, в котором точка с координатами X и Y – это центр окружности. В классе ОКРУЖНОСТЬ создайте protected вещественное поле диаметр (длина диаметра), public метод perimetr(), вычисляющий периметр окружности по формуле: P = Pi*D. Создайте в классе ОКРУЖНОСТЬ конструктор с параметрами, который вызывает конструктор родительского класса, а сам доинициализирует остальные поля своего экземпляра класса. Создайте в классе ОКРУЖНОСТЬ public метод show(), который печатает содержимое полей экземпляра своего класса, вызывая метод show() родительского класса, и допечатывая значения остальных полей. От класса ОКРУЖНОСТЬ унаследуйте класс, подходящий по логике наследования, со своим конструктором, методом show(), прочими членами.
5	Создать базовый класс ТОЧКА с protected полями вещественного типа X и Y (это координаты точки по оси X и Y), public методом show(), который печатает значения всех полей экземпляра данного класса и

	конструктором, принимающим 2 параметра для инициализации полей экземпляра класса. От класса ТОЧКА унаследуйте класс ОКРУЖНОСТЬ, в котором точка с координатами X и Y – это центр окружности. В классе ОКРУЖНОСТЬ создайте protected вещественное поле диаметр (длина диаметра), public метод square(), вычисляющий площадь внутри окружности по формуле: S = (Pi*D^2) / 4. Создайте в классе ОКРУЖНОСТЬ конструктор с параметрами, который вызывает конструктор родительского класса, а сам доинициализирует остальные поля своего экземпляра класса. Создайте в классе ОКРУЖНОСТЬ public метод show(), который печатает содержимое полей экземпляра своего класса, вызывая метод show() родительского класса, и допечатывая значения остальных полей. От класса ОКРУЖНОСТЬ унаследуйте класс, подходящий по логике наследования, со своим конструктором, методом show(), прочими членами.
6	Создать базовый класс ЛИНИЯ с protected полем вещественного типа L (это длина линии), public методом show(), который печатает значения всех полей экземпляра данного класса и конструктором, принимающим 1 параметр для инициализации полей экземпляра класса. От класса ЛИНИЯ унаследуйте класс КВАДРАТ, в котором кроме поля L будут еще поля S (площадь квадрата) и P (периметр квадрата). Создайте в классе КВАДРАТ конструктор с параметрами, который вызывает конструктор родительского класса, а сам доинициализирует остальные поля своего экземпляра класса, вызывая нижеописанные методы. Создайте в классе КВАДРАТ public метод show(), который печатает содержимое полей экземпляра своего класса, вызывая метод show() родительского класса, и допечатывая значения остальных полей. Создайте в классе КВАДРАТ public методы perimetr(), вычисляющий периметр квадрата по формуле (P = 4 * L) и square(), вычисляющий площадь квадрата по формуле (S = L^2). От класса КВАДРАТ унаследуйте класс, подходящий по логике наследования, со своим конструктором, методом show(), прочими членами.
7	Создать базовый класс ЛИНИЯ с protected полем вещественного типа L (это длина линии), public методом show(), который печатает значения всех полей экземпляра данного класса и конструктором, принимающим 1 параметр для инициализации полей экземпляра класса. От класса ЛИНИЯ унаследуйте класс ПРЯМОУГОЛЬНИК, в котором кроме поля L (одна сторона) будет еще protected поле L2 (вторая сторона). Создайте в классе ПРЯМОУГОЛЬНИК конструктор с параметрами, который вызывает конструктор родительского класса, а сам доинициализирует остальные поля своего экземпляра класса. Создайте в классе ПРЯМОУГОЛЬНИК public метод show(), который печатает содержимое полей экземпляра своего класса, вызывая метод show() родительского класса, и допечатывая значения остальных полей. Создайте в классе ПРЯМОУГОЛЬНИК public методы perimetr(), вычисляющий периметр прямоугольника по формуле (P = 2 * (L1 + L2)) и square(), вычисляющий площадь прямоугольника по формуле (S = L1 * L2). От класса ПРЯМОУГОЛЬНИК унаследуйте класс, подходящий по логике наследования, со своим конструктором, методом show(), прочими членами.
8	Создать базовый класс ЛИНИЯ с protected полем вещественного типа L (это длина линии), public методом show(), который печатает значения всех полей экземпляра данного класса и конструктором, принимающим 1 параметр для инициализации полей экземпляра класса. От класса ЛИНИЯ унаследуйте класс РОМБ, в котором кроме поля L будет еще поле H (высота ромба). Создайте в классе РОМБ конструктор с параметрами, который вызывает конструктор родительского класса, а сам доинициализирует остальные поля своего экземпляра класса. Создайте в классе РОМБ public метод show(), который печатает содержимое полей экземпляра своего класса, вызывая метод show() родительского класса, и допечатывая значения остальных полей. Создайте в классе РОМБ public методы perimetr(), вычисляющий периметр ромба по формуле (P = 4 * L) и square(), вычисляющий площадь ромба по формуле (S = L * H). От класса РОМБ унаследуйте класс, подходящий по логике наследования, со своим конструктором, методом show(), прочими членами.
9	Создать базовый класс ЛИНИЯ с protected полем вещественного типа L (это длина линии), public методом show(), который печатает значения всех полей экземпляра данного класса и конструктором, принимающим 1 параметр для инициализации полей экземпляра класса. От класса ЛИНИЯ унаследуйте класс ПАРАЛЛЕЛОГРАММ, в котором кроме поля L будут еще protected поля L2 (вторая сторона параллелограмма) и H (высота параллелограмма). Создайте в классе ПАРАЛЛЕЛОГРАММ конструктор с параметрами, который вызывает конструктор родительского класса, а сам доинициализирует остальные поля своего экземпляра класса. Создайте в классе ПАРАЛЛЕЛОГРАММ public метод show(), который печатает содержимое полей экземпляра своего класса, вызывая метод show() родительского класса, и допечатывая значения остальных полей. Создайте в классе ПАРАЛЛЕЛОГРАММ public методы perimetr(), вычисляющий периметр параллелограмма по формуле (P = 2 * (L1 + L2)) и square(), вычисляющий площадь параллелограмма по формуле (S = L * H).

	От класса ПАРАЛЛЕЛОГРАММ унаследуйте класс, подходящий по логике наследования, со своим конструктором, методом show(), прочими членами.
10	Создать базовый класс ЛИНИЯ с protected полем вещественного типа L (это длина линии), public методом show(), который печатает значения всех полей экземпляра данного класса и конструктором, принимающим 1 параметр для инициализации полей экземпляра класса. От класса ЛИНИЯ унаследуйте класс ТРЕУГОЛЬНИК, в котором кроме поля L будет еще поле H (высота треугольника). Создайте в классе ТРЕУГОЛЬНИК конструктор с параметрами, который вызывает конструктор родительского класса, а сам доинициализирует остальные поля своего экземпляра класса. Создайте в классе ТРЕУГОЛЬНИК public метод show(), который печатает содержимое полей экземпляра своего класса, вызывая метод show() родительского класса, и допечатывая значения остальных полей. Создайте в классе ТРЕУГОЛЬНИК public метод square(), вычисляющий площадь треугольника по формуле (S = (L * H) / 2). От класса ТРЕУГОЛЬНИК унаследуйте класс, подходящий по логике наследования, со своим конструктором, методом show(), прочими членами.
11	Создать базовый класс ЛИНИЯ с protected полем вещественного типа L (это длина линии), public методом show(), который печатает значения всех полей экземпляра данного класса и конструктором, принимающим 1 параметр для инициализации полей экземпляра класса. От класса ЛИНИЯ унаследуйте класс ТРАПЕЦИЯ, в котором кроме поля L будут еще protected поля L2 (вторая сторона трапеции) H (высота трапеции). Создайте в классе ТРАПЕЦИЯ конструктор с параметрами, который вызывает конструктор родительского класса, а сам доинициализирует остальные поля своего экземпляра класса. Создайте в классе ТРАПЕЦИЯ public метод show(), который печатает содержимое полей экземпляра своего класса, вызывая метод show() родительского класса, и допечатывая значения остальных полей. Создайте в классе ТРАПЕЦИЯ public метод square(), вычисляющий площадь трапеции по формуле (S = ((L1 + L2) * H) / 2). От класса ТРАПЕЦИЯ унаследуйте класс, подходящий по логике наследования, со своим конструктором, методом show(), прочими членами.
12	Создать базовый класс ЛИНИЯ с protected полем вещественного типа L (это длина линии), public методом show(), который печатает значения всех полей экземпляра данного класса и конструктором, принимающим 1 параметр для инициализации полей экземпляра класса. От класса ЛИНИЯ унаследуйте класс ШЕСТИУГОЛЬНИК, в котором кроме поля L будет еще protected поле H (высота (радиус) шестиугольника). Создайте в классе ШЕСТИУГОЛЬНИК конструктор с параметрами, который вызывает конструктор родительского класса, а сам доинициализирует остальные поля своего экземпляра класса. Создайте в классе ШЕСТИУГОЛЬНИК public метод show(), который печатает содержимое полей экземпляра своего класса, вызывая метод show() родительского класса, и допечатывая значения остальных полей. Создайте в классе ШЕСТИУГОЛЬНИК public методы perimetr(), вычисляющий периметр шестиугольника по формуле (P = 6 * L) и square(), вычисляющий площадь шестиугольника по формуле (S = 3 * L * H). От класса ШЕСТИУГОЛЬНИК унаследуйте класс, подходящий по логике наследования, со своим конструктором, методом show(), прочими членами.
13	Создать базовый класс ТОЧКА с protected полями вещественного типа X и Y (это координаты точки по оси X и Y), public методом show(), который печатает значения всех полей экземпляра данного класса и конструктором, принимающим 2 параметра для инициализации полей экземпляра класса. От класса ТОЧКА унаследуйте класс ОКРУЖНОСТЬ, в котором точка с координатами X и Y – это центр окружности. В классе ОКРУЖНОСТЬ создайте protected вещественное поле радиус (длина радиуса), public метод square(), вычисляющий площадь внутри окружности по формуле: S = Pi*R^2. Создайте в классе ОКРУЖНОСТЬ конструктор с параметрами, который вызывает конструктор родительского класса, а сам доинициализирует остальные поля своего экземпляра класса. Создайте в классе ОКРУЖНОСТЬ public метод show(), который печатает содержимое полей экземпляра своего класса, вызывая метод show() родительского класса, и допечатывая значения остальных полей. От класса ОКРУЖНОСТЬ унаследуйте класс СЕКТОР окружности, со своим protected полем A (угол сектора окружности). Создайте в классе СЕКТОР конструктор с параметрами, который вызывает конструктор родительского класса, а сам доинициализирует остальные поля своего экземпляра класса. Создайте в классе СЕКТОР public метод show(), который печатает содержимое полей экземпляра своего класса, вызывая метод show() родительского класса, и допечатывая значения остальных полей. Создайте в классе СЕКТОР public метод square(), вычисляющий площадь сектора по формуле S = Pi*R^2*A / 180.
14	Создать базовый класс ТОЧКА с protected полями вещественного типа X и Y (это координаты точки по оси X и Y), public методом show(), который печатает значения всех полей экземпляра данного класса и конструктором, принимающим 2 параметра для инициализации полей экземпляра класса. От класса ТОЧКА унаследуйте класс ОКРУЖНОСТЬ, в котором точка с координатами X и Y – это центр

	окружности. В классе ОКРУЖНОСТЬ создайте protected вещественное поле радиус (длина радиуса), public метод square(), вычисляющий площадь внутри окружности по формуле: S = Pi*R^2. Создайте в классе ОКРУЖНОСТЬ конструктор с параметрами, который вызывает конструктор родительского класса, а сам доинициализирует остальные поля своего экземпляра класса. Создайте в классе ОКРУЖНОСТЬ public метод show(), который печатает содержимое полей экземпляра своего класса, вызывая метод show() родительского класса, и допечатывая значения остальных полей. От класса ОКРУЖНОСТЬ унаследуйте класс СЕГМЕНТ окружности, со своим protected полем A (угол сегмента окружности). Создайте в классе СЕГМЕНТ конструктор с параметрами, который вызывает конструктор родительского класса, а сам доинициализирует остальные поля своего экземпляра класса. Создайте в классе СЕГМЕНТ public метод show(), который печатает содержимое полей экземпляра своего класса, вызывая метод show() родительского класса, и допечатывая значения остальных полей. Создайте в классе СЕГМЕНТ public метод square(), вычисляющий площадь сегмента по формуле S = Pi*R^2*A / 180 – ((R^2*sin(A)) / 2).
15	Создать базовый класс ЛИНИЯ с protected полем вещественного типа L (это длина линии), public методом show(), который печатает значения всех полей экземпляра данного класса и конструктором, принимающим 1 параметр для инициализации полей экземпляра класса. От класса ЛИНИЯ унаследуйте класс ТРЕУГОЛЬНИК, в котором кроме поля L будут еще protected поля L2 (другая сторона треугольника) и A (угол напротив одной из сторон треугольника). Создайте в классе ТРЕУГОЛЬНИК конструктор с параметрами, который вызывает конструктор родительского класса, а сам доинициализирует остальные поля своего экземпляра класса. Создайте в классе ТРЕУГОЛЬНИК public метод show(), который печатает содержимое полей экземпляра своего класса, вызывая метод show() родительского класса, и допечатывая значения остальных полей. Создайте в классе ТРЕУГОЛЬНИК public метод square(), вычисляющий площадь треугольника по формуле (S = (L1 * L2 * sin(A)) / 2). От класса ТРЕУГОЛЬНИК унаследуйте класс, подходящий по логике наследования, со своим конструктором, методом show(), прочими членами.
16	Создать базовый класс ТОЧКА с protected полями вещественного типа X и Y (это координаты точки по оси X и Y), public методом show(), который печатает значения всех полей экземпляра данного класса и конструктором, принимающим 2 параметра для инициализации полей экземпляра класса. От класса ТОЧКА унаследуйте класс ЛИНИЯ, в котором кроме полей X и Y будут еще поля X1 и Y1, так как линия имеет координаты двух своих крайних точек. Создайте в классе ЛИНИЯ конструктор с параметрами, который вызывает конструктор родительского класса, а сам доинициализирует остальные поля своего экземпляра класса. Создайте в классе ЛИНИЯ public метод show(), который печатает содержимое полей экземпляра своего класса, вызывая метод show() родительского класса, и допечатывая значения остальных полей. Создайте в классе ЛИНИЯ public метод length(), вычисляющий длину линии по формуле: L = sqrt((X1 – X2)^2 + (Y1 – Y2)^2). От класса ЛИНИЯ унаследуйте класс, подходящий по логике наследования, со своим конструктором, методом show(), прочими членами.
17	Создать базовый класс ТОЧКА с protected полями вещественного типа X и Y (это координаты точки по оси X и Y), public методом show(), который печатает значения всех полей экземпляра данного класса и конструктором, принимающим 2 параметра для инициализации полей экземпляра класса. От класса ТОЧКА унаследуйте класс ОКРУЖНОСТЬ, в котором точка с координатами X и Y – это центр окружности. В классе ОКРУЖНОСТЬ создайте protected вещественное поле радиус (длина радиуса), public метод perimetr(), вычисляющий периметр окружности по формуле: P = 2*Pi*R. Создайте в классе ОКРУЖНОСТЬ конструктор с параметрами, который вызывает конструктор родительского класса, а сам доинициализирует остальные поля своего экземпляра класса. Создайте в классе ОКРУЖНОСТЬ public метод show(), который печатает содержимое полей экземпляра своего класса, вызывая метод show() родительского класса, и допечатывая значения остальных полей. От класса ОКРУЖНОСТЬ унаследуйте класс, подходящий по логике наследования, со своим конструктором, методом show(), прочими членами.
18	Создать базовый класс ТОЧКА с protected полями вещественного типа X и Y (это координаты точки по оси X и Y), public методом show(), который печатает значения всех полей экземпляра данного класса и конструктором, принимающим 2 параметра для инициализации полей экземпляра класса. От класса ТОЧКА унаследуйте класс ОКРУЖНОСТЬ, в котором точка с координатами X и Y – это центр окружности. В классе ОКРУЖНОСТЬ создайте protected вещественное поле радиус (длина радиуса), public метод square(), вычисляющий площадь внутри окружности по формуле: S = Pi*R^2. Создайте в классе ОКРУЖНОСТЬ конструктор с параметрами, который вызывает конструктор родительского класса, а сам доинициализирует остальные поля своего экземпляра класса. Создайте в классе ОКРУЖНОСТЬ public метод show(), который печатает содержимое полей экземпляра своего класса, вызывая метод show() родительского класса, и допечатывая значения остальных полей. От класса ОКРУЖНОСТЬ унаследуйте класс, подходящий по логике наследования, со своим

	конструктором, методом show(), прочими членами.
19	Создать базовый класс ТОЧКА с protected полями вещественного типа X и Y (это координаты точки по оси X и Y), public методом show(), который печатает значения всех полей экземпляра данного класса и конструктором, принимающим 2 параметра для инициализации полей экземпляра класса. От класса ТОЧКА унаследуйте класс ОКРУЖНОСТЬ, в котором точка с координатами X и Y – это центр окружности. В классе ОКРУЖНОСТЬ создайте protected вещественное поле диаметр (длина диаметра), public метод perimetr(), вычисляющий периметр окружности по формуле: P = Pi*D. Создайте в классе ОКРУЖНОСТЬ конструктор с параметрами, который вызывает конструктор родительского класса, а сам доинициализирует остальные поля своего экземпляра класса. Создайте в классе ОКРУЖНОСТЬ public метод show(), который печатает содержимое полей экземпляра своего класса, вызывая метод show() родительского класса, и допечатывая значения остальных полей. От класса ОКРУЖНОСТЬ унаследуйте класс, подходящий по логике наследования, со своим конструктором, методом show(), прочими членами.
20	Создать базовый класс ТОЧКА с protected полями вещественного типа X и Y (это координаты точки по оси X и Y), public методом show(), который печатает значения всех полей экземпляра данного класса и конструктором, принимающим 2 параметра для инициализации полей экземпляра класса. От класса ТОЧКА унаследуйте класс ОКРУЖНОСТЬ, в котором точка с координатами X и Y – это центр окружности. В классе ОКРУЖНОСТЬ создайте protected вещественное поле диаметр (длина диаметра), public метод square(), вычисляющий площадь внутри окружности по формуле: S = (Pi*D^2) / 4. Создайте в классе ОКРУЖНОСТЬ конструктор с параметрами, который вызывает конструктор родительского класса, а сам доинициализирует остальные поля своего экземпляра класса. Создайте в классе ОКРУЖНОСТЬ public метод show(), который печатает содержимое полей экземпляра своего класса, вызывая метод show() родительского класса, и допечатывая значения остальных полей. От класса ОКРУЖНОСТЬ унаследуйте класс, подходящий по логике наследования, со своим конструктором, методом show(), прочими членами.
21	Создать базовый класс ЛИНИЯ с protected полем вещественного типа L (это длина линии), public методом show(), который печатает значения всех полей экземпляра данного класса и конструктором, принимающим 1 параметр для инициализации полей экземпляра класса. От класса ЛИНИЯ унаследуйте класс КВАДРАТ, в котором кроме поля L будут еще поля S (площадь квадрата) и P (периметр квадрата). Создайте в классе КВАДРАТ конструктор с параметрами, который вызывает конструктор родительского класса, а сам доинициализирует остальные поля своего экземпляра класса, вызывая нижеописанные методы. Создайте в классе КВАДРАТ public метод show(), который печатает содержимое полей экземпляра своего класса, вызывая метод show() родительского класса, и допечатывая значения остальных полей. Создайте в классе КВАДРАТ public методы perimetr(), вычисляющий периметр квадрата по формуле (P = 4 * L) и square(), вычисляющий площадь квадрата по формуле (S = L^2). От класса КВАДРАТ унаследуйте класс, подходящий по логике наследования, со своим конструктором, методом show(), прочими членами.
22	Создать базовый класс ЛИНИЯ с protected полем вещественного типа L (это длина линии), public методом show(), который печатает значения всех полей экземпляра данного класса и конструктором, принимающим 1 параметр для инициализации полей экземпляра класса. От класса ЛИНИЯ унаследуйте класс ПРЯМОУГОЛЬНИК, в котором кроме поля L (одна сторона) будет еще protected поле L2 (вторая сторона). Создайте в классе ПРЯМОУГОЛЬНИК конструктор с параметрами, который вызывает конструктор родительского класса, а сам доинициализирует остальные поля своего экземпляра класса. Создайте в классе ПРЯМОУГОЛЬНИК public метод show(), который печатает содержимое полей экземпляра своего класса, вызывая метод show() родительского класса, и допечатывая значения остальных полей. Создайте в классе ПРЯМОУГОЛЬНИК public методы perimetr(), вычисляющий периметр прямоугольника по формуле (P = 2 * (L1 + L2)) и square(), вычисляющий площадь прямоугольника по формуле (S = L1 * L2). От класса ПРЯМОУГОЛЬНИК унаследуйте класс, подходящий по логике наследования, со своим конструктором, методом show(), прочими членами.
23	Создать базовый класс ЛИНИЯ с protected полем вещественного типа L (это длина линии), public методом show(), который печатает значения всех полей экземпляра данного класса и конструктором, принимающим 1 параметр для инициализации полей экземпляра класса. От класса ЛИНИЯ унаследуйте класс РОМБ, в котором кроме поля L будет еще поле H (высота ромба). Создайте в классе РОМБ конструктор с параметрами, который вызывает конструктор родительского класса, а сам доинициализирует остальные поля своего экземпляра класса. Создайте в классе РОМБ public метод show(), который печатает содержимое полей экземпляра своего класса, вызывая метод show() родительского класса, и допечатывая значения остальных полей. Создайте в классе РОМБ public методы perimetr(), вычисляющий периметр ромба по формуле (P = 4 * L) и square(), вычисляющий

	площадь ромба по формуле (S = L * H). От класса РОМБ унаследуйте класс, подходящий по логике наследования, со своим конструктором, методом show(), прочими членами.
24	Создать базовый класс ЛИНИЯ с protected полем вещественного типа L (это длина линии), public методом show(), который печатает значения всех полей экземпляра данного класса и конструктором, принимающим 1 параметр для инициализации полей экземпляра класса. От класса ЛИНИЯ унаследуйте класс ПАРАЛЛЕЛОГРАММ, в котором кроме поля L будут еще protected поля L2 (вторая сторона параллелограмма) и H (высота параллелограмма). Создайте в классе ПАРАЛЛЕЛОГРАММ конструктор с параметрами, который вызывает конструктор родительского класса, а сам доинициализирует остальные поля своего экземпляра класса. Создайте в классе ПАРАЛЛЕЛОГРАММ public метод show(), который печатает содержимое полей экземпляра своего класса, вызывая метод show() родительского класса, и допечатывая значения остальных полей. Создайте в классе ПАРАЛЛЕЛОГРАММ public методы perimetr(), вычисляющий периметр параллелограмма по формуле (P = 2 * (L1 + L2)) и square(), вычисляющий площадь параллелограмма по формуле (S = L * H). От класса ПАРАЛЛЕЛОГРАММ унаследуйте класс, подходящий по логике наследования, со своим конструктором, методом show(), прочими членами.
25	Создать базовый класс ЛИНИЯ с protected полем вещественного типа L (это длина линии), public методом show(), который печатает значения всех полей экземпляра данного класса и конструктором, принимающим 1 параметр для инициализации полей экземпляра класса. От класса ЛИНИЯ унаследуйте класс ТРЕУГОЛЬНИК, в котором кроме поля L будет еще поле H (высота треугольника). Создайте в классе ТРЕУГОЛЬНИК конструктор с параметрами, который вызывает конструктор родительского класса, а сам доинициализирует остальные поля своего экземпляра класса. Создайте в классе ТРЕУГОЛЬНИК public метод show(), который печатает содержимое полей экземпляра своего класса, вызывая метод show() родительского класса, и допечатывая значения остальных полей. Создайте в классе ТРЕУГОЛЬНИК public метод square(), вычисляющий площадь треугольника по формуле (S = (L * H) / 2). От класса ТРЕУГОЛЬНИК унаследуйте класс, подходящий по логике наследования, со своим конструктором, методом show(), прочими членами.
26	Создать базовый класс ЛИНИЯ с protected полем вещественного типа L (это длина линии), public методом show(), который печатает значения всех полей экземпляра данного класса и конструктором, принимающим 1 параметр для инициализации полей экземпляра класса. От класса ЛИНИЯ унаследуйте класс ТРАПЕЦИЯ, в котором кроме поля L будут еще protected поля L2 (вторая сторона трапеции) H (высота трапеции). Создайте в классе ТРАПЕЦИЯ конструктор с параметрами, который вызывает конструктор родительского класса, а сам доинициализирует остальные поля своего экземпляра класса. Создайте в классе ТРАПЕЦИЯ public метод show(), который печатает содержимое полей экземпляра своего класса, вызывая метод show() родительского класса, и допечатывая значения остальных полей. Создайте в классе ТРАПЕЦИЯ public метод square(), вычисляющий площадь трапеции по формуле (S = ((L1 + L2) * H) / 2). От класса ТРАПЕЦИЯ унаследуйте класс, подходящий по логике наследования, со своим конструктором, методом show(), прочими членами.
27	Создать базовый класс ЛИНИЯ с protected полем вещественного типа L (это длина линии), public методом show(), который печатает значения всех полей экземпляра данного класса и конструктором, принимающим 1 параметр для инициализации полей экземпляра класса. От класса ЛИНИЯ унаследуйте класс ШЕСТИУГОЛЬНИК, в котором кроме поля L будет еще protected поле H (высота (радиус) шестиугольника). Создайте в классе ШЕСТИУГОЛЬНИК конструктор с параметрами, который вызывает конструктор родительского класса, а сам доинициализирует остальные поля своего экземпляра класса. Создайте в классе ШЕСТИУГОЛЬНИК public метод show(), который печатает содержимое полей экземпляра своего класса, вызывая метод show() родительского класса, и допечатывая значения остальных полей. Создайте в классе ШЕСТИУГОЛЬНИК public методы perimetr(), вычисляющий периметр шестиугольника по формуле (P = 6 * L) и square(), вычисляющий площадь шестиугольника по формуле (S = 3 * L * H). От класса ШЕСТИУГОЛЬНИК унаследуйте класс, подходящий по логике наследования, со своим конструктором, методом show(), прочими членами.
28	Создать базовый класс ТОЧКА с protected полями вещественного типа X и Y (это координаты точки по оси X и Y), public методом show(), который печатает значения всех полей экземпляра данного класса и конструктором, принимающим 2 параметра для инициализации полей экземпляра класса. От класса ТОЧКА унаследуйте класс ОКРУЖНОСТЬ, в котором точка с координатами X и Y – это центр окружности. В классе ОКРУЖНОСТЬ создайте protected вещественное поле радиус (длина радиуса), public метод square(), вычисляющий площадь внутри окружности по формуле: S = Pi*R^2. Создайте в

	классе ОКРУЖНОСТЬ конструктор с параметрами, который вызывает конструктор родительского класса, а сам доинициализирует остальные поля своего экземпляра класса. Создайте в классе ОКРУЖНОСТЬ public метод show(), который печатает содержимое полей экземпляра своего класса, вызывая метод show() родительского класса, и допечатывая значения остальных полей. От класса ОКРУЖНОСТЬ унаследуйте класс СЕКТОР окружности, со своим protected полем A (угол сектора окружности). Создайте в классе СЕКТОР конструктор с параметрами, который вызывает конструктор родительского класса, а сам доинициализирует остальные поля своего экземпляра класса. Создайте в классе СЕКТОР public метод show(), который печатает содержимое полей экземпляра своего класса, вызывая метод show() родительского класса, и допечатывая значения остальных полей. Создайте в классе СЕКТОР public метод square(), вычисляющий площадь сектора по формуле S = Pi*R^2*A / 180.
29	Создать базовый класс ТОЧКА с protected полями вещественного типа X и Y (это координаты точки по оси X и Y), public методом show(), который печатает значения всех полей экземпляра данного класса и конструктором, принимающим 2 параметра для инициализации полей экземпляра класса. От класса ТОЧКА унаследуйте класс ОКРУЖНОСТЬ, в котором точка с координатами X и Y – это центр окружности. В классе ОКРУЖНОСТЬ создайте protected вещественное поле радиус (длина радиуса), public метод square(), вычисляющий площадь внутри окружности по формуле: S = Pi*R^2. Создайте в классе ОКРУЖНОСТЬ конструктор с параметрами, который вызывает конструктор родительского класса, а сам доинициализирует остальные поля своего экземпляра класса. Создайте в классе ОКРУЖНОСТЬ public метод show(), который печатает содержимое полей экземпляра своего класса, вызывая метод show() родительского класса, и допечатывая значения остальных полей. От класса ОКРУЖНОСТЬ унаследуйте класс СЕГМЕНТ окружности, со своим protected полем A (угол сегмента окружности). Создайте в классе СЕГМЕНТ конструктор с параметрами, который вызывает конструктор родительского класса, а сам доинициализирует остальные поля своего экземпляра класса. Создайте в классе СЕГМЕНТ public метод show(), который печатает содержимое полей экземпляра своего класса, вызывая метод show() родительского класса, и допечатывая значения остальных полей. Создайте в классе СЕГМЕНТ public метод square(), вычисляющий площадь сегмента по формуле S = Pi*R^2*A / 180 – ((R^2*sin(A)) / 2).
30	Создать базовый класс ЛИНИЯ с protected полем вещественного типа L (это длина линии), public методом show(), который печатает значения всех полей экземпляра данного класса и конструктором, принимающим 1 параметр для инициализации полей экземпляра класса. От класса ЛИНИЯ унаследуйте класс ТРЕУГОЛЬНИК, в котором кроме поля L будут еще protected поля L2 (другая сторона треугольника) и A (угол напротив одной из сторон треугольника). Создайте в классе ТРЕУГОЛЬНИК конструктор с параметрами, который вызывает конструктор родительского класса, а сам доинициализирует остальные поля своего экземпляра класса. Создайте в классе ТРЕУГОЛЬНИК public метод show(), который печатает содержимое полей экземпляра своего класса, вызывая метод show() родительского класса, и допечатывая значения остальных полей. Создайте в классе ТРЕУГОЛЬНИК public метод square(), вычисляющий площадь треугольника по формуле (S = (L1 * L2 * sin(A)) / 2). От класса ТРЕУГОЛЬНИК унаследуйте класс, подходящий по логике наследования, со своим конструктором, методом show(), прочими членами.