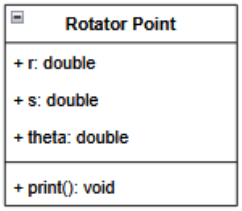
ВАРИАНТ 01: Перегрузка операций (100 баллов)

Определяет класс Rotator Point, определенный диаграммой UML ниже:



[1]. Построить класс **Rotator Point**, конструктор по умолчанию, по параметрам, по копированию объектов, методы **setters** и **getters**. Учитывать, что все атрибуты имеют привилегию доступа **private**. Определить функцию **print()** для отображения значений каждого атрибута.

[2]. Создайте 2 объекта класса **Rotator Point**. Сделайте перегрузку операций **+=** и ***=**. Реализуйте глобальную функцию с привилегированным **friend**, вызываемым по ссылке (рассмотрите объекты **rp1** и **rp2** класса **Rotator Point**):

a) определить операцию **+=** по **rp1 += rp2**

$$rp1.r = (rp1.r + rp2.r) * \cos(rp1.theta)$$

$$rp1.s = (rp1.s + rp2.s) * \sin(rp2.theta)$$

$$r1.theta = rp1.theta + rp2.theta$$

b) определить операцию ***=** по **rp1 *= rp2**

$$rp1.r = rp1.r * \cos(rp1.theta) + rp2.r * \sin(rp2.theta)$$

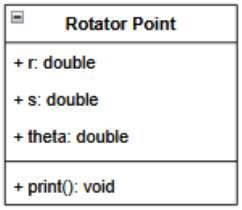
$$rp1.s = rp1.s * \sin(rp1.theta) + rp2.s * \cos(rp2.theta)$$

$$r1.theta = rp1.theta + rp2.theta$$

c) Распечатать атрибуты результата операции **rp1*=rp2.**

ВАРИАНТ 01: Перегрузка операций (100 баллов)

Определяет класс Rotator Point, определенный диаграммой UML ниже:



[1]. Построить класс **Rotator Point**, конструктор по умолчанию, по параметрам, по копированию объектов, методы **setters** и **getters**. Учитывать, что все атрибуты имеют привилегию доступа **private**. Определить функцию **print()** для отображения значений каждого атрибута.

[2]. Создайте 2 объекта класса **Rotator Point**. Сделайте перегрузку операций **+=** и ***=**. Реализуйте глобальную функцию с привилегированным **friend**, вызываемым по ссылке (рассмотрите объекты **rp1** и **rp2** класса **Rotator Point**):

a) определить операцию **+=** по **rp1 += rp2**

$$rp1.r = (rp1.r + rp2.r) * \cos(rp1.theta)$$

$$rp1.s = (rp1.s + rp2.s) * \sin(rp2.theta)$$

$$r1.theta = rp1.theta + rp2.theta$$

b) определить операцию ***=** по **rp1 *= rp2**

$$rp1.r = rp1.r * \cos(rp1.theta) + rp2.r * \sin(rp2.theta)$$

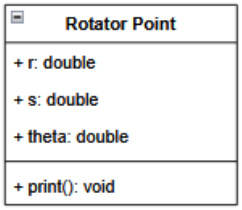
$$rp1.s = rp1.s * \sin(rp1.theta) + rp2.s * \cos(rp2.theta)$$

$$r1.theta = rp1.theta + rp2.theta$$

c) Распечатать атрибуты результата операции **rp1*=rp2.**

ВАРИАНТ 01: Перегрузка операций (100 баллов)

Определяет класс Rotator Point, определенный диаграммой UML ниже:



[1]. Построить класс **Rotator Point**, конструктор по умолчанию, по параметрам, по копированию объектов, методы **setters** и **getters**. Учитывать, что все атрибуты имеют привилегию доступа **private**. Определить функцию **print()** для отображения значений каждого атрибута.

[2]. Создайте 2 объекта класса **Rotator Point**. Сделайте перегрузку операций **+=** и ***=**. Реализуйте глобальную функцию с привилегированным **friend**, вызываемым по ссылке (рассмотрите объекты **rp1** и **rp2** класса **Rotator Point**):

a) определить операцию **+=** по **rp1 += rp2**

$$rp1.r = (rp1.r + rp2.r) * \cos(rp1.theta)$$

$$rp1.s = (rp1.s + rp2.s) * \sin(rp2.theta)$$

$$r1.theta = rp1.theta + rp2.theta$$

b) определить операцию ***=** по **rp1 *= rp2**

$$rp1.r = rp1.r * \cos(rp1.theta) + rp2.r * \sin(rp2.theta)$$

$$rp1.s = rp1.s * \sin(rp1.theta) + rp2.s * \cos(rp2.theta)$$

$$r1.theta = rp1.theta + rp2.theta$$

c) Распечатать атрибуты результата операции **rp1*=rp2.**

ВАРИАНТ 02: Вектор объектов (50 баллов)

Рассмотрим некую веб-службу, реализованную в финансовой компании **traders.org**, которая имеет **кластеры** и **серверы баз данных**. Каждый **сервер баз данных** имеет следующие характеристики:

- **server name** (имя сервера): строка с максимум 20 буквами

- **database engine** (ядро базы данных): перечисление, вызываемое с 3 возможными вариантами строк (PostgreSQL, MySQL и Oracle)

- host name (имя хоста): строка, объединяющая имя компании «**traders.org**» + некоторую дополнительную информацию о строке, определенную пользователем.

- **IP address** (IP-адрес): имеет формат: «xxx.xxx.x.x» (пример: 192.168.1.1). Первые 3 группы чисел точно такие же, как у кластера (скопируйте IP-адрес соответствующего кластера и рассмотрите только первые группы чисел, используя copy[strlen(copy)-4] = '\0';), и добавьте к строке последнее число, которое определяется пользователем.

- **port** (порт): целочисленное значение

Действительно, у компании есть 2 **кластера**: **первичный кластер (Primary cluster)** и **кластер-реплика (Replica cluster)**. Каждый **кластер** имеет следующие характеристики:

- **cluster name** (имя кластера): строка, содержащая максимум 20 букв

- **total number of nodes** (общее количество узлов): целочисленное значение количества зарегистрированных серверов баз данных.

- **maximum capacity** (максимальная емкость): целочисленное значение максимального количества серверов баз данных.

- **IP address** (IP-адрес): имеет формат: «xxx.xxx.x.0/24» (пример: 192.168.1.0/24). Последняя цифра адреса всегда равна 0, а 24 указывает максимальную емкость.

- **vector of servers** (вектор серверов): содержит вектор серверов баз данных.

Следуйте следующим инструкциям:

[1]. Создайте классы **Cluster** и **Database Server** и нарисуйте диаграмму UML. Определите конструктор по умолчанию, конструктор по параметрам, конструктор по копии объекта, методы **setters** и **getters**. Все атрибуты для обоих классов имеют **private** доступ. Каждый кластер (первичный кластер и кластер реплики) являются объектами класса Cluster. В main.cpp необходимо определить эти 2 объекта с помощью конструктора кластера.

[2]. Определите в классе **Cluster** методы **set_vec_servers()** и **print_cluster()**. Первый метод требует вставки вектора объектов сервера базы данных в некоторый кластер. Второй метод требует вывода имени кластера и всех серверов базы данных с их атрибутами в формате, похожем на таблицу. Действительно, надо печать общего количества серверов.

ВАРИАНТ 02: Вектор объектов (50 баллов)

Рассмотрим некую веб-службу, реализованную в финансовой компании **traders.org**, которая имеет **кластеры** и **серверы баз данных**. Каждый **сервер баз данных** имеет следующие характеристики:

- **server name** (имя сервера): строка с максимум 20 буквами

- **database engine** (ядро базы данных): перечисление, вызываемое с 3 возможными вариантами строк (PostgreSQL, MySQL и Oracle)

- host name (имя хоста): строка, объединяющая имя компании «**traders.org**» + некоторую дополнительную информацию о строке, определенную пользователем.

- **IP address** (IP-адрес): имеет формат: «xxx.xxx.x.x» (пример: 192.168.1.1). Первые 3 группы чисел точно такие же, как у кластера (скопируйте IP-адрес соответствующего кластера и рассмотрите только первые группы чисел, используя copy[strlen(copy)-4] = '\0';), и добавьте к строке последнее число, которое определяется пользователем.

- **port** (порт): целочисленное значение

Действительно, у компании есть 2 **кластера**: **первичный кластер (Primary cluster)** и **кластер-реплика (Replica cluster)**. Каждый **кластер** имеет следующие характеристики:

- **cluster name** (имя кластера): строка, содержащая максимум 20 букв

- **total number of nodes** (общее количество узлов): целочисленное значение количества зарегистрированных серверов баз данных.

- **maximum capacity** (максимальная емкость): целочисленное значение максимального количества серверов баз данных.

- **IP address** (IP-адрес): имеет формат: «xxx.xxx.x.0/24» (пример: 192.168.1.0/24). Последняя цифра адреса всегда равна 0, а 24 указывает максимальную емкость.

- **vector of servers** (вектор серверов): содержит вектор серверов баз данных.

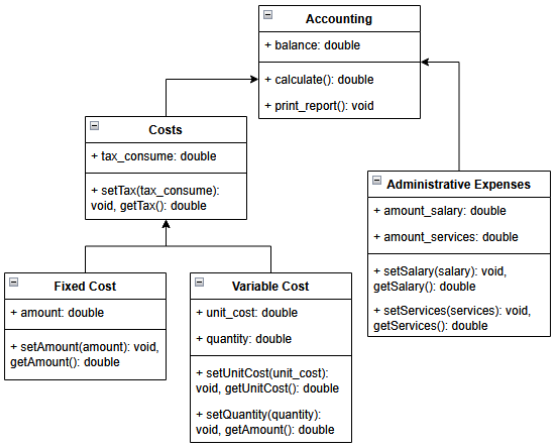
Следуйте следующим инструкциям:

[1]. Создайте классы **Cluster** и **Database Server** и нарисуйте диаграмму UML. Определите конструктор по умолчанию, конструктор по параметрам, конструктор по копии объекта, методы **setters** и **getters**. Все атрибуты для обоих классов имеют **private** доступ. Каждый кластер (первичный кластер и кластер реплики) являются объектами класса Cluster. В main.cpp необходимо определить эти 2 объекта с помощью конструктора кластера.

[2]. Определите в классе **Cluster** методы **set_vec_servers()** и **print_cluster()**. Первый метод требует вставки вектора объектов сервера базы данных в некоторый кластер. Второй метод требует вывода имени кластера и всех серверов базы данных с их атрибутами в формате, похожем на таблицу. Действительно, надо печать общего количества серверов.

ВАРИАНТ 03: Наследование классов (100 баллов)

В системе учета (Accounting) компании мы производим учет затрат (Costs) и административных расходов (Administrative Expenses). Затраты определяются двумя типами: фиксированные затраты и переменные затраты. На диаграмме UML ниже определены атрибуты каждого класса и их отношения наследования:



- Бухгалтерский учет (class Accounting): баланс; чистый виртуальный метод calculate() и виртуальный метод print_report()

- Затраты (class Costs): налог на потребление (в процентах)

- Постоянные затраты (class Fixed Cost): фиксированная сумма- Переменные затраты (class Variable Cost): себестоимость единицы продукции и количество продуктов.

- Административные расходы (class Administrative Expenses): размер заработной платы и платы услуг.

Следуйте следующим инструкциям:

[1]. Создайте классы, разработайте конструкторы: конструктор по умолчанию, по параметрам и по копии объекта; setters и getters, следуя отношениям наследования. Атрибуты класса Accounting имеют protected привилегию доступа.

[2]. Реализуйте виртуальные функции calculate() — чисто виртуальную функцию и print_report() — простой виртуальный метод, учитывая следующие условия:

метод calculate()

- для класса Fixed Cost:

$$balance = balance * (1 - tax\_consume) - amount$$

- для класса Variable Cost:

$$balance = \big(balance - (unit_{cost} * quantity)\big) * (1 - tax\_consume)$$

- для класса Administrative Expenses:

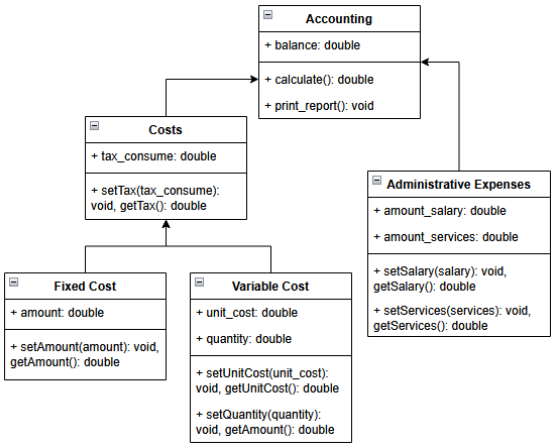
$$balance = balance - amount_{salary} - amount_{services}$$

метод print_transaction()

Распечатать все атрибуты, связанные с созданным объектом в обоих производных классах. Включить результаты, полученные в функции calculate(), в подотчетность в зависимости от производного класса. Создать 1 объект для классов Fixed cost, Variable Cost и Administrative Expenses и показать эти результаты для каждого объекта.

ВАРИАНТ 03: Наследование классов (100 баллов)

В системе учета (Accounting) компании мы производим учет затрат (Costs) и административных расходов (Administrative Expenses). Затраты определяются двумя типами: фиксированные затраты и переменные затраты. На диаграмме UML ниже определены атрибуты каждого класса и их отношения наследования:



- Бухгалтерский учет (class Accounting): баланс; чистый виртуальный метод calculate() и виртуальный метод print_report()

- Затраты (class Costs): налог на потребление (в процентах)

- Постоянные затраты (class Fixed Cost): фиксированная сумма- Переменные затраты (class Variable Cost): себестоимость единицы продукции и количество продуктов.

- Административные расходы (class Administrative Expenses): размер заработной платы и платы услуг.

Следуйте следующим инструкциям:

[1]. Создайте классы, разработайте конструкторы: конструктор по умолчанию, по параметрам и по копии объекта; setters и getters, следуя отношениям наследования. Атрибуты класса Accounting имеют protected привилегию доступа.

[2]. Реализуйте виртуальные функции calculate() — чисто виртуальную функцию и print_report() — простой виртуальный метод, учитывая следующие условия:

метод calculate()

- для класса Fixed Cost:

$$balance = balance * (1 - tax\_consume) - amount$$

- для класса Variable Cost:

$$balance = \big(balance - (unit_{cost} * quantity)\big) * (1 - tax\_consume)$$

- для класса Administrative Expenses:

$$balance = balance - amount_{salary} - amount_{services}$$

метод print_transaction()

Распечатать все атрибуты, связанные с созданным объектом в обоих производных классах. Включить результаты, полученные в функции calculate(), в подотчетность в зависимости от производного класса. Создать 1 объект для классов Fixed cost, Variable Cost и Administrative Expenses и показать эти результаты для каждого объекта.