Билет 12

Задание 1

Познакомьтесь с предложенной реализацией классов. Определите тип отношения между классами (предоставьте описание в виде UML), ответ обоснуйте.

```
class ClassB;
  class ClassA
  {
  private:
        bool m_update;
        ClassB* m_ptrB[4]; //массив указателей на связанные объекты
  public:
        ClassA();
        ~ClassA();
        //вернуть количество связей
        const int getMultiplicity() const;
        //вернуть указатели на связанные объекты
        const ClassB** getB() const;
        bool hasB() const; // проверить наличие связей
        bool hasB(const ClassB&) const; //проверить связь
        void addB(ClassB&);//установить связь
        void removeB(ClassB&);//разорвать связь
        void removeB(); //разорвать все связи
  };
```

Реализация класса ClassA

```
#include "ClassA.h"
#include"ClassB.h"
ClassA::ClassA() : m_update(false)
{
       for (int j(0); j < 4; j++)
              m_ptrB[j] = 0;
}
ClassA :: ~ClassA()
{
       for (int j(0); j < 4; j++)
              m_ptrB[j] = 0;
}
//вернуть количество связей
const int ClassA::getMultiplicity() const
{
       int n(0);
       while (m_ptrB[n] && n < 4)
              ++n;
       return n;
}
//вернуть указатели на связанные объекты
const ClassB** ClassA::getB() const
{
       return const_cast <const ClassB**> (m_ptrB);
}
//проверить наличие связей
bool ClassA::hasB() const
{
       return m_ptrB[0] != 0;
}
//проверить связь с объектом
bool ClassA::hasB(const ClassB& r) const
{
       int j(0);
       while (m_ptrB[j])
       {
              if (m_ptrB[j] == &r)
                     return true;
              j++;
       }
```

```
	return false;
}
//установить связь с объектом
void ClassA::addB(ClassB& r)
{
	if (hasB(r))
		return;
	if (m_update)
		return;
	//должно быть место для добавления
	if (getMultiplicity() >= 4) return;
	//r должен быть свободен
	if (r.hasA())
		return;
	m_update = true;
	//запрос r об обновлении связи
	r.addA(*this);
	//обновление своих связей
	int n = getMultiplicity();
	m_ptrB[n] = &r;
	m_update = false;
}
//разорвать связь с объектом
void ClassA::removeB(ClassB& r)
{
	if (m_update) return;
	if (!hasB(r))
		return;
	m_update = true;
	//запрос r о разрыве связи
	r.removeA();
	//разрыв своей связи
	int n = getMultiplicity();
	int j;
	for (j = 0; j < n && m_ptrB[j] != &r; j++);
	for (int i(j); i < n - 1; i++)
		m_ptrB[i] = m_ptrB[i + 1];
	m_ptrB[n - 1] = 0;
	m_update = false;
}
//разорвать все связи
void ClassA::removeB()
{
	if (m_update) return;
	m_update = true;
	int j(0);
	while (j < getMultiplicity())
	{
		m_ptrB[j]->removeA(); j++;
	}
	for (int j(0); j < 4; j++)
		m_ptrB[j] = 0;
	m_update = false;
}
```

Класс ClassB

```
class ClassA;
class ClassB {
	ClassA* m_ptrA; //указатель на связанный объект
	bool m_update; //признак обновления связи
public: ClassB();
		~ClassB(); //вернуть указатель на связанный объект
		const ClassA* getA() const;
		bool hasA() const;//проверить наличие связи
		void addA(ClassA&);//установить связь
		void removeA(); //разрушить связь
};
```

Реализация класса ClassB

```
ClassB::ClassB() : m_update(false), m_ptrA(0) {}
ClassB :: ~ClassB()
{

	m_ptrA = 0;
}
//вернуть указатель на связанный объект
const ClassA* ClassB::getA() const
{
	return m_ptrA;
}
//проверить наличие связи
bool ClassB::hasA() const
{
	return m_ptrA != 0;
}
//установить связь
void ClassB::addA(ClassA& r)
{
	if (m_update) return;
	if (m_ptrA == &r) return;
	//объект r должен быть свободен от этого объекта
	if (r.hasB(*this)) return;
	if (hasA()) removeA();
	//разрушение своей связи
	//модификация связи
	m_update = true;
	r.addB(*this);
	m_ptrA = &r;
	m_update = false;
}
//разрушить связь
void ClassB::removeA()
{
	if (!hasA()) return;
	if (m_update) return;
	m_update = true;
	m_ptrA->removeB();
	//разрушение своей связи
	m_ptrA = 0;
	m_update = false;
}
```

Задание 2

Базовые принципы проектирования (SOLID). Принцип открытости/закрытости. На представленном примере определить, существует ли нарушение принципа открытости/закрытости. Если да, **то предложите решение**. Ответ обоснуйте.

Легенда

У нас есть приложение, которое *должно* уметь управлять различными фигурами (например, кругами, квадратами, многоугольниками и т. Д.) И рисовать их в стандартном графическом интерфейсе. Имеется исходная реализация.

```

#include <iostream>
#include <list>
#include <vector>
// Структура описывающая точку
struct Point
{
	int x;
	int y;
	Point(int _x = 0, int _y = 0) { x = _x; y = _y; }
};
```

```
// Базовый класс фигура
class Shape
{
public:
	//Здесь конструкторы, геттеры и сеттеры  Get / Set
	Shape(Point c) { center = c; type = "BaseFigure"; }
	std::string GetType() const
	{ return type; }
protected:
	Point center;
	std::string type;
};

class Circle : public Shape
{
public:
	//Здесь конструкторы, геттеры и сеттеры  Get / Set
	Circle(Point cnt, int r) :Shape(cnt) { radius = r; type = "Circle"; }
private:
	int radius;
};

class Square : public Shape
{
public:
	//Здесь конструкторы, геттеры и сеттеры  Get / Set
	Square(Point cnt, int r) :Shape(cnt) { side = r; type = "Square"; }
private:
		int side;
};

class DrwManager
{
private:
	std::list<Shape> shapeList;
public:
	// Здесь различные конструкторы
	DrwManager()
	{
		//Такая инициализация только для примера
		Point p(0,0);
		std::list<Shape> sh = { Square(p,3),Circle(p,3) };
	};

	// Метод рисует все фигуры из списка shapeList
	void drawShapes()
	{

		for (const auto &elem : shapeList)
		{
			if (elem.GetType() =="Circle")
			{
				drawCirle(elem);
			}
			else if (elem.GetType() == "Square")
			{
				drawSquare(elem);
			}
		}
	}
private:
	void drawCirle(const Shape c) { std::cout << "Draw Cirle!\n"; };
	void drawSquare(const Shape c) { std::cout << "Draw Square!\n"; };
};
```

Задание 3

Паттерн «Абстрактная фабрика». Рассмотреть назначение и архитектуру ***абстрактной фабрики***. Применить к предложенной легенде рассматриваемый паттерн. Решение представить в виде кода с подробными объяснениями в онлайн компиляторе, также предоставить UML диаграмму для предложенного решения.

Легенда

Предположим, что на ***складе*** хранятся мобильные телефоны различных производителей. Будем рассматривать два ***типа*** телефонов

1. **Смартфон;**
2. **Простой телефон;**

Для простоты предположим, что у нас есть 3 производителя:

1. **Nokia;**
2. **Samsung;**
3. **HTC;**

Необходимо реализовать программу, которая будет выдавать информацию о телефонах заданного производителя.

Например:

Производитель – **Samsung**;
Смартфон - **Galaxy2**;
Простой телефон - **Primo**;

Требования к ответам

Задание представляете в файле типа word. Каждое задание снабжаете подробными комментариями (обоснованиями предложенного решения). UML диаграмму отдельной картинкой (можно от руки нарисовать), также с объяснениями + ссылка на код в онлайн компиляторе. https://www.onlinegdb.com/

Ответ, скопированный откуда-либо в качестве ответа на задание, не рассматривается, такая работа не зачитывается.