# Лабораторная работа 3. Отношения классов. Ассоциации.

## Ассоциация.

**Композиция**. Предполагает создание вложенных объектов в конструкторе класса-контейнера. (Не стоит создавать продавцов вместе с магазином.) Если логически должна быть связь 1-∞, для композиции использовать массив.

Для главного класса-контейнера разработать несколько перегруженных конструкторов:

1) без параметров,

2) с параметрами для полей этого класса

3) с параметрами для полей этого класса и подчиненного класса-композиции

**Агрегация**. Предполагает связь по ссылке. Ссылка может быть null.

Здесь возможны варианты:

1. Создать массив объектов в классе-контейнере (массив машин в гараже). Изначально все ссылки null.
2. Вместо того, чтобы делать массив ссылок в главном классе, можно в каждом подчиненном объекте сделать ссылку на главный класс (например, каждая машина имеет ссылку на свой гараж – эта ссылка может быть null).
3. То и другое – при добавлении машины к гаражу, в гараже меняется соответствующее место в массиве, у машины меняется ссылка на гараж. Соответствующие методы должны быть в том и другом классе и каждый метод должен изменять объекты обоих классов.

Предусмотреть методы для регистрации подчиненного объекта (клиент прописывается в гостиницу, самолет запрашивает посадку и т.д.)

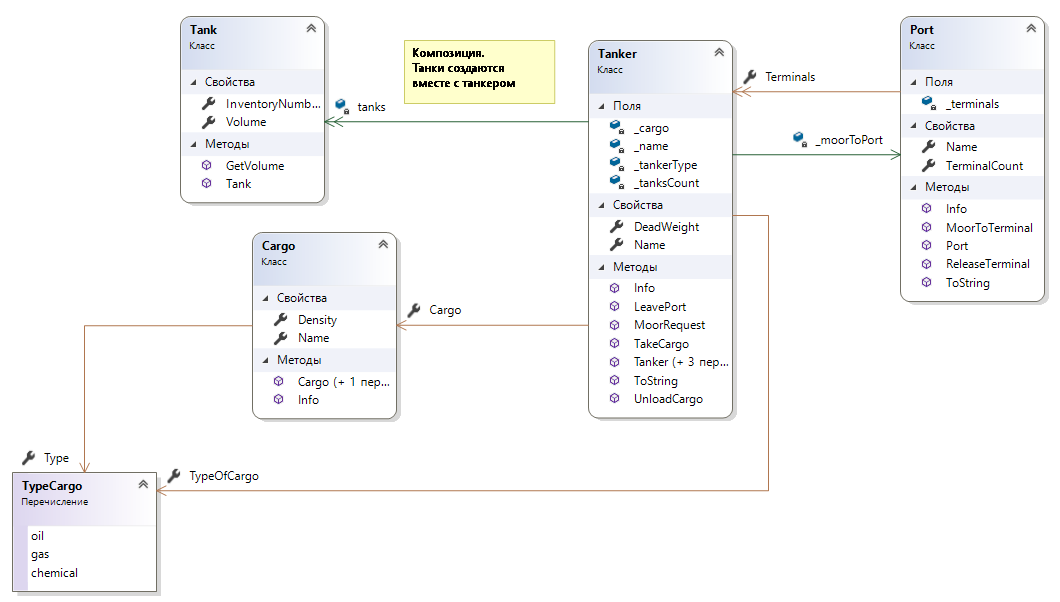
## Задание.

Спроектировать UML-диаграммы классов согласно варианту индивидуального задания. В каждом классе предусмотреть поля (2-4), конструкторы, свойства (2-4) и методы (1-2). Между парами классов спроектировать отношения ­– композицию и агрегацию. Причем для агрегации спроектируйте оба варианта связи: подчиненный класс имеет ссылку на главный, либо массив ссылок в главном классе (либо a и b, либо только c). Продемонстрировать работу классов.

Из предложенной предметной области вы можете использовать все перечисленные объекты, либо часть (минимум – 3). Вы можете добавить свои объекты.

**Варианты заданий.**

1. Факультет – Кафедра – Аудитория – Студент.
2. Автомобиль – Двигатель – Гараж – Водитель.
3. Аэропорт – Взлетная полоса – Самолет – Диспетчер.
4. Танкер – Танк – Порт – Груз.
5. Гостиница – Номер – Клиент – Мебель.
6. Город – Достопримечательность – Турист – Гид.
7. Магазин – Секция – Продавец – Товар.
8. Поезд – Вагон – Груз – Вокзал.
9. Интернет-оператор – Услуга – Абонент – Тарифный план.
10. Отдел – Сотрудник – Организация – Проект.



enum TypeCargo : int

{

oil,

gas,

chemical

}

class Cargo

{

public string Name { get; set; }

public double Density { get; set; }

public TypeCargo Type { get; }

public Cargo(string name, TypeCargo type) : this(name, 1, type) { }

public Cargo(string name, double density, TypeCargo type)

{

Name = name;

Density = density;

Type = type;

}

public void Info()

{

Console.WriteLine($" {Name} {Density} {Type}");

}

}

class Tank

{

public int InventoryNumber { get; }

public int Volume{ get; }

public Tank(TypeCargo forCargoTypes, int num)

{

InventoryNumber = num;

Volume = GetVolume(forCargoTypes);

}

// статический метод для определения объема танка в зависимости от предназначения (типа грузов)

public static int GetVolume(TypeCargo typeT)

{

switch (typeT)

{

case TypeCargo.oil:

return 100;

case TypeCargo.gas:

return 40;

case TypeCargo.chemical:

return 60;

default: return -1;

}

}

}

class Tanker

{

private string \_name; // название танкера

private Tank[] tanks; // массив танков

private int \_tanksCount; // количество танков

private string \_tankerType;

private Cargo \_cargo;

private Port \_moorToPort;

public string Name

{

get => $"\"{\_name}\"";

set { \_name = value; }

}

public int DeadWeight { get; } // дедвейт

public TypeCargo TypeOfCargo { get; }

public Cargo Cargo { get => \_cargo; }

public Tanker() : this("NoName") { }

public Tanker(string name) : this(name, 999) { }

public Tanker(string name, int deadweight) : this(name, deadweight, TypeCargo.oil) { }

public Tanker(string name, int deadweight, TypeCargo typeCargo)

{

\_name = name;

DeadWeight = deadweight;

TypeOfCargo = typeCargo;

\_tanksCount = DeadWeight / Tank.GetVolume(typeCargo);// количество танков при заданом дедвейте для определенного типа грузов

tanks = new Tank[\_tanksCount];

for (int i = 0; i < \_tanksCount; i++)

tanks[i] = new Tank(typeCargo,i+1);

if (typeCargo == TypeCargo.oil)

\_tankerType = "нефтевоз";

else if (typeCargo == TypeCargo.gas)

\_tankerType = "газовоз";

else if (typeCargo == TypeCargo.chemical)

\_tankerType = "химовоз";

}

public override string ToString()

{

string location = \_moorToPort == null ? "в море" : $" в порту \"{\_moorToPort.Name}\"";

string cargo = \_cargo == null ? "пустой" : \_cargo.Name;

return $"-------\nтанкер-{\_tankerType} {Name} количество танков:{\_tanksCount} по {tanks[0].Volume}, дедвейт судна:{DeadWeight}\n" +

$" Местонахождение: {location}; наличие груза: {cargo}\n-------";

}

public void Info()

{

Console.WriteLine(this.ToString());

}

// запрос швартовки

public void MoorRequest(Port port)

{

\_moorToPort = port.MoorToTerminal(this);

}

// покинуть терминал

public void LeavePort(Port port = null) // если в процедуру не передается объект порта, значит инициатор - танкер. Иначе инициатор Port.ReleaseTerminal

{

if (\_moorToPort != null && port == null)

{

\_moorToPort.ReleaseTerminal(this);

\_moorToPort = null;

}

else if (port == \_moorToPort)

\_moorToPort = null;

}

public void TakeCargo(Cargo cargo)

{

if (\_moorToPort != null)

{

if (this.\_cargo != null)

Console.WriteLine($"! Танкер {this.Name} уже загружен!");

else

{

if (this.TypeOfCargo == cargo.Type)

{

this.\_cargo = cargo;

Console.WriteLine($"!Завершилась погрузка груза({cargo.Name}) на танкер {this.Name}");

}

else

Console.WriteLine($"! Танкер {this.Name} не может взять груз ({cargo.Name})!");

}

}else

Console.WriteLine("! Погрузка может осуществляться только в порту!");

}

public void UnloadCargo()

{

if (\_moorToPort != null)

{

if (this.\_cargo != null)

{

Console.WriteLine($"!Завершилась разгрузка груза({this.\_cargo.Name}) с танкера {this.Name}");

this.\_cargo = null;

}

else

Console.WriteLine($"! Танкер {this.Name} пустой!");

}

else

Console.WriteLine("! Разгрузка может осуществляться только в порту!");

}

}

class Port

{

private Tanker[] \_terminals; // т.к. танкеры будем привязывать к терминалу, то \_terminals - это массив танкеров, а порядковый номер в массиве - номер терминала. Если терминал не занят, соответствующий элемент в массиве = null

public string Name { get; set; }

//public string WaterArea { get; set; }

public int TerminalCount { get; }

public Tanker[] Terminals

{

get => \_terminals;

}

public Port(string name, int terminalCount)

{

Name = name;

TerminalCount = terminalCount;

\_terminals = new Tanker[TerminalCount];

}

public Port MoorToTerminal(Tanker tanker) // пришвартовать танкер к свободному терминалу

{

if (tanker != null)

{

// если танкер уже пришвартован

for (int i = 0; i < TerminalCount; i++)

if (\_terminals[i] == tanker)

return this;

// ищем свободный терминал

Console.WriteLine($"! танкер {tanker.Name} запрашивает швартовку в порту \"{Name}\"");

for (int i = 0; i < TerminalCount; i++)

{

if (\_terminals[i] == null)

{

\_terminals[i] = tanker;

Console.WriteLine("! Разрешена швартовка к терминалу № " + (i + 1));

tanker.MoorRequest(this);

return this;

}

}

// все занято

Console.WriteLine("! Отказ. Все терминалы заняты");

return null;

}

return null;

}

public void ReleaseTerminal(Tanker tanker) // освободить терминал

{

if (tanker != null)

{

for (int i = 0; i < TerminalCount; i++)

if (\_terminals[i] == tanker)

{

\_terminals[i] = null;

Console.WriteLine($"! танкер {tanker.Name} покинул терминал № {i + 1} порта \"{this.Name}\"");

}

tanker.LeavePort(this);

}

}

public override string ToString()

{

string str\_terminals = string.Empty;

int i = 1;

foreach (Tanker tanker in \_terminals) {

str\_terminals += $"терминал №{i} ";

if (tanker == null)

str\_terminals += "свободен \n";

else

str\_terminals += $"Занят танкером {tanker.Name} \n";

i++;

}

return $"\*\*\*\nназвание порта: \"{Name}\", количество терминалов - {TerminalCount} \n" + str\_terminals + "\*\*\*";

}

public void Info()

{

Console.WriteLine(this.ToString());

}

}

class Program

{

static void Main(string[] args)

{

Cargo cargoOil = new Cargo("Сырая нефть", TypeCargo.oil);

Cargo cargoGas = new Cargo("Природный газ", TypeCargo.gas);

cargoGas.Info();

Tanker tankerPobeda = new Tanker("Победа", 1000, TypeCargo.oil);

tankerPobeda.Info();

Tanker tankerSibir = new Tanker("Сибирь", 800, TypeCargo.gas);

tankerSibir.Info();

Tanker tankerDerbent = new Tanker("Дербент", 1500, TypeCargo.chemical);

tankerDerbent.Info();

Port portBaltiiskii = new Port("Балтийский", 4);

portBaltiiskii.Info();

Console.WriteLine("попытка загрузить в море");

tankerSibir.TakeCargo(cargoOil);

// Швартовка

portBaltiiskii.MoorToTerminal(tankerSibir);

portBaltiiskii.Info();

tankerSibir.Info();

Console.WriteLine("попытка загрузить не подходящим типом груза");

tankerSibir.TakeCargo(cargoOil);

Console.WriteLine("погрузка газа");

tankerSibir.TakeCargo(cargoGas);

// Швартовка

tankerDerbent.MoorRequest(portBaltiiskii);

portBaltiiskii.Info();

tankerDerbent.Info();

tankerSibir.LeavePort(); // можно и так: portBaltiiskii.ReleaseTerminal(tankerSibir)

portBaltiiskii.Info();

tankerSibir.Info();

tankerPobeda.MoorRequest(portBaltiiskii);

portBaltiiskii.Info();

Console.WriteLine("погрузка нефти");

tankerPobeda.TakeCargo(cargoOil);

portBaltiiskii.ReleaseTerminal(tankerPobeda); // освобождаем терминал

tankerPobeda.Info();

Port portChernomorskii = new Port("Черноморский", 2);

portChernomorskii.MoorToTerminal(tankerPobeda);

portChernomorskii.Info();

tankerPobeda.Info();

tankerPobeda.UnloadCargo();

tankerPobeda.LeavePort();

tankerPobeda.Info();

Console.ReadLine();

}

}

