Object relationship

1) Composition (bir obyekt coxlu xirda hisselerden

yaranirsa bu composition elaqesidir mes. masil

masin yox olsa onun hisseleri de yox olmalidir

masin hemin hisselerden asilidir) elaqe coxdur

2) Aggregation (obyekt digerinin icinde ola da biler olmayda da biler

mes. her banka geden adamin bank hesabi olmasi mecbur deyil)

3) Association (bir obyekt digeri ile elaqe qurur lakin onun icinde olmasi mecbur deyil

mes. patient doctor) elaqe daha azdir

4) Dependency (eger bir obyekt yarananda diger obyektin her hansi funksiyasini oyadirsa

hemin obyekt digerinden asilidir(asililiq - dependency))

Nested (Embedded) Classes

{

class List

{

public:

private:

class Node

{

public:

int data;

Node\* next;

Node\* prev;

};

private:

Node\* head;

Node\* tail;

};

Node tipi yalniz List classinin daxilinde gorunur

Oxuynarligi artirmaq ucundur

Ex.

class Product

{

string name;

public:

struct ProductType

{

enum Types

{

FANCY,AWESOME,USEFUL

};

};

struct ProductBoxType

{

enum Types {

BOX,BAG,CRATE

};

};

Product(ProductType::Types type, ProductBoxType::Types boxtype,string name)

{

}

};

void main()

{

Product p(Product::ProductType::AWESOME, Product::ProductBoxType::BOX, "APPLE");

}

}

Inheritance (bir classin digerinden torenmesi)

{

struktura qazandirir;

yigcamliq qazandirir;

kod artiqliginin qarsisini alir;

Every derived class is base class;

But not every base class derived class;

base class/ super class/ parent class;

derived class/ subclass/ child class/

Istenilen classdan istenilen classi kodda toretmek olar, lakin mentiqi uygunluq olmalidir;

}