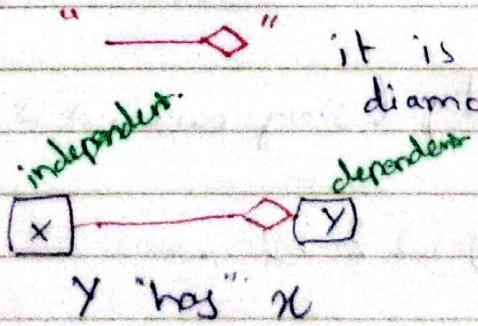
Day: Date: / /

Aggregation:

In this we use simple objects inside complex objects but are not responsible for them.

- It is "Has" or "uses a" relationship
- It is a very weak relationship and there is no ownership
- In this a dependent class contains another independent class but it is NOT responsible for its lifeline.
- The dependent object does NOT create or destroy the independent object, so even if the dependent object is destroyed, independent remains.

Symbols



"◇" it is represented by a hollow diamond.

y "has" x

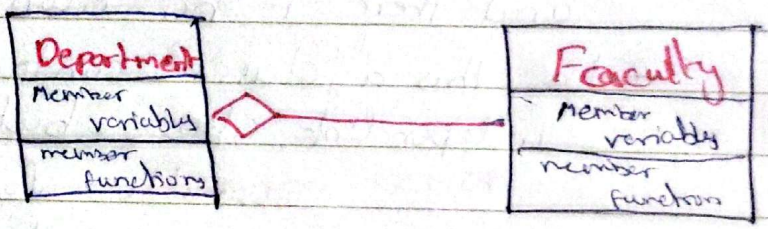
Day: Date: / /

Examples:

1) First way to do aggregation:

- First way is to use a pointer.

classes I will be developing:




```
class Faculty {

        string  name;
        int     id;

public:
        Faculty ( string name , int id) : name(name) , id(id) { };

    ~Faculty () { };

};
```

Yousaf

Day: Date: / /

```
class Department {

    int size ;
    Faculty ** ptr;

Public:
    Department (int s=3) : size(s) {

        Faculty ptr = new Faculty * [size];
    }.

    void add Faculty ( Faculty *f, int Index) {

        ptr [index] = f;
    }

    ~ Department () {
        delete [ ] ptr ;
    }

};

int main () {
    Faculty f1("Haider", "24"), f2("Ali", 92);
    Department d(2);
    d.add faculty ( &f1);
    d. add faulty (&f2);
}
```

Yousaf

Page: Date: / /

Explanation:

Inside Department and inside the addFaculty function:

we are not creating any new faculty we are just pointing faculty towards that.

This is aggregation.
we are not creating the faculty rather we are just using it.

In destructor we only destroy ptr array which is an array of pointers there is no value which is being destroyed.

- Inside main we have to send addresses because in addFaculty we require *pointers as parameters.

Yousaf

Date: / /

2) 2nd Way to do aggregation:

The second way is to use an alias (&) variable.

Using the same Faculty class.

```
class Department {
    Faculty & Fac;
public:
    Department (Faculty & F) : Fac(F) { }
};
```

Explanation:

This code now takes in a Faculty object and makes an alias.

now in main we only will pass objects, no pointer, no reference as inside parameters we will get the reference.

- This way the code for only one faculty member.

- This only work for single because an array of reference is not possible.

Date: / /

- Also very important:
  we have to initialize the reference variable in member initializer or it cannot be created without initializing.

- Do examples on your own or look at example 6 from composition