Types of relationships:-

عرفنا ان عندنا حاجه اسمها objects ,class وان ممكن يكون عندي اكتر من class في الprogram بتاعي وان ممكن اعمل connection بين الclasses وبعضها واعمل object من class و استدعيه في class تاني وهكذا ...
ومن هنا ظهر توبيك ال types of relationships.

اول نوع عندنا هو ال association relationship :
ودا كل فكرته ان بيبقي عندي (class A , class B)
وهعمل من كل class —> object

1. Unary

بتبقي العلاقه ماشيه ف اتجاه واحد
ودا فيه ان هعمل اوبجيكت من احد الكلاسيز ف الكلاس التاني (نفترض اننا هنعمل اوبجيكت من class A جوا class B)
وبالتالي كدا ال class B بيعرف حاجات عن ال class A لكنه مش بيمتلكه او بيأثر عليه
وبتتمثل بالشكل دا وناخد بالنا مش شكل السهم لانه بيختلف من نوع لنوع

ودهالمثال بالكود

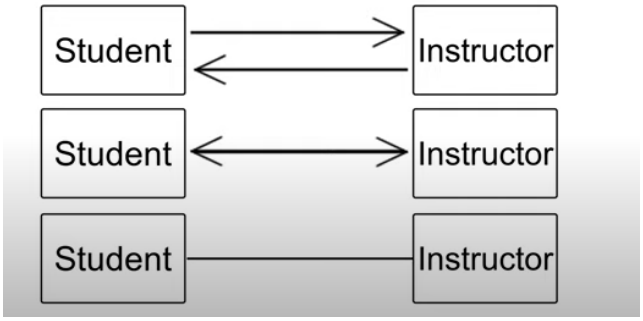
```
class Library {
    String name = "Central Library";
}

class Student {
    Library library; // Association: Student uses Library

    Student(Library lib) {
        this.library = lib;
    }
```

2. Binary

ودي ريليشن في الاتجاهين , وبتتمثل باكتر من شكل زي كدا



وهي ان اعمل اوبجيكت من الكلاسين جوا بعض ونفس الفكره برضو هما بيعرفوا عن بعض لكن مفيش ملكيه ل كلاس علي التاني زي كدا

```
class Teacher {
    String name;
    Student student;
}

class Student {
    String name;
    Teacher teacher;
}
```

وهنا بقي ممكن العلاقه تكون ->

One to one , many to many , one to many

النوع التاني وهو ال agregation:-

هي نوع خاص من Association بيبقي الاوبجيكت معتمد علي اوبجيكت تاني بس مش اعتماد كلي

➕عندنا ال (company) و ال (employee) كل كلاس منهم مستقل بذاته لكن عدم وجود واحد مش هياثر علي وجود التاني .

بمعني تاني ان انا عندي ال company جواها employees كتير ف هي اي نعم معتمده علي وجود الموظف بس عادي لو واحد مشي وساب الشركه في غيره او يعني الشركه مش هتقع .

ودي علاقة "has-a" وبتترسم بالشكل دا

وهنا ال class A بيعبر عن الwhole اللي هو الكلاس الكبيير او الاب(company)

ودا مثال بالكود :

```java
import java.util.*;

// كلاس الموظف
class Employee {
    String name;
    int id;

    Employee(String name, int id) {
        this.name = name;
        this.id = id;
    }

    void showInfo() {
        System.out.println("Employee Name: " + name + ", ID: " + id);
    }
}

// كلاس الشركة
class Company {
    String name;
    List<Employee> employees;

    Company(String name, List<Employee> employees) {
        this.name = name;
        this.employees = employees;
    }
```

النوع التالت وهو ال composition : -

دا بقا بيبقي الكلاس معتمد بشكل كلي علي الكلاس التاني

ودي علاقه has a

بتترسم بالشكل اللي تحت دا ومعناها ان class B بيعتمد كليا علي ال class A

ناخد مثال بالعربي 👍

لو انا قولت ان انا عندي كتاب وكل كتاب بيتكون من شباتر

هلوجود الشباتر بتعتمد علي وجود الكتاب ؟

اه لان انا لو مسحتت الكتاب مبقاش عندي شباتر وبالتالي وجود الشباتر مرتبط بوجود الكتاب

اما في النوع اللي فات كان وجود الموظف معتمد علي وجود الشركه ؟ لا الشركه لو قفلت الموظف هيشتغل ف حته تانيه ولو الوظف مشي غيره هيجي والشركه هتكمل عادي ف اي نعم الموظف مهم للشركه لكن تاثيره مش كلي (اعتماد جزئي)

ناخد مثال بالكود 👍

```java
class Chapter {
    String title;
}

class Book {
    private List<Chapter> chapters;

    Book() {
        chapters = new ArrayList<>();
        chapters.add(new Chapter());
    }
}
```

يبقي نلم لمه بقا 😀

➕ال association بيبقي ال;لاسين مش معتمدين علي بعض خالص ، بيبقوا عارفين عن بعض بس (هستخدم اوبجيكت من كلاس ف كلاس تاني)

ممكن تكون unary لو كلاس واحد منهم اللي يعرف عن التاني (لو عملت اوبجيكت من واحد واستخدمته ف التاني لكن معملتش من التاني ف الاول)

وممكن يبقي binary لو العلاقه رايحه جايه الاول عنده اوبجيكت من التاني والتاني عنده اوبجيكت من الاول)

➕ال agregation بيبقي الكلاس الصغيربياثر علي الكبير بس مش بشكل كلي يعني هياثر اه لكن هيعتمد بشكل كلي لا ودا زي مثال الموظف والشركه

➕ال composition بيبقي الاعتماد كلي ووجود التاني بيثر بشكل كلي علي وجود الاول زي مثال الكتاب والشباتر كدا .

Final Keyword: -

دي keyword بتتكتب قبل ال(variable , parameter,method , class)

كحاجه بتمتنع التعديل عليها يعني لو بصينا علي الفاريبل 👍

الفاريابل العادي بيتكتب كابتل او اسمول لكن الfinal int VARIABLE كل حروفه بتبقي كابتل وقدام حطيت قبله الكيوورد دي معناها ان دي خر قيمه هتتحفظ في المتغير دا ومش هقدر اغيرها او اعدل عليها تاني

ملحوظه هي مش زي ال access modefire لا انا هقدر اوصلها اه لكن مش هقدر اعدل عيهاا

هي عامله زي ال const variable كدا

اي الفرق بين ال final , const؟ 🤔

اول حاجه مفيش حاجه اسمها كونست في الجافا ف بنستخدم بدالها ال فاينل بس فعليا الكونست عباره عن static final

بمعني ان الفاينال لوحدها لو للفاريابل معناها الفاريابل دا ثابت لكل اوبجيكت لوحده يعني كل اوبجيكت هيبقي عنده فاريابل مش هيقدر يعدل في قيمته

اما ال كونست اللي هي فعليا بتساوي ال static final دي معناها ان الفاريابل دا مشترك لكل الاوبجيكتس ومش هقدر اعدل عليه .

Note: -

في تعريف ال final variable

```
Final int VAR_A=15;              // التعريف في نفس السطر
Final int VAR_B;
VAR_B=12;      //  blank final var هنا التعريف من خلال الاستستمينت ف دا اسمه
```

وهنا لو التعريف عنجي هتبع طريقه ال blank عندنا طريقتين :

```
Final int VAR_B;      // اكتب دي عادي
{VAR_B=12;}           // ادي القيمه الابتدائيه جوا بلوك
```

الطريقه التانيه ان ادي الاقيمه الابتدائيه جوا كونستراكتور

```
Public class Student{
Final int VAR_M;
student(){
VAR_M=20;
}
```

👍ال block ليه الاولويه عن ال constructor يعني بيتنفذ الاول .

طب لو عايزاه static blank final variable ؟

مفيش غير طريقه واحده لتعريفه وهي ال static block .

```
Public class Student{
Final  static int VAR_M;
Static {
VAR_M=20;
}                                    // بالطريقه دي
```

يعني اي بقا final parameter

عندي ميثود او كونستراكتو عادي بتستقبل باراميتيرز وجيت ف ال body عملت تغير علي احد الباراميتيرز دي وغيرتها وجيت اطبعها هنا هيطبعلي القيمه بعد التغيير عادي

انا بقا مش عايزه دا يحصل ف هعمل اي ؟؟ اعمل الباراميتير دا final بحيث مقدرش اعدل عليه

```
Public student (int ID , String Name){
ID=0;      //  انا هنا غيرت القيمه اللي اليوزر هيدخلها وخليتها بصفر
}
```

الحل اي بقا علشان مسمحش بالحركه دي ؟؟ استخدم ال final keyword

```
Public student (final int ID , String Name){
ID=0;      // هيجيبلي ايرور علي السطر دا لانه مش مسموح
}
```

Final methods : -

لوعندي كلاس الاب فيه ميثود معينه وكلاس الابن وارث من الاب (وارث كل الاتتربيوتس و الميثودز)دا العادي ف الوراثه

انا بقا عايزه اعمل سيكيوريتي علي الميثود المعينه دي و مخليش اي حد يقدر يعمل عليها override

اقوم اعمل اي ؟؟ احطلها ال keyword final

مثاااال 👍

```
// كلاس الأب
class Animal {
    public final void makeSound() {
        System.out.println("Animal makes a sound");
    }
}
// كلاس الإبن
class Dog extends Animal {
```

```
@overriding                                  // لأن الميثود final ده هيعمل Error
// public void makeSound() {
//     System.out.println("Dog barks");
// }
}
public class Main {
    public static void main(String[] args) {
        Dog d = new Dog();
        d.makeSound();                  // هيطبع: Animal makes a sound
    }
}
```

نفس الكلام في ال final class

ان الكلاس دا مينفعش يتورث

Example 👍

```
final class Vehicle {
    public void start() {
        System.out.println("Vehicle is starting...");
    }
}

class Car extends Vehicle {    // errooooooooorrrrrrrrrrr
    public void start() {
        System.out.println("Car is starting...");
    }
}
```

يبقي نلم لمه 😂

Final variable

دا مينفعش اعدل علي قيمته وممكن اعمله انيشياليزيشن في نفس سطر الانشاء

Final blank variable

دا مينفعش اعدل علي قيمته وممكن اعمله انيشياليزيشن بطريقتين البلوك او الكونستراكتور لو مش استاتتيك , لكن لو استاتيك هعمله انيشياليزيشن بالاستاتيك بلوك بس

Final parameter

مش هقدر اعدل علي البراميتير اللي اليوزر هيدخله

Final method

مش هقدر اعمل overriding علي الميثود دي

Final class

مش هقدر اورث الكلاس دا لاي كلاسيز تانيه

By : MarTina Mina