OOP

1 **Typed languages**

String, number, boolean : primitive
string[], number[]
type MyTpe { name: string, age: number }

2 Object/Class

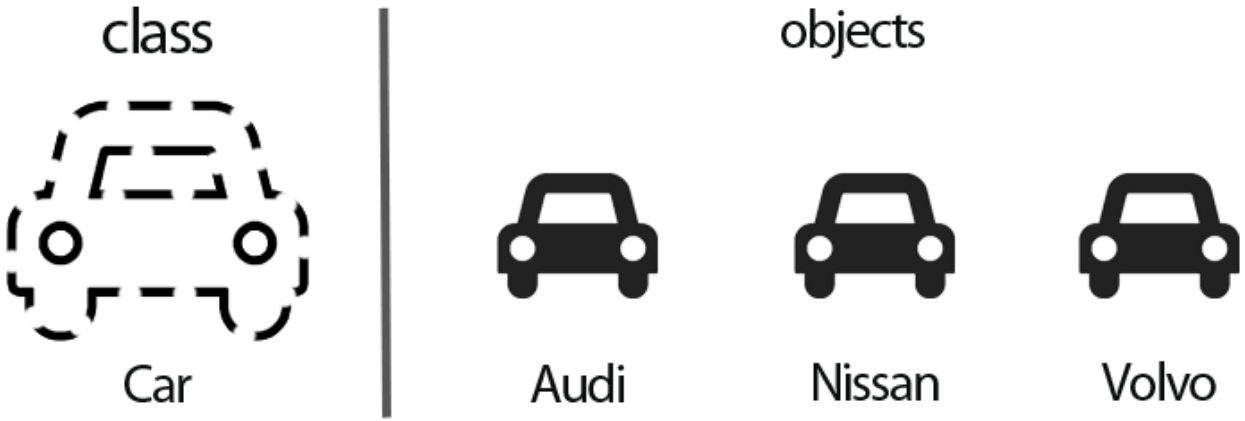
3 **Encapsulation / Aggregation**

4 Polymorphism

5 Abstraction

SO FAR YOU KNOW HOW TO
CREATE AN OBJECT FROM A CLASS

```
Let myCar = new Car("audi");
```



Does the constructor have a return type ?

```
class User {
  name: string;

  constructor(name: string) {
    this.name = name;
  }

  profile() {
    console.log(`User name: ${this.name}`);
  }
}
```

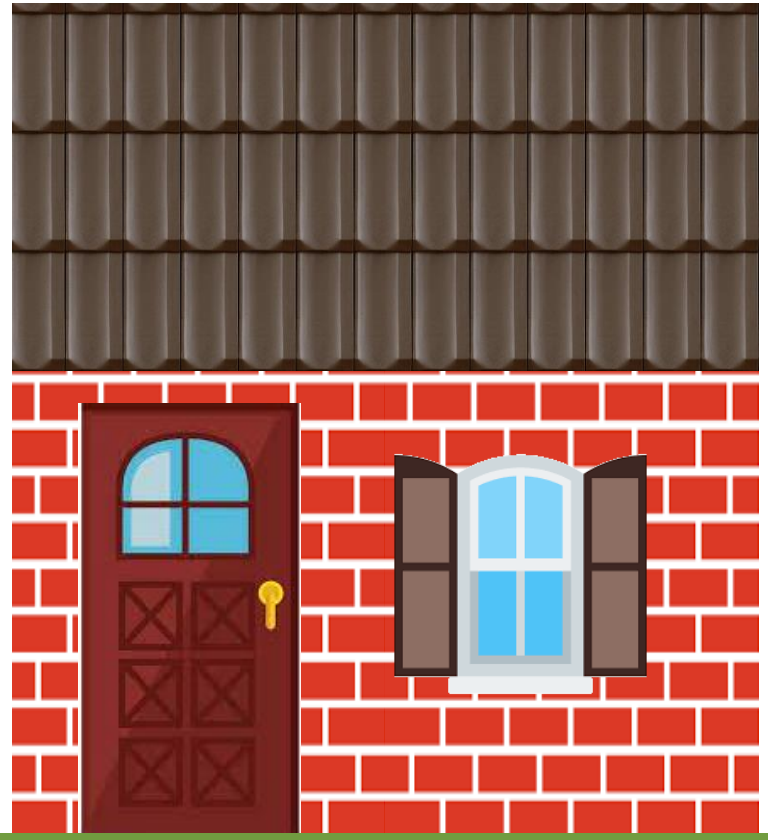
A - YES B - NO

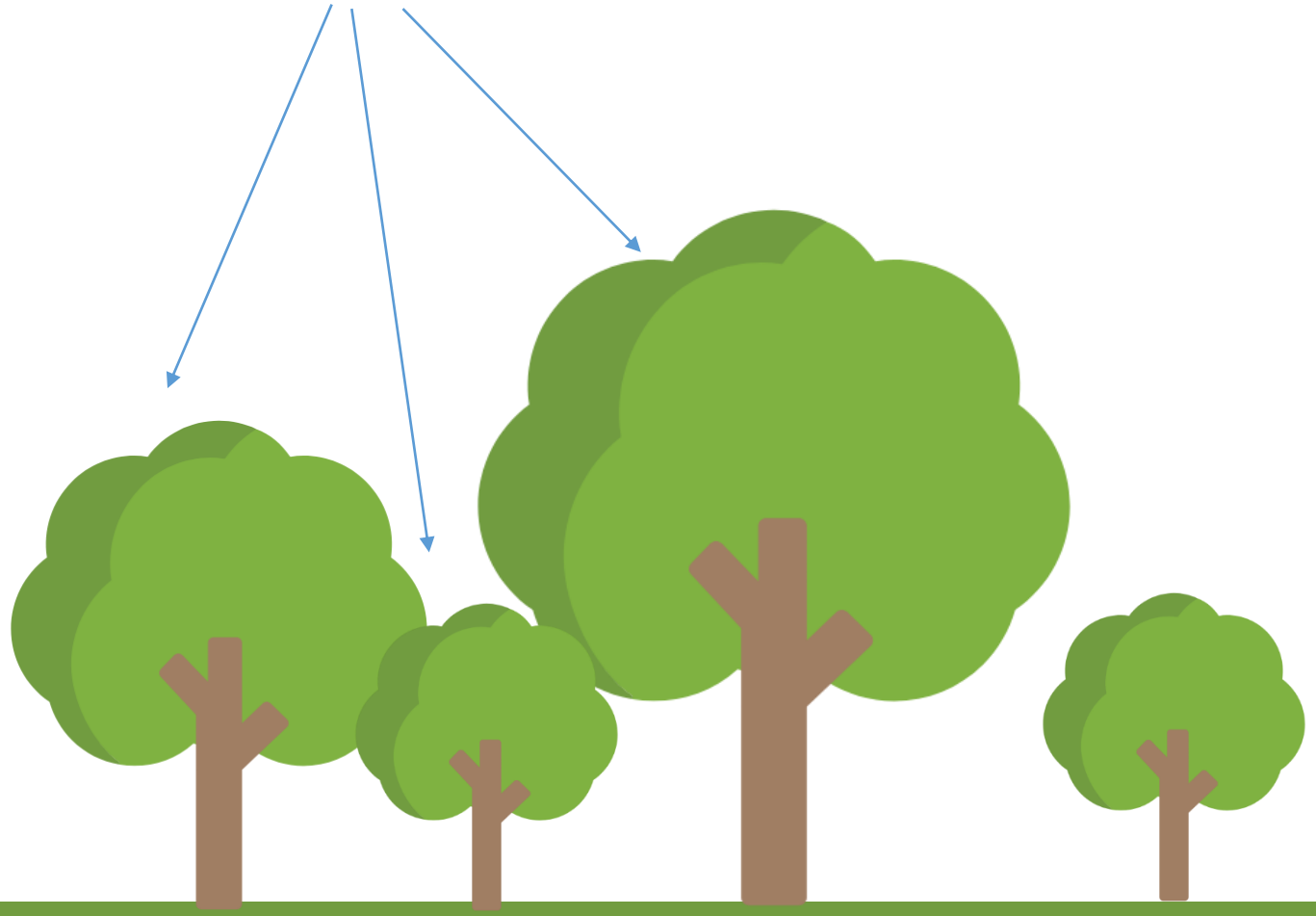
NOW LET’S COMBINE
OBJECTS TOGETHER

AGREGATION

A house can have MANY tree

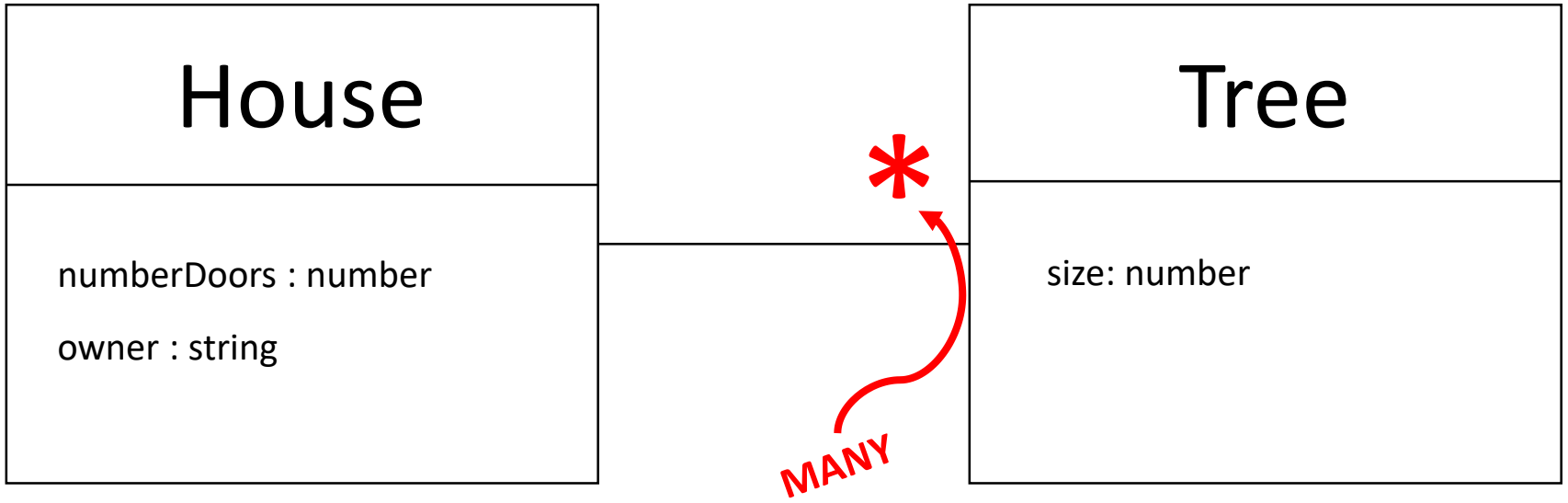
Each tree has a specific SIZE





In OOP, we represent these 2 classes as follows:

Class diagram



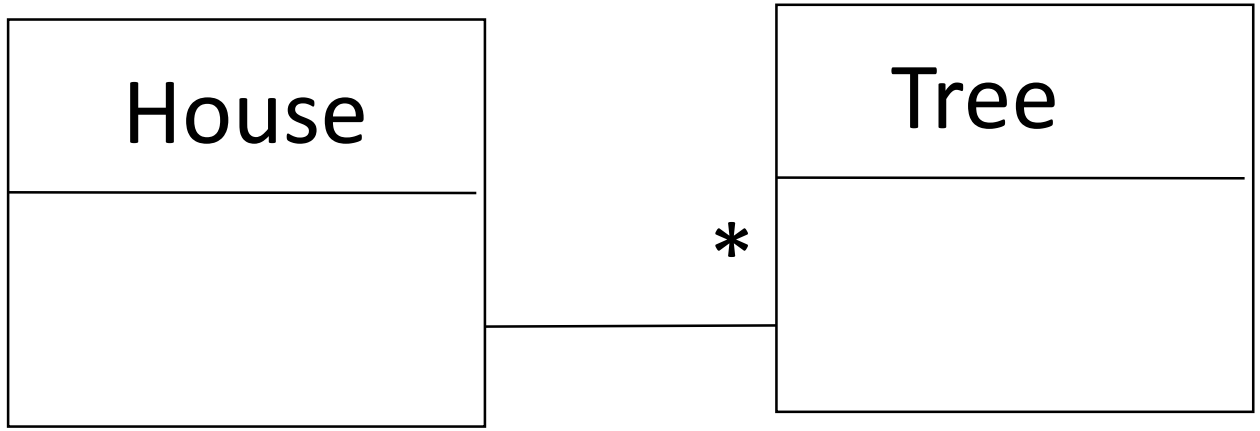
In typescript , we represent these 2 classes as follows:

Code :

```
class House {
  numberDoor: number;
  owner: string;

  constructor(numberDoor: number, owner: string) {
    this.numberDoor = numberDoor;
    this.owner = owner;
  }
}
```

```
class Tree {
  size: number;

  constructor(size: number
    this.size = size;
  }
}
```

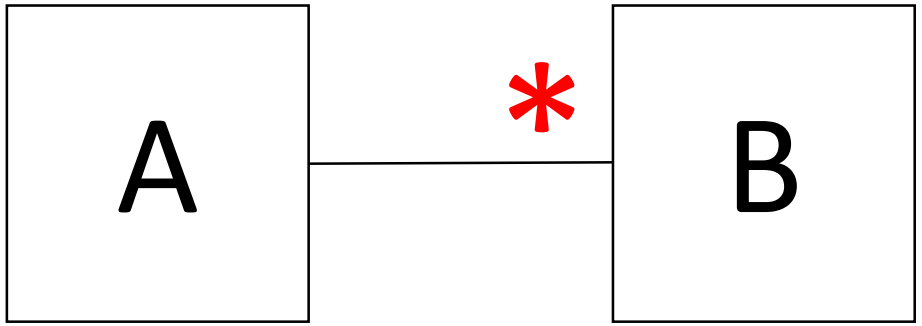
ACTIVITY 1




```
//  1-
 update the class to allow the house
to have "many" tree


// 2-
 Add the 2 trees to the houseRonan
```

‘A is composed of many B’

Class diagram



Code :

```
class A {
  theB: B[];
}
```

```
Class B {
}
```

VERSION 1

Add an objet to another objet

```
class House {
  allRooms: Room[] = [];
}

// Create a house
let house1 = new House();

// Create a room
let room1 = new Room();

// Add the room to the house
house1.allRooms.push(room1);
```

By default the list of room
Is empty

We add the room to the house

VERSION 2

Add an objet to another objet

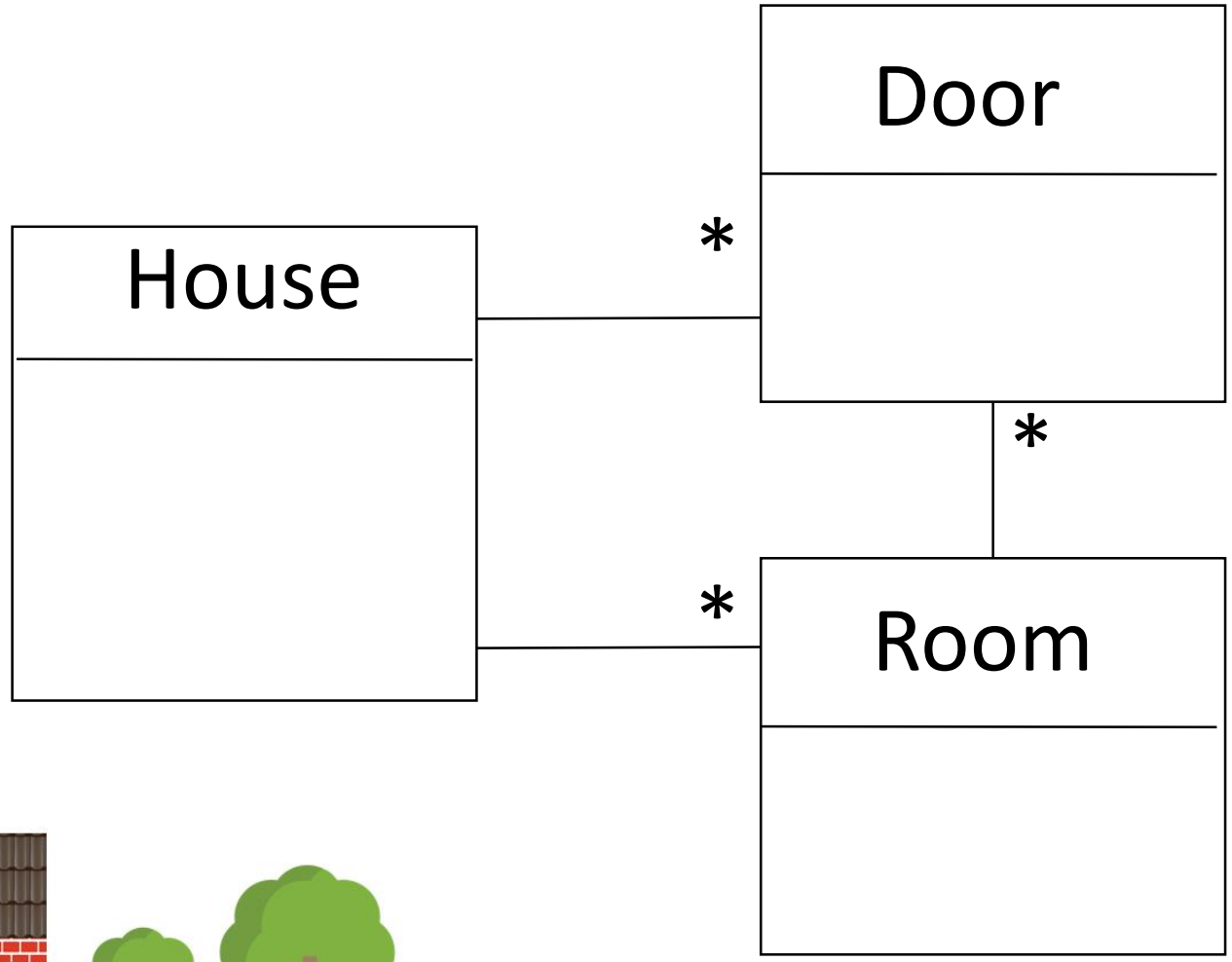
```
class House {
  allRooms: Room[] = [];

  addRoom(room: Room) {
      this.allRooms.push(room)
  }
}

// Create a house
let house1 = new House();

// Create a room
let room1 = new Room();

// Add the room to the house
house1.addRoom(room1);
```

By default the list of room
Is empty

We create a method inside the class
To add the room

We add the room to the house

ACTIVITY 2




```
//  1- Create:
 - 1 house
 - 3 doors
 - 2 rooms
```

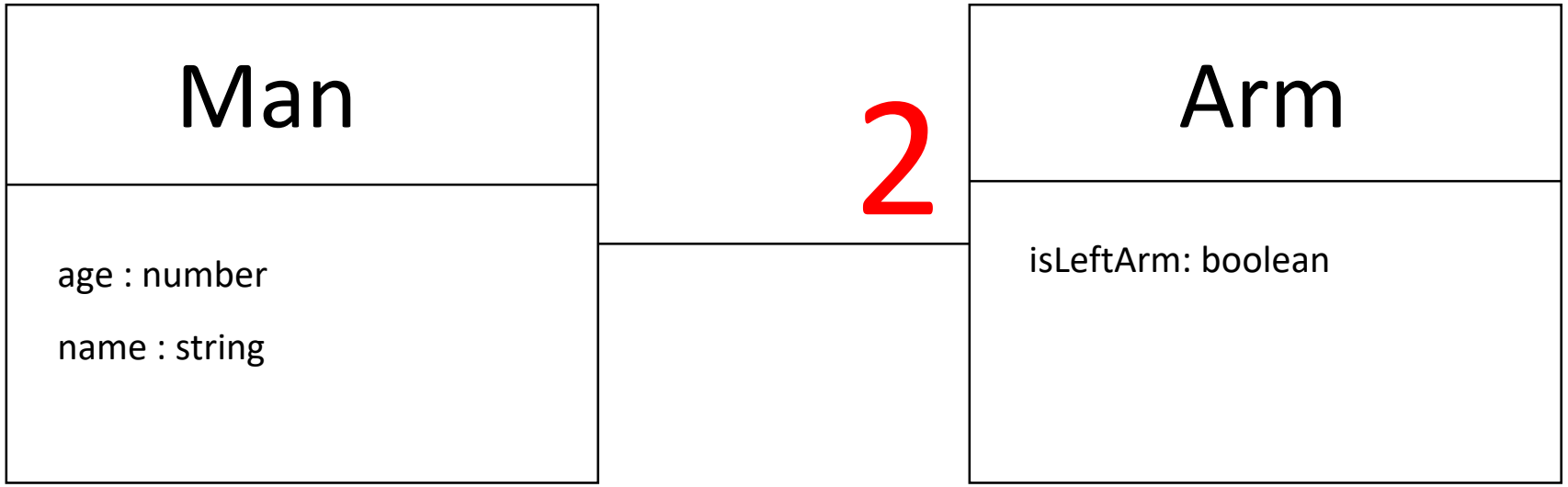
```
// 2- Add the 2 rooms to the house
```

```
// 3 – Add a door to the house and rooms
```

A man has **1 left** arm and **1 right** arm

In OOP, we represent these 2 classes as follows:

Class diagram

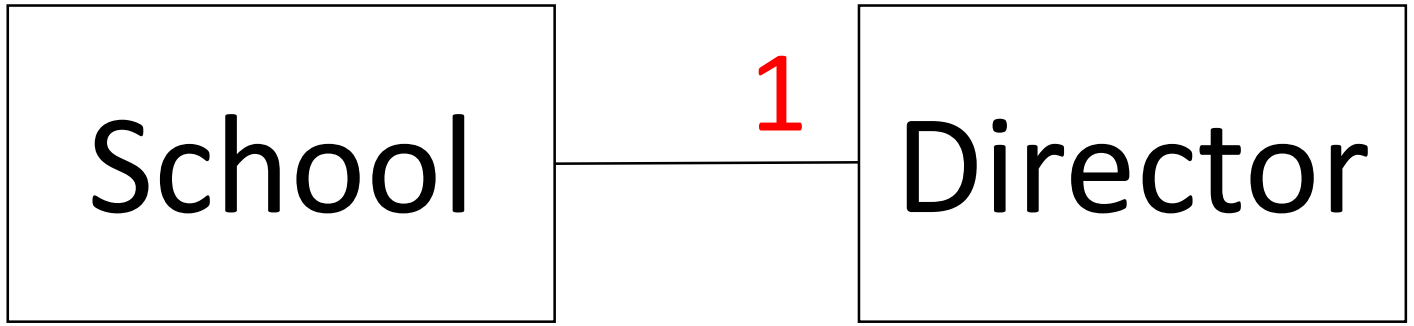


In typescript , we represent these 2 classes as follows:

```
class Man {
  leftArm: Arm = new Arm(true);
  rightArm: Arm = new Arm(false);

}
```

```
class Arm {
  isLeft: boolean;


  constructor(isLeft: boolean){
       this.isLeft = isLeft
  }
}
```

‘A has 1 B’

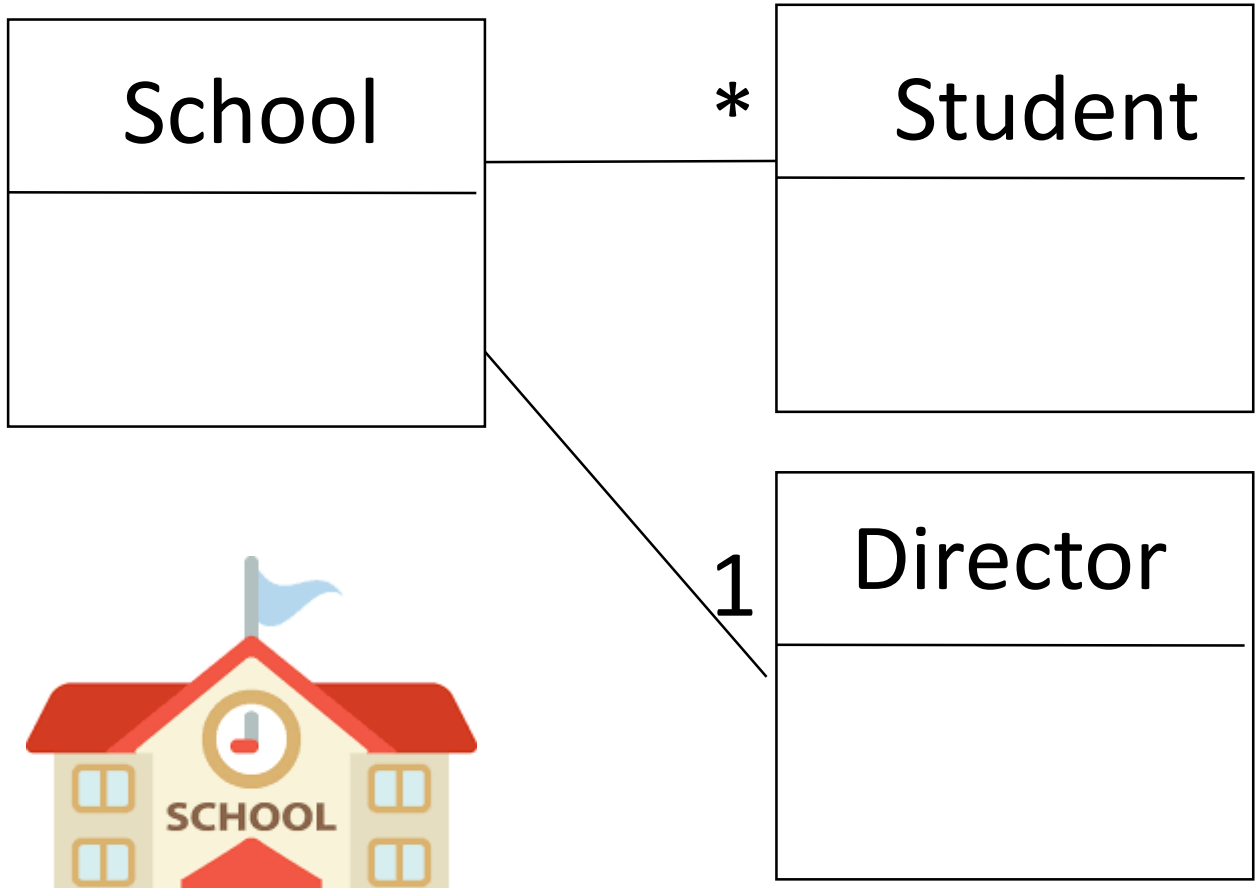



```
class School {
  director: Director;
}
```

```
Class Director {
}
```

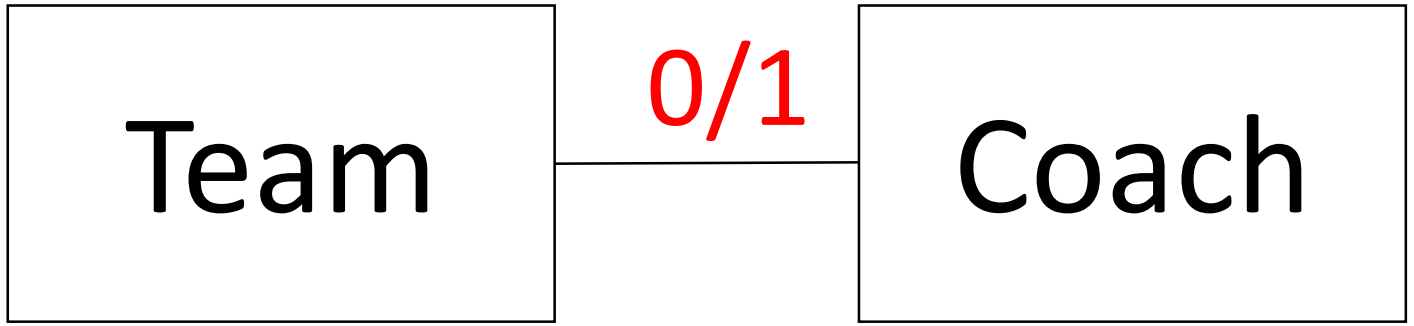
ACTIVITY 3

```
//  1- Update the classes to mange
//          - a school has many students
//          - as school has 1 director
```




```
// 2 - Create a school with a director,
 and students
```

A team can have or not a coach

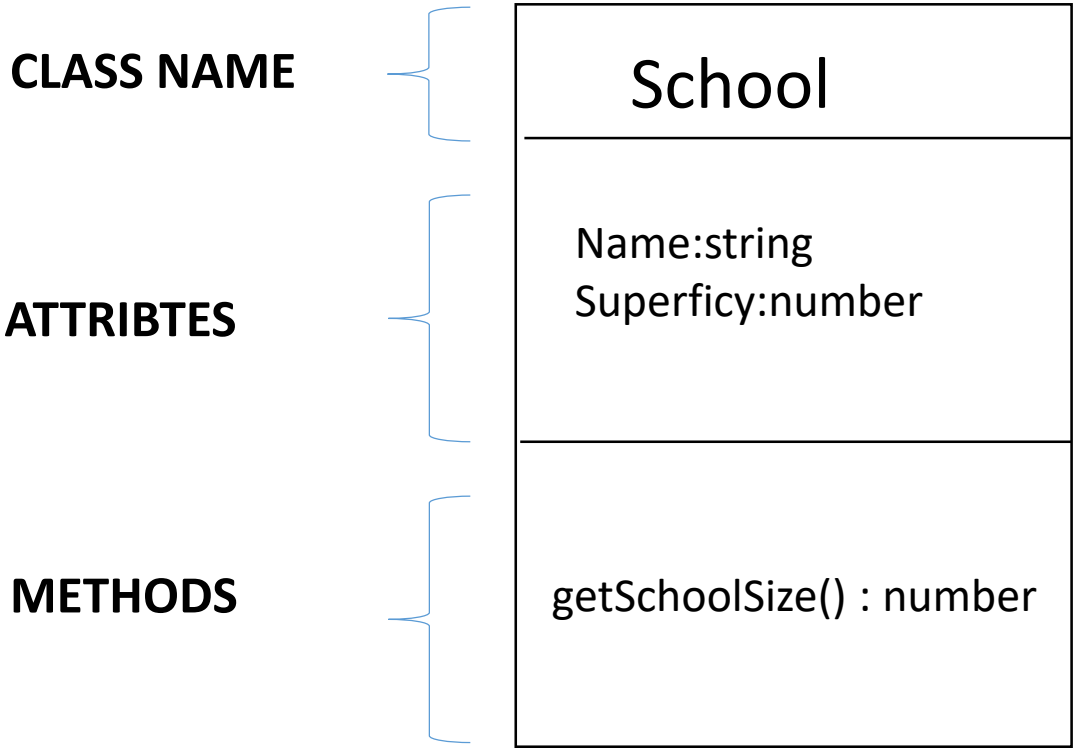



```
class Team {
  coach?:  Coach;
}
```

```
Class Coach {
}
```

The coach
Is optional

What is a UML class diagram ?

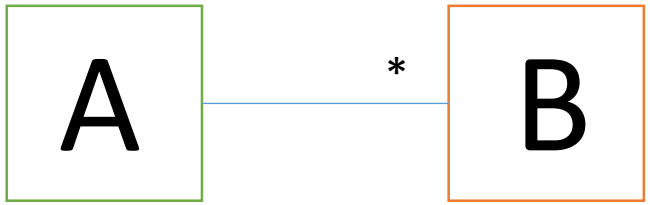


- ✓ A way to represent the class graphically
- ✓ Similar to ERDs

TO SUM UP

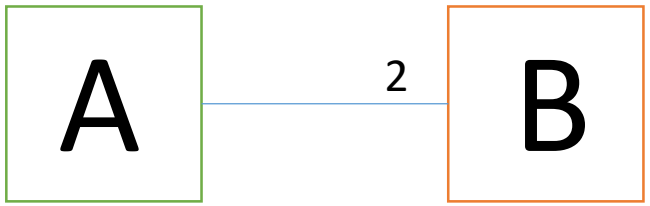
We saw **3 types of aggregations**

"A HAS MANY B"



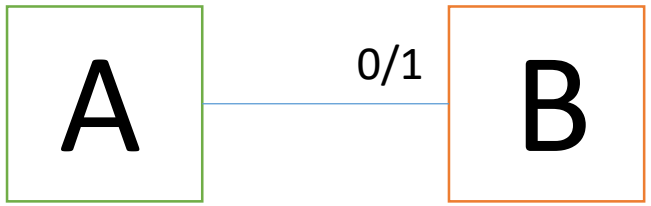
We create an array
In A

"A HAS ALWAYS 2 B"



We create 2 attributes B
In A

"A HAS 0 or 1 B"



We create 1 attribute B
In A
But it is optional

WANT TO GO FURTHER ?

CLASSES & OBJECTS IN TYPESCRIPT

https://www.typescriptlang.org/docs/handbook/classes.html