PROBLEM: **How to implement the below relationships in the following use cases?**

Relationships

- **One to one**
- **One to many**
- **Many to One**
- **Many to Many**

Use case 1: **Database Schema modelling**

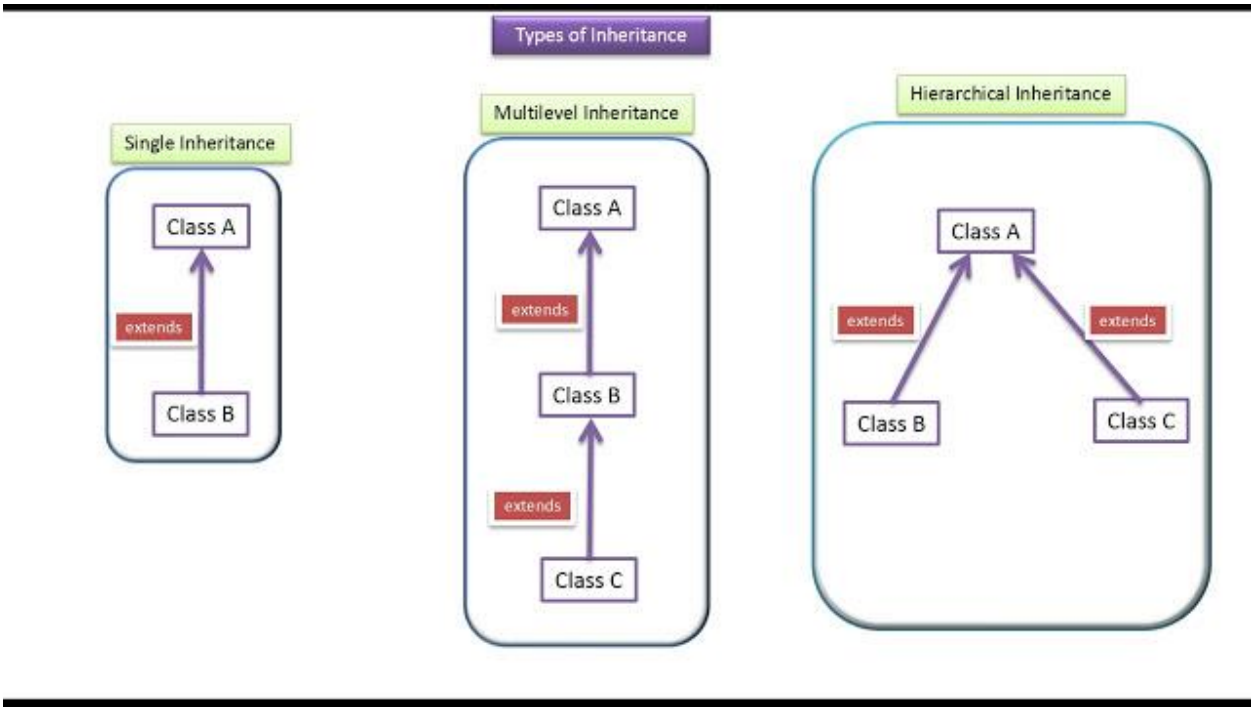
Use case 2: **JSON Schema modelling for REST API's (Microservices)**

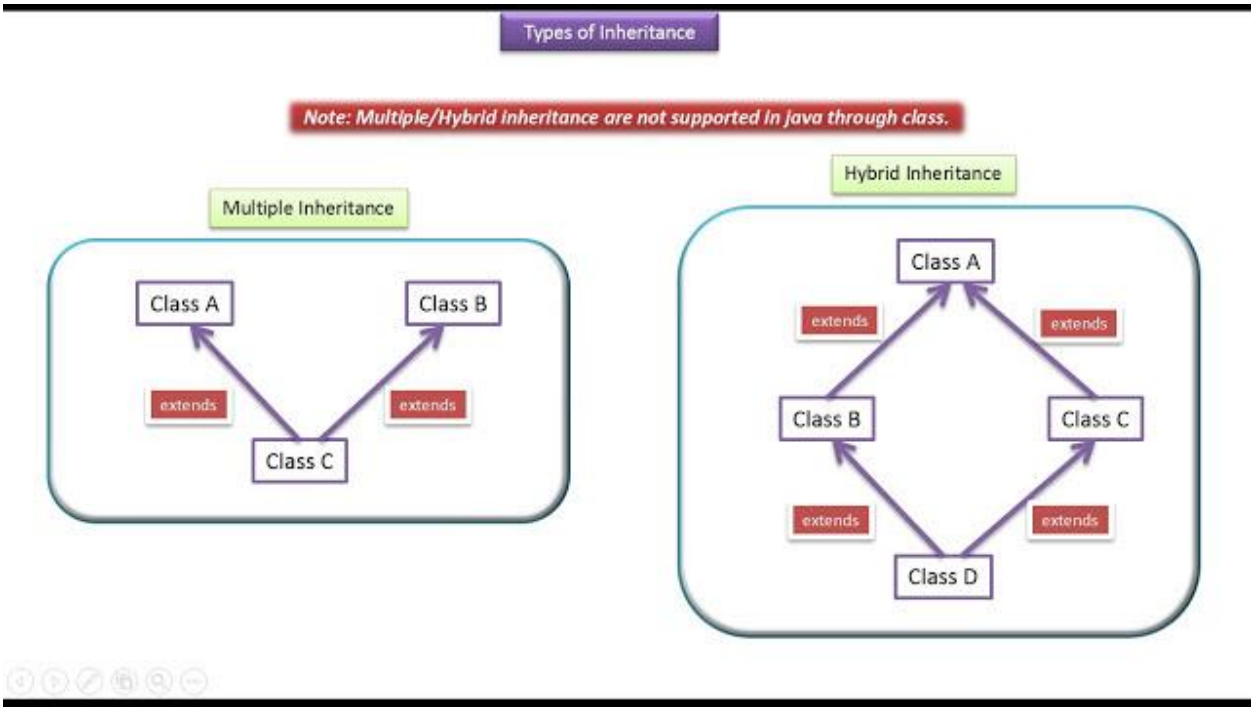
Use case 3: **XML Schema modelling for SOAP API's**

Use case 4: **Entity Objects in DAO Layer/ DTO objects to display data in UI**

Use case 5: **Code reusability while implementing the business logic**

INHERITANCE TYPES

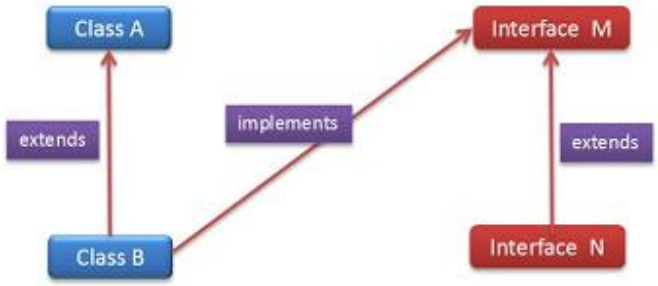




WHAT IS INHERITANCE?

Java Inheritance(IS-A)

- ✓ Inheritance is one of the key features of Object Oriented Programming. Inheritance provided mechanism that allowed **a class to inherit property of another class**.
- ✓ When a Class extends another class it inherits all non-private members including fields and methods
- ✓ Inheritance in Java can be best understood in terms of Parent and Child relationship, also known as **Super class**(Parent) and **Sub class**(child) in Java language.
- ✓ Inheritance defines **is-a** relationship between a Super class and its Sub class. extends and implements keywords are used to describe inheritance in Java.



WHY DO WE NEED INHERITANCE?

Java Inheritance(IS-A)

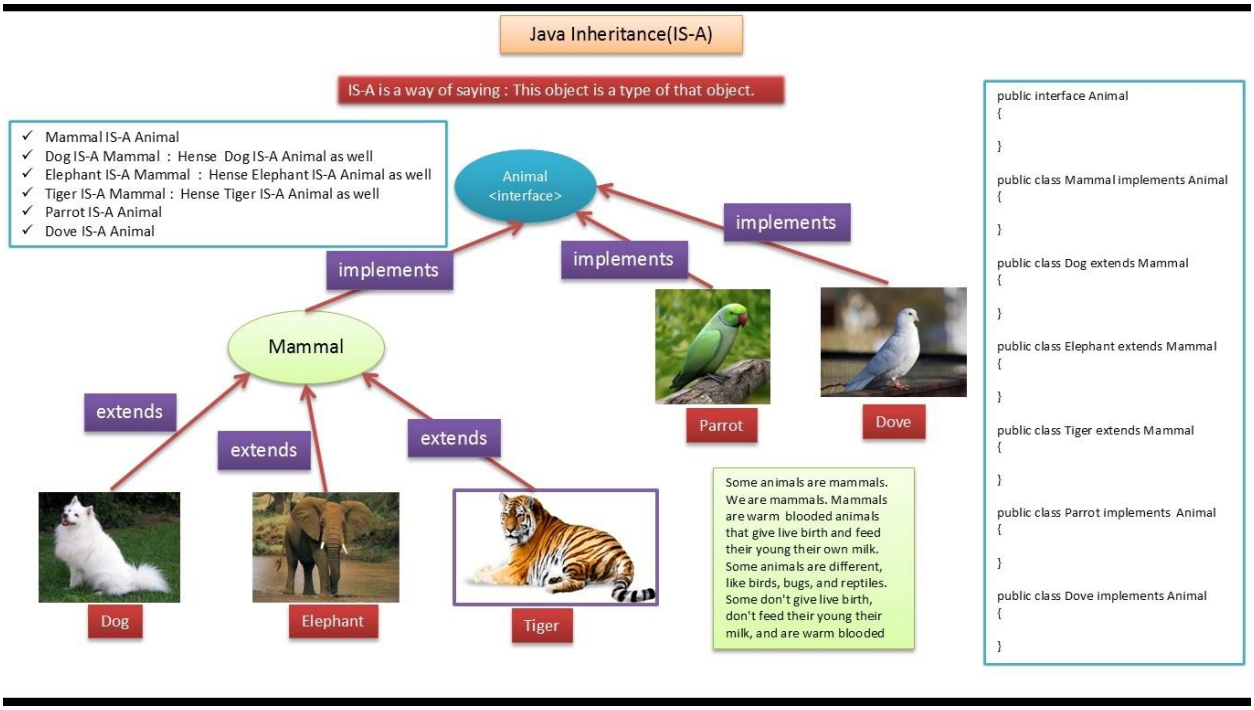
```
class Vehicle.
{
  ......
}
class Car extends Vehicle
{
 .......  //extends the property of vehicle class.
}
```

Vehicle is super class of **Car**.
Car is sub class of **Vehicle**.
Car IS-A **Vehicle**.

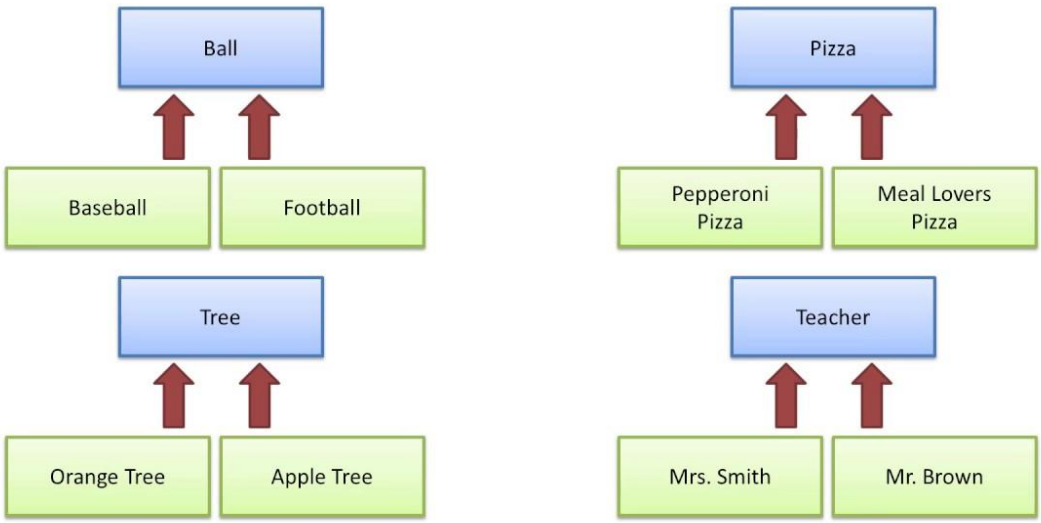
Purpose of Inheritance

- ✓ To promote code reuse.
- ✓ To use Polymorphism.

EXAMPLES


Java Inheritance(IS-A)
IS-A is a way of saying : This object is a type of that object.
✓ Mammal IS-A Animal
✓ Dog IS-A Mammal : Hense Dog IS-A Animal as well
✓ Elephant IS-A Mammal : Hense Elephant IS-A Animal as well
✓ Tiger IS-A Mammal : Hense Tiger IS-A Animal as well
✓ Parrot IS-A Animal
✓ Dove IS-A Animal
Animal <interface>
implements
implements
implements
Mammal
extends
extends
extends
Dog
Elephant
Tiger
Parrot
Dove
Some animals are mammals. We are mammals. Mammals are warm blooded animals that give live birth and feed their young their own milk. Some animals are different, like birds, bugs, and reptiles. Some don't give live birth, don't feed their young their milk, and are warm blooded
public interface Animal
{
}
public class Mammal implements Animal
{
}
public class Dog extends Mammal
{
}
public class Elephant extends Mammal
{
}
public class Tiger extends Mammal
{
}
public class Parrot implements Animal
{
}
public class Dove implements Animal
{
}


INHERITANCE: SUPERCLASS AND SUBCLASS


Ball
Baseball
Football
Pizza
Pepperoni Pizza
Meal Lovers Pizza
Tree
Orange Tree
Apple Tree
Teacher
Mrs. Smith
Mr. Brown

HOW MANY WAYS WE CAN CODE INHERITANCE?

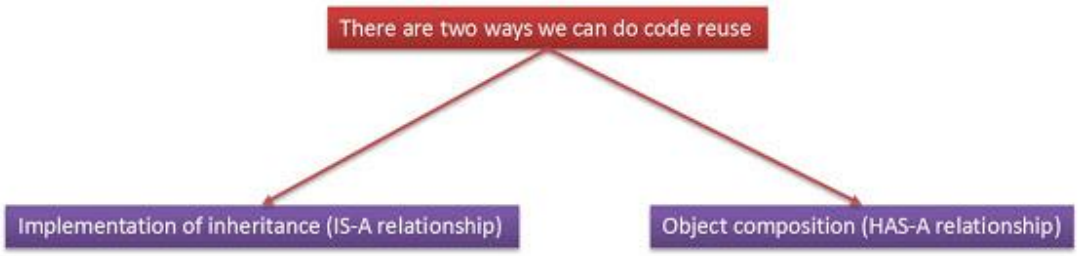
Java Inheritance(IS-A)

IS-A Relationship:

- ✓ In object oriented programming, the concept of IS-A is a totally based on Inheritance, which can be of two types
  1. Class Inheritance
  2. Interface Inheritance
- ✓ IS-A relationship is just like saying "A is a B type of thing".
  For example, Apple is a Fruit, Car is a Vehicle etc. Inheritance is uni-directional. For example House is a Building. But Building is not a House.
- ✓ We can easily identify the IS-A relationship. Wherever we see an extends keyword or implements keyword in a class declaration, then this class is said to have IS-A relationship.

Java Inheritance(IS-A)

- ✓ One of the advantages of Object-Oriented programming language is code reuse.
- ✓ There are two ways we can do code reuse either by implementation of inheritance (IS-A relationship), or object composition (HAS-A relationship).
- ✓ The compiler and Java virtual machine (JVM) will do a lot of work for you when you use inheritance, you can also get at the functionality of inheritance when you use composition.



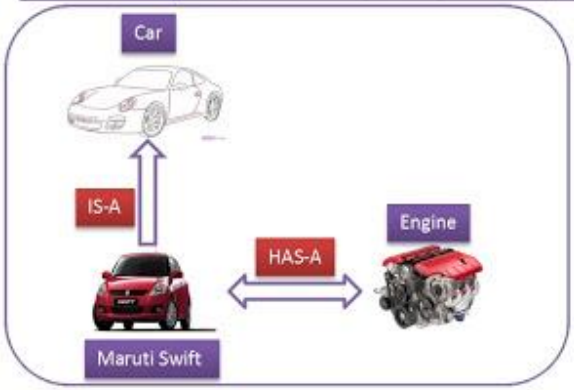
Aggregation (HAS-A)

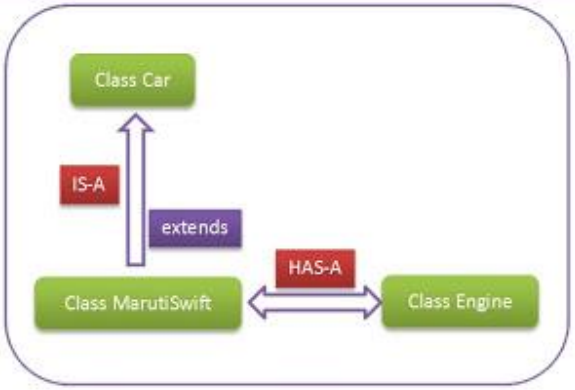
Why and when to use Aggregation?

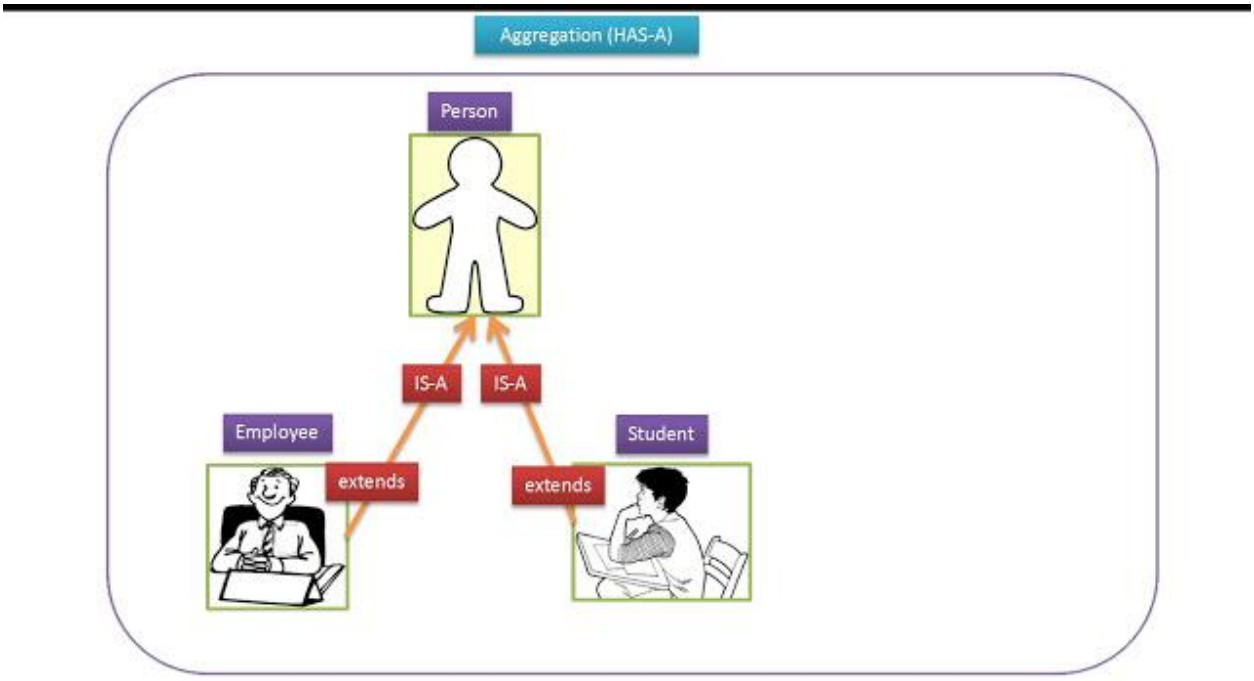
- ✓ For Code Reusability.
- ✓ Code reuse is also best achieved by aggregation when there is no IS-A relationship.
- ✓ Inheritance should be used only if the relationship IS-A is maintained throughout the lifetime of the objects involved; otherwise, aggregation is the best choice.

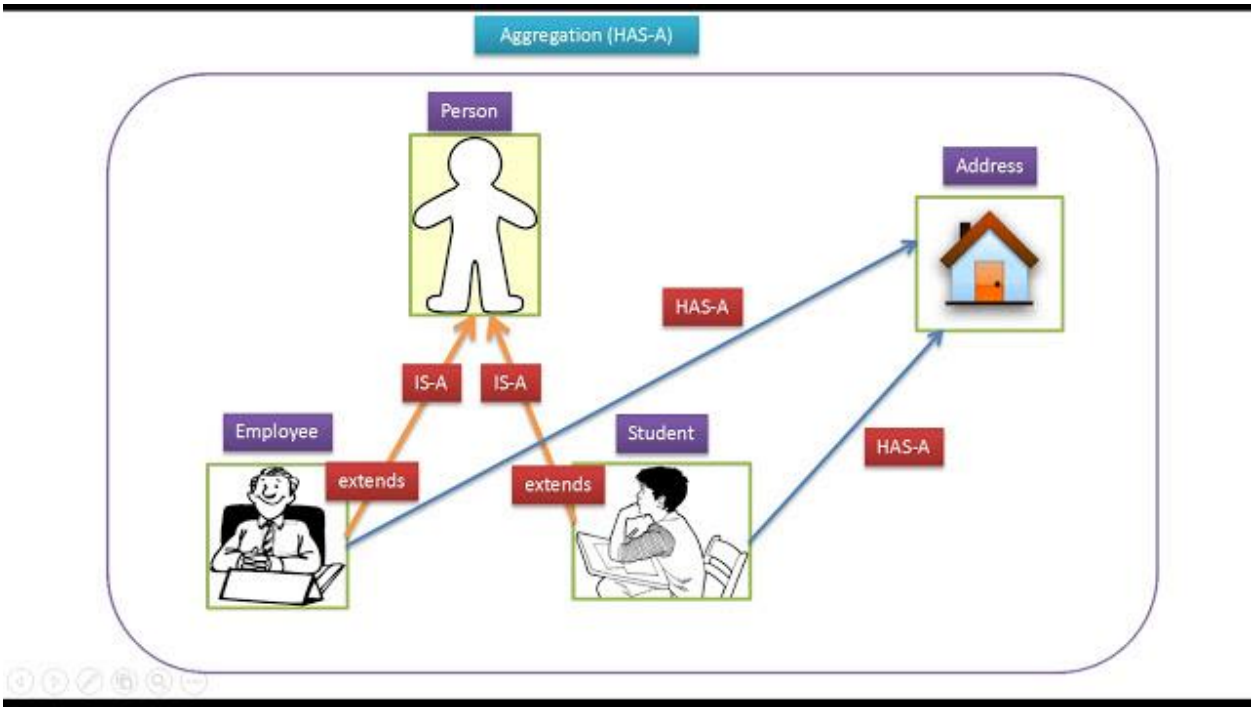
Java Inheritance(Has-A)

- ✓ Composition(HAS-A) simply mean use of instance variables that are references to other objects.
  For example: **MarutiSwift** Car has **Engine**, **House** has **Bathroom** , **Person** has **Address**.
- ✓ Has-A means an instance of one class "has a" reference to an instance of another class or another instance of same class.
- ✓ It is also known as "composition" or "aggregation".
- ✓ There is no specific keyword to implement HAS-A relationship but mostly we are depended upon "**new**" keyword.
- ✓ Has-a relationship is composition relationship which is productive way of code reuse.

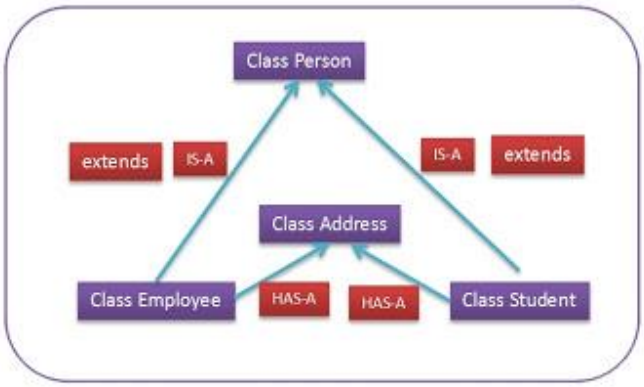





Aggregation (HAS-A)
Person
IS-A
IS-A
Employee
extends
extends
Student


Aggregation (HAS-A)
Person
Address
HAS-A
IS-A
IS-A
Employee
extends
extends
Student
HAS-A

Aggregation (HAS-A)



Aggregation (HAS-A)

- ✓ If a class have an entity reference, it is known as Aggregation. Aggregation represents HAS-A relationship.
- ✓ Employee object contains many information's such as name,age,employeeid, departmentName etc. It contains one more object named address, which contains its own information's such as city, state, country, zipcode etc. In such case, Employee has an entity reference address, so relationship is Employee HAS-A address.

IS-A VS HAS-A RELATIONSHIPS

'Is-a' vs. 'has-a' relationship

In English
This person **has a** name, address, age

In Programming

```
class Person
{
String name;
String address;
int age;
}
```

A class can have references to objects of other classes as members called as **composition or *has-a* relationship**

In English
This person **is a** businessman

In Programming

SUMMARY

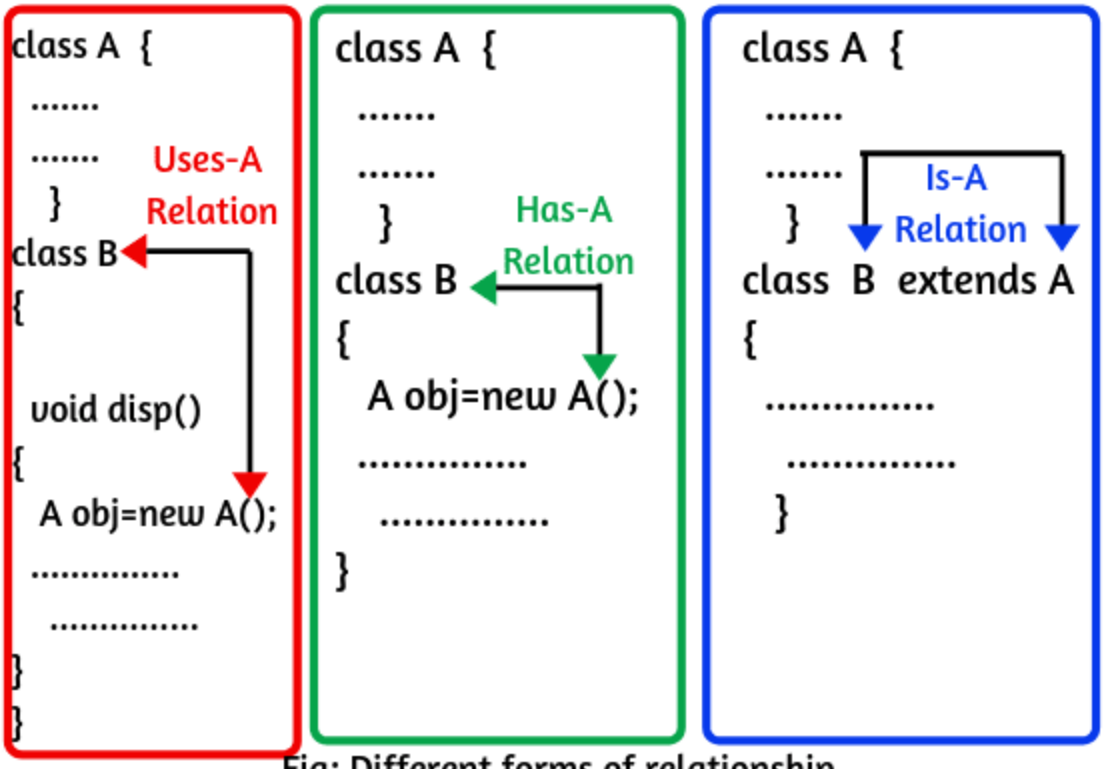


Fig: Different forms of relationship