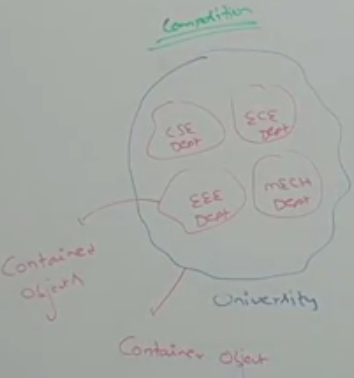
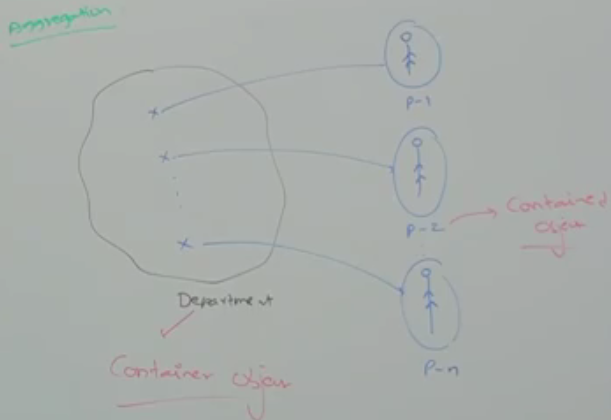
has-a relationship

1. **Also known** as **Composition and Aggregation**.
2. There is no specific keyword (like in case of is-a relationship) to implement this relation. But most of the time, we are depending on **new keyword**.   
   If the services methods are static then we don’t need to use new keyword but use the class name directly.
3. The main advantage of **has-a relationship** is reusability of the code.
4. **Example**:  
   Engine Class, Car Class **has** Engine.
5. **Difference b/w composition and aggregation?  
   NOTE**: Even though both are has-a relationship.  
   **Differenc**e:   
   **Composition**: Without existing container object, if there is no chance of existence of contained objects then container and contained objects are **strongly associated (tightly coupled)** and this strong association is nothing but **composition**.   
   **Example**: University consists of several departments. Without existence of university, there is no chance of existence of department. Hence, University and departments are strongly associated and this strong association is called composition.

  
**Aggregate**: Without existence of Container object, if there is chance of existence of contained objects, then container and contained objects are **weakly associated** and this weak association is nothing but **aggregation**.   
**Example**: Departments consists of several professors. Without existence of a department, there is a chance of existence of professors. Hence, Department and professors are **weakly associated** and this weak association is nothing but **aggregation**.

  
**NOTE**: In **composition**, container object holds directly contained objects. Whereas in **aggregation**, container object holds just references of contained objects.

1. dd

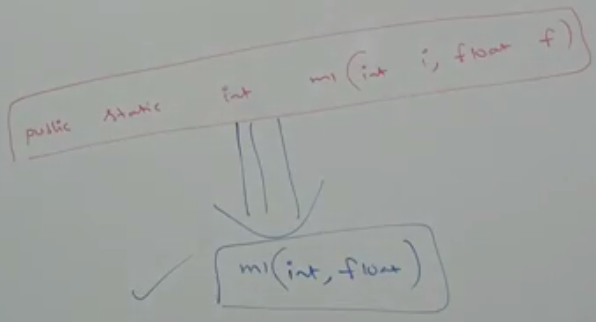
is-a vs has-a relationship

# When to use which one relationship?

**is-a relationship**: When we want **total functionality** of a class automatically, then go for it. Student class requires complete functionality of Person class. So Student extends Person.

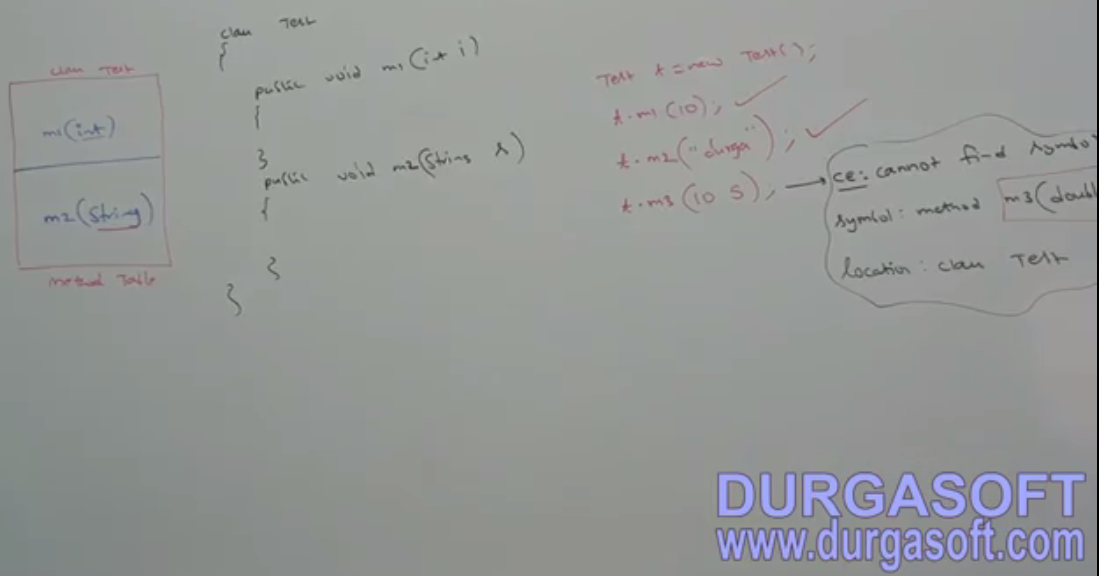
**has-a relationship**: If we want **part of the functionality**, then we should go for it. Test class contains 100 methods but Demo class needs only a few methods.

Method Signature

1. In java, method signature consists of method name followed by argument types  
     
   **NOTE**: In C++, return type is also part of method signature but in java, it’s not.

method signature---🡪

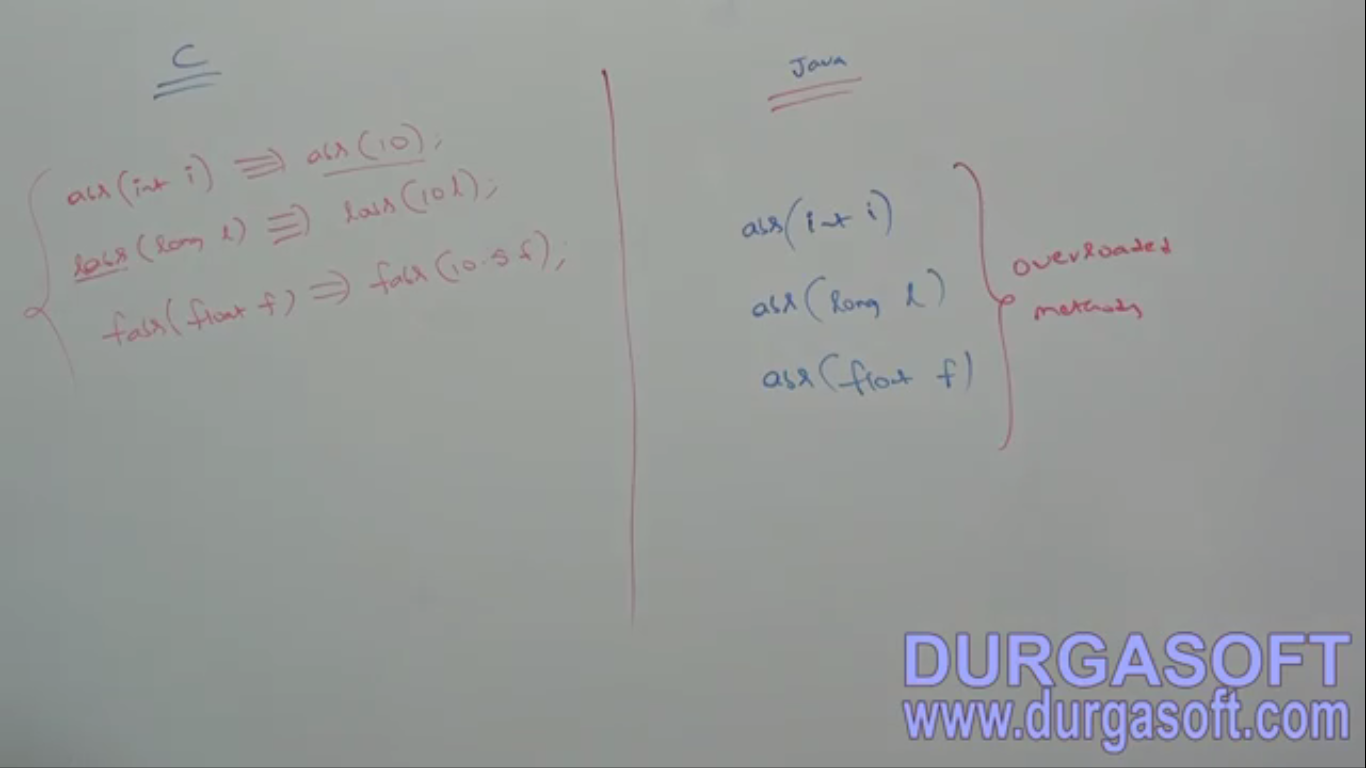
# Who will use method signature?

Compiler will use method signature to resolve method calls.   


See, if method call is not resolved by compiler by using the method signature table, compiler will return error msg but the error msg contains method signature which is not found in method signature table.

Table having method signatures used by compiler.

Overloading

1. C and Java languages in case of overloading   
   

Having overloading concept in java reduces complexity of programming.

C language doesn’t support overloading, so we can’t declare multiple methods with same name but different argument types. If there is change in argument type compulsory we should go for new method name which **increases complexity of programming**.

1. **\*\*\*NOTE**: In overloading, method resolution always is taken care of by compiler based on reference type. Hence, overloading is also considered **compile time polymorphism or static polymorphism or early binding**