Aggregation in Java

If a class have an another class as reference, it is known as Aggregation. Aggregation represents HAS-A relationship.

```
class Employee{
int id;
String name;
Address address;//Address is a class
...
}
```

—

```
public class Address {
String city,state,country;

public Address(String city, String state, String country) {
   this.city = city;
   this.state = state;
   this.country = country;
}

}
```

—--

Composition

Composition is a "**belong-to**" type of relationship in which one object is logically related with other objects. It is also referred to as "**has-a**" relationship.

A classroom belongs to a school, or we can say a school has a classroom.

Composition is a strong type of "**has-a**" relationship because the containing object is its owner. So, objects are tightly coupled, which means if we delete the parent object, the child object will also get deleted with it.

```
class Car {
    private Engine engine;

    public Car() {
        this.engine = new Engine();
    }

    // Other methods and attributes of the Car class
}

class Engine {
    // Engine implementation
}
```

Aggregation

Aggregation relationship is also a "**has-a**" relationship. The only difference between Aggregation and Composition is that in Aggregation, objects are not tightly coupled or don't involve owning. All the objects are independent of each other and can exist even if the parent object gets deleted.

Real-time example: Consider a University class that has an aggregation relationship with Student class. The University class has a collection of Student objects, but the Student objects can exist even if the University is destroyed.

Code :

```
class University {
    private List<Student> students;

    public University() {
        this.students = new ArrayList<>();
    }

    public void addStudent(Student student) {
        students.add(student);
    }

    // Other methods and attributes of the University class
}

class Student {
    // Student implementation
}
```

In this example, the University class aggregates Student objects. The University has a collection of Student objects, but the lifecycle of the Student objects is not tied to the University object. If the University is destroyed, the Student objects can still exist.

In summary, composition represents a strong relationship where the composed objects cannot exist independently, while aggregation represents a weaker

relationship where the aggregated objects can exist independently of the container object.

Object class in Java

The **Object class** is the parent class of all the classes in java by default. In other words, it is the topmost class of java.

The Object class is beneficial if you want to refer any object whose type you don't know. Notice that parent class reference variable can refer the child class object.

The Object class provides some common behaviors to all the objects such as object can be compared, object can be cloned, object can be notified etc.

Methods of Object class

public final Class getClass()
returns the Class class object of this object. The Class class can further be used to get the metadata of this class.

public int hashCode()

returns the hashcode number for this object.

The `hashCode()` method is used to generate a unique numeric identifier (hash code) for an object based on its internal state. The default implementation of `hashCode()` in the `Object` class computes the hash code based on the memory address of the object.

public boolean equals(Object obj)

compares the given object to this object.

protected Object clone() throws CloneNotSupportedException

creates and returns the exact copy (clone) of this object.

public String toString()
returns the string representation of this object.

protected void finalize()throws Throwable
is invoked by the garbage collector before object is being garbage collected.

Wait , Notify , Notify all will be discussed in Multithreading.