*Aggregation* is an association that represents a part of a whole relationship where a part can exist without a whole. It has a weaker relationship.

**1. Intent/Definition**

*Aggregation* is an association that represents a part of a whole relationship where a part can exist without a whole. It has a weaker relationship.

It is a specialized form of Association where all object has their own lifecycle but there is ownership. This represents a “whole-part or a-part-of” relationship.

Let’s take an example of the relationship between the *Department* and the *Teacher*. A *Teacher* may belong to multiple departments. Hence *Teacher* is a part of multiple departments. But if we delete a *Department*, the *Teacher* Object will not destroy.

****

**Key Points**

* It represents the Has-A relationship.
* It is a unidirectional association i.e. a one-way relationship. For example, the department can have students but vice versa is not possible and thus unidirectional in nature.
* In Aggregation, both the entries can survive individually which means ending one entity will not affect the other entity.

**2. Implementation**

Let's take the example of *LineItem* and its *Products*. If line-item HAS-A product, then a line item is a whole and product is a part.

If a line item is deleted, then the corresponding product needs not to be deleted.

**Step 1:**Create a *Product* class.

class Product {

private int id;

private String name;

private String description;

public Product(int id, String name, String description) {

super();

this.id = id;

this.name = name;

this.description = description;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getDescription() {

return description;

}

public void setDescription(String description) {

this.description = description;

}

@Override

public String toString() {

return "Product [id=" + id + ", name=" + name + ", description=" + description + "]";

}

}

**Step 2:** This is *LineItem* class, which *HAS-A* aggregation associated with the *Product* class. That means, if you delete *LineItem*, then the associated *Product* can exist.

class LineItem {

private int id;

private int quantity;

private Product p;

public LineItem(int id, int quantity, Product p) {

super();

this.id = id;

this.quantity = quantity;

this.p = p;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public int getQuantity() {

return quantity;

}

public void setQuantity(int quantity) {

this.quantity = quantity;

}

public Product getP() {

return p;

}

public void setP(Product p) {

this.p = p;

}

@Override

public String toString() {

return "LineItem [id=" + id + ", quantity=" + quantity + ", p=" + p + "]";

}

}

Step 3: Let's test an *Aggregation*.

public class Aggregation {

public static void main(String[] args) {

// Create Products

Product p1 = new Product(1, "Pen", "This is red pen");

Product p2 = new Product(2, "Pencil", "This is pencil");

Product p3 = new Product(3, "ColorBox", "This is color box");

// Create lineItem and add quntity of the products

LineItem item1 = new LineItem(1, 2, p1);

LineItem item2 = new LineItem(1, 2, p2);

LineItem item3 = new LineItem(1, 2, p3);

// Before deleting line item 1

System.out.println(item1.getId());

System.out.println(item1.getQuantity());

System.out.println(item1.getP());

item1 = null;

// Still product exist and not deleted

System.out.println(p1);

}

}