

```
Aggregation (Weak Reference) :
Aggregation is less a another form of association between classes that represents a "HAS-A"
relationship, but with a weaker bond compared to composition.
In aggregation, one class contains an object of another class, but the contained object can exist
independently of the container. If the container object is destroyed, the contained object can still exist.
package com.ravi.aggregation;

public class College
{
    private String collegeName;
    private String collegeLocation;

    public College(String collegeName, String collegeLocation)
    {
        super();
        this.collegeName = collegeName;
        this.collegeLocation = collegeLocation;
    }

    @Override
    public String toString()
    {
        return "College [collegeName: " + collegeName + ", collegeLocation: " + collegeLocation +
        " ]";
    }
}

package com.ravi.aggregation;

public class Student
{
    private int studentId;
    private String studentName;
    private String studentAddress;
    private College college; //HAS-A Relation

    public Student(int studentId, String studentName, String studentAddress, College college)
    {
        super();
        this.studentId = studentId;
        this.studentName = studentName;
        this.studentAddress = studentAddress;
        this.college = college;
    }

    @Override
    public String toString() {
        return "Student [studentId: " + studentId + ", studentName: " + studentName + ",
        studentAddress: " + studentAddress
        + ", college: " + this.college + " ]";
    }
}

package com.ravi.aggregation;

public class Aggregation {
    void main()
    {
        College c1g = new College("VIT", "Vellore");
        College c2g = new College("MIT", "Waz");

        Student s1 = new Student(1, "Siddh", "Mangalore", c1g);
        System.out.println(s1);

        Student s2 = new Student(2, "Siddh", "9 8 Nagar", c1g);
        System.out.println(s2);
    }
}
```

```
Creates a Shopping Mall Application project by using Method Overriding Concept to
display and accept different kinds of discount given to their customers like PrimeCustomer
and VVIPCustomer (No discount for General Customer)

Summary :
[Super class : Customer
Subclasses: GeneralCustomer, PrimeCustomer, VVIPCustomer]
Override the calculateBill() method to implement different discount rules as well as
override printDetails() method for printing the details.

Coding Requirements :
=====
Create a RUC class Customer
Fields :
name String protected
total double protected
Use a parameterized constructor to initialize the fields, In this constructor
provide error message, if name is null or empty (see test cases for more details)

Methods :
1) Method Name : calculateBill()
Parameter : One Parameter of type double (see RUC) [double... prices]
Return Type : double
Access modifier : public
In this method using one RUC receive item price, Give an error message and
exit, if the price is negative. Calculate the total bill by adding all the item price
received through one RUC.

2) Method Name : printDetails()
Parameter : No parameter
Return Type : void
Access modifier : public
In this method print customer name, total cost and no discount for general
customer.

Create another RUC class GeneralCustomer which is the sub class of Customer
Fields : No Fields
Take a parameterized constructor to initialize super class properties.

Method :
1) Method Name : calculateBill()
Parameter : One Parameter of type double (see RUC) [double... prices]
Return Type : double
Access modifier : public
In this overridden method, call the super class calculateBill() method to
calculate the price and return the total amount.

Create another RUC class PrimeCustomer which is the sub class of Customer
Fields :
protected double discountRate = 10.0;
Take a parameterized constructor to initialize super class properties.

Method :
1) Method Name : calculateBill()
Parameter : One Parameter of type double (see RUC) [double... prices]
Return Type : double
Access modifier : public
In this overridden method, call the super class calculateBill() method to
calculate the price and return the total amount.

2) Method Name : printDetails()
Parameter : No parameter
Return Type : void
Access modifier : public
In this method give 10% discount on total bill, calculate final bill after
subtracting the discount amount from the final bill. Print name, total amount,
discount amount and final amount (see the test cases)

Create another RUC class VVIPCustomer which is the sub class of Customer
Fields :
protected double discountRate = 15.0;
Take a parameterized constructor to initialize super class properties.

Method :
1) Method Name : calculateBill()
Parameter : One Parameter of type double (see RUC) [double... prices]
Return Type : double
Access modifier : public
In this overridden method, call the super class calculateBill() method to
calculate the price and return the total amount.

2) Method Name : printDetails()
Parameter : No parameter
Return Type : void
Access modifier : public
In this method give 15% discount on total bill, calculate final bill after
subtracting the discount amount from the final bill. Print name, total amount,
discount amount and final amount (see the test cases)

Create an RUC class ShoppingMall with main method to test this application.

Method :
1) Method Name : generateBill()
Argument : Two arguments (Customer obj, double... prices)
Return Type : void
Access modifier : public and static
This method will receive Customer object and item price using one RUC and call
appropriate object overridden method of sub classes to generate the Bill.
For item price, Take an array variable to store multiple item prices.

Write Switch case with Scanner class in the main method
to Test the application as shown in the below Test Cases.

Test Cases for Output :
=====
Test Case 1 :
Please select the Customer Type to get additional Discount :
1) General Customer
2) Prime Customer
3) VVIP Customer

System.out.println("Please enter Customer Type :");
Scanner sc = new Scanner(System.in);
int number of Items :
Please Enter the Item Name and Price :
Item Name : Shirt
Item Price : 1000
Item Name : Jeans
Item Price : 800
Welcome to Superbilled Mall :
Customer: Mr. Siddhant
Total cost is : 1800.0
Discount: No discount for general customers.

Test Case 2 :
Please select the Customer Type to get additional Discount :
1) General Customer
2) Prime Customer
3) VVIP Customer

Please enter Customer Type :
Mr. Siddh
Enter number of Items :
Please Enter the Item Name and Price :
Item Name : Shirt
Item Price : 1000
Item Name : Jeans
Item Price : 800
Welcome to Superbilled Mall :
Total price cannot be negative.
```