Motivation:

The double negative in the code is very confusing. This makes the code unreadable. It can damage the code which in turn leads to bugs. So, fix them on sight. Here in the example you can see the use of double negative in the codes. In the checkout method there is a if condition which is (! customer.IsNotFlagged) and another one is in the Customer class which is IsNotFlagged. These conditions are really confusing and hard to understand.

**Before:**

class Account

{

private List<Entry> lstEntries {get; set;}

public double GetResult (DateTime start, DateTime end)

{

double result = 0;

foreach (var entry in lstEntries)

{

if (start <= entry.ChargeDate && end >= entry.ChargeDate)

{

result += entry.value;

}

}

return result;

}

}

**Mechanics:**

Here I am going to remove all the negative logic by converting the logic negative to positive and reversing the sense. **(! customer.IsNotFlagged)** this condition will be converted to **(customer.IsFlagged).** And **IsNotFlagged** will be converted to **IsFlagged**

**After:**

class Account

{

private List<Entry> lstEntries { get; set; }

public double GetResult(DateRange range)

{

double result = 0;

foreach (var entry in lstEntries)

{

if (range.Includes(entry.ChargeDate))

{

result += entry.value;

}

}

return result;

}

}

class DateRange

{

public DateTime start { get; set; }

public DateTime end { get; set; }

public bool Includes(DateTime givenDate)

{

return start <= givenDate && end >= givenDate;

}

}