Motivation:

Sometime identical groups of parameters are often used in multiple methods. For this reason, code duplication happens. Also, methods with several parameter makes the method unreadable. To refactor this its better to create a class with those parameters. Here you see the method GetResult (DateTime start, DateTime end). It has same types of parameter.

**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:**

To refactor this code, I am going create a class called DateRange where all the parameter will be stored. So, in those methods I can use DateRange object. It makes the code flexible. If more parameter needed the I just have added another property to the class.

**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;

}

}