Work day calculator

The task is to implement a calculator for work days in .NET Core:

```
WorkdayCalendar workdayCalendar = new WorkdayCalendar();
```

The calculator will take a date (DateTime), a number of work days to add or subtract (decimal), and return a new date that is within business hours on a work day. Work days are defined as any day from Monday to Friday that isn't a holiday. There are two methods in the calendar which define holidays:

```
workdayCalendar.SetHoliday(DateTime date)
```

means that this specific date is a holiday and should not be considered a work day.

```
workdayCalendar.SetRecurringHoliday(int month, int day)
```

means that this day and month is to be considered a holiday every year.

```
workdayCalendar.SetWorkdayStartAndStop(int startHours, int startMinutes,
int stopHours, int stopMinutes)
```

defines the business hours of the work day. For instance from 08:00 to 16:00:

```
workdayCalendar.SetWorkdayStartAndStop(8, 0, 16, 0);
```

The core of the solution will be this method:

```
public DateTime GetWorkdayIncrement(Date startDate, decimal
incrementInWorkdays)
```

The method must return a DateTime between the business hours defined in SetWorkdayStartAndStop, even when the startDate is outside of business hours. For instance, if business hours are 8 to 16, 15:07 + 0,25 work days will be 09:07, and 04:00 + 0,5 work days will be 12:00.

In other words, the class will implement the following interface:

```
namespace WorkdayNet
{
    public interface IWorkdayCalendar
    {
        void SetHoliday(DateTime date);
        void SetRecurringHoliday(int month, int day);
        void SetWorkdayStartAndStop(int startHours, int startMinutes,
int stopHours, int stopMinutes);
        DateTime GetWorkdayIncrement(DateTime startDate, decimal
incrementInWorkdays);
    }
}
```

The class can be used like this:

```
IWorkdayCalendar calendar = new WorkdayCalendar();
calendar.SetWorkdayStartAndStop(8, 0, 16, 0);
calendar.SetRecurringHoliday(5, 17);
calendar.SetHoliday(new DateTime(2004, 5, 27));
string format = "dd-MM-yyyy HH:mm";
var start = new DateTime(2004, 5, 24, 18, 5, 0);
decimal increment = -5.5m;
var incrementedDate = calendar.GetWorkdayIncrement(start, increment);
Console.WriteLine(
 start.ToString(format) +
 " with an addition of " +
 increment +
 " work days is " +
 incrementedDate.ToString(format));
```

This should given the following result:

```
24-05-2004 18:05 with an addition of -5.5 work days is 14-05-2004 12:00
```

Other correct results:

```
24-05-2004 19:03 with an addition of 44.723656 work days is 27-07-2004 13:47
24-05-2004 18:03 with an addition of -6.7470217 work days is 13-05-2004 10:02
24-05-2004 08:03 with an addition of 12.782709 work days is 10-06-2004 14:18
24-05-2004 07:03 with an addition of 8.276628 work days is 04-06-2004 10:12
```

The implementation will be tested against these cases as well as randomly generated values. A good solution will be easy to understand, relatively short, (typically less than 250 lines of code), and it should be possible to write it in about four hours.

Good luck!