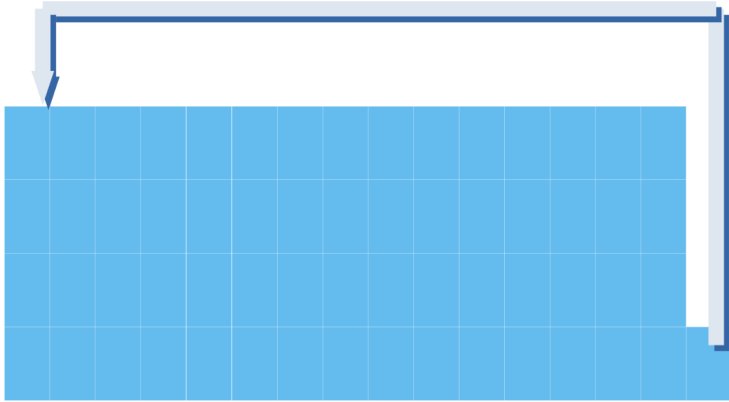
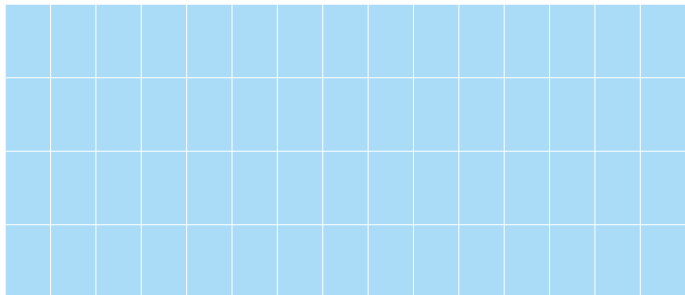


# Circular Buffer for TimeSeries



Fixed capacity: `CircularBuffer<Data>(N)` stores no more than  $N$  elements

# Computing Averages



```
(from row in rows select row.Value).Average()
```

Static Extension Methods

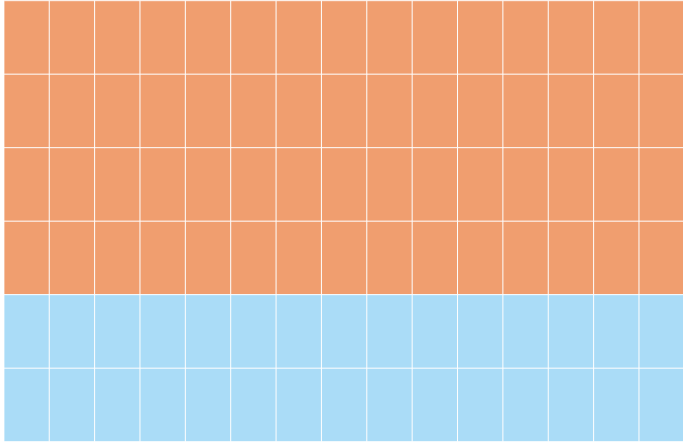
<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/extension-methods>

Language-Integrated Query (LINQ)

<https://docs.microsoft.com/en-us/dotnet/standard/linq/>

[http://www.java2s.com/Tutorial/CSharp/0450\\_\\_LINQ/Catalog0450\\_\\_LINQ.htm](http://www.java2s.com/Tutorial/CSharp/0450__LINQ/Catalog0450__LINQ.htm)

# Computing Averages

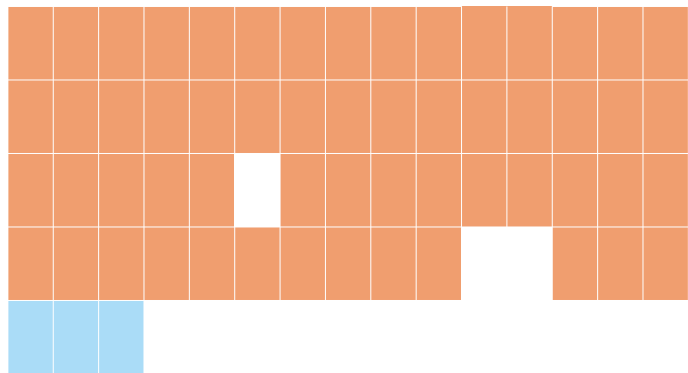


```
class Data {  
    DateTime timeStamp;  
    Int32 value;  
}
```

```
(from row in rows  
where (now - row.TimeStamp).TotalMinutes <= interval  
select row.Value).Average()
```

Extend the buffer to more than 60 elements - compute 1 minute average

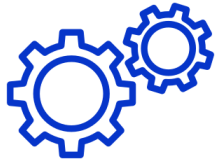
# Computing Averages



```
(from row in rows
where (now - row.Timestamp).TotalMinutes <= interval
select row.Value).Average()
```

Missed info and buffer spans longer than 1 minute - compute 1 minute average

# Design



Windows Service with two timers:

- collect System Performance Metric every second
- calculate Averages on demand
- optionally calculate Averages once per minute

Services Controller manages install, start, end of a service

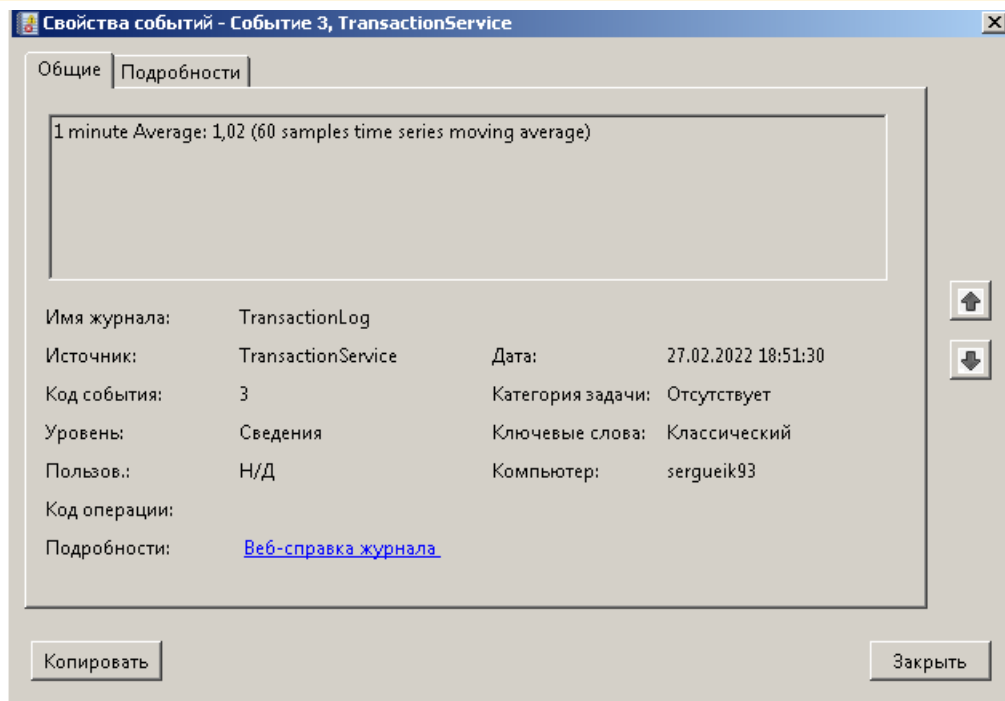
does not need user to be logged on

Services respond to custom commands

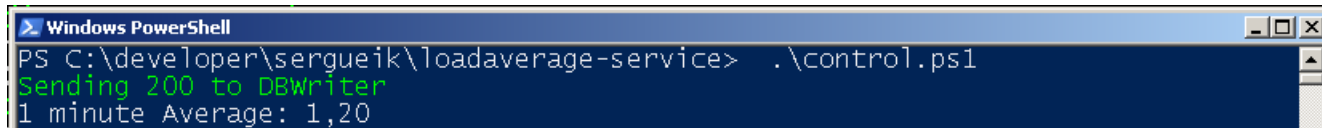
[https://en.wikipedia.org/wiki/Windows\\_service](https://en.wikipedia.org/wiki/Windows_service)

[https://en.wikipedia.org/wiki/Performance\\_Monitor](https://en.wikipedia.org/wiki/Performance_Monitor)

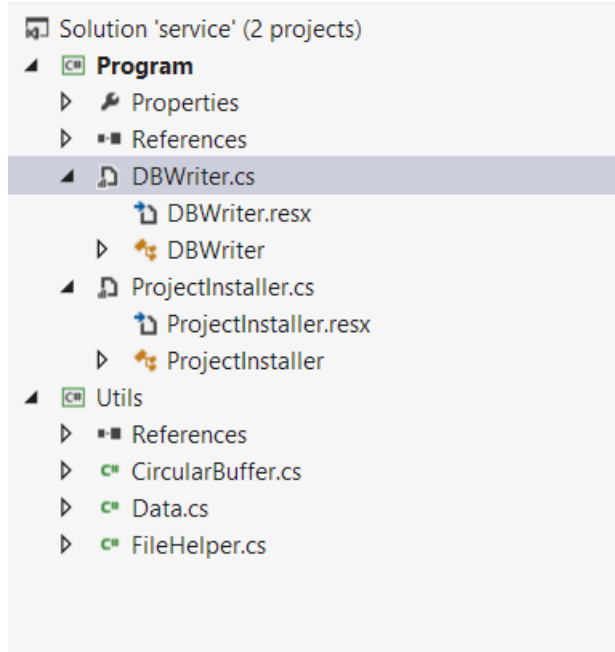
# Execution Demo



- LoadAverage Service is running continuously on Windows machine
- PS Borg app calls the service and posts the data to Borg Server



# Service Component



Every *Windows Service* is a subclass of `System.ServiceProcess.ServiceBase`

The code is largely IDE-generated:  
`Program`, `Service`, `ProjectInstaller` classes

One is required to provide the constructor, override `OnStart`, `OnStop`, `OnCustomCommand` and `Dispose`

One can add method named `Main` with no special meaning: *Windows Service* cannot be run directly

One can instantiate a `System.Timers.Timer` and create method conventionally named `OnElapsedTimer` to handle `Elapsed` event

<https://docs.microsoft.com/en-us/dotnet/api/system.serviceprocess.servicebase>

<https://docs.microsoft.com/en-us/dotnet/api/system.timers.timer>

# Data Collector Invocation

control.ps1\* X

```
1 param (
2     [String]$name = 'DBWriter',
3     [String]$filepath = 'C:\temp\loadaverage.txt',
4     [int]$command = 200
5 )
6 [System.Reflection.Assembly]::LoadWithPartialName('System.ServiceProcess')
7 $controller = new-object System.ServiceProcess.ServiceController($name)
8 $controller.ExecuteCommand($command)
9 get-content -path $filepath
10
```

## Powershell script

- load **System.ServiceProcess.dll**
- find Windows Service
- invoke **customCommand**
- read the results file

```
protected override void OnCustomCommand(int command) {
    base.OnCustomCommand(command);
    switch (command) {
        case(200):
            AverageData();
            Commit();
            break;
    }
}
```

## Application

- processes the **customCommand**
- compute load averages
- save information to the file



# Application Configuration

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <appSettings>
    <add key="Noop" value="False"/>
    <add key="CollectInterval" value="1000"/>
    <add key="Datafile" value="c:\temp\loadaverage.txt"/>
  </appSettings>
</configuration>
```

Provided Installer script,  
Settings loaded at start of  
service

reinstall.ps1\* X

```
1 param(
2     [string]$SERVICENAME = 'DBWriter' ,
3     [string]$APPNAME = 'TransactionService.exe',
4     [switch]$debug
5 )
6 $config_file = ".\Program\bin\Debug\${APPNAME}.config"
7 [xml]$config = [xml]((get-content $config_file) -join '')
8 $config.configuration.appSettings.add | foreach-object { if( $_.Key -eq 'Debug' ) { $_.Value = $debug.ToBool().ToString() }}
9 $config.Save($config_file)
10
11 invoke-expression -command "InstallUtil.exe -install Program\bin\Debug\${APPNAME}"
12 invoke-expression -command "sc.exe start ${SERVICENAME}"
13
14
```

# Application Configuration

entry	description	default
Autorun	perform periodic Average calculation	FALSE
Noop	Mock up / read \\System\\Processor Queue Length	FALSE
Debug	Log additional information to EventLog	FALSE
CollectInterval	Interval between reads	1000
AutoAverageInterval	Interval between computing averages	120000
Datafile	Filename for Counters	c:\\temp\\loadaverage.txt