Andrew Armstrong

SIS System Example: Heart Rate Monitor

Motivation: The motivation behind this project was to build a small example system that emulates a heart rate monitor being a part of a larger health care system. A monitor (SIS component not the heart rate monitor) listens to the entire system and makes decisions based on the readings it receives. These decisions involve deciding upon the status of the patient and determining if any actions need to be taken in order to maximize the patient's health.

Larger Role: In a fully implemented system this would be a single part that is in charge of recording data. In order to simulate this functionality, I have the monitor writing data to a file labeled 'database_fake.txt.' This simulates performing writes to a database that would house information for the entire system. From here another component, such as a notification system would read this database and update another webpage that is in charge of distributing resources (i.e. ambulances and personnel) to appropriate situations – such as a heart rate that signifies an emergency.

Improvements: Future improvements to this would include implementing the actual database in order to retain the data in a more structured format. As well as increasing the amount and scope of data being recorded. By this I mean that the decisions made by the monitor right now as to how to classify the readings is incomplete and naïve. Instead, I should store more information such as the age of the patient and also their physical condition. These and many more factors influence at what heart rate level certain notifications need to take place.

System Diagram

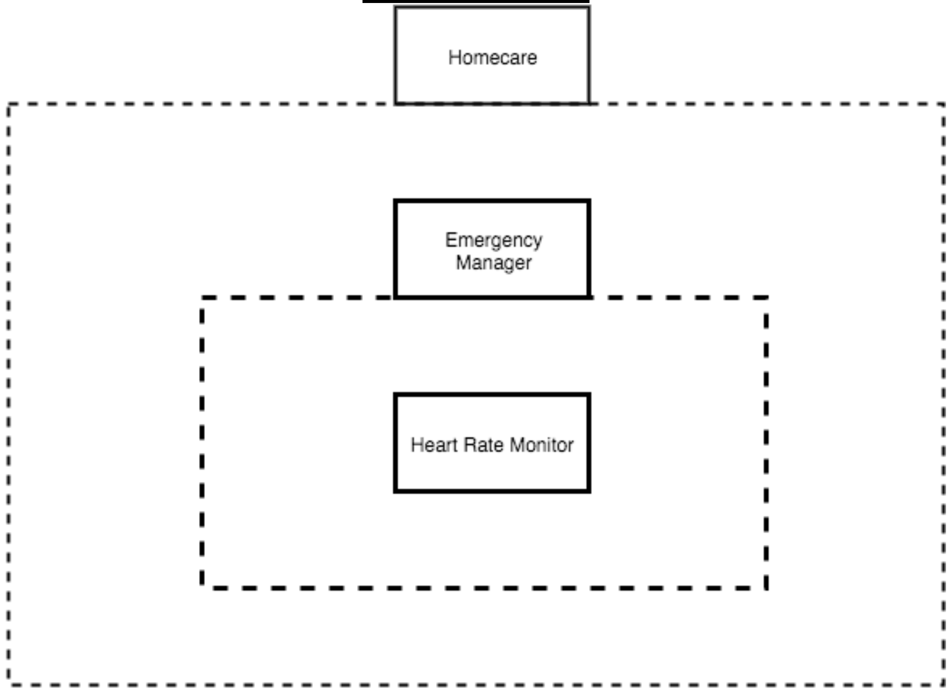


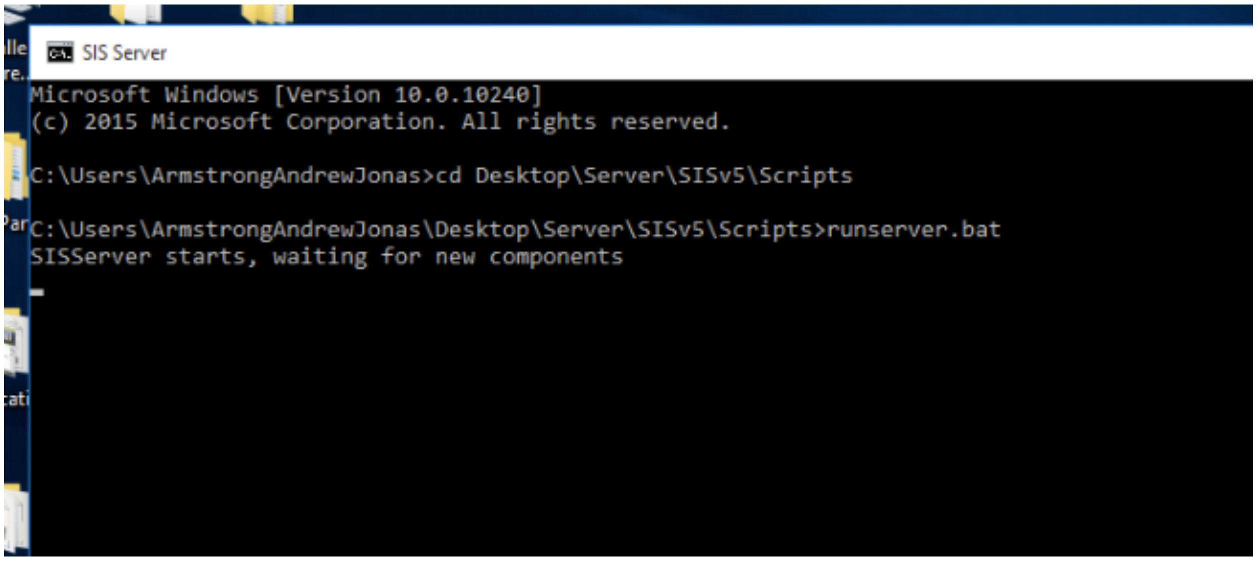
Figure 1: System Diagram

What *Figure 1* is showing is a Super component (Emergency Manager) with a Basic component (Heart Rate Monitor) associated with it. Both are on the same scope which has a Monitor component (Homecare) associated with it. This Monitor will listen for all readings across the scope and process them. In this case that means listening to the Heart Rate Monitor and processing the readings produced by it.

<u>**Example Scenario**</u>

***Note: This scenario covers the standard setup for the system and sending of a message. It then discusses the handling of one specific message category in general – namely Dangerous. ***

The first step to take is to start up the SIS Server. This is accomplished using the script *runserver.bat*.

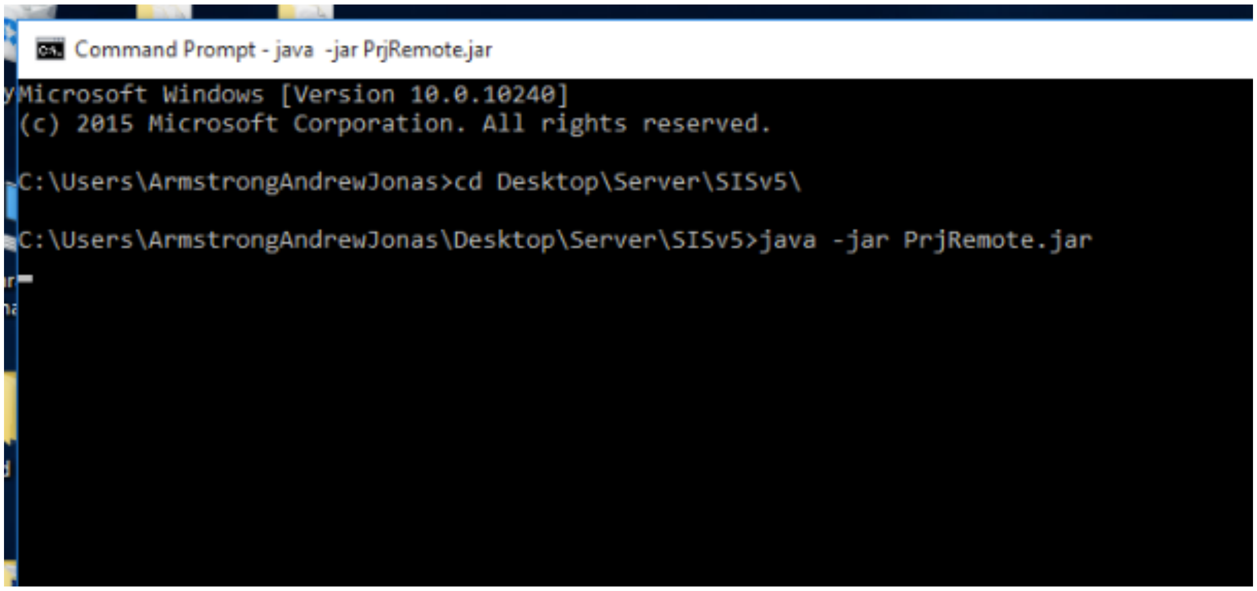


```
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\ArmstrongAndrewJonas>cd Desktop\Server\SISv5\Scripts

C:\Users\ArmstrongAndrewJonas\Desktop\Server\SISv5\Scripts>runserver.bat
SISServer starts, waiting for new components
```

Figure 2: runserver.bat script

After starting the SIS Server, the PrjRemote needs to be started so messages can be sent. This is achieved via the command *java –jar PrjRemote.jar*.

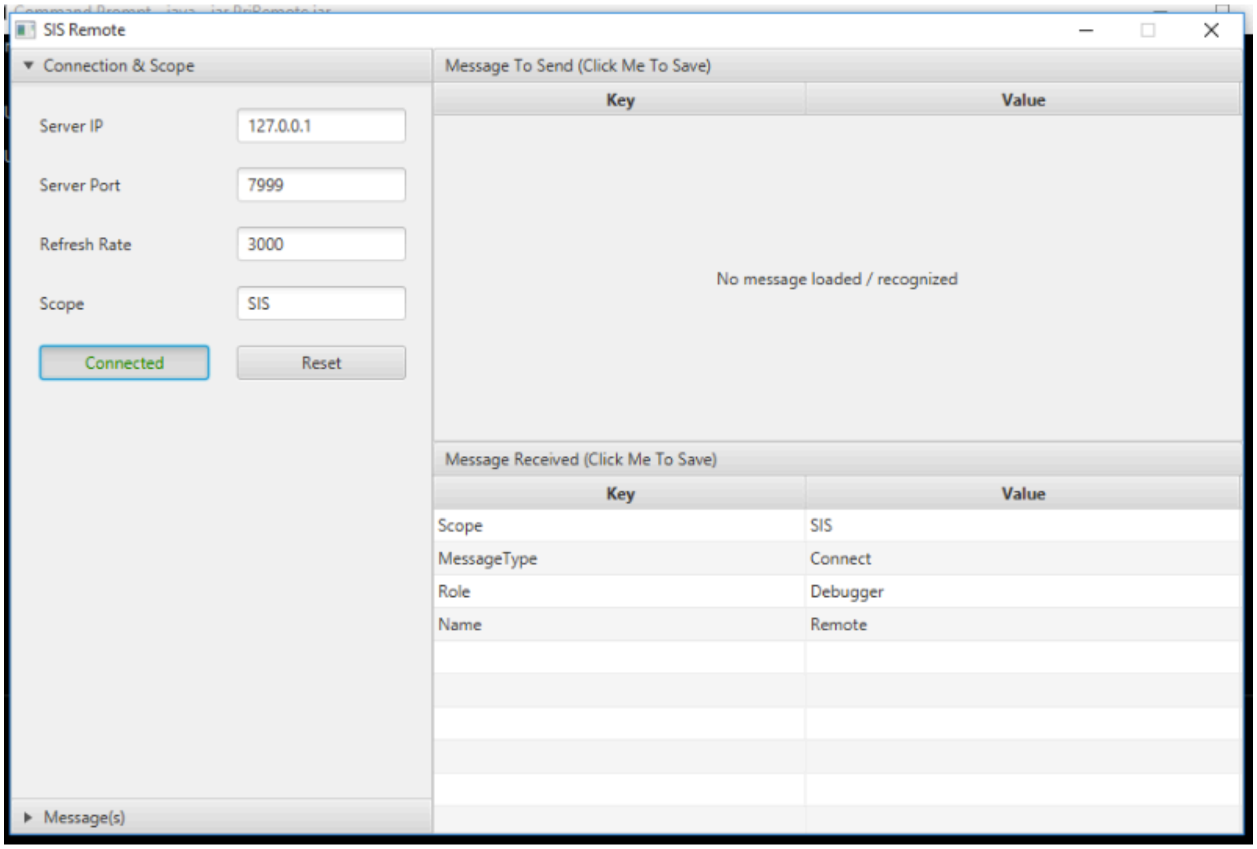
Andrew Armstrong



```
Command Prompt - java  -jar PrjRemote.jar
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\ArmstrongAndrewJonas>cd Desktop\Server\SISv5\

C:\Users\ArmstrongAndrewJonas\Desktop\Server\SISv5>java -jar PrjRemote.jar
```

Figure 3: PrjRemote command

After executing this command the following GUI appears.



SIS Remote

Connection & Scope

Server IP	127.0.0.1
Server Port	7999
Refresh Rate	3000
Scope	SIS

Message To Send (Click Me To Save)

Key	Value

No message loaded / recognized

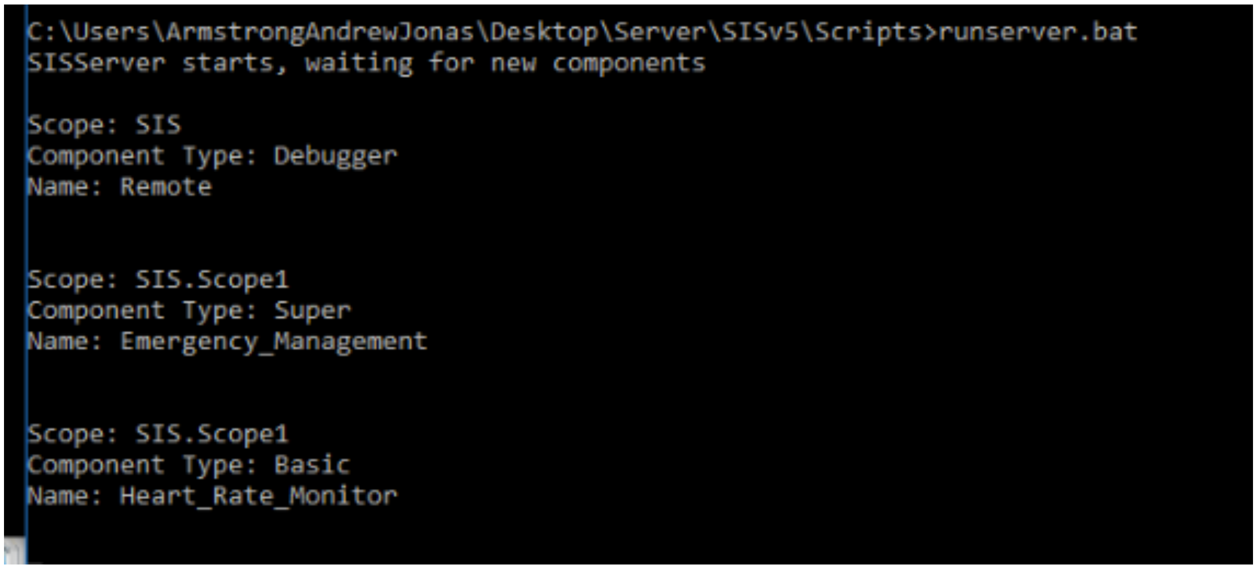
Message Received (Click Me To Save)

Key	Value
Scope	SIS
MessageType	Connect
Role	Debugger
Name	Remote

Figure 4: PrjRemote User Interface

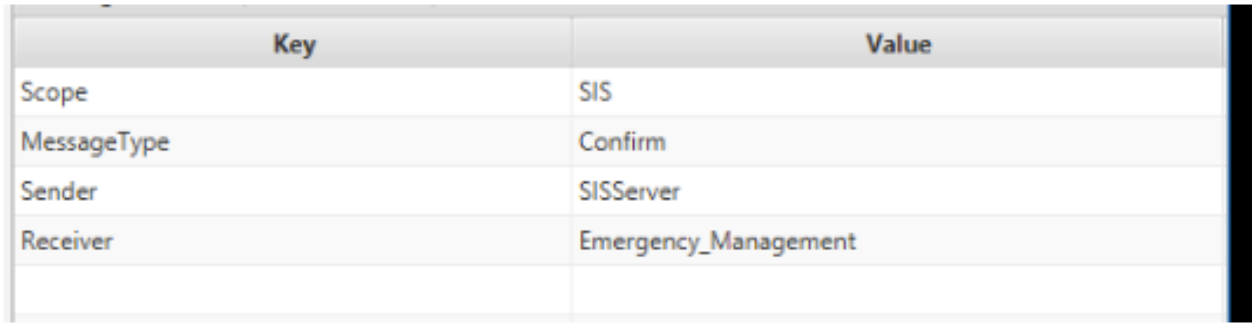
Notice that in *Figure 4* the 'Connected' button is pressed and highlighted in green. This means that the Debugger component named Remote has been connected to the SIS Server. This is verified by the message in the bottom right corner which says it has been connected.

From here we need to connect the different components that are necessary for the system to run. This includes the Super component Emergency Manager and the Basic component Heart Rate Monitor. The following figures show the SIS Server receiving register and connect messages from these components and responding with a confirm message.



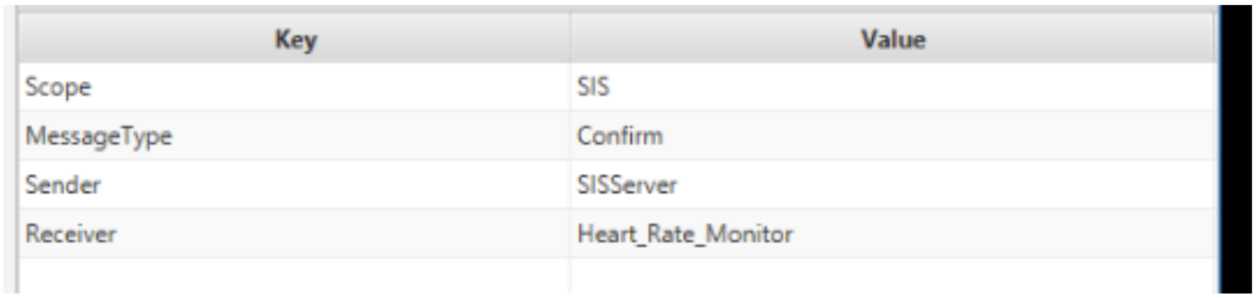
```
C:\Users\ArmstrongAndrewJonas\Desktop\Server\SISv5\Scripts>runserver.bat
SISServer starts, waiting for new components

Scope: SIS
Component Type: Debugger
Name: Remote


Scope: SIS.Scope1
Component Type: Super
Name: Emergency_Management


Scope: SIS.Scope1
Component Type: Basic
Name: Heart_Rate_Monitor
```

Figure 5: SIS Server Output



Key	Value
Scope	SIS
MessageType	Confirm
Sender	SISServer
Receiver	Emergency_Management

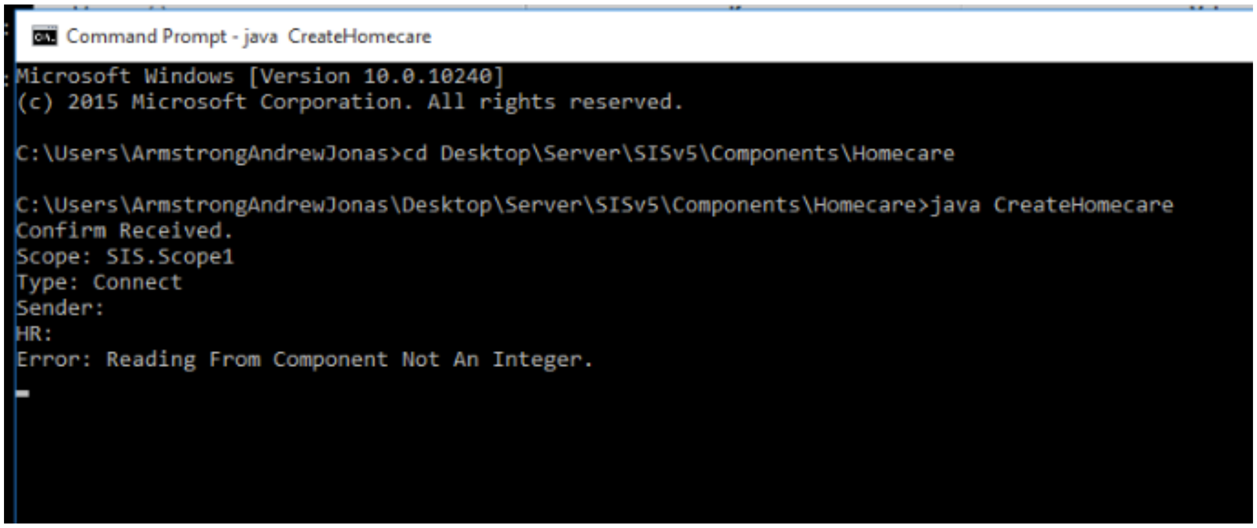
Figure 6: Emergency Management Confirmation



Key	Value
Scope	SIS
MessageType	Confirm
Sender	SISServer
Receiver	Heart_Rate_Monitor

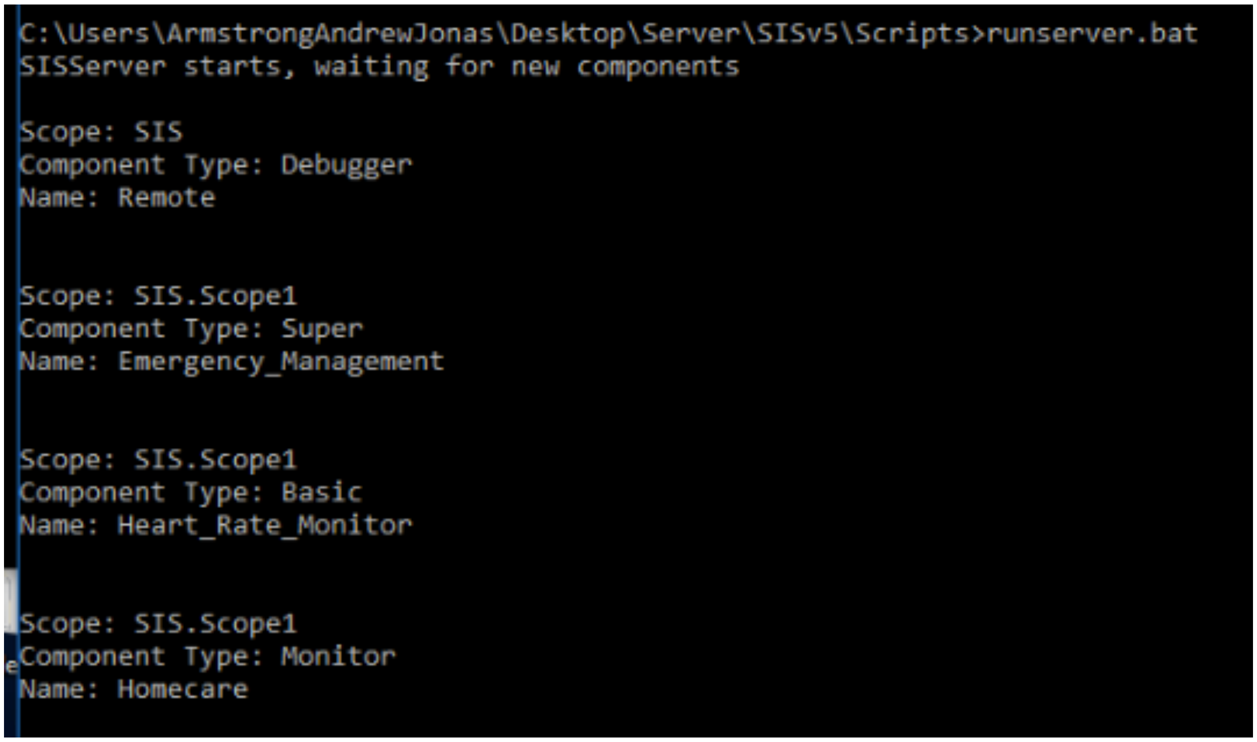
Figure 7: Heart Rate Monitor Confirmation

Now our system has been set up it is time to introduce the monitor component that I implemented – Homecare. The Homecare component, which I implemented as a Java class, contains code to send both a register and connect message before waiting to process reading messages from components. Therefore, by running the command 'java CreateHomecare' the Monitor component will register and connect itself to the SIS Server. This can be seen in the following two figures.



```
Command Prompt - java  CreateHomecare
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\ArmstrongAndrewJonas>cd Desktop\Server\SISv5\Components\Homecare

C:\Users\ArmstrongAndrewJonas\Desktop\Server\SISv5\Components\Homecare>java CreateHomecare
Confirm Received.
Scope: SIS.Scope1
Type: Connect
Sender:
HR:
Error: Reading From Component Not An Integer.
```

Figure 8: Homecare Monitor Command



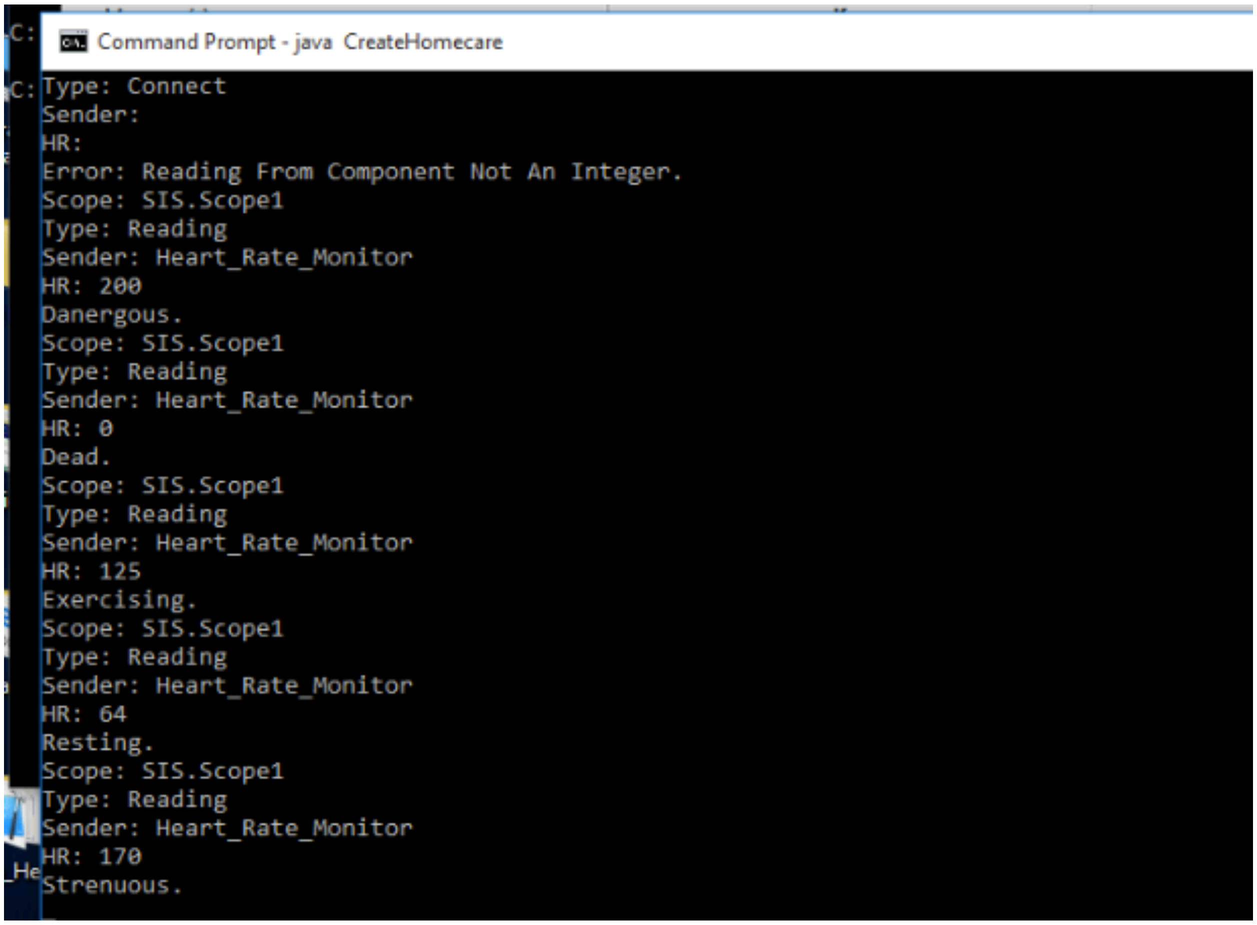
```
C:\Users\ArmstrongAndrewJonas\Desktop\Server\SISv5\Scripts>runserver.bat
SISServer starts, waiting for new components

Scope: SIS
Component Type: Debugger
Name: Remote


Scope: SIS.Scope1
Component Type: Super
Name: Emergency_Management


Scope: SIS.Scope1
Component Type: Basic
Name: Heart_Rate_Monitor


Scope: SIS.Scope1
Component Type: Monitor
Name: Homecare
```

Figure 9: SIS Homecare Confirmation

So a quick recap. Everything up to this moment has been setting up the system for a scenario involving the Homecare Monitor. The SIS Server was started along with PrjRemote in order to facilitate message passing. I then connected a Super and Basic component to the system before running the necessary Java class to register and connect my own component. Now for the scenario I want to run.

Dangerous Scenario

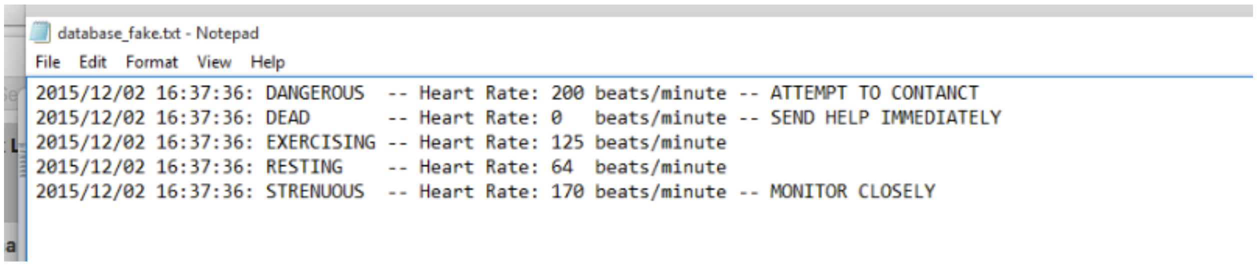
For my Monitor component I programmed different categories of notifications based on the Heart Rate Monitor's reading. In this scenario the reading from the Heart Rate Monitor falls in to the 'Dangerous range. Below is a screenshot of numerous messages being received by the Homecare Monitor that have been sent via the PrjRemote GUI. These messages cover the full range of categories, although we are still only interested in the Dangerous category.



```
Type: Connect
Sender:
HR:
Error: Reading From Component Not An Integer.
Scope: SIS.Scope1
Type: Reading
Sender: Heart_Rate_Monitor
HR: 200
Danergous.
Scope: SIS.Scope1
Type: Reading
Sender: Heart_Rate_Monitor
HR: 0
Dead.
Scope: SIS.Scope1
Type: Reading
Sender: Heart_Rate_Monitor
HR: 125
Exercising.
Scope: SIS.Scope1
Type: Reading
Sender: Heart_Rate_Monitor
HR: 64
Resting.
Scope: SIS.Scope1
Type: Reading
Sender: Heart_Rate_Monitor
HR: 170
Strenuous.
```

Figure 10: Homecare Monitor Output

Perform describing the functionality further here is a screenshot of the contents of the *database_fake.txt* file.



```
2015/12/02 16:37:36: DANGEROUS  -- Heart Rate: 200 beats/minute -- ATTEMPT TO CONTANCT
2015/12/02 16:37:36: DEAD       -- Heart Rate: 0   beats/minute -- SEND HELP IMMEDIATELY
2015/12/02 16:37:36: EXERCISING -- Heart Rate: 125 beats/minute
2015/12/02 16:37:36: RESTING    -- Heart Rate: 64  beats/minute
2015/12/02 16:37:36: STRENUOUS  -- Heart Rate: 170 beats/minute -- MONITOR CLOSELY
```

Figure 11: Fake Database Contents

What the system has done is take the heart rate output by the Heart Rate Monitor (200 beats / minute) and classify this as a ‘Dangerous’ heart rate before writing the output to the file. What a more complicated and fully integrated system would od would be to take this output and read it in so that an action can be taken in response to the ‘Dangerous’ classification. More likely than not this would require sending an ambulance to attend to the patient.

Conclusions

Overall this system provides a simple overview of a possible implementation for a more complicated system. In order to improve upon this, components would be added to provide real-time data from a living patient as well as allowing for more patient information to be available in order to make better decisions. The notification system would also be expanded to allow for constant monitoring of patients and / or automated response to particular events. While these changes would improve the system, the current configuration still provides a informative abstraction as to the workings of the system and the steps necessary to process input.

Andrew Armstrong