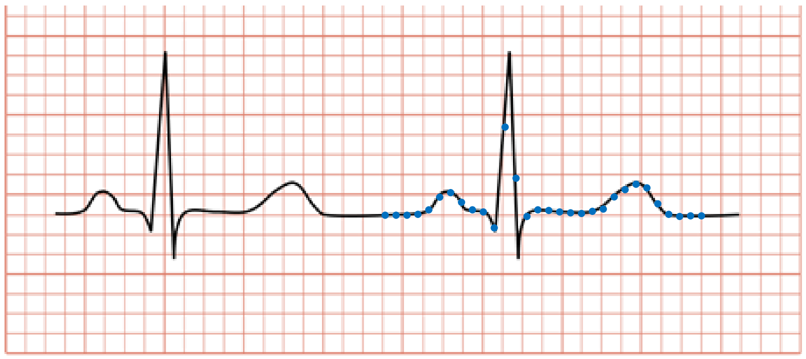
AARHUS UNIVERSITET
INGENIØRHØJSKOLEN

Exercise: Producer – Consumer, Electrocardiography (ECG)

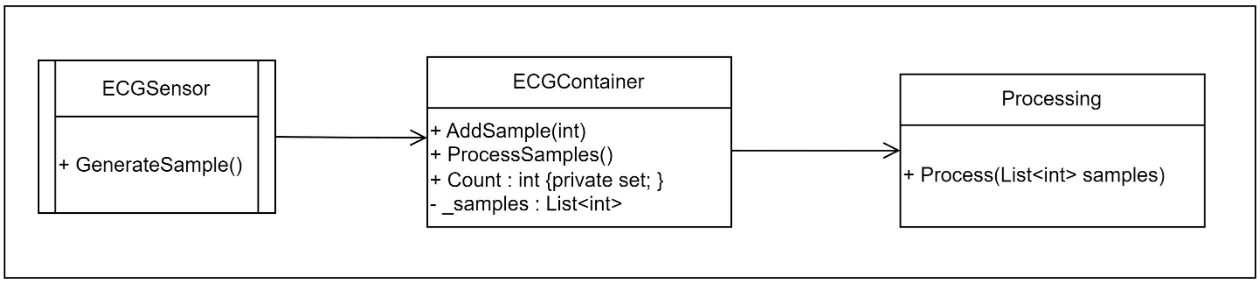
In this exercise, you will work with the collection and processing of *electrocardiography* (ECG) samples. In short, ECG is the process of recording the heart's electrical activity over time.

This exercise focuses on the producer-consumer pattern, which we will use to decouple the collection of samples from the processing of samples and allow collection and processing to run in separate threads.

An *ECG signal* is a continuous signal with – in a healthy person – looks similar to the one below. We will create a small system that can receive and process samples of such a signal periodically.



A self-taught l33t programmer has previously worked on the system and left a design for you:



Being the brilliant and educated engineer you are, you immediately see that this is a bad design: The sensor, which provides samples know the ECGContainer, which consumes samples. The ECGSensor is highly coupled to the ECGContainer.

Your job is to create a better design and implement it. You should work together with one or more other students when you do the design.

Exercise 1:
Consider the design above. How can you use the producer-consumer pattern and a queue (BlockingCollection) to decouple the ECGSensor from the ECGContainer?

Create a design for you system in the form of an UML class diagram.

*Hint: You will probably need to introduce new classes for the **ECGReadingProducer**, the **ECGReadingConsumer** and a **DataContainer** for the data being read from the ECGSensor and sent to the ECGContainer.*

Exercise 2:
Show your design to another group and get their feedback. Decide if you want to change anything.

Exercise 3:
Implement your design as a console application and create a simulated version of the sensor.
The simulated sensor should be polled, i.e., you have to ask the sensor for a new sample and it shall return a value between 0 and 50. The polling frequency shall be once every second.

The producer and the consumer parts of your system shall run in separate threads. Every time the consumer receives a new sample it shall be printed to the console.

Exercise 4:
The 'Process' method on ECGContainer shall be called from the Main thread, when the button 'p' is pressed.
The default processing is to print the complete list of samples.

Exercise 5:
Printing of the received samples should not be done in the consumer. We want a separate **Log** class for that.

Use the GoF Observer pattern to decouple the reception of sample from printing of samples. The Log class shall have the role of an *Observer* and the ECGReadingConsumer, which receives data from the BlockingCollection, shall have the role of a *Subject*.

Exercise 6:
If you have not done so already, the ECGContainer should also be decoupled from the ECGReadingConsumer, by making it an Observer of the ECGReadingConsumer.