Now you know how the plastic ware was made, but the plastic ware itself won't do anything. It needs to be controlled by some software. So we could just start programming but that would probably end up in a big mess. Thus we started to think about what operations the machine had to do. And we started making sates of those and their connections. First off it were just some basic states like: abort, sorting, sorting after start/stop button and the resting state. And then we started to make it more precise. And we ended up with the state transition diagram. Which you can see here. I will give you some explanation about how the parts of the diagram.

The first part is usually the resting state in which the machine is booted up. But we don't know if the sorter mechanism is in the right position when it is booted. So we have a couple of states in the beginning for calibrating the sorter. The second part is the sorting part. Sorting is being done by waiting for a white disc to arrive at the colour detector. When that happens we stop the feeder engine, and the sorter mechanism moves up. Then the disc passes through and the sorting mechanism moves back down. Then the feeder will start again. Another thing the machine has to do in this part is to stop after all discs are sorted. This has been implemented by us using a timer interrupt. When the machine starts the timer is set to 2 seconds. Because then we know that no discs are on the conveyor belt anymore. This timer gets reset after a disc is spotted by the position sensor.

The entire running part copied for the part where the machine has to stop. The only difference is that the feeder engine is off all the time, and the timer interrupt doesn't get reset after a disc is at the position sensor.

The last part is for when the machine has to abort. This part is done by checking for the abort button in every state. So now we knew how the machine had to operate so we started creating java code.

And I will explain how we translated a state to a piece of java code. Here you see that the state 'running' with the transitions and also the corresponding java code. As you can see we first call a function called timerManage. That is the function which regulates the power of the engines. Then we check if the start stop button is pressed. If it is we make the transition and set the variables for the outputs and we go to the next state. So you can see that the transition is represented by the if statement and the state by the function. Then you can see we have the other transition which occurs if the position sensor is triggered. Just like the transition. But there is one transition which you don't see as an if statement and that is the timer interrupt. That one is not represented by an if statement because when the interrupt occurs the machine will automatically take that transition. Another thing you may have noticed is that in the java code the variables have some dollar signs in front of them. That has to do with the implementation of the code which will be explained by wigger.