Pseudo Code for WorkStationAgent

Data:

```
	public enum StationStatus
{EMPTY,NOTHINGTODO,PARTONSTATION,NEEDSPROCESSING,NONEEDPROCESSING,PROCESSING,NEEDNOTIFICATION,
PREPARETOLOAD,PARTSENT};
	public StationStatus status;
	public enum NextCFState{READY,BUSY};
	public NextCFState stateOfnextCF;
	private Conveyor conveyor;
	private ConveyorFamily family;
	private ConveyorFamily nextCF;
	private List<Part> parts = new ArrayList<Part>(); //list of all parts
	private String name;
	private Transducer transducer;
	private TChannel channel;
	private int workStation;
	public WorkStationAgent(String name,Transducer transducer,TChannel channel,int
workStation)
	{
		super();
		this.name = name;
		this.transducer = transducer;
		this.workStation = workStation;
		this.transducer.register(this, channel);
		this.channel = channel;
		this.status = StationStatus.EMPTY;
		this.stateOfnextCF = NextCFState.BUSY;
	}
```

Messages:

```
	//gets part from conveyor and sends msg back to conveyor to verify
	public void msgHereIsPart(Part p)
	{
		this.parts.add(p);
		System.out.println("size of parts in workstation: " + this.parts.size());
		stateChanged();
	}

	//machine ready
	public void msgImReady()
	{
		stateOfnextCF = NextCFState.READY;
		System.out.println("next conveyor is ready");
		stateChanged();
```

```
        }

        //part received from previous conveyor, so we can now delete the previous part off
the list
        public void msgPartReceived()
        {
                //conveyor.msgImFree();
                this.status = StationStatus.NOTHINGTODO;
                parts.remove(0);
                System.out.println("remove parts size: " + parts.size());
                System.out.println("c14 state is: " + stateOfnextCF);
                stateChanged();

        }

        //part is now on machine and lets conveyor know that he received the part
        public void PartOnMachine()
        {
                System.out.println("Part is on machine");
                //stateOfnextCF = NextCFState.READY;
                status = StationStatus.PARTONSTATION;
                conveyor.msgPartReceived(parts.get(0));
                stateChanged();
        }

        //preparing to UV LAMP the part
        public void ProcessingPart()
        {
                System.out.println("preparing to process part");
                status = StationStatus.PROCESSING;
                stateChanged();
        }

        //preparing to load to next CF
        public void PreparingToload()
        {
                System.out.println("preparing to load to next CF");
                status = StationStatus.PREPARETOLOAD;
                stateChanged();
        }

        //successfully sent the part to the next conveyor
        public void PartSentToNextConveyor()
        {
                System.out.println("part sucessfully sent to other conveyor");
                status = StationStatus.PARTSENT;
                //conveyor.msgMachineFree();
                stateChanged();
```

```
        }

        //stall the machine
        public void StallMachine()
        {
                System.out.println("Machine stalled!");
                stateOfnextCF = NextCFState.BUSY;
        }
```

Scheduler:

```
    public boolean pickAndExecuteAnAction()
    {
          if(status == StationStatus.NOTHINGTODO) //&& stateOfnextCF != NextCFState.BUSY)
        {
                NotifyConveyorImFree();
                return true;
        }
         else if(status == StationStatus.PARTONSTATION)
        {
                CheckIfPartNeedsProcessing();
                return true;
        }

        else if(status == StationStatus.PREPARETOLOAD && stateOfnextCF == NextCFState.READY)
        {
                UnloadToNextCF();
                return true;
        }
        else if(status == StationStatus.PROCESSING)
        {
                ProcessThePart();
                return true;
        }
        else if(status == StationStatus.PARTSENT)
        {
                status = StationStatus.NEEDNOTIFICATION;
                return true;
        }


        return false;
    }
```

Actions:

```
//Check to see if part needs processing
private void CheckIfPartNeedsProcessing()
{
	if(parts.get(0).getRecipe().charAt(workStation) == '1')
	{
		status = StationStatus.NEEDSPROCESSING;
		System.out.println("PART NEEDS TO BE PROCESSED!");
		this.ProcessingPart();
	}
	else
	{
		status = StationStatus.NONEEDPROCESSING;
		System.out.println("PART DOES NOT NEED TO BE PROCESSED!");
		this.PreparingToload();
	}
}

//UV LAMPING PART
private void ProcessThePart()
{
	System.out.println("Processing part");
	transducer.fireEvent(channel, TEvent.WORKSTATION_DO_ACTION, null);
	status = StationStatus.NEEDNOTIFICATION;
}

//Load to next CF
private void UnloadToNextCF()
{
	System.out.println("loading to next conveyor");
	transducer.fireEvent(channel,TEvent.WORKSTATION_RELEASE_GLASS, null);
	status = StationStatus.NEEDNOTIFICATION;
	this.nextCF.msgHereIsNewPart(nextCF, parts.get(0));
	//stateOfnextCF = NextCFState.BUSY;
	stateChanged();

}
//check if next conveyor family is ready, if ready, send msg
private void NotifyConveyorImFree()
{
	System.out.println("notifying conveyor machine is free");
	//if(stateOfnextCF == NextCFState.READY)
	conveyor.msgImFree();
	status = StationStatus.NEEDNOTIFICATION;
}
```

```
	public void eventFired(TChannel channel, TEvent event, Object[] args)
	{
		if (channel == this.channel && event == TEvent.WORKSTATION_LOAD_FINISHED)
		{
			this.PartOnMachine();
		}

		else if (channel == this.channel && event ==
TEvent.WORKSTATION_GUI_ACTION_FINISHED)
		{
			this.PreparingToload();
		}
		else if (channel == this.channel && event ==
TEvent.WORKSTATION_RELEASE_FINISHED)
		{
			this.PartSentToNextConveyor();
		}

	}
```