Edward Fuks

Conveor

Data

**private** **int** myIndex;

**private** String name;

**private** Transducer myTransducer;

**private** ConveyorFamily cFBack; // COnveyorFamily from the left

// ------------------only used in the ConveyorSimpleFamily-------

**private** ConveyorFamily cFFront; // COnveyorFamily from the left

// -----------------------

**private** ConveyorFamily myCF; // my ConveorFamily

**private** Machine myMachine; // machine from the right

// states of conveyor

// broken state used when the sensor is broken

// brokenConveyor is used when the entire conveyor is broken

**private** **boolean** brokenConveyor;

**private** **enum** MyState {

*broken*, *working*

};

// convayoe can be broken or working normally

**private** MyState myState;

**private** **enum** MachineState {

*busy*, *ready*, *broken*

};

**private** **boolean** debug\_mode;

**private** MachineState machineRightState;

**enum** SensorState {

*pressed*, *released*, *BROKEN*

};

SensorState leftSensorState;

SensorState rightSensorState;

**private** **boolean** moving; // moving conveyor or not moving

**private** List<Part> partsOnMe = Collections

.*synchronizedList*(**new** ArrayList<Part>()); // parts on the conveyor

// keeps track of how many parts are on the conveyor.

**private** **int** numberOfPartsAgent;

**private** **int** numberOfPartsSensor;

// when the machine tells that it received the part (load finished)

**boolean** receivedMsgPartReceived;

// used to remember whever notified the back CF that this CF received the

// part

// is false when received the message hereISNewPart

// and true when notifyCFReaidy is called

**boolean** notifiedBackCF;

Messages

/\*\*

\* the machine notifies us that it received the part

\*

\*/

**public** **void** conveyorPartReceived(ConveyorFamily cf) {

**synchronized** (partsOnMe) {

partsOnMe.remove(0);

print("msg: I received message part received now I have"

+ partsOnMe.size());

machineRightState = MachineState.*busy*;

receivedMsgPartReceived = **true**;

stateChanged();

}

}

/\*\*

\* the machine sends this message when it is ready

\*/

**public** **void** iAmReady() {

machineRightState = MachineState.*ready*;

stateChanged();

print("msg: the my machine says that it is ready");

}

/\*\*

\* the Conveyor Family sends the part from the left to us

\*/

**public** **void** hereIsNewPart(Part part) {

**synchronized** (partsOnMe) {

partsOnMe.add(part);

print("msg: the back Conveyor Family passes me the part "

+ part.toString());

numberOfPartsAgent++;

}

}

/\*\*

\* for non-normative stopping of the conveyor

\*/

**public** **void** msgConveyorStopping() {

// **TODO** Auto-generated method stub

}

/\*\*

\* this message is sent by the front machine regarding it's failure conveyor

\* should not send any parts to it untill it notifies about its readiness

\*/

**public** **void** iAmBroken() {

machineRightState = MachineState.*busy*;

System.*err*.println(name + " the machine tells that it is broken");

stateChanged();

}

/\*\*

\* the gui notified the conveyor that it is fixed trhough the transducer

\*/

**private** **void** fixConveyor() {

myState = MyState.*working*;

stateChanged();

}

/\*\*

\* the gui notified the conveyor that it is broken trhough the transducer

\*/

**private** **void** conveyorIsBroken() {

myState = MyState.*broken*;

stateChanged();

}

Scheduler

**if** (myState == MyState.*broken*) {

stopConveyor();

}

**else** **if** (leftSensorState is *pressed* and !moving

and rightSensorState is *released*

and myState is *working*) {

startConveyor();

} **else** **if** (rightSensorState is *pressed*

and machineRightState is *busy* and moving and myState is *working*) {

stopConveyor();

} **else** **if** (!partsOnMe.isEmpty()

and rightSensorState is *released* and !moving and myState is *working*) {

startConveyor();

} **else** **if** (rightSensorState is *pressed*

and machineRightState is *ready* and !moving and myState is *working*) {

startConveyor();

**this**.passThePartTotheMachine();

} **else** **if** (rightSensorState is *pressed*

and machineRightState is *ready* and moving and receivedMsgPartReceived and myState is *working*) {

**this**.passThePartTotheMachine();

} **else** **if** (rightSensorState is *pressed*

and machineRightState is *busy* and moving and receivedMsgPartReceived and myState is *working*) {

stopConveyor();

} **else** **if** (partsOnMe.isEmpty() and moving

and myState is *working*) {

stopConveyor();

}

Actions

/\*\*

\* the function stops the coneyor

\*/

**private** **void** stopConveyor() {

moving = **false**;

Integer args[] = **new** Integer[1];

args[0] = myIndex;

transducer.fireEvent(TChannel.*CONVEYOR*, TEvent.*CONVEYOR\_DO\_STOP*, args);

print("act: stopping conveyor");

}

/\*\*

\* the conveyor passes the first part in the list to the popUp

\*/

**private** **void** passThePartTotheMachine() {

**synchronized** (partsOnMe) {

**if** (myMachine != **null**) {

myMachine.machinePart(partsOnMe.get(0));

print(" act: passing the part to machine "

+ partsOnMe.get(0).toString() + " total num i have "

+ partsOnMe.size());

// will become true when the CF in front will notify that it

// received the part

**this**.receivedMsgPartReceived = **false**;

} **else** {

**this**.cFFront.msgHereIsNewPart(myCF, partsOnMe.get(0));

print(" act: passing the part to conveyor "

+ partsOnMe.get(0).toString() + " total num i have "

+ partsOnMe.size());

// will become true when the CF in front will notify that it

// received the part

**this**.receivedMsgPartReceived = **false**;

}

}// end synchronized

}

/\*\*

\* the function starts the conveyor

\*/

**private** **void** startConveyor() {

moving = **true**;

Integer args[] = **new** Integer[1];

args[0] = myIndex;

transducer.fireEvent(TChannel.*CONVEYOR*, TEvent.*CONVEYOR\_DO\_START*, args);

print("act : conveyor started");

}

/\*\*

\* the conveyor tels the CF from the left that it can accept the parts.

\*/

/\*\*

\* used to output the messages in debugging mode

\*

\* **@param** string

\*/

**private** **void** print(String string) {

**if** (debug\_mode)

System.*out*.println(**this**.name + ":" + string);

}

/\*\*

\* by calling this public function you can see print statements when t he

\* agent runs

\*/

@Override

**public** **void** setPrintEnabled() {

debug\_mode = **true**;

}

/\*\*

\* used to output the error messages in debugging mode

\*

\* **@param** string

\*/

**private** **void** printErr(String string) {

**if** (debug\_mode)

System.*err*.println(**this**.name + ":" + string);

}

**public** **void** eventFired(TChannel channel, TEvent event, Object[] args) {

**if** ((Integer) args[0] / 2 == myIndex and channel == TChannel.*SENSOR*) {

**if** (channel == TChannel.*SENSOR*

and event == TEvent.*SENSOR\_GUI\_RELEASED*) {

**if** ((Integer) args[0] % 2 == 0) {

**this**.leftSensorState = SensorState.*released*;

printErr("msg: the left sensor is released");

stateChanged();

cFBack.msgConveyorReady(myCF);

printErr("sending message that I am ready");

print(name

+ " notifying Back CF that my CF is ready to accept the part");

} **else** **if** ((Integer) args[0] % 2 == 1) {

**this**.rightSensorState = SensorState.*released*;

print("msg: the right sensor is released");

stateChanged();

}

} **else** **if** (channel == TChannel.*SENSOR*

and event == TEvent.*SENSOR\_GUI\_PRESSED*) {

**if** ((Integer) args[0] % 2 == 0) {

**this**.leftSensorState = SensorState.*pressed*;

print("msg: the left sensor is pressed and the number of parts is "

+ partsOnMe.size());

cFBack.msgConveyorPartReceived(myCF); // telling that CF

// received the part

// when the sensor

// is released

stateChanged();

} **else** **if** ((Integer) args[0] % 2 == 1) {

**this**.rightSensorState = SensorState.*pressed*;

print("msg: the right sensor is pressed ");

stateChanged();

}

}

} **else** **if** ((Integer) args[0] == myIndex and channel == TChannel.*CONVEYOR*) {

**if** (channel == TChannel.*CONVEYOR* and event == TEvent.*CONVEYOR\_BROKEN*) {

**this**.conveyorIsBroken();

stateChanged();

System.*err*.println(name + " conveyor is broken");

} **else** **if** (channel == TChannel.*CONVEYOR*

and event == TEvent.*CONVEYOR\_FIXED*) {

**this**.fixConveyor();

stateChanged();

System.*err*.println(name + " conveyor is fixed");

}

}

}

Workstation

Data

**private** **enum** MyOtherState {*broken*,*working*};

MyOtherState otherState;

// only used for debigging printing

**private** **boolean** debug\_mode;

**private** Integer args[] = **new** Integer[1];

// index of the machine

**private** **int** myIndex;

// name of the machine

**private** String name;

**private** Transducer myTransducer;

**private** **enum** AgentState {

*busy*, *ready*,

};

**private** **enum** MyState {

*loadFinished*, *releaseFinished*, *actionFinished*, *noAction*, *waiting*, *checkRecipe*, *doAction*, *Broken*

};

**private** MyState myState;

// Conveyor Family from the right

**private** ConveyorFamily frontCF;

// conveyor from the right

**private** Conveyor conveyorBack;

// conveyor Family from the right

**private** ConveyorFamily myCF;

**private** AgentState frontCFState;

// part on the machine

**private** Part myPart;

// channel

**private** TChannel myChannel;

**private** **int** myRecipe;

Messages

/\*\*

\* the machine gets this message when the front CF gets the part

\*/

**public** **void** conveyorPartReceived(ConveyorFamily frontCF) {

myPart = **null**;

myState = MyState.*noAction*;

**this**.frontCFState = AgentState.*busy*;

stateChanged();

print("msg: the CF in front of me received the part");

}

/\*\*

\* the machine gets the message from the front CF

\*

\* **@Override**

\*/

**public** **void** iAmReady() {

**this**.frontCFState = AgentState.*ready*;

print("msg: the CF in front of me says that it is ready");

stateChanged();

}

/\*\*

\* the machine needs to treat the part

\*/

**private** **void** msgDoAction() {

**this**.myState = MyState.*doAction*;

print("msg: asked to doAction");

}

**public** **void** machinePart(Part part) {

myState = MyState.*waiting*;

myPart = part;

print("msg: my conveyor asked to machine part "+part.toString()+"-waiting till the loading finishes");

}

/\*\*

\* stopping the agent thread

\*/

**public** **void** stopThread() {

**super**.stopAgent();

}

/\*\*

\* starting the agent thread

\*/

**public** **void** startThread() {

**super**.startThread();

}

Scheduler

**if** (myState is *Broken*) {

notifyMachineBreaks();

}

**if** (myState is *actionFinished*

and frontCFState is *ready* and otherState is *working*) {

releasePart();

}

**else** **if** (myState is *releaseFinished* and otherState is *working*) {

myState = MyState.*waiting*;

} **else** **if** (myState is *loadFinished* and otherState is *working*) {

doCheckRecipe();

} **else** **if** (myState is *doAction* and otherState is *working*) {

doAction();

} **else** **if** (myState is *noAction* and otherState is *working*) {

notifyReady();

}

Actions

**private** **void** notifyMachineBreaks() {

conveyorBack.iAmBroken();

}

**private** **void** doCheckRecipe() {

**if** (myPart.getRecipe().charAt(myRecipe) == '1') {

myState = MyState.*waiting*;

**this**.msgDoAction();

} **else** {

myState = MyState.*actionFinished*;

stateChanged();

}

print("act: I do check recipe");

}

/\*\*

\* the machine notifies the conveyor from the back that the machine is ready

\*/

**private** **void** notifyReady() {

**this**.conveyorBack.iAmReady();

myState = MyState.*waiting*;

print("act: I am notifying the conveyor ready");

}

/\*\*

\* the machine's scheduler calls this action to treat the part

\*/

**private** **void** doAction() {

transducer.fireEvent(myChannel, TEvent.*WORKSTATION\_DO\_ACTION*, args);

myState = MyState.*waiting*;

print("act: I am doing action");

}

/\*\*

\* used for debugging

\*

\* **@param** string

\*/

**private** **void** print(String string) {

**if** (debug\_mode)

System.*out*.println(**this**.name + " " + string);

}

**private** **void** printErr(String string) {

**if** (debug\_mode)

System.*err*.println(**this**.name + " " + string);

}

/\*\*

\* the scheduler calls this action to release the part

\*/

**private** **void** releasePart() {

transducer.fireEvent(myChannel, TEvent.*WORKSTATION\_RELEASE\_GLASS*, args);

myState = MyState.*waiting*;

**this**.frontCF.msgHereIsNewPart(myCF, myPart);

print("act: I am releasing the part");

}

@Override

**public** **void** eventFired(TChannel channel, TEvent event, Object[] args) {

**if** (channel == myChannel and event == TEvent.*WORKSTATION\_LOAD\_FINISHED*) {

**if** (myPart != **null**) {

myState = MyState.*loadFinished*;

conveyorBack.conveyorPartReceived(myCF);

stateChanged();

print("msg: the animation says load finishesd and I am notifying my conveyor that part received");

} **else** {

System.*err*

.println(name

+ " the error has occured- I am staying in noAction state-the front"

+ " sensor of the conveyor is broken");

myState = MyState.*waiting*;

conveyorBack.iAmBroken();

}

}

**else** **if** (channel == myChannel

and event == TEvent.*WORKSTATION\_GUI\_ACTION\_FINISHED*) {

myState = MyState.*actionFinished*;

stateChanged();

print("msg: the animation says work finished");

} **else** **if** (channel == myChannel

and event == TEvent.*WORKSTATION\_RELEASE\_FINISHED*) {

myState = MyState.*releaseFinished*;

stateChanged();

print("msg the animation says release finished");

}

**else** **if** (channel == myChannel

and event == TEvent.*WORKSTATION\_BREAK\_GLASS*) {

//this.msgBreakGlass();

}

**else** **if** (channel == myChannel

and event == TEvent.*WORKSTATION\_BROKEN*) {

otherState = MyOtherState.*broken*;

printErr("workstation is broken");

stateChanged();

}

**else** **if** (channel == myChannel

and event == TEvent.*WORKSTATION\_FIXED*) {

otherState=MyOtherState.*working*;

printErr("workstation is fixed");

stateChanged();

}

}

/\*\*

\* used for debugging

\*/

@Override

**public** **void** setPrintEnabled() {

debug\_mode = **true**;

}