//Sensor pseudo-code

Data:

ConveyorFamily cf, pcf;

SensorState state;

private List<Glass> glasses;

;

boolean pcfIsWaiting = false;

boolean stopConveyor = false;

private enum SensorState {

NULL, STOP\_CONVEYOR, EMPTY, TELL\_CONVEYOR\_TO\_SEND\_GLASS, OCCUPIED, WAIT\_CONVEYOR\_TO\_START,SENDING\_GLASS\_TO\_CONVEYOR, TELL\_POPUP\_WAITING\_FOR\_CLEAR,WAITING\_FOR\_POPUP\_SENDING\_GLASS, OCCUPIED\_BUT\_POPUP\_IS\_NOT\_OCCUPIED,OCCUPIED\_AND\_SO\_DOES\_POPUP, EMPTY\_BUT\_POPUP\_IS\_NOT\_EMPTY, EMPTY\_AND\_SO\_DOES\_POPUP

}

Messages:

public void msgCanISendGlass(Conveyor conveyor) {

print("State: " + state);

if (state == SensorState.EMPTY

|| state == SensorState.EMPTY\_AND\_SO\_DOES\_POPUP

|| state == SensorState.EMPTY\_BUT\_POPUP\_IS\_NOT\_EMPTY)

state = SensorState.TELL\_CONVEYOR\_TO\_SEND\_GLASS;

if (state == SensorState.OCCUPIED\_AND\_SO\_DOES\_POPUP

|| state == SensorState.OCCUPIED

|| state == SensorState.OCCUPIED\_BUT\_POPUP\_IS\_NOT\_OCCUPIED) {

state = SensorState.STOP\_CONVEYOR;

}

stateChanged();

}

msgIAmOccupied(Conveyor conveyor) {

state = SensorState.WAIT\_CONVEYOR\_TO\_START;

stateChanged();

}

msgIAmEmpty(Conveyor conveyor) {

if (glasses.size() > 0)

state = SensorState.SENDING\_GLASS\_TO\_CONVEYOR;

stateChanged();

}

msgIAmEmpty() {

if (state == SensorState.OCCUPIED

|| state == SensorState.OCCUPIED\_AND\_SO\_DOES\_POPUP

|| state == SensorState.OCCUPIED\_BUT\_POPUP\_IS\_NOT\_OCCUPIED) {

state = SensorState.OCCUPIED\_BUT\_POPUP\_IS\_NOT\_OCCUPIED;

} else

state = SensorState.EMPTY\_AND\_SO\_DOES\_POPUP;

// fire even here in the real code:

stateChanged();

}

msgHereIsGlass(Popup popup, Glass glass) {

glasses.add(glass);

state = SensorState.OCCUPIED;

stateChanged();

// fire event in the real code here:

}

msgIAmOccupied() {

if (state == SensorState.EMPTY) {

state = SensorState.EMPTY\_BUT\_POPUP\_IS\_NOT\_EMPTY;

} else if (state == SensorState.OCCUPIED) {

state = SensorState.OCCUPIED\_AND\_SO\_DOES\_POPUP;

}

stateChanged();

}

msgGlassIsWaiting(Conveyor conveyor) {

// do nothing, this is test msg for conveyor

stateChanged();

}

msgCanISendGlass() {

if (state == SensorState.EMPTY) {// if sensor is cleared

state = SensorState.WAITING\_FOR\_POPUP\_SENDING\_GLASS;

} else if (state == SensorState.OCCUPIED) {

state = SensorState.TELL\_POPUP\_WAITING\_FOR\_CLEAR;

}

stateChanged();

}

msgHereIsGlass(Conveyor conveyor, Glass glass) {

// here, sensor suppose to know every glass passed to it.

glasses.add(glass);

state = SensorState.OCCUPIED;

// fire event here

stateChanged();

}

Scheduler:

if (state == SensorState.OCCUPIED) {

// notify the conveyor/popup that I am occupied. can I send a glass?

if (name == "Sensor2") {

// sensor2 is the back end sensor which connect to the

// workstation/popup directly, then notify popups

cf.popup.msgCanISendGlass(this, glasses.get(0));

} else if (name == "Sensor1") {

//sensor1 is front end sensor, then, send glass to conveyor

cf.conveyor1.msgCanISendGlass(this, glasses.get(0));

}

return true;

}

if (state == SensorState.TELL\_POPUP\_WAITING\_FOR\_CLEAR) {

// tell popup to wait

notifyPopupToWait();

return true;

}

if (state == SensorState.WAITING\_FOR\_POPUP\_SENDING\_GLASS) {

notifyPopupToSend();

return true;

}

if (state == SensorState.OCCUPIED\_BUT\_POPUP\_IS\_NOT\_OCCUPIED) {

passGlassToPopup();

if (stopConveyor)

activeConveyor();

return true;

}

if (state == SensorState.SENDING\_GLASS\_TO\_CONVEYOR) {

sendGlassToConveyor();

return true;

}

if (state == SensorState.WAIT\_CONVEYOR\_TO\_START) {

notifyPopupToWait();

return true;

}

if (state == SensorState.TELL\_CONVEYOR\_TO\_SEND\_GLASS) {

notifyConveyorToSend();

return true;

}

if (state == SensorState.STOP\_CONVEYOR) {

tellConveyorToStop();

return true;

}

Methods:

activeConveyor() {

cf.conveyor1.msgStart();

stopConveyor = false;

stateChanged();

}

tellConveyorToStop() {

cf.conveyor1.msgStop();

stopConveyor = true;

state = SensorState.OCCUPIED\_AND\_SO\_DOES\_POPUP;

stateChanged();

}

notifyConveyorToSend() {

cf.conveyor1.msgIAmEmpty();

stateChanged();

}

sendGlassToConveyor() {

cf.conveyor1.msgHereIsGlass(this, glasses.remove(0));

// fire event here

state = SensorState.EMPTY;

stateChanged();

}

passGlassToPopup() {

cf.popup.msgHereIsGlass(this, glasses.remove(0));

state = SensorState.EMPTY;

stateChanged();

}

notifyPopupToSend() {

if (pcf != null) {

pcf.popup.msgIAmEmpty(this);

pcfIsWaiting = false;

}

stateChanged();

}

notifyPopupToWait() {

if (pcf != null) {

pcf.popup.msgIAmOccupied(this);

pcfIsWaiting = true;

}

state = SensorState.OCCUPIED;// change the state back to occupied

stateChanged();

}

getName() {

return name;

}