Class Server {

Public:

Server() {}

Selector getSelector() {return selector;}

EventManger getEventManger() { return eventMgr;}

Private:

Selector selector;

EventManger eventMgr;

}

Class GNC : public Server {

Public:

GNC() {

SCComms sc(getSelector(), getEventManger());

ImageProcessorComms ip(getSelector(), getEventManger());

}

Class ImageProccsor : public Server {

Public:

GNC() {

SCComms sc(getSelector(), getEventManger());

ImageProcessorComms ip(getSelector(), getEventManger());

}

Class WatchDOg : public Server {

Public:

GNC() {

SCComms sc(getSelector(), getEventManger());

ImageProcessorComms ip(getSelector(), getEventManger());

}

Class SCComms : public Server {

Public:

GNC() {

SCComms sc(getSelector(), getEventManger());

ImageProcessorComms ip(getSelector(), getEventManger());

}

class SocketToSCComms {

public:

SCComms(Selector& selector, EventManger& eventMgr) {

Sc = createSocketToSC();

Selector.registerForRead(sc);

Selector.registerForRead(watchdog);

}

private:

Socket\* sc;

Socket\* watchdog;

}

Small World:

SC and S/C Comms

Server Class

SCComms and SC classes

Socket(SC)

Socket(SCComms)

Class S/C Comms {

}

int main{

S/C Coms sccom;

Sccom.connect()

}

**client**

a =socket()

bind(a)

connect(s)

write()

read()

**Server**

S = Socket()

Bind(s)

Listen(s) 1

New\_s = Accept(s) 2

Read(new\_s) & write(new\_s)