class Publisher

{

public ICommand command;

Publisher (ICommand cmd)

{

command=cmd;

}

public void executeCommand()

{

command.execute();

}

};

--------------------------------------------------------------------------------------------------------------------------------

//Interface

class ICommand

{

public void execute();

};

--------------------------------------------------------------------------------------------------------------------------------

class CompositeCommand:ICommand

{

std:: list<ICommand> commandList;

public void add (ICommand cmd)

{

commandList.push\_back(cmd);

}

public void execute()

{

std:: list<ICommand>::iterator itr;

for(itr=commandList.begin(); itr !=commandList.end(); itr++)

{

itr->execute();

}

}

};

------------------------------------------------------------------------------------------------------------------------------------

class Command1:ICommand

{

private Subscriber1 subscriber;

public void execute()

{

subscriber.action1();

}

};

---------------------------------------------------------------------------------------------------------------------------------------

class Command2:ICommand

{

private Subscriber2 subscriber;

public void execute()

{

subscriber.action2();

}

};

-----------------------------------------------------------------------------------------------------------------------------------------

class Subscriber1{

public void action1()

{

//perform 1st action

}

};

-----------------------------------------------------------------------------------------------------------------------------------

class Subscriber2{

public void action2()

{

//perform 2nd action

}

};

------------------------------------------------------------------------------------------------------------------------------