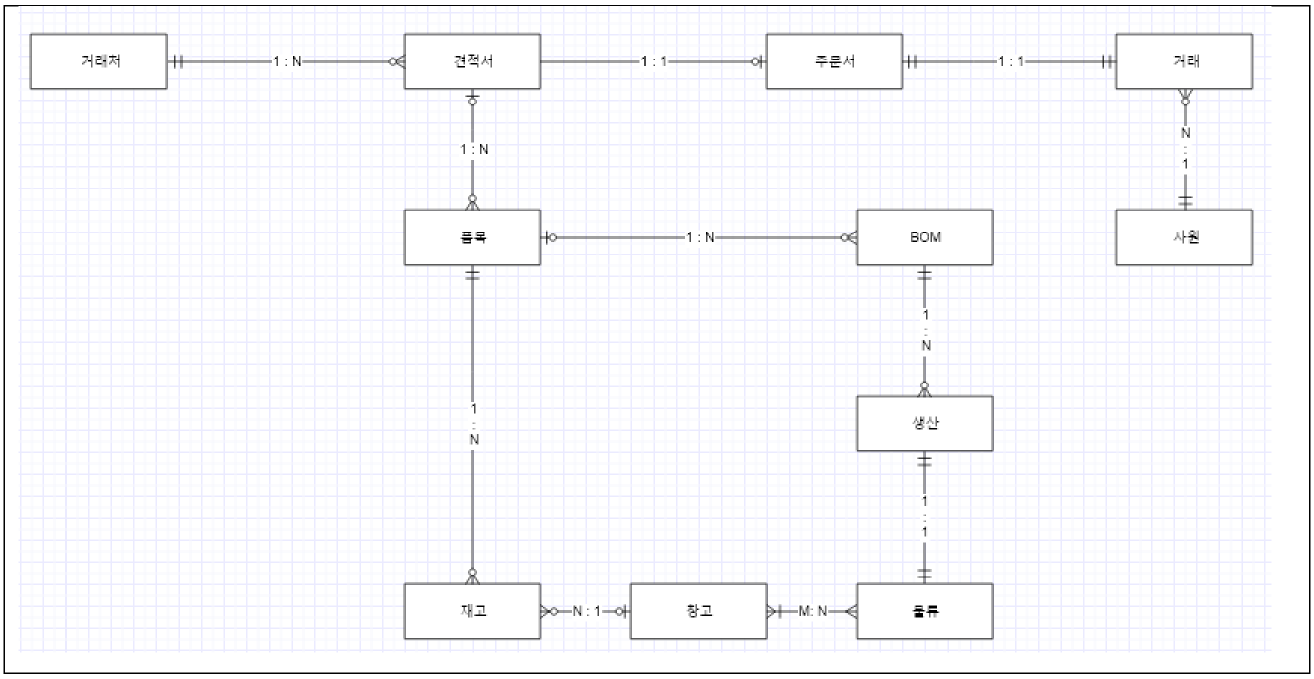
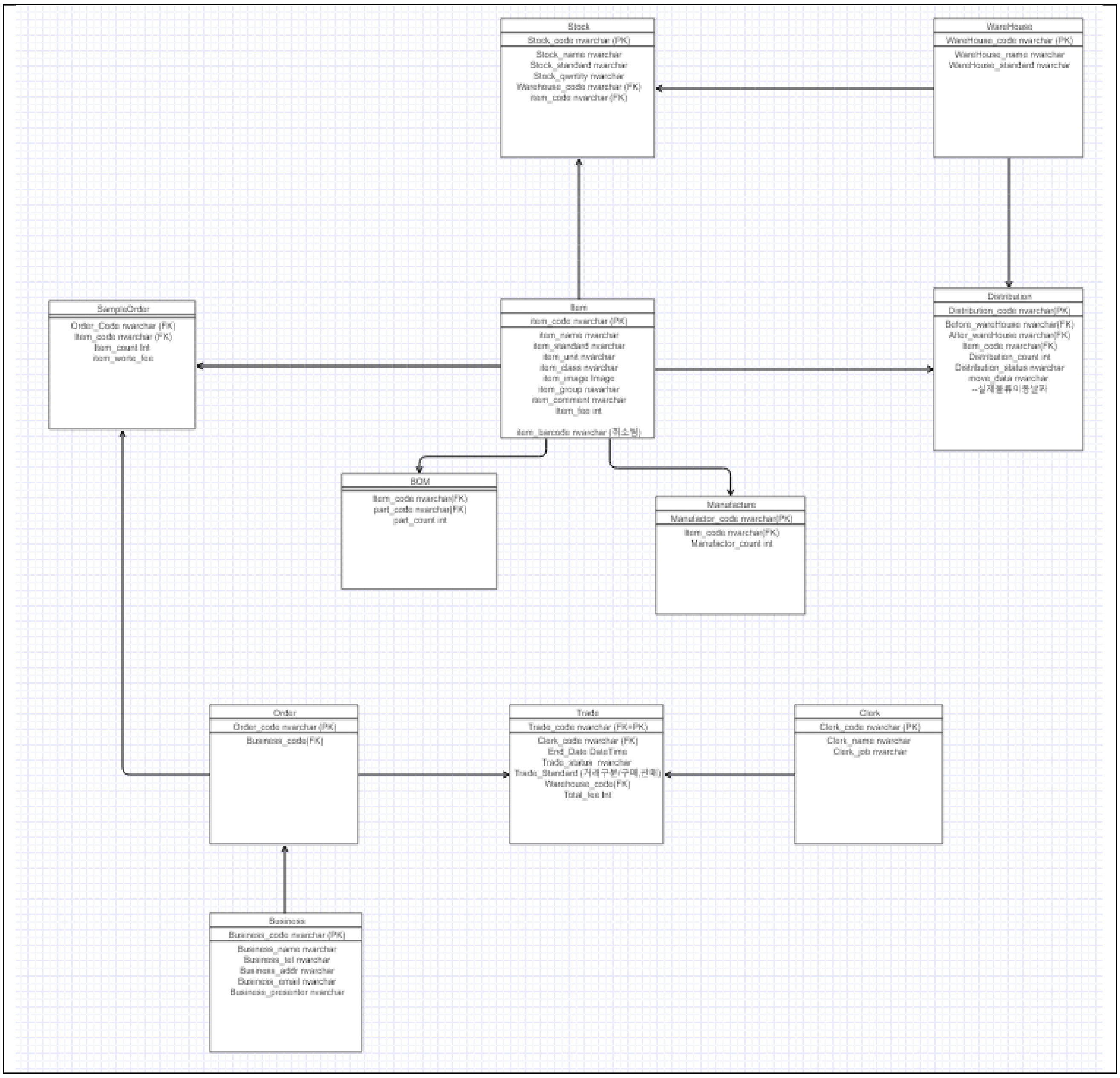
- ERD




Stock
Stock_code nvarchar (PK)
Stock_name nvarchar
Stock_standard nvarchar
Stock_qentity nvarchar
Warehouse_code nvarchar (FK)
item_code nvarchar (FK)
WareHouse
WareHouse_code nvarchar (PK)
WareHouse_name nvarchar
WareHouse_standard nvarchar
SampleOrder
Order_Code nvarchar (FK)
Item_code nvarchar (FK)
Item_count Int
item_works_fee
Item
item_code nvarchar (PK)
item_name nvarchar
item_standard nvarchar
item_unit nvarchar
item_class nvarchar
item_image Image
item_group navarhar
item_comment nvarchar
Item_fee int
item_barcode nvarchar (취소됨)
Distribution
Distribution_code nvarchar(PK)
Before_wareHouse nvarchar(FK)
After_wareHouse nvarchar(FK)
Item_code nvarchar(FK)
Distribution_count int
Distribution_status nvarchar
move_data nvarchar
--실제물류이동날짜
BOM
Item_code nvarchar(FK)
part_code nvarchar(FK)
part_count int
Manufacture
Manufactor_code nvarchar(PK)
Item_code nvarchar(FK)
Manufactor_count int
Order
Order_code nvarchar (PK)
Business_code(FK)
Trade
Trade_code nvarchar (FK+PK)
Clerk_code nvarchar (FK)
End_Date DateTime
Trade_status nvarchar
Trade_Standard (거래구분(구매,판매)
Warehouse_code(FK)
Total_fee Int
Clerk
Clerk_code nvarchar (PK)
Clerk_name nvarchar
Clerk_job nvarchar
Business
Business_code nvarchar (PK)
Business_name nvarchar
Business_tel nvarchar
Business_addr nvarchar
Business_email nvarchar
Business_presenter nvarchar

- 테이블 명세서

테이블정의서	Database	ERP_INFO						
	스키마	dbo						
테이블명	Business							
코멘트	거래처 테이블							
Col	Name	Type	Length	Key	Null	Identity	Default	Comments
1	Business_code	nvarchar	20	PK				거래처 번호 (사용자 입력)
2	Business_name	nvarchar	50					거래처 이름
3	Business_tel	nvarchar	13					거래처 전화번호
4	Business_addr	nvarchar	Max					거래처 주소
5	Business_email	nvarchar	50		NULL			
6	Business_presenter	nvarchar	20					대표자명

테이블정의서	Database	ERP_INFO						
	스키마	dbo						
테이블명	SampleOrder							
코멘트								
Col	Name	Type	Length	Key	Null	Identity	Default	Comments
1	Order_code	nvarchar	20	PK,FK				Order 테이블
2	Item_code	nvarchar	20	PK,FK				Item 테이블
3	Item_count	int						음수 불가 추후 기능구현시 설계시 참고
4	Item_wrote_fee	int						주문시점 당시의 물품가격

테이블정의서	Database	ERP_INFO						
	스키마	dbo						
테이블명	SampleOrder_BOM							
코멘트	견적서 테이블							
Col	Name	Type	Length	Key	Null	Identity	Default	Comments
1	Order_code	nvarchar	20	PK,FK				Order 테이블
2	Item_code	nvarchar	20	PK,FK				Item 테이블
3	Item_count	int						음수 불가 추후 기능구현시 설계시 참고
4	Item_wrote_fee	int						주문시점 당시의 물품가격

테이블정의서	Database	ERP_INFO						
	스키마	dbo						
테이블명	Ordered							
코멘트	주문 테이블							
Col	Name	Type	Length	Key	Null	Identity	Default	Comments
1	Order_code	nvarchar	20	PK				날짜_시퀀스번호(ex: 20170102_1) (자동생성)
2	Business_code	nvarchar	20	FK				거래처 번호

테이블정의서	Database	ERP_INFO						
	스키마	dbo						
테이블명	Trade							
코멘트	거래 테이블							
Col	Name	Type	Length	Key	Null	Identity	Default	Comments
1	Trade_code	nvarchar	20	PK,FK				Order 테이블
2	Clerk_code	nvarchar	20	FK				Clerk 테이블
3	Warehouse_code	nvarchar	20	FK	O			출고 or 입고 창고 / Warehouse 테이블
4	Trade_standard	nvarchar	20					거래구분 (구매/판매)
5	Trade_status	nvarchar	20				승인 전	진행사항 (승인전, 진행완료 등) ,
6	Total_fee	int						
7	End_Date	DateTime			O			거래종료일

테이블정의서	Database	ERP_INFO						
	스키마	dbo						
테이블명	Clerk							
코멘트	사원 테이블							
Col	Name	Type	Length	Key	Null	Identity	Default	Comments
1	Clerk_code	nvarchar	20	PK				사번 (사용자 입력?)
2	Clerk_name	nvarchar	50					사원명 (외노자 포함)
3	Clerk_job	nvarchar	30					직급
4	Clerk_password	varbinary	32					비밀번호(SHA2_256)

테이블정의서	Database	ERP_INFO						
	스키마	dbo						
테이블명	Distribution							
코멘트	물류 테이블							
Col	Name	Type	Length	Key	Null	Identity	Default	Comments
1	Distribution_code	nvarchar	20	PK		O		날짜_시퀀스번호(ex: 20170102_1) (자동생성)
2	trade_code	nvarchar	20	FK	O			관련 거래코드
3	Item_code	nvarchar	20	FK				Item 테이블
4	Before_wareHouse	nvarchar	20	FK				이전 위치 / warehouse 테이블
5	After_wareHouse	nvarchar	20	FK				이후 위치 / warehouse 테이블
6	move_date	datetime			O			이동날짜
7	Distribution_Count	int						이동될 물류의 개수
8	distribution_status	nvarchar	20				대기	물류 상태(출고중, 완료) 등

테이블정의서	Database	ERP_INFO
	스키마	dbo
테이블명	Warehouse	
코멘트	창고 테이블	

Col	Name	Type	Length	Key	Null	Identity	Default	Comments
1	WareHouse_code	nvarchar	20	PK				거래처창고는 거래처코드로 입력(그외 사용자입력)
2	WareHouse_name	nvarchar	50					거래처창고는 거래처이름으로 입력
3	WareHouse_standard	nvarchar	20					구분 (창고 / 공장 / 거래처)

테이블정의서	Database	ERP_INFO
	스키마	dbo
테이블명	Stock	
코멘트	재고	

Col	Name	Type	Length	Key	Null	Identity	Default	Comments
1	item_code	nvarchar	20	FK				Item 테이블
2	Warehouse_code	nvarchar	20	FK				Warehouse 테이블
3	Stock_name	nvarchar	50					
4	Stock_count	int						재고 수량
5	Stock_standard	nvarchar	20		O			규격(박스,롤,컨테이너....)

테이블정의서	Database	ERP_INFO
	스키마	dbo
테이블명	Item	
코멘트	품목 테이블	

Col	Name	Type	Length	Key	Null	Identity	Default	Comments
1	item_code	nvarchar	20	PK				해당 필드로 바번호 생성할 것
2	item_name	nvarchar	50					
3	item_standard	nvarchar	20		O			규격(박스, 롤, 컨테이너 등)
4	item_unit	nvarchar	20		O			단위 (미터, 리터, 킬로그램)
5	item_class	nvarchar	20					구분(완제품,자재,반제품...)
6	item_fee	int						품목 현재가격 (지속적으로 업데이트)
7	stock_count	int						품목 재고 갯수
8	item_image	Image			O			이미지
9	item_group	nvarchar	20		O			해당품목그룹 (식음료,컴퓨터셋트)
10	item_comment	nvarchar	Max		O			비고

테이블정의서	Database	ERP_INFO
	스키마	dbo
테이블명	Bom	
코멘트	자재 청구 테이블	

Col	Name	Type	Length	Key	Null	Identity	Default	Comments
1	Item_code	nvarchar	20	FK				Item 테이블
2	part_code	nvarchar	20	Fk				Item 테이블
3	part_count	int						필요 개수

테이블정의서	Database	ERP_INFO
	스키마	dbo
테이블명	Manufacture	
코멘트	생산 테이블	

Col	Name	Type	Length	Key	Null	Identity	Default	Comments
1	Manufactor_code	nvarchar	20	PK		O		날짜_시퀀스번호(ex: 20170102_1) (자동생성)
2	Item_code	nvarchar	20	FK				Item 테이블
3	Manufactor_count	int						생산 갯수

테이블정의서	Database	ERP_INFO
	스키마	dbo
테이블명	DB_date	
코멘트	시퀀스초기화를 위한 테이블	

Col	Name	Type	Length	Key	Null	Identity	Default	Comments
1	dbdate	date		PK		O		날짜

- 소스코드

```
using MiniERP.Model.DAO.Message;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace MiniERP.View
{
    public partial class RealTimeMonitor : Form
    {
        Machine_Monitoring machine_Server;

        public RealTimeMonitor()
        {
            InitializeComponent();
            this.ShowInTaskbar = false;                        // 작업표시줄X
            this.FormBorderStyle = FormBorderStyle.SizableToolWindow;   // 프로그램 전환기 숨기기
        }

        private void RealTimeMonitor_Load(object sender, EventArgs e)
        {
            machine_Server = new Machine_Monitoring(txt_Log);
            machine_Server.Start();

        }

        private void txt_Log_TextChanged(object sender, EventArgs e)
        {
            txt_Log.SelectionStart = txt_Log.Text.Length;
            richTextBox1.AppendText(txt_Log.Text + Environment.NewLine);

            //ClientConnectingCheck(txt_Log.Text);
        }
        private void ClientConnectingCheck(string msg)
        {
            string temp = "";
            if (msg.Contains("[command]") && msg.Contains("is connecting"))
                temp = msg.Replace("[command]", "").Replace("is connecting", "");    //[pc1]


            foreach (Control item in panel1.Controls)
            {
                if (item.Text == temp)
                {
                    ((CheckBox)item).Checked = true;
                }
            }

        }

        private void RealTimeMonitor_FormClosing(object sender, FormClosingEventArgs e)
        {
            machine_Server.CloseSever();
        }

        private void btn_inputCountRequest_Click(object sender, EventArgs e)
        {
            string command = "[command]";
            switch (((Button)sender).Text)
            {
                case "종료": command += selectPc + "exit"; machine_Server.SendMsg(command); break;
                case "재부팅": command += selectPc + "restart"; machine_Server.SendMsg(command); break;
                case "투입 자재 개수": command += selectPc + "barcode"; machine_Server.SendMsg(command); break;
                case "머신 설정 변경": Machine_Info_Change change = new Machine_Info_Change(machine_Server, selectPc); change.ShowDialog(); break;
                default:
                    break;
            }
        }

        string selectPc = ""; bool clickSwitch = false;
        private void pic_pc1_MouseClick(object sender, MouseEventArgs e)
        {
            if (e.Button == MouseButtons.Left && clickSwitch == false)
            {
                if (((PictureBox)sender).BackColor != SystemColors.ButtonShadow)
                {
                    ((PictureBox)sender).BackColor = SystemColors.ButtonShadow;
                    selectPc = "[" + ((PictureBox)sender).Name + "]";
                }
                clickSwitch = true;
            }

            else if (((PictureBox)sender).BackColor == SystemColors.ButtonShadow && clickSwitch == true)
            {
                clickSwitch = false;
                if (((PictureBox)sender).BackColor == SystemColors.ButtonShadow)
                {
                    ((PictureBox)sender).BackColor = SystemColors.ButtonHighlight;
                }
            }
            else
                MessageBox.Show("한번에 하나의 명령만 가능합니다.");
        }
        /// <summary>
        /// 리치텍스트박스 변경시 일어날 메소드
        /// </summary>
        private void richTextBox1_TextChanged(object sender, EventArgs e)
         {
            ServerStateChecker();
        }
        /// <summary>
        /// 텍스트박스의 서버의 커맨드를 이용해 체크박스 체크해줍니다.
        /// </summary>
        private void ServerStateChecker()
        {
            string temp = txt_Log.Text;
            if (temp.Contains("[command]") && temp.Contains("is connecting"))
            {
                temp = temp.Replace("[command]", "").Replace("is connecting",
"");
                foreach (Control item in panel1.Controls)
                {
                    if (item.Text == temp)
                    {
                        ((CheckBox)item).Checked = true;
                    }
                }
            }

            else if (temp.Contains("[command]") && temp.Contains("is
endconnecting"))
            {
                temp = temp.Replace("[command]", "").Replace("is endconnecting",
"");
                foreach (Control item in panel1.Controls)
                {
                    if (item.Text == temp)
                    {
                        ((CheckBox)item).Checked = false;
                    }
                }
            }
        }
        /// <summary>
        /// 텍스트박스의 서버 상태를 체크하고, 버튼의 enable속성을 변경해줍니다.
        /// </summary>
        private void check_pc1_CheckedChanged(object sender, EventArgs e)
        {
            string temp = ((Control)sender).Text;                       //
[pc1]
            temp = temp.Replace("[", "").Replace("]", "");              //  pc1

            foreach (Control item in panel1.Controls)
            {
                if (item.Name == temp&&((CheckBox)sender).Checked)
                    ((PictureBox)item).Enabled = Enabled;

●                   else if (item.Name == temp && !((CheckBox)sender).Checked)
●                       ((PictureBox)item).Enabled = false;
●               }
●           }
●
●           //  테스트 위한 메서드
●           private void btn_Tester_Click(object sender, EventArgs e)
●           {
●               pc1.Enabled = true;
●               pc2.Enabled = true;
●               pc3.Enabled = true;
●               pc4.Enabled = true;
●               pc5.Enabled = true;
●           }
●       }
●   }
●
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Sockets;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace MiniERP.Model.DAO.Message
{
    public class Machine_Monitoring
    {

        TcpClient client = new TcpClient();//tcpclient를 미리 초기화해놓음
        NetworkStream stream = default(NetworkStream);
        Thread thread;
        object txtBox;



        string ip = "192.168.0.6";
        public string Ip { get { return ip; } set { this.ip = value; } }

        string readData = null;

        public Machine_Monitoring(object txtBox)
        {
            this.txtBox = txtBox;
        }

        public void Start()
        {
            IAsyncResult access = null;
            try
            {
                access = client.BeginConnect(ip, 4444, null, null);
                var result = access.AsyncWaitHandle.WaitOne(TimeSpan.FromSeconds(1));

                stream = client.GetStream();

                byte[] name = Encoding.UTF8.GetBytes("master");     //  접속 닉네임?
주라고하드라 추후수정
                stream.Write(name, 0, name.Length);
                stream.Flush();

                thread = new Thread(getMsg);
                thread.Start();
            }
            catch (Exception)
            {
                return;
            }
        }

        private void getMsg()
        {
            while (true)
            {
                stream = client.GetStream();
                Byte[] byteFrom = new byte[client.SendBufferSize];
                stream.Read(byteFrom, 0, client.SendBufferSize);
                readData = Encoding.UTF8.GetString(byteFrom);       //  getString

                ((TextBox)txtBox).Text = readData;

            }
        }

        public void SendMsg(string msg)
        {
            Byte[] byteFrom = Encoding.UTF8.GetBytes(msg);
            stream.Write(byteFrom, 0, byteFrom.Length);
            stream.Flush();
        }

        public void CloseSever()
        {
            byte[] msgTemp = Encoding.UTF8.GetBytes("접속종료합니다");
            stream.Write(msgTemp, 0, msgTemp.Length);
            stream.Flush();

            client.Close();
            stream.Close();
        }
    }
}
```

```
using MiniERP.Model.DAO.Message;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace MiniERP.View
{
    public partial class Machine_Info_Change : Form
    {
        string name;
        string ip;
        Machine_Monitoring server;

        public Machine_Info_Change(Machine_Monitoring server,string name)
        {
            InitializeComponent();
            this.name = name;
            this.ip = server.Ip;
            this.server = server;
        }

        private void radioButton1_CheckedChanged(object sender, EventArgs e)
        {
            if (radio_Name.Checked == true)
            {
                txt_Name.Enabled = true;
                txt_Ip.Enabled = false;
            }
            else
            {
                txt_Name.Enabled = false;
                txt_Ip.Enabled = true;
            }
        }

        private void Machine_Info_Change_Load(object sender, EventArgs e)
        {
            txt_Name.Enabled = true;
            txt_Ip.Enabled = false;

            txt_Name.Text = this.name;
            txt_Ip.Text = this.ip;
        }

        private void btn_Submit_Click(object sender, EventArgs e)
        {
            string command = "[command]";
            if (radio_Name.Checked == true)
            {
                command += this.name + "[name]" + txt_Name.Text;
                server.SendMsg(command);
            }
            else
            {
                command += this.name + "[ip]" + txt_Ip.Text;
                server.SendMsg(command);
            }

            Close();
        }
    }
}
```

```
using MiniERP.Model.DAO;
using MiniERP.VO;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace MiniERP.View
{
    public partial class FrmDashBoard : Form
    {
        //private bool menu_OpenChk = false;

        List<Trade> trades = new List<Trade>();

        public FrmDashBoard()
        {
            InitializeComponent();
            lbl_ToDay.Text = "오늘은 " + DateTime.Today.ToShortDateString() + "
입니다.";
            TradeListShow();
        }

        /// <summary>
        /// 트레이드 리스트 다시그립니다.
        /// </summary>
        public void TradeListShow()
        {
            trades.Clear();
            trades = GetTreade();
            ControlAdd(trades);
        }
         /// <summary>
        /// '판매' 거래에대한 정보를 리스트에 초기화
        /// </summary>
        private List<Trade> GetTreade()
        {
            TradeDAO tradeDAO = new TradeDAO();
            return tradeDAO.GetProgTrade("판매");
        }

        /// <summary>
        /// 거래리스트에 있는 모든 정보에 대한 컨트롤을 동적생성합니다.
        /// </summary>
        private void ControlAdd(List<Trade> list)
        {
            panel_TradeList.Controls.Clear();

            int x = 11; int y = 5;
            foreach (var item in list)
            {
                ToDoList temp = new ToDoList(item,split.Panel2,this);
                temp.Location = new Point(x, y);
                y += 48;

                panel_TradeList.Controls.Add(temp);
                temp.Show();
            }
        }

        private void FrmDashBoard_Load(object sender, EventArgs e)
        {
            panel1.AutoScroll = true;
        }



        private void timer1_Tick(object sender, EventArgs e)
        {
            lbl_Time.Text = "현재시간 : " + DateTime.Now.ToLongTimeString();
        }

        private void btn_Refresh_Click(object sender, EventArgs e)
        {
            TradeListShow();
        }
    }
}
```

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Net;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

//  taskkill /im minierp_client_jsu.exe /f
```

```
namespace MiniErp_Client_jsu
{
    public partial class Form1 : Form
    {
        Machine machine = new Machine();
        Chatting chatting;

        //  erro , command 리스트
        List<Erro> erros = new List<Erro>();
        List<Command> commands = new List<Command>();
        List<Barcode> codes = new List<Barcode>();

        public Form1()
        {
            InitializeComponent();
            this.Text = machine.Name;
            this.TopMost = true;
            chatting = new Chatting(machine.Ip, machine.Name, codes);

            //StringBuilder sb = new StringBuilder();
            //sb.AppendLine("---------[투입 현황]");
            //sb.AppendLine("---------[2019-03-02 6:44:50]");
            //sb.AppendLine("*[1234]      1");
            //sb.AppendLine("*[1235]      1");
            //sb.AppendLine("----------------------------");

            ////  바코드 문자열잘라줄때쓰자
            //string temp = sb.ToString().Remove(0,
sb.ToString().IndexOf('*')).Replace("-", "").Trim();
            //string[] temp2 = temp.Split('*');
            //MessageBox.Show(temp);
            //foreach (var item in temp2)
            //{
            //    MessageBox.Show(item);
            //}
        }



        private void txt_Barcode_KeyDown(object sender, KeyEventArgs e)
       {
            if (e.KeyCode == Keys.Enter)
            {
                string temp = txt_Barcode.Text;
                txt_Barcode.Clear();

                foreach (var item in codes)
                {
                    if (item.Barcode_Code == temp)
                    {
                        item.Barcode_Count += 1;    // 같다면 카운트
                        return;
                    }
                }
                //  없다면 추가
                codes.Add(new Barcode(temp));
            }

        }

        private void Form1_Load(object sender, EventArgs e)
        {
            #region erro test 모듈
            //Erro testErro = new Erro(1);
            //MessageBox.Show(testErro.Erro_Code + "₩n" + testErro.Head +
testErro.Erro_String);
            #endregion

            if (chatting.Start())
                chk_Server.Checked = true;

            toolStripTextBox1.Text = AppConfiguration.GetAppConfig("name");
            toolStripTextBox2.Text = AppConfiguration.GetAppConfig("ip");
        }

        private void BarcodeMonitor()
        {
            StringBuilder sb = new StringBuilder();
            sb.AppendLine("---------[투입 현황]"+machine.Name);
            sb.AppendLine(NowTime());
            foreach (var item in codes)
            {
                txt_Log.Clear();
                sb.AppendLine("*["+item.Barcode_Code+"]" + "₩t" + item.Barcode_Count);
            }
            sb.AppendLine("----------------------------");
            txt_Log.Text = sb.ToString();
            this.txt_Barcode.Focus();
        }

        private string NowTime()
        {
            DateTime dt = DateTime.Now;
            return
                "---------[" + dt.ToShortDateString() + " " + dt.Hour + ":" + dt.Minute
+ ":" + dt.Second + "]";
        }

        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
            chatting.CloseServer();
        }

        private void menuStrip1_ItemClicked(object sender, ToolStripItemClickedEventArgs
e)
        {
            switch (e.ClickedItem.ToString())
            {
                case "화면고정":
                    if (this.TopMost != true)
                        this.TopMost = true;
                    else
                        this.TopMost = false;
                    break;

                case "종료":  this.Close();   break;
                case "재시작":  Application.Restart();   break;
                case "자재투입현황": BarcodeMonitor();   break;

                default:
                    break;
            }
        }

        private void toolStripTextBox1_KeyDown(object sender, KeyEventArgs e)
        {   //name 변경
            if (e.KeyCode == Keys.Enter)
            {
                AppConfiguration.SetAppConfig("name", toolStripTextBox1.Text);
                MessageBox.Show("변경되었습니다.");
            }
        }

        private void toolStripTextBox2_KeyDown(object sender, KeyEventArgs e)
        {   //  ip  변경
            if (e.KeyCode == Keys.Enter)
            {
                if (IsValidIp(toolStripTextBox2.Text))
                {
                    AppConfiguration.SetAppConfig("ip", toolStripTextBox2.Text);
                    MessageBox.Show("변경되었습니다.");
                }
                else
                {
                    toolStripTextBox2.Text = AppConfiguration.GetAppConfig("ip");
                    MessageBox.Show("올바른 IP 주소가 아닙니다.");
                    return;
                }

            }
        }
        public bool IsValidIp(string addr)
        {
            IPAddress ip;
            bool valid = !string.IsNullOrEmpty(addr) && IPAddress.TryParse(addr, out
ip);
            return valid;
        }

        private void checkBox1_CheckedChanged(object sender, EventArgs e)
        {
            if (chatting.Client == null)
            {
                txt_Log.Text += "erro server is not connected.₩n";
                return;
            }
            if (((CheckBox)sender).Text == "err_1")
            {
                Erro tempErro = new Erro(1);
                erros.Add(tempErro);
                chatting.SendMsg(tempErro.Erro_String);
                txt_Log.Text += "erro_1 Exeption₩n";
            }else if (((CheckBox)sender).Text == "err_2")
            {
                Erro tempErro = new Erro(2);
                erros.Add(tempErro);
                chatting.SendMsg(tempErro.Erro_String);
                txt_Log.Text += "erro_2 Exeption₩n";
            }

        }

        private void button1_Click_1(object sender, EventArgs e)
        {

        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Linq;
using System.Net.Sockets;
using System.Text;
using System.Threading.Tasks;

namespace MiniErp_Client_jsu
{
    class Machine
    {
        private string name = AppConfiguration.GetAppConfig("name");

        public string Name
        {
            get { return name; }
            set { name = value; }
        }

        private string ip = AppConfiguration.GetAppConfig("ip");

        public string Ip
        {
            get { return ip; }
            set { ip = value; }
        }

        public Machine()
        {
        }

    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MiniErp_Client_jsu
{
    class Erro
    {
        private string head;

        public string Head
        {
            get { return head; }
            set { head = value; }
        }
        private string erro_String;

        public string Erro_String
        {
            get { return erro_String; }
            set { erro_String = value; }
        }
        private int erro_Code;

        public int Erro_Code
        {
            get { return erro_Code; }
            set { erro_Code = value; }
        }


        public Erro(int err_Code)
        {
            this.erro_Code = err_Code;
            this.head = "[erro] ";
            ErroSetting();
        }

        private void ErroSetting()
        {
            switch (this.erro_Code)
            {
                case 1: this.erro_String = "[erro]" +
AppConfiguration.GetAppConfig("name") + "작업자 일시정지";break;
                case 2: this.erro_String = "[erro]" +
AppConfiguration.GetAppConfig("name") + "라인 에러 발생";break;
                default:
                    break;
                }
            }
        }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

//  커맨드 생성
//  올바른 커멘드인지 체크
//  커멘드 펑션 실행 ( 정지, 재시작 )
namespace MiniErp_Client_jsu
{
    class Command
    {
        private string head;                //  [command]

        public string Head
        {
            get { return head; }
            set { head = value; }
        }

        private string name;                //  [pc1]

        public string Name
        {
            get { return name; }
            set { name = value; }
        }

        private string command_Value;       //  프로그램종료

        public string Command_Value
        {
            get { return command_Value; }
            set { command_Value = value; }
        }

        private Chatting chatinfo;

        private List<Barcode> barcodes;

        public Command(string name, string command_Head, string command_Value,Chatting
chatinfo,object barcodelist)
        {
            this.name = name;
            this.head = command_Head;
            this.command_Value = command_Value;
            this.chatinfo = chatinfo;
            this.barcodes = (List<Barcode>)barcodelist;
        }

        public Command(string command_Value)
        {
            this.command_Value = command_Value;
        }

        //  바코드 리스트를 보내는 메소드
        public void BarcodeMsgMaker(List<Barcode> barcodes)
        {
            StringBuilder sb = new StringBuilder();
            sb.AppendLine("---------[투입 현황]" + this.Name);
            sb.AppendLine(NowTime());
            foreach (var item in barcodes)
            {
                sb.AppendLine("*[" + item.Barcode_Code + "]" + "₩t" +
item.Barcode_Count);
            }
            sb.AppendLine("----------------------------");

            chatinfo.SendMsg(sb.ToString());
        }

        /// <summary>
        /// 현재 시간의 스트링을 만드는 메서드
        /// </summary>
        private string NowTime()
        {
            DateTime dt = DateTime.Now;
            return
                "---------[" + dt.ToShortDateString() + " " + dt.Hour + ":" + dt.Minute
+ ":" + dt.Second + "]";
        }

        public void ChangeIp()
        {
            //  [command][this.name][ip]192.168.0.8
            if (new Form1().IsValidIp(this.command_Value))
            {
                AppConfiguration.SetAppConfig("ip", this.command_Value);
                chatinfo.SendMsg(this.name + "ip change ok");
            }
            else
            {
                chatinfo.SendMsg(this.name + "ip change not ok");
            }
        }
        public void ChangeName()
        {
            //  [command][this.name][name]pc2
            AppConfiguration.SetAppConfig("name", this.command_Value);
        }

        public  void CommandRunning()
        {
            if (this.command_Value.Contains("[ip]"))                            //
ip변경 커맨드
            {
                this.command_Value = Command_Value.Replace("[ip]", "");
                ChangeIp();
                return;
            }
            else if (this.command_Value.Contains("[name]"))                      //
이름변경 커맨드
            {
                this.command_Value = Command_Value.Replace("[name]", "");
                ChangeName();
                Application.Restart();                                          //
이름변경후 재시작
                return;
            }

            switch (this.command_Value)                                          //
커맨드 선택부
            {
                case "test_module": System.Windows.Forms.MessageBox.Show("Test"); break;
                case "exit": Application.Exit(); break;
                case "restart": Application.Restart(); break;
                case "barcode": BarcodeMsgMaker(barcodes); break;

                default:
                    break;
            }
        }


    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Sockets;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace MiniErp_Client_jsu
{
    class Chatting
    {
        #region 멤버 변수

        TcpClient client;
        NetworkStream stream = default(NetworkStream);              //  기본값 할당(해당
객체의 기본값 참조형은 null)
        Thread thread;                                              //  서버 쓰레드
        string readData = null;                                     //  서버의 메시지
        private string ip;
        private string name;
        object barcodeList;                                     //  바코드정보를
담는 리스트

        List<Command> command = new List<Command>();            //  명령어 리스트
        List<string> erro = new List<string>();                 //  에러 리스트
        public List<Command> Command { get { return command; } }
        public List<string> Erro { get { return erro; } }
        public TcpClient Client { get { return client; } }
        #endregion

        public Chatting(string ipAddr, string name, object barcodelist)
        {
            this.ip = ipAddr; this.name = name;
            this.barcodeList = barcodelist;
        }



        /// <summary>
        /// 머신 클라이언트를 시작합니다.
        /// 머신 서버와 접속합니다..
        /// </summary>
        public bool Start()
        {
            IAsyncResult access = null;
            try
            {
                if (client == null)
                {
                    client = new TcpClient();
                    access = client.BeginConnect(this.ip, 4444, null, null);
                    var result =
access.AsyncWaitHandle.WaitOne(TimeSpan.FromSeconds(1));
                    stream = client.GetStream();

                    SendMsg(this.name);          //  접속클라이언트 이름 보냄
                }
                else if (client.Connected == false)
                {
                    access = client.BeginConnect(this.ip, 4444, null, null);
                    var result =
access.AsyncWaitHandle.WaitOne(TimeSpan.FromSeconds(1));
                    stream = client.GetStream();
                }

                if (thread == null)
                {
                    thread = new Thread(getMsg);
                    thread.Start();
                    if (client.Connected)
                        SendMsg("[command]" + this.name + "is connecting");
//접속성공했다면 메시지
```

```
                }
                return true;
            }
            catch (Exception)
            {
                return false;
            }
        }

        private void DisplayText(string text)
        {
            Byte[] outStream = Encoding.UTF8.GetBytes(text);
            stream.Write(outStream, 0, outStream.Length);
            stream.Flush();
        }

        /// <summary>
        /// 서버가 보내오는 메시지를 수신합니다.    쓰레드호출 메서드
        /// </summary>
        private void getMsg()
        {
            while (true)
            {
                stream = client.GetStream();
                Byte[] byteFrom = new byte[client.SendBufferSize];
                stream.Read(byteFrom, 0, client.SendBufferSize);
                readData = Encoding.UTF8.GetString(byteFrom);
                readData = readData.Replace("₩0", "");      //  바이트배열에
빈값(쓰레기값 제거)
                CommandChacker(readData);//  올바른커맨드 판별
            }
        }

        /// <summary>
        /// 올바른 command 인지 인식합니다.
        /// 올바르다면 list에 추가함
        /// </summary>
        /// <param name="readData">서버에서 msg</param>
        private void CommandChacker(string readData)
        {
            if (readData.Contains("[command]") != true ||
readData.Contains(this.name) != true)
                return;
            else if (readData.Contains("접속") == true)   //  서버접속시 접속이라고
보내기에 이를 무시
                return;

            string tempHead = readData.Substring(readData.IndexOf("[command]"),
"[command]".Length);
            string tempName = readData.Substring(readData.IndexOf(this.name),
this.name.Length);
            string tempValue = readData.Replace(tempHead, "").Replace(tempName,
"").Replace("₩0","").Trim();

            //  임시커맨드 객체 생성후 리스트에 삽입, 그 후 커맨드 실행함
            Command tempCommand = new Command(tempHead, tempName, tempValue, this,
barcodeList);
            command.Add(new Command(tempHead, tempName, tempValue, this, barcodeList));
            tempCommand.CommandRunning();
        }

        /// <summary>
        /// 메시지 보내기
        /// </summary>
        public void SendMsg(string msg)
        {
            stream = client.GetStream();
            Byte[] byteFrom = Encoding.UTF8.GetBytes(msg);
            stream.Write(byteFrom, 0, byteFrom.Length);
            stream.Flush();
        }

        //  서버종료
        public void CloseServer()
        {
            // ex) [command][pc1]is endconnecting
            if (client.Connected != false)
            {
                byte[] msgTemp = Encoding.UTF8.GetBytes("[command]" + this.name + "is
endconnecting");
                stream = client.GetStream();
                stream.Write(msgTemp, 0, msgTemp.Length);
                stream.Flush();
                client.Close();
                stream.Close();
            }

        }


    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MiniErp_Client_jsu
{
    class Barcode
    {
        private string barcode_Code;

        public string Barcode_Code
        {
            get { return barcode_Code; }
            set { barcode_Code = value; }
        }
        private int barcode_Count;

        public Barcode(string barcode_Code)
        {
            this.barcode_Code = barcode_Code;
            this.barcode_Count = 1;
        }

        public int Barcode_Count
        {
            get { return barcode_Count; }
            set { barcode_Count = value; }
        }
    }
}
```

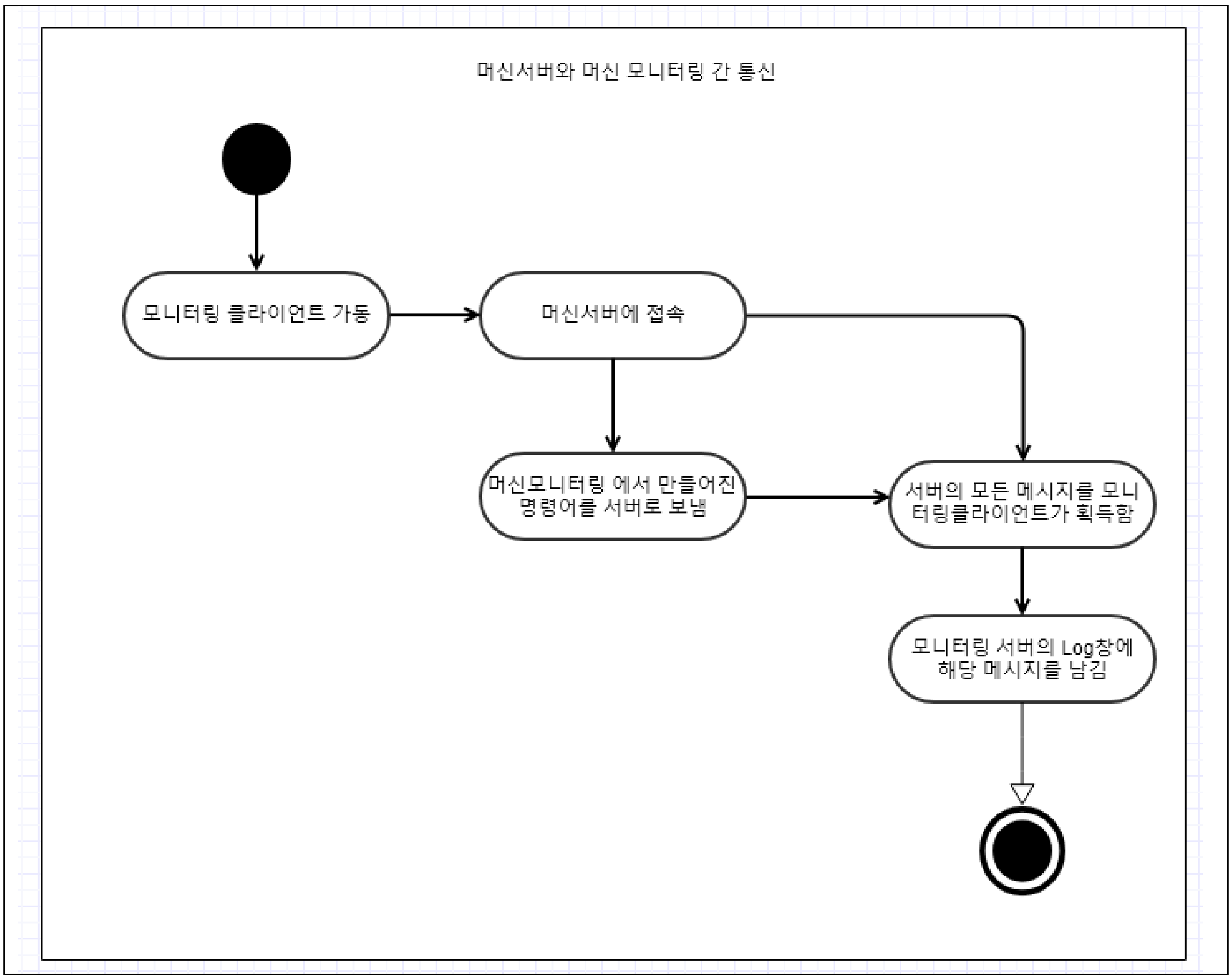
```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MiniErp_Client_jsu
{
    class AppConfiguration
    {
        public static string GetAppConfig(string key)
        {
            return ConfigurationManager.AppSettings[key];
        }

        public static void SetAppConfig(string key, string value)
        {
            Configuration config =
ConfigurationManager.OpenExeConfiguration(ConfigurationUserLevel.None);
            KeyValueConfigurationCollection cfgCollection = config.AppSettings.Settings;

            cfgCollection.Remove(key);
            cfgCollection.Add(key, value);

            config.Save(ConfigurationSaveMode.Modified);

ConfigurationManager.RefreshSection(config.AppSettings.SectionInformation.Name);
        }

        public static void AddAppConfig(string key, string value)
        {
            Configuration config =
ConfigurationManager.OpenExeConfiguration(ConfigurationUserLevel.None);
            KeyValueConfigurationCollection cfgCollection = config.AppSettings.Settings;

            cfgCollection.Add(key, value);
            config.Save(ConfigurationSaveMode.Modified);

ConfigurationManager.RefreshSection(config.AppSettings.SectionInformation.Name);
        }

        public static void RemoveAppConfig(string key)
        {
            Configuration config =
ConfigurationManager.OpenExeConfiguration(ConfigurationUserLevel.None);
            KeyValueConfigurationCollection cfgCollection = config.AppSettings.Settings;

            try
            {
                cfgCollection.Remove(key);

                config.Save(ConfigurationSaveMode.Modified);

ConfigurationManager.RefreshSection(config.AppSettings.SectionInformation.Name);
            }
            catch { }
        }
    }
}
```

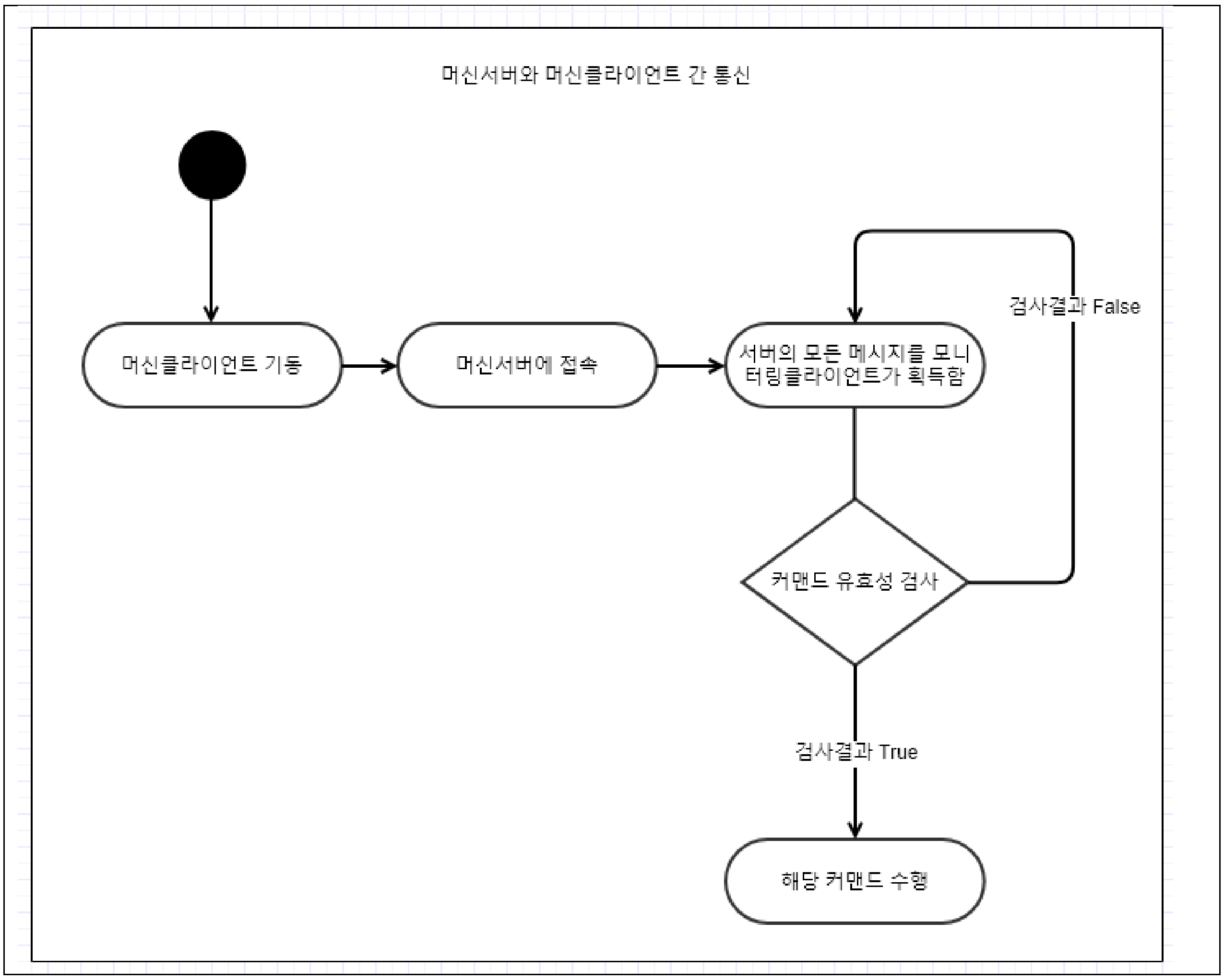
```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using BarcodeLib;

/*
 *  License : Apache-2.0 license
 *  해당 라이브러리 프로젝트 주소
 *  https://github.com/barnhill/barcodelib
 *
 */
namespace MiniERP.VO
{
    class Barcode_Module
    {
        /// <summary>
        /// 바코드 이미지생성함
        /// </summary>
        /// <param name="bacodeValue">바코드 값</param>
        /// <param name="lblChk">바코드에 코드번호 스위치</param>
        /// <param name="size">바코드 사이즈</param>
        /// <returns>바코드 이미지를 반환</returns>
        public Image MakeBarcode(string bacodeValue,bool lblChk,Size size)
        {
            Debug.WriteLine("bacode : MakeBarcode is running");

            Barcode code = new Barcode();
            Image barcode_img;

            if (String.IsNullOrEmpty(bacodeValue))  //  바코드값이 널일경우
            {
                Debug.WriteLine("err_bacode : bacodeValue is null or empty");
                return null;
            }

            if (lblChk != false)                    //  바코드 라벨
            {
                code.IncludeLabel = lblChk;
                code.LabelPosition = BarcodeLib.LabelPositions.BOTTOMCENTER;
            }

            try
            {
                code.RawData = bacodeValue;             //  바코드 정보
                code.Encode(TYPE.CODE128, bacodeValue, Color.Black, Color.White,
size.Width, size.Height);
            }
            catch (Exception)
            {
                Debug.WriteLine("err_bacode : bacode Encoding Exception");
                throw;
            }

            barcode_img = code.EncodedImage;
            Debug.WriteLine("bacode : bacode image make finish");

            return barcode_img;
        }

    }

}
```

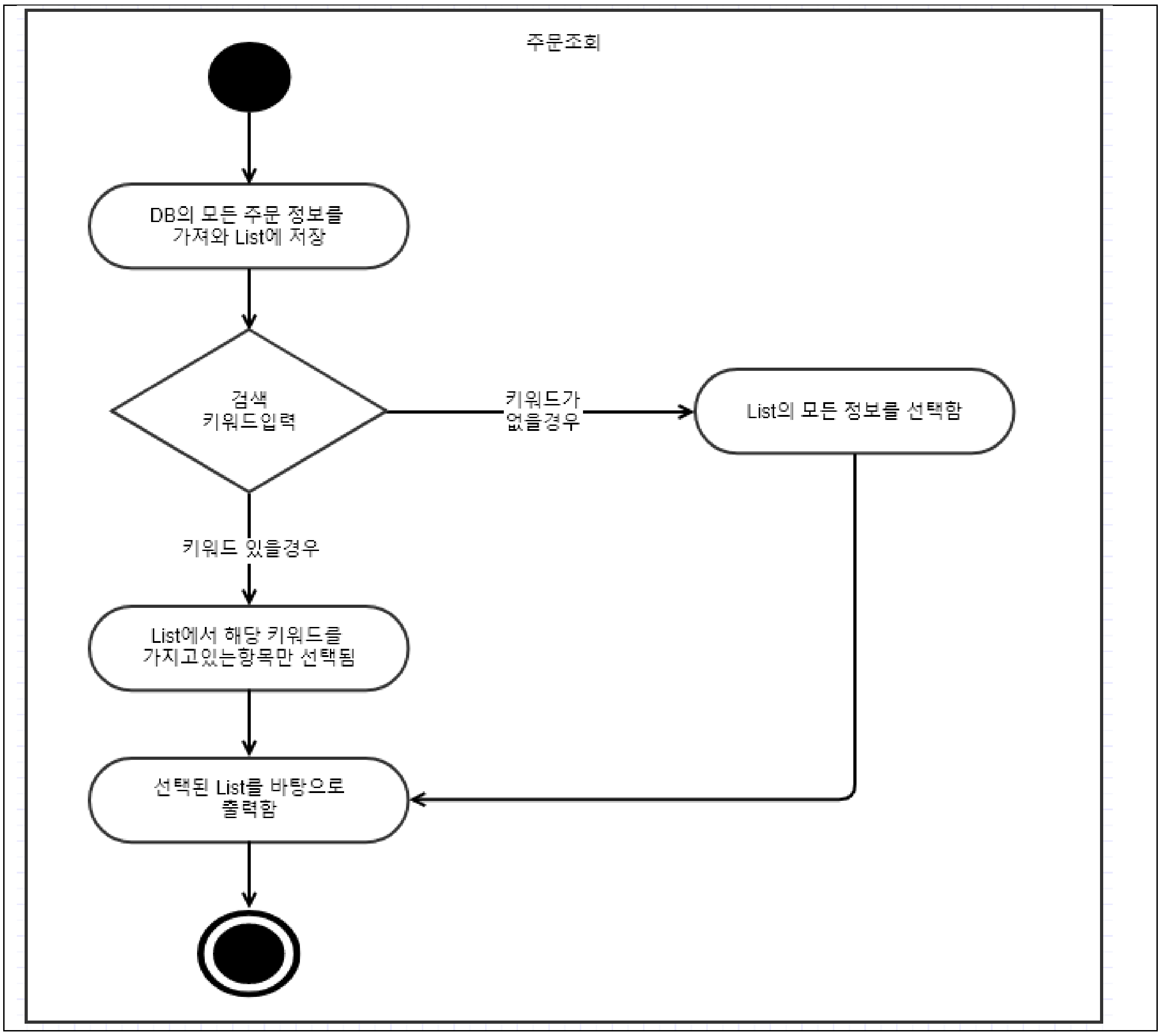
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using MiniERP.Model.DAO;
using MiniERP.VO;
namespace MiniERP.View
{
    public partial class Frm_PrintDisplay : Form
    {
        Barcode_Module barcode = new Barcode_Module();
        List<Item> items = new List<Item>();

        public Frm_PrintDisplay()
        {
            InitializeComponent();
            saveFileDialog1.FileName = DateTime.Today.ToShortDateString() + "_Barcodes";

            for (int i = 1; i < 43; i++)
            {
                combo_Count.Items.Add(i);
            }
        }

        private void Frm_PrintDisplay_Load(object sender, EventArgs e)
        {
            items = new ItemDAO().GetItems("");
            Display();
        }

        /// <summary>
        /// 현재 클래스의 List를 이용해 DataGridView에 내용을 출력합니다.
        /// </summary>
        private void Display()
        {
            DataTable dataTable = new DataTable();
            DataColumn[] dataColumns = new DataColumn[4]
            {
                new DataColumn("아이템코드"),
                new DataColumn("아이템명"),
                new DataColumn("규격"),
                new DataColumn("단위")
            };
            dataTable.Columns.AddRange(dataColumns);
            foreach (var item in items)
            {
                DataRow dataRow = dataTable.NewRow();
                dataRow["아이템코드"] = item.Item_code;
                dataRow["아이템명"] = item.Item_name;
                dataRow["규격"] = item.Item_standard;
                dataRow["단위"] = item.Item_unit;
                dataTable.Rows.Add(dataRow);
            }

            dataGridView1.DataSource = dataTable;
            for (int i = 0; i < dataGridView1.Columns.Count; i++)
            {
                dataGridView1.Columns[i].Width = dataGridView1.Size.Width /
dataGridView1.Columns.Count-1;
            }
        }

        private void dataGridView1_CellContentDoubleClick(object sender,
DataGridViewCellEventArgs e)
        {
            foreach (var item in items)
            {
                if (item.Item_code ==
dataGridView1.SelectedRows[0].Cells["아이템코드"].Value.ToString())
```

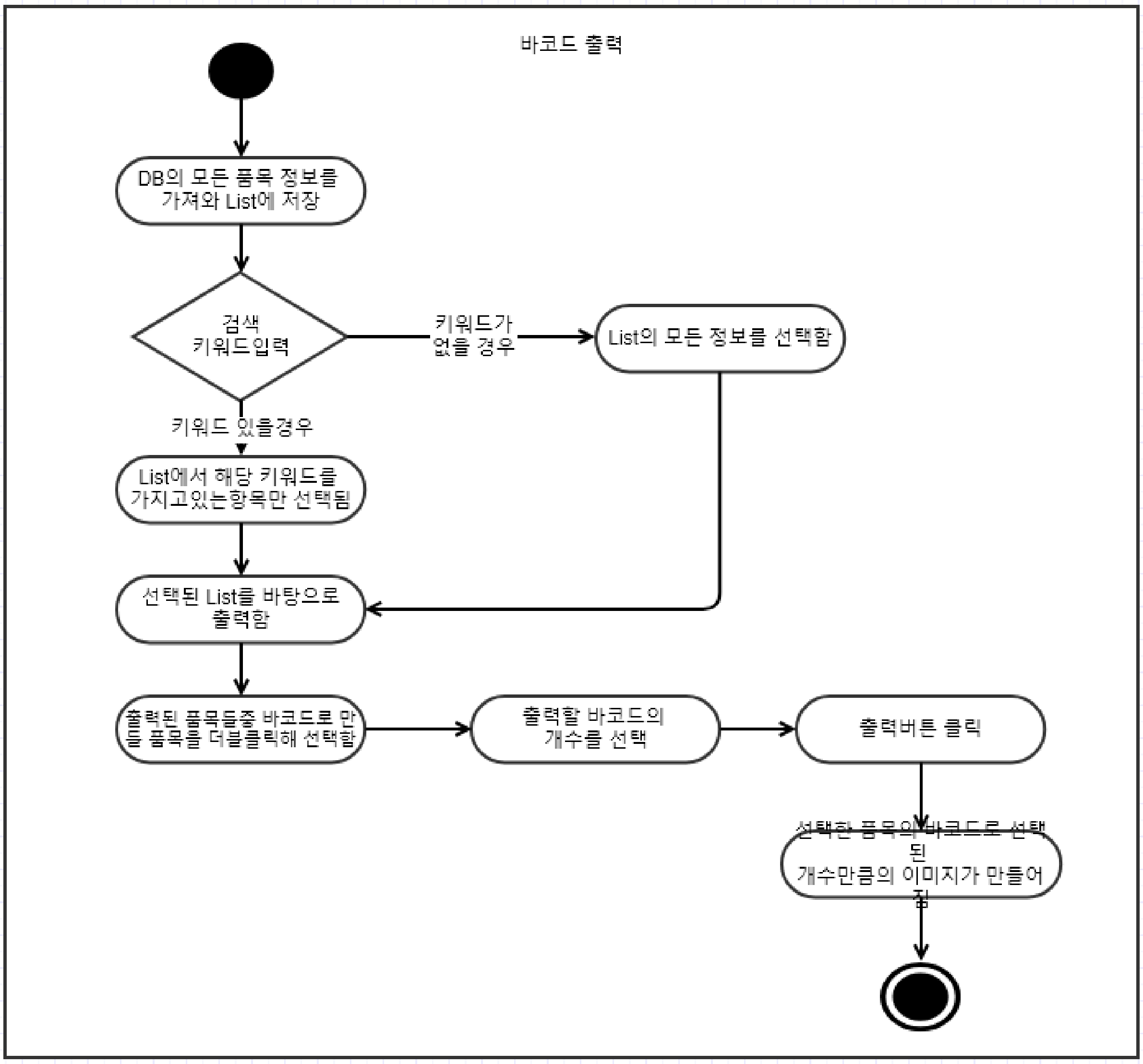
```
                {
                    pictureBox1.Image = null;
                    pictureBox1.Image = barcode.MakeBarcode(item.Item_code, true, new
Size(300, 50));
                    break;
                }
            }
        }

        private void btn_Search_Click(object sender, EventArgs e)
        {
            items = new ItemDAO().GetItems(txt_Search.Text);
            Display();
        }

        private void txt_Search_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.KeyCode == Keys.Enter)
            {
                btn_Search_Click(null, null);
            }
        }
        /// <summary>
        /// 바코드 이미지 내보냅니다.
        /// 출력 경로 지정할 것.
        /// </summary>
        private void btn_Print_Click(object sender, EventArgs e)
        {
            try
            {
                if (pictureBox1.Image == null)
                {
                    MessageBox.Show("바코드를 선택하여 주세요");
                    return;
                }            //  이미지 예외분기
                if (Int32.Parse(combo_Count.Text) > 43)
                {
                    MessageBox.Show("42개 이상불가능합니다.");
                    return;
                }      //  카운트 예외분기
                else if (Int32.Parse(combo_Count.Text) == 0)
                {
                    MessageBox.Show("0은 입력이 불가능합니다.");
                    combo_Count.Text = "1";
                    return;
                }
                if (saveFileDialog1.ShowDialog() != DialogResult.OK)
                {
                    return;
                }

                #region 이미지 이어붙이기
                Bitmap A4 = new Bitmap(1240, 1754);    //  a4 용지 크기
                Size size = pictureBox1.Image.Size;
                Image img = pictureBox1.Image;
                Graphics g = Graphics.FromImage(A4);
                int y = 0; int x = 0;
                for (int i = 0; i < Int32.Parse(combo_Count.Text); i++)
                {
                    //  한 줄에 16개씩 찍히도록..
                    if (i == 14)
                    { x += 400; y = 0; }
                    else if (i == 28)
                    { x += 400; y = 0; }
                    else if (i == 42)       //  3줄 ,갯수 42개 끝
                    { x += 400; y = 0; }

                    g.DrawImage(img, x, y, size.Width, size.Height);
                    y += 130;
                }

                A4.Save(saveFileDialog1.FileName);
                MessageBox.Show("완료");
                #endregion
            }
            catch (Exception)
            {
                MessageBox.Show("숫자만 입력해주세요");
            }
        }
    }
}
```

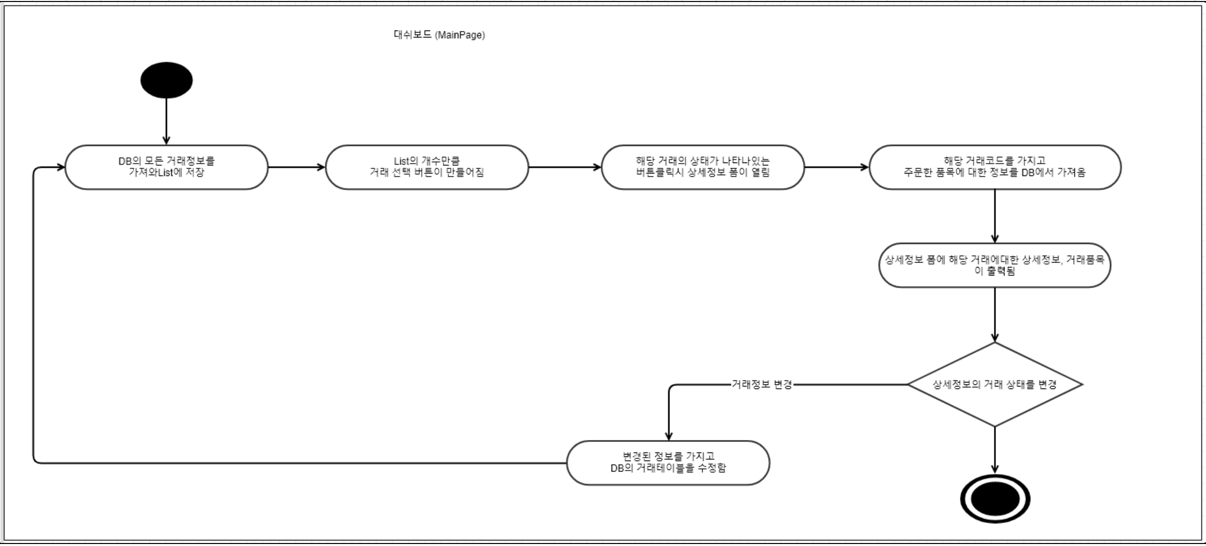
- 액티비티 다이어그램


머신서버와 머신 모니터링 간 통신
모니터링 클라이언트 가동
머신서버에 접속
머신모니터링 에서 만들어진 명령어를 서버로 보냄
서버의 모든 메시지를 모니터링클라이언트가 획득함
모니터링 서버의 Log창에 해당 메시지를 남김

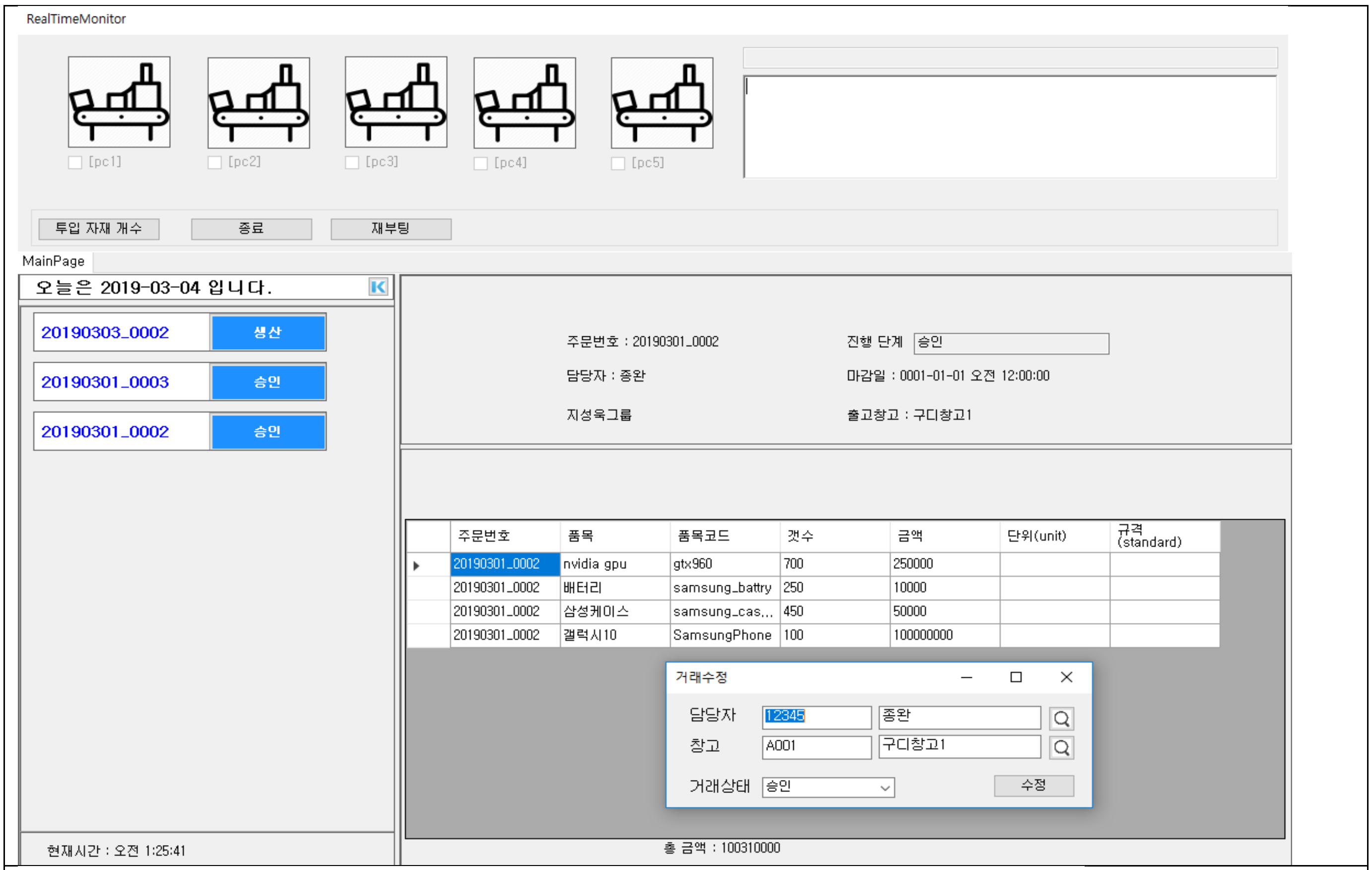

머신서버와 머신클라이언트 간 통신
머신클라이언트 기동
머신서버에 접속
서버의 모든 메시지를 모니터링클라이언트가 획득함
검사결과 False
커맨드 유효성 검사
검사결과 True
해당 커맨드 수행


주문조회
DB의 모든 주문 정보를
가져와 List에 저장
검색
키워드입력
키워드가
없을경우
List의 모든 정보를 선택함
키워드 있을경우
List에서 해당 키워드를
가지고있는항목만 선택됨
선택된 List를 바탕으로
출력함


바코드 출력
DB의 모든 품목 정보를 가져와 List에 저장
검색 키워드입력
키워드가 없을 경우
List의 모든 정보를 선택함
키워드 있을경우
List에서 해당 키워드를 가지고있는항목만 선택됨
선택된 List를 바탕으로 출력함
출력된 품목들중 바코드로 만들 품목을 더블클릭해 선택함
출력할 바코드의 개수를 선택
출력버튼 클릭
선택한 품목의 바코드로 선택된 개수만큼의 이미지가 만들어짐


대쉬보드 (MainPage)
DB의 모든 거래정보를 가져와List에 저장
List의 개수만큼 거래 선택 버튼이 만들어짐
해당 거래의 상태가 나타나있는 버튼클릭시 상세정보 폼이 열림
해당 거래코드를 가지고 주문한 품목에 대한 정보를 DB에서 가져옴
상세정보 폼에 해당 거래에대한 상세정보, 거래품목이 출력됨
상세정보의 거래 상태를 변경
거래정보 변경
변경된 정보를 가지고 DB의 거래테이블을 수정함

- 실행 사진



오늘은 2019-03-04 입니다.

20190303_0002	생산
20190301_0003	승인
20190301_0002	승인

현재시간 : 오전 1:25:41

주문번호 : 20190301_0002　　진행 단계 승인

담당자 : 종완　　마감일 : 0001-01-01 오전 12:00:00

지성욱그룹　　출고창고 : 구디창고1

주문번호	품목	품목코드	갯수	금액	단위(unit)	규격 (standard)
20190301_0002	nvidia gpu	gtx960	700	250000		
20190301_0002	배터리	samsung_battry	250	10000		
20190301_0002	삼성케이스	samsung_cas...	450	50000		
20190301_0002	갤럭시10	SamsungPhone	100	100000000		

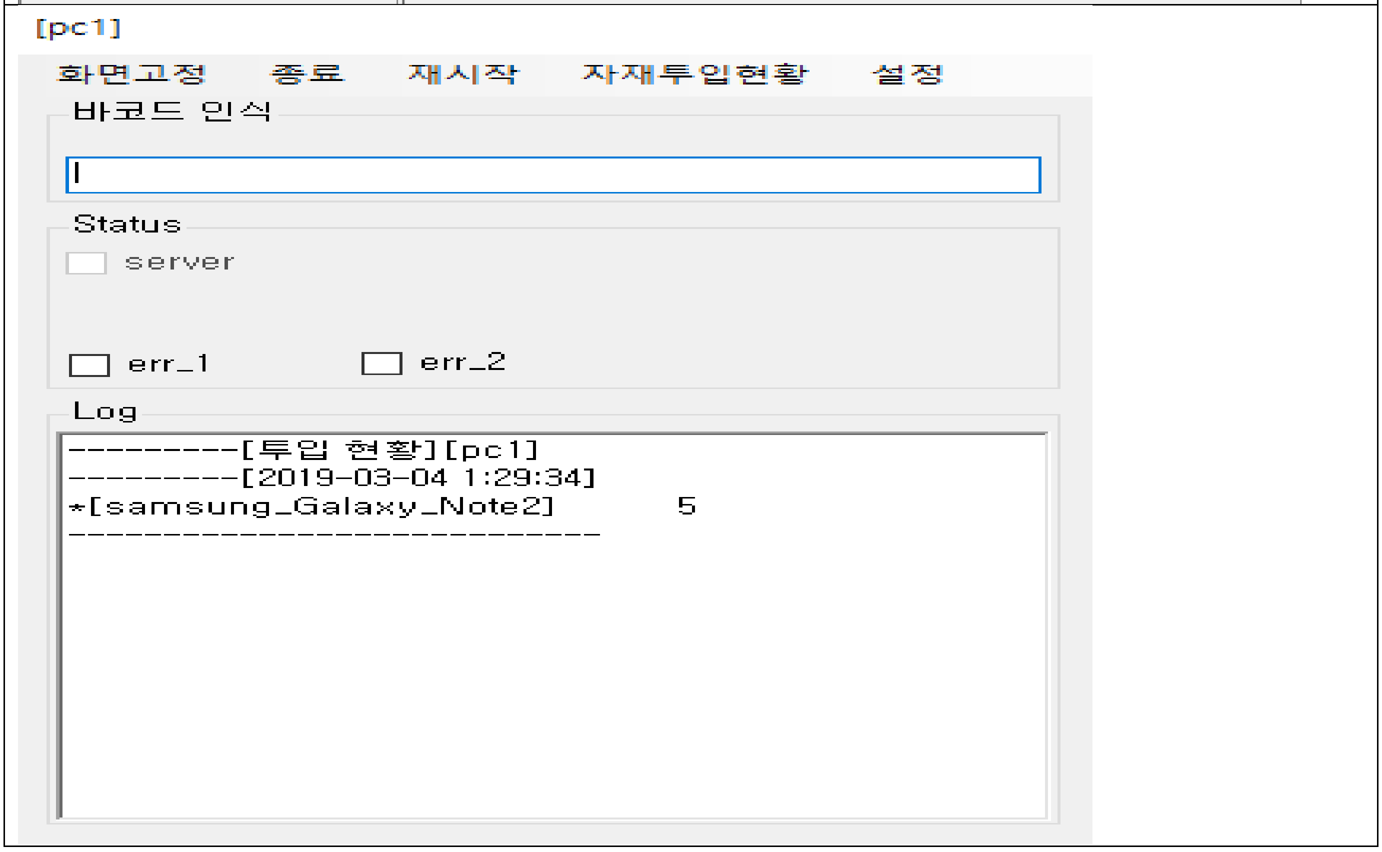
거래수정

담당자 12345 종완

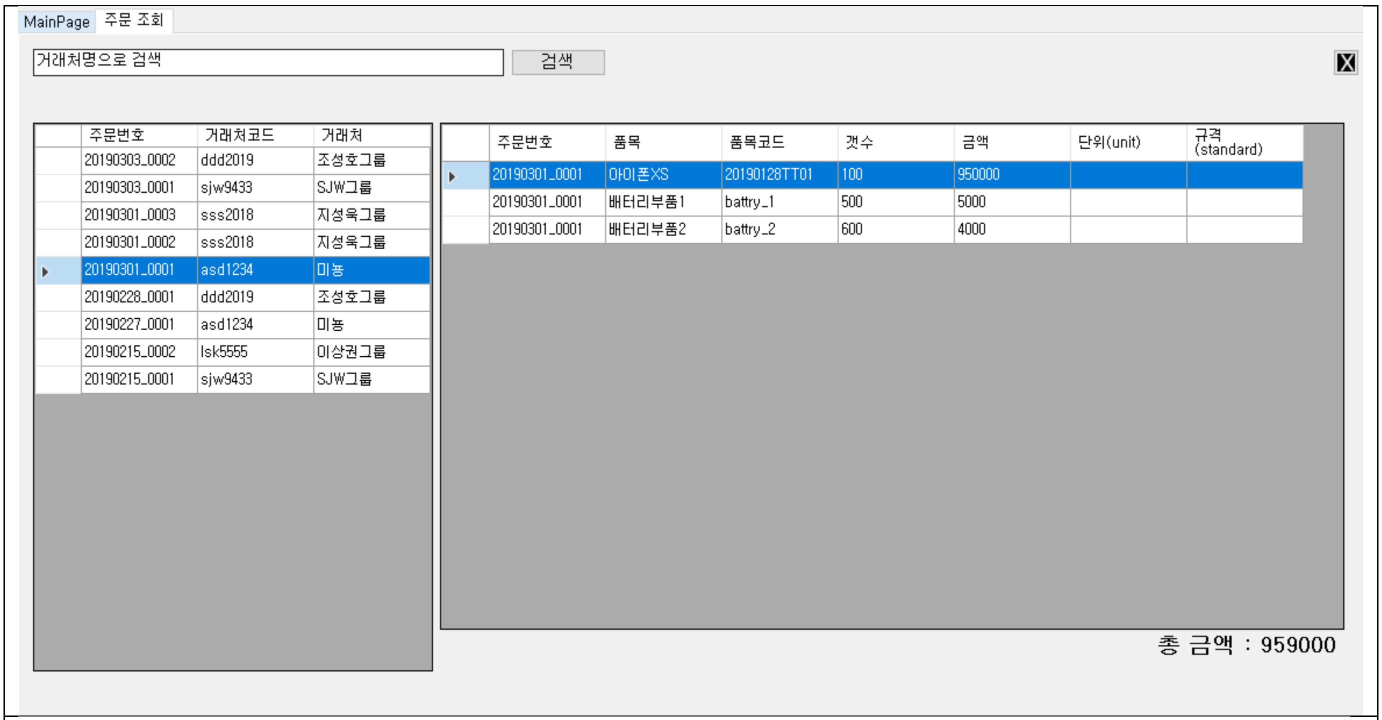
창고 A001 구디창고1

거래상태 승인　수정

총 금액 : 100310000



```
----------[투입 현황][pc1]
----------[2019-03-04 1:29:34]
*[samsung_Galaxy_Note2]        5
------------------------------
```

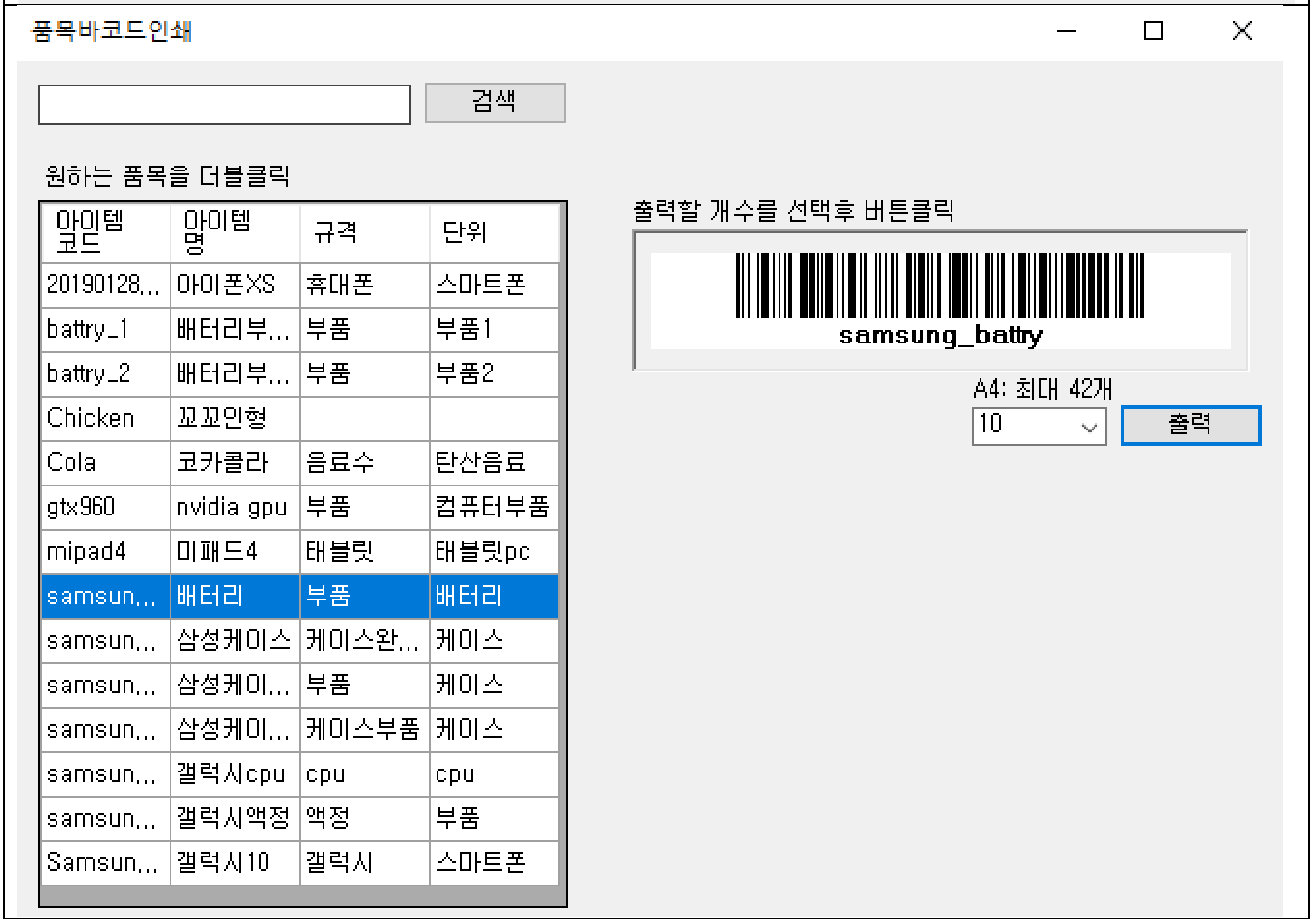

MainPage | 주문 조회

거래처명으로 검색 | 검색

	주문번호	거래처코드	거래처
	20190303_0002	ddd2019	조성호그룹
	20190303_0001	sjw9433	SJW그룹
	20190301_0003	sss2018	지성욱그룹
	20190301_0002	sss2018	지성욱그룹
▸	20190301_0001	asd1234	미농
	20190228_0001	ddd2019	조성호그룹
	20190227_0001	asd1234	미농
	20190215_0002	lsk5555	이상권그룹
	20190215_0001	sjw9433	SJW그룹

	주문번호	품목	품목코드	갯수	금액	단위(unit)	규격 (standard)
▸	20190301_0001	아이폰XS	20190128TT01	100	950000		
	20190301_0001	배터리부품1	battry_1	500	5000		
	20190301_0001	배터리부품2	battry_2	600	4000		

총 금액 : 959000


품목바코드인쇄

검색

원하는 품목을 더블클릭

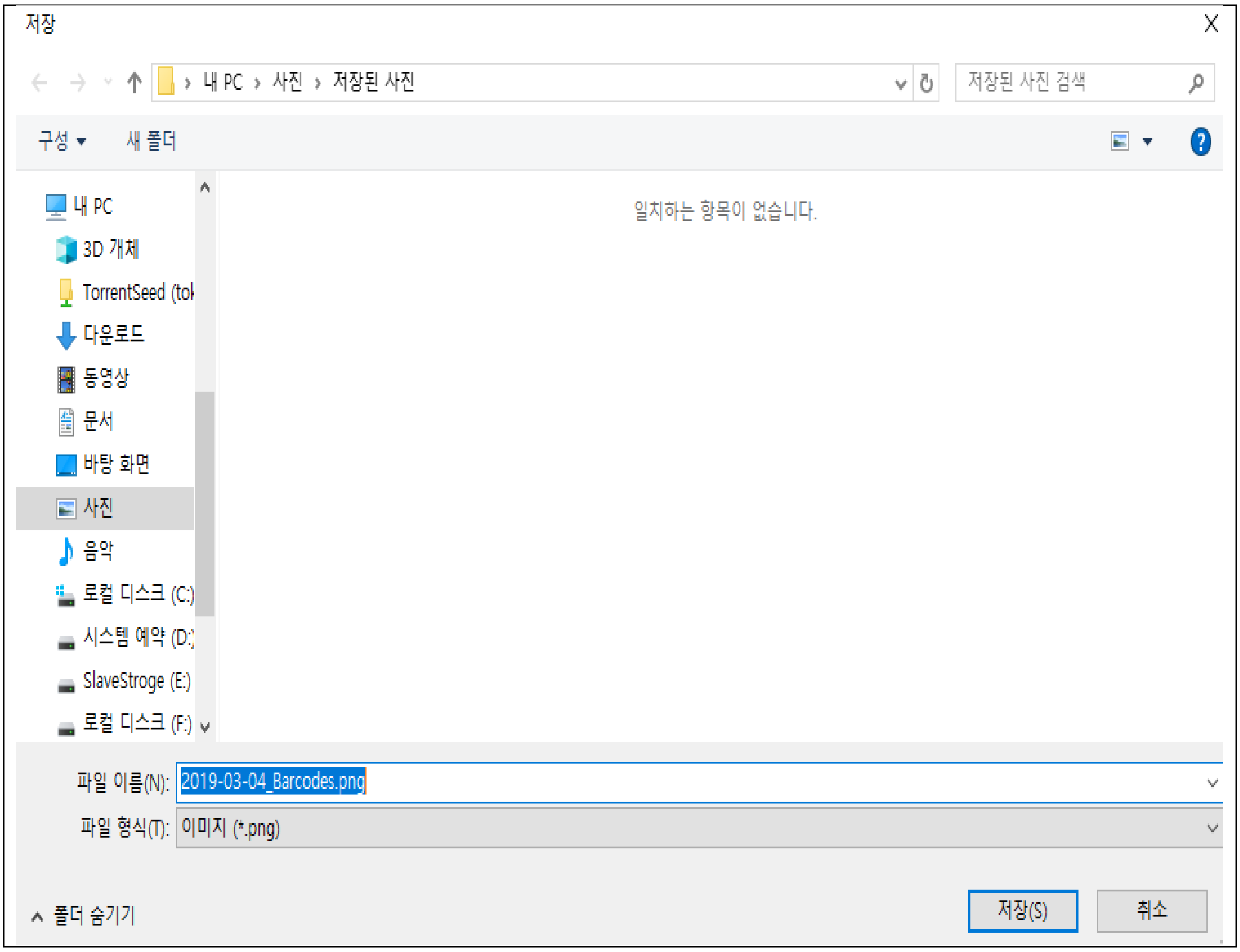
아이템코드	아이템명	규격	단위
20190128...	아이폰XS	휴대폰	스마트폰
battry_1	배터리부...	부품	부품1
battry_2	배터리부...	부품	부품2
Chicken	꼬꼬인형		
Cola	코카콜라	음료수	탄산음료
gtx960	nvidia gpu	부품	컴퓨터부품
mipad4	미패드4	태블릿	태블릿pc
samsun...	배터리	부품	배터리
samsun...	삼성케이스	케이스완...	케이스
samsun...	삼성케이...	부품	케이스
samsun...	삼성케이...	케이스부품	케이스
samsun...	갤럭시cpu	cpu	cpu
samsun...	갤럭시액정	액정	부품
Samsun...	갤럭시10	갤럭시	스마트폰

출력할 개수를 선택후 버튼클릭

samsung_battry

A4: 최대 42개

10 | 출력



파일 이름(N): 2019-03-04_Barcodes.png

파일 형식(T): 이미지 (*.png)

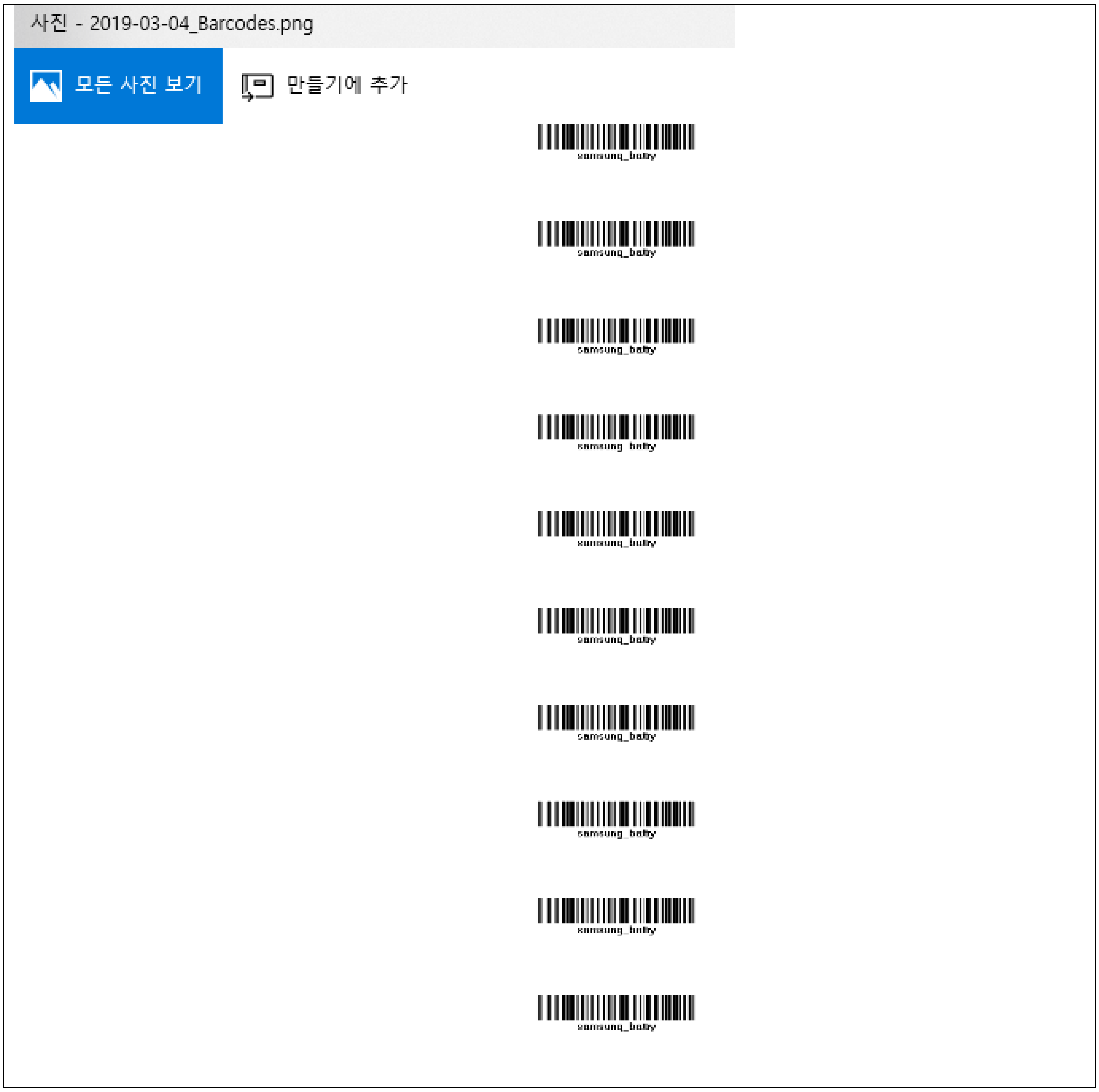


사진 - 2019-03-04_Barcodes.png

samsung_baby

samsung_baby

samsung_baby

samsung_baby

samsung_baby

samsung_baby

samsung_baby

samsung_baby

samsung_baby

samsung_baby