병원 관리 프로그램

- C# DB 연동 프로젝트

김우기

목차

01 개발 환경

02 프로그램 구조

03 프로그램 실행

04 추가/보완 사항

개발 환경

01 사용 프로그램

C#

Visual Studio

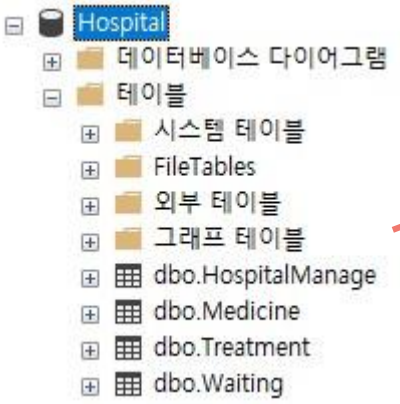
MSSQL

프로그램 구조

02 실행 흐름

프로그램 구조

02 DB 생성 및 연결



DB생성

```
private static SqlConnection conn = new SqlConnection();
public static SqlDataAdapter da;
public static DataSet ds;
public static DataTable dt;

참조 4개
public static void ConnectDB()
{
    conn.ConnectionString = String.Format("Data Source={0};" + "initial Catalog={1};" + "integrated Security={2};" + "Timeout=3", "local", "Hospital", "SSPI");
    conn = new SqlConnection(conn.ConnectionString);
    conn.Open();
}
```

DBHelper로 DB연결

프로그램 구조

02 테이블을 리스트로

```
public class DataManager
{
    public static List<Manager> managers = new List<Manager>();
    public static List<Medicine> medicines = new List<Medicine>();
    public static List<Treat> treatments = new List<Treat>();
    public static List<Waiting> waitings = new List<Waiting>();

    참조 0개
    static DataManager()
    {
        Load();
        MLoad();
        TLoad();
        WLoad();
    }
```

```
public static void Load()
{
    try
    {
        DBHelper.selectQuery();
        managers.Clear();

        foreach (DataRow item in DBHelper.dt.Rows)
        {
            Manager manager = new Manager();
            manager.id = int.Parse(item["Id"].ToString());
            manager.password = item["Password"].ToString();
            manager.name = item["Name"].ToString();
            manager.position = item["Position"].ToString();
            managers.Add(manager);
        }
    }
    catch (Exception ex)
    {
        System.Windows.Forms.MessageBox.Show(ex.Message);
        //System.Windows.Forms.MessageBox.Show("Load오류");
    }
```

```
public static void MLoad()
{
    try
    {
        MDBHelper.selectQuery();
        medicines.Clear();

        foreach (DataRow item in MDBHelper.dt.Rows)
        {
            Medicine medicine = new Medicine();
            medicine.code = item["Code"].ToString();
            medicine.name = item["Name"].ToString();
            medicine.amount = int.Parse(item["Amount"].ToString());
            medicines.Add(medicine);
        }
    }
    catch (Exception ex)
    {
        System.Windows.Forms.MessageBox.Show(ex.Message);
        //System.Windows.Forms.MessageBox.Show("MLoad오류");
    }
```

```
public static void TLoad()
{
    try
    {
        TDBHelper.selectQuery();
        treatments.Clear();

        foreach (DataRow item in TDBHelper.dt.Rows)
        {
            Treat treat = new Treat();
            treat.chartNum = int.Parse(item["ChartNum"].ToString());
            treat.pCode = int.Parse(item["Code"].ToString());
            treat.pName = item["Name"].ToString();
            treat.pBirth = DateTime.Parse(item["Birth"].ToString());
            treat.pGen = item["Gender"].ToString();
            treat.pNum = item["Phone"].ToString();
            treat.pAddress = item["Address"].ToString();
            treat.pVisit = DateTime.Parse(item["Visit"].ToString());
            treat.pDiagnosis = item["Diagnosis"].ToString();
            treat.pMedicine = item["Medicine"].ToString();
            treatments.Add(treat);
        }
    }
    catch (Exception ex)
    {
        System.Windows.Forms.MessageBox.Show(ex.Message);
        //System.Windows.Forms.MessageBox.Show("TLoad오류");
```

```
public static void WLoad()
{
    try
    {
        WDBHelper.selectQuery();
        waitings.Clear();

        foreach (DataRow item in WDBHelper.dt.Rows)
        {
            Waiting wait = new Waiting();
            wait.ChartNum = int.Parse(item["ChartNum"].ToString());
            wait.Code = int.Parse(item["Code"].ToString());
            wait.Name = item["Name"].ToString();
            wait.Birth = DateTime.Parse(item["Birth"].ToString());
            wait.Gender = item["Gender"].ToString();
            wait.Phone = item["Phone"].ToString();
            wait.Address = item["Address"].ToString();
            waitings.Add(wait);
        }
    }
    catch (Exception ex)
    {
        System.Windows.Forms.MessageBox.Show(ex.Message);
        //System.Windows.Forms.MessageBox.Show("WLoad오류");
```

각 테이블을 get/set을 통해 리스트로 저장

프로그램 구조

02 쿼리문

```
public static void selectQuery(int chartNum = -1)
{
    try
    {
        ConnectDB();

        SqlCommand cmd = new SqlCommand();
        cmd.Connection = conn;
        if (chartNum == -1)
        {
            cmd.CommandText = "select * from Treatment order by ChartNum";
        }
        else
        {
            cmd.CommandText = "select * from Treatment " + " where ChartNum =" + chartNum;
        }

        da = new SqlDataAdapter(cmd);
        ds = new DataSet();
        da.Fill(ds, "Treatment");
        dt = ds.Tables[0];
    }
    catch (Exception)
    {

    }
    finally
    {
        conn.Close();
```

DB를 연결하여 테이블의 데이터를 불러옴

DB에 있는 4개의 테이블을 동일한 방식으로 연결

프로그램 구조

02 쿼리문

```
public static void dataInsertQuery(string chartNum, string pCode, string pName, string pBirth, string pGen, string pNum, string pAddress, string pVisit, string pDiagnosis, string pMedicine, string command)
{
    string sqlcommand = "";
    if (command == "insert")
    {
        sqlcommand = "insert into Treatment values (@p1, @p2, @p3, @p4, @p5, @p6, @p7, @p8, @p9, @p10)";
    }
    try
    {
        ConnectDB();
        SqlCommand cmd = new SqlCommand();
        cmd.Connection = conn;
        cmd.CommandType = CommandType.Text;
        cmd.Parameters.AddWithValue("@p1", chartNum);
        cmd.Parameters.AddWithValue("@p2", pCode);
        cmd.Parameters.AddWithValue("@p3", pName);
        cmd.Parameters.AddWithValue("@p4", pBirth);
        cmd.Parameters.AddWithValue("@p5", pGen);
        cmd.Parameters.AddWithValue("@p6", pNum);
        cmd.Parameters.AddWithValue("@p7", pAddress);
        cmd.Parameters.AddWithValue("@p8", pVisit);
        cmd.Parameters.AddWithValue("@p9", pDiagnosis);
        cmd.Parameters.AddWithValue("@p10", pMedicine);
        cmd.CommandText = sqlcommand;
        cmd.ExecuteNonQuery();
    }
    catch (Exception)
    {

    }
    finally
    {
        conn.Close();
    }
```

DB에 데이터를 추가하기 위한 쿼리 메소드

```
public static void dataDeleteQuery(string chartNum, string command)
{
    string sqlcommand = "";
    if (command == "delete")
    {
        sqlcommand = "delete from Treatment where ChartNum=@p1";
    }
    try
    {
        ConnectDB();
        SqlCommand cmd = new SqlCommand();
        cmd.Connection = conn;
        cmd.CommandType = CommandType.Text;
        cmd.Parameters.AddWithValue("@p1", chartNum);
        cmd.CommandText = sqlcommand;
        cmd.ExecuteNonQuery();
    }
    catch (Exception)
    {

    }
    finally
    {
        conn.Close();
    }
```

DB에서 데이터를 삭제하기 위한 쿼리 메소드

프로그램 구조

02 쿼리문

```
public static void dataUpdateQuery(string pCode, string pVisit, string pDiagnosis, string pMedicine, string command)
{
    string sqlcommand = "";
    if (command == "update")
    {
        sqlcommand = "update Treatment set Visit=@p1, Diagnosis=@p2, Medicine=@p3 where Code=@p4";
    }
    try
    {
        ConnectDB();
        SqlCommand cmd = new SqlCommand();
        cmd.Connection = conn;
        cmd.CommandType = CommandType.Text;
        cmd.Parameters.AddWithValue("@p1", pVisit);
        cmd.Parameters.AddWithValue("@p2", pDiagnosis);
        cmd.Parameters.AddWithValue("@p3", pMedicine);
        cmd.Parameters.AddWithValue("@p4", pCode);
        cmd.CommandText = sqlcommand;
        cmd.ExecuteNonQuery();
    }
    catch (Exception)
    {

    }
    finally
    {
        conn.Close();
    }
}
```

```
public static void dataUpdateQuery(string pCode, string pName, string pGen, string pNum, string pAddress, string command)
{
    string sqlcommand = "";
    if (command == "update")
    {
        sqlcommand = "update Treatment set Name=@p1, Gender=@p2, Phone=@p3, Address=@p4 where Code=@p5";
    }
    try
    {
        ConnectDB();
        SqlCommand cmd = new SqlCommand();
        cmd.Connection = conn;
        cmd.CommandType = CommandType.Text;
        cmd.Parameters.AddWithValue("@p1", pName);
        cmd.Parameters.AddWithValue("@p2", pGen);
        cmd.Parameters.AddWithValue("@p3", pNum);
        cmd.Parameters.AddWithValue("@p4", pAddress);
        cmd.Parameters.AddWithValue("@p5", pCode);
        cmd.CommandText = sqlcommand;
        cmd.ExecuteNonQuery();
    }
    catch (Exception)
    {

    }
    finally
    {
        conn.Close();
    }
```

DB에 저장된 데이터를 수정하기 위한 쿼리 메소드
- 변수를 통해 다양한 업데이트 쿼리 메소드 구현

```
public static void dataViewQuery(string pName)
{
    try
    {
        ConnectDB();
        SqlCommand cmd = new SqlCommand();
        cmd.Connection = conn;
        cmd.CommandText = "SELECT * FROM Treatment where Name=@p1";
        cmd.Parameters.AddWithValue("@p1", pName);

        da = new SqlDataAdapter(cmd);
        ds = new DataSet();
        da.Fill(ds, "Treatment");
        dt = ds.Tables[0];
    }
    catch (Exception ex)
```

검색을 위한 쿼리 메소드

프로그램 구조

02 DataManager

```
public static void Save(string command, string id, string password, string name, string position, out string contents)
{
    contents = "";

    if (command == "insert")
    {
        DBInsert(id, password, name, position, ref contents);
    }
}

참조 1개
public static void Save(string command, string id, out string contents)
{
    contents = "";

    if (command == "delete")
    {
        DBDelete(id, ref contents);
```

```
private static bool DBDelete(string id, ref string contents)
{
    if (DBHelper.dt.Rows.Count != 0)
    {
        DBHelper.deleteQuery(id);
        contents = $"관리자 번호 {id}이/가 삭제되었습니다.";
        return true;
    }
    else
    {
        contents = $"해당 관리자 번호 {id}가 존재하지 않습니다.";
        return false;
    }
}

참조 1개
private static bool DBInsert(string id, string password, string name, string position, ref string contents)
{
    if (DBHelper.dt.Rows.Count !=0)
    {
        DBHelper.insertQuery(id, password, name, position);
        contents = $"관리자 번호 {id}이/가 추가되었습니다.";
        return true;
    }
    else
    {
        contents = $"해당 관리자 번호 {id}가 이미 존재합니다.";
        return false;
```

DataManager를 통해 DBHelper를 연결하고
DB에 접근하여 테이블을 확인 후
메시지 박스를 띄움

프로그램 구조

02 로그 기록

```
public static void printLog(string contents)
{
    DirectoryInfo di = new DirectoryInfo("LogHistory");

    if (di.Exists == false)
    {
        di.Create();
    }

    using (StreamWriter w = new StreamWriter("LogHistory₩₩LogHistory.txt", true))
    {
        w.WriteLine(contents);
    }
```

로그 기록을 생성하여, 오류 생성시 대처 가능

LogHistory - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

[2022-09-07 02:13:07 오후]관리자 번호 1001가 이미 존재합니다.
[2022-09-07 02:13:20 오후]관리자 번호 1003이/가 추가되었습니다.
[2022-09-07 02:13:40 오후]관리자 번호 1004이/가 추가되었습니다.
[2022-09-07 02:13:56 오후]관리자 번호 1005이/가 추가되었습니다.
[2022-09-07 02:14:13 오후]관리자 번호 1006이/가 추가되었습니다.
[2022-09-08 11:58:49 오전]의약품 번호 3002이/가 추가되었습니다.
[2022-09-08 11:59:08 오전]의약품 번호 3003이/가 추가되었습니다.
[2022-09-08 12:00:46 오후]의약품 번호 3003이/가 삭제되었습니다.
[2022-09-08 12:00:49 오후]의약품 번호 3003이/가 삭제되었습니다.
[2022-09-08 12:03:44 오후]의약품 번호 3003이/가 추가되었습니다.
[2022-09-08 12:05:05 오후]의약품 번호 3004이/가 추가되었습니다.

프로그램 구조

02 추가/수정/삭제

```
private void wait_new(string chartNum, string pCode, string p
{
    chartNum = chartNum.Trim();

    string contents = "";

    DataManager.WSave(command, chartNum, pCode, pName, pBirth
    button_Refresh.PerformClick();
    MessageBox.Show(contents);
    writeLog(contents);
}

참조 1개
private void wait_delete(string pCode, string command)
{
    pCode = pCode.Trim();

    string contents = "";

    DataManager.WSave(command, pCode, out contents);
    button_Refresh.PerformClick();
    MessageBox.Show(contents);
    writeLog(contents);
}
```

데이터 수정 및 삭제 버튼 메소드

```
private string lookUpChartNum(int chartNum)
{
    string usedChartNum = "";
    try
    {
        foreach (var item in DataManager.treatments)
        {
            if (item.chartNum == int.Parse(chartNum.ToString()))
            {
                usedChartNum = item.chartNum.ToString();
                break;
```

데이터 추가할 때 동일한 데이터가
삽입되는 것을 방지

```
private void treat_new(string chartNum, string pCode, string pName, string pBirth, strin
{
    chartNum = chartNum.Trim();
    int.TryParse(chartNum, out int pChartNum);
    int.TryParse(pCode, out int ppCode);
    string contents = "";
    string usedcn = lookUpChartNum(int.Parse(chartNum));
    string usedco = lookUppCode(int.Parse(pCode));
    if (pChartNum >= 10000)
    {
        if (ppCode >= 5000)
        {

        }
        else
        {
            writeLog("환자 번호는 5000이상의 숫자여야 합니다.");
            MessageBox.Show("환자 번호는 5000이상의 숫자여야 합니다.");
            return;
        }
    }
    else
    {
        writeLog("차트 번호는 10000이상의 숫자여야 합니다.");
        MessageBox.Show("차트 번호는 10000이상의 숫자여야 합니다.");
        return;
    }

    if (chartNum == usedcn)
    {
        writeLog($"차트 번호 {chartNum}가 이미 존재합니다.");
        MessageBox.Show($"차트 번호 {chartNum}가 이미 존재합니다.");
        return;
    }
    else if (pCode == usedco)
    {
        writeLog($"환자 번호 {pCode}가 이미 존재합니다.");
        MessageBox.Show($"환자 번호 {pCode}가 이미 존재합니다.");
        return;
    }
    else
    {
        DataManager.TSave(command, chartNum, pCode, pName, pBirth, pGen, pNum, pAddress,
        button_Refresh.PerformClick();
        MessageBox.Show(contents);
        writeLog(contents);
```

데이터 추가 버튼 메소드

프로그램 구조

02 검색 / 키보드 효과

```
private void find_name(string pName)
{
    if (pName == textBox_Finder.Text)
    {
        DataManager.FindLoad(pName);
        MessageBox.Show($"{pName} 환자를 찾았습니다.");
        writeLog($"{pName} 환자를 찾았습니다.");
    }
    else
    {
        MessageBox.Show($"{pName} 환자는 존재하지 않습니다.");
        writeLog($"{pName} 환자는 존재하지 않습니다.");
    }
```

검색을 통해 환자를
찾을 수 있는 기능

```
private void button_Find_Click(object sender, EventArgs e)
{
    dataGridView_Find.DataSource = null;
    find_name(textBox_Finder.Text);
    dataGridView_Find.DataSource = DataManager.treatments;
}
```

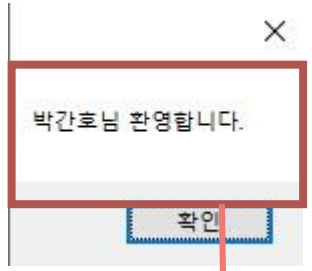
```
public Nurse()
{
    InitializeComponent();
    label_Now.Text = DateTime.Now.ToString("yyyy년 MM월 dd일 hh시 mm분 ss초 tt");
    this.textBox_Finder.KeyDown += button_Find_KeyDown;
```

편한 사용을 위해
키보드 이벤트를 활용하여
Enter를 누르면 클릭과 같은 효과 발생

```
private void button_Find_KeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Enter)
    {
        button_Find_Click(sender, e);
    }
}
```

프로그램 실행

03 로그인 기능



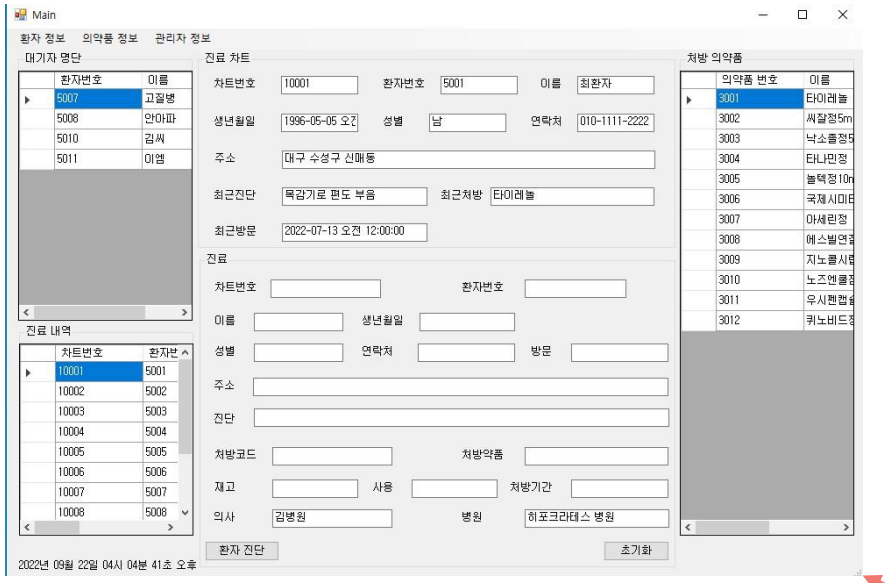
사용자 이름을 메시지 박스로 표기

로그인 Form을 사용해 프로그램 접속

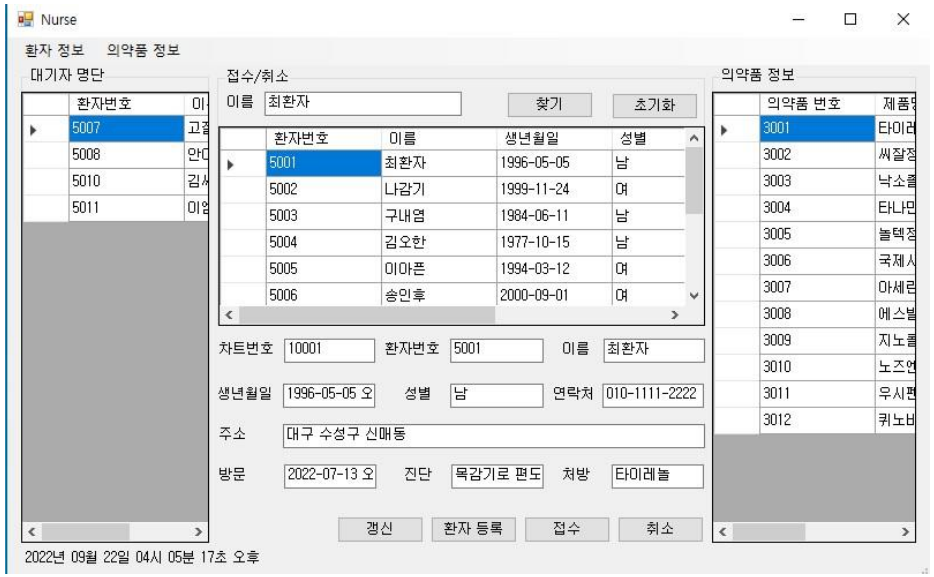
프로그램 실행

03 사용자 구분

의사 및 관리자 창



간호사 창

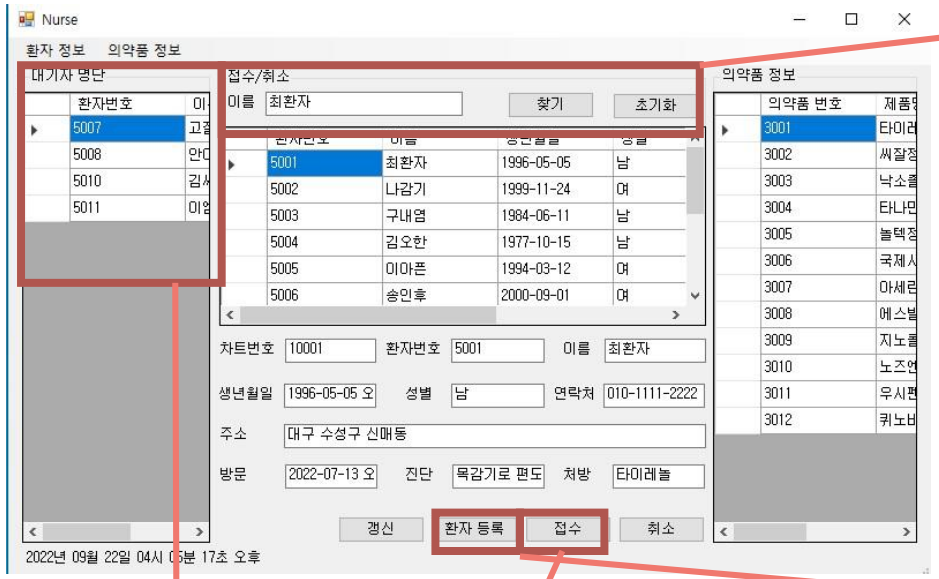


의사와 간호사를 구분하여
서로 다른 창으로 접속

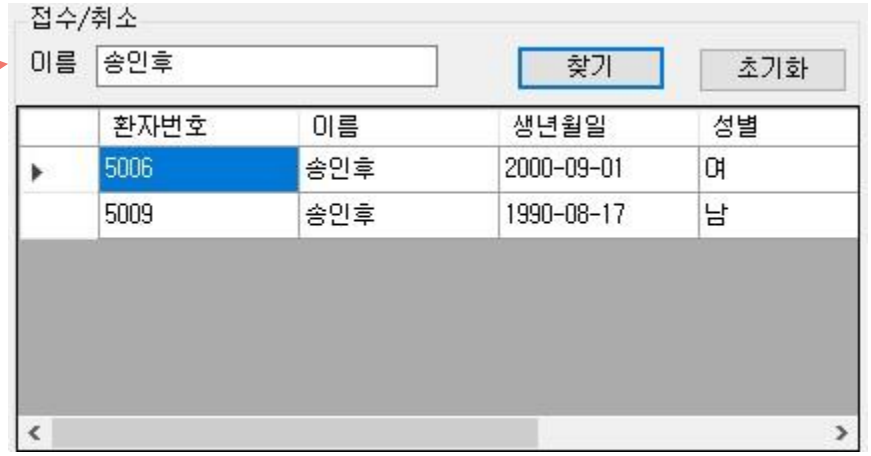
```
private void button_Login_Click(object sender, EventArgs e)
{
    if(textBox_Id.Text.Trim() == "")
    {
        MessageBox.Show("ID를 입력해주세요.");
    }
    else
    {
        try
        {
            Manager manage=DataManager.managers.Single(x => x.id.ToString() == textBox_Id.Text);
            if (textBox_Password.Text.Trim() == "")
            {
                MessageBox.Show("비밀번호를 입력해주세요.");
            }
            else
            {
                if (manage.id==int.Parse(textBox_Id.Text))
                {
                    if(manage.password==textBox_Password.Text)
                    {
                        if (manage.position == "간호사")
                        {
                            MessageBox.Show($"{manage.name}님 환영합니다.");
                            Nurse nurse = new Nurse();
                            nurse.ShowDialog();
                            DataManager.Load();
                            textBox_Id.Text = "";
                            textBox_Password.Text = "";
                        }
                        else
                        {
                            MessageBox.Show($"{manage.name}님 환영합니다.");
                            Main main = new Main();
                            main.doc = manage.name;
                            main.ShowDialog();
                            DataManager.Load();
                            textBox_Id.Text = "";
                            textBox_Password.Text = "";
                        }
                    }
                    else
                    {
                        MessageBox.Show("비밀번호가 다릅니다.");
                    }
                }
                else
                {
                    MessageBox.Show("존재하지 않는 ID입니다.");
```

프로그램 실행

03 간호사 창

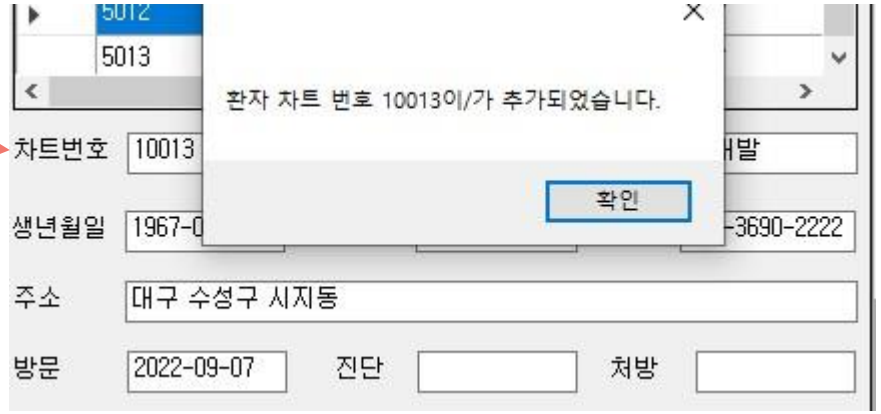


등록된 환자 검색 기능



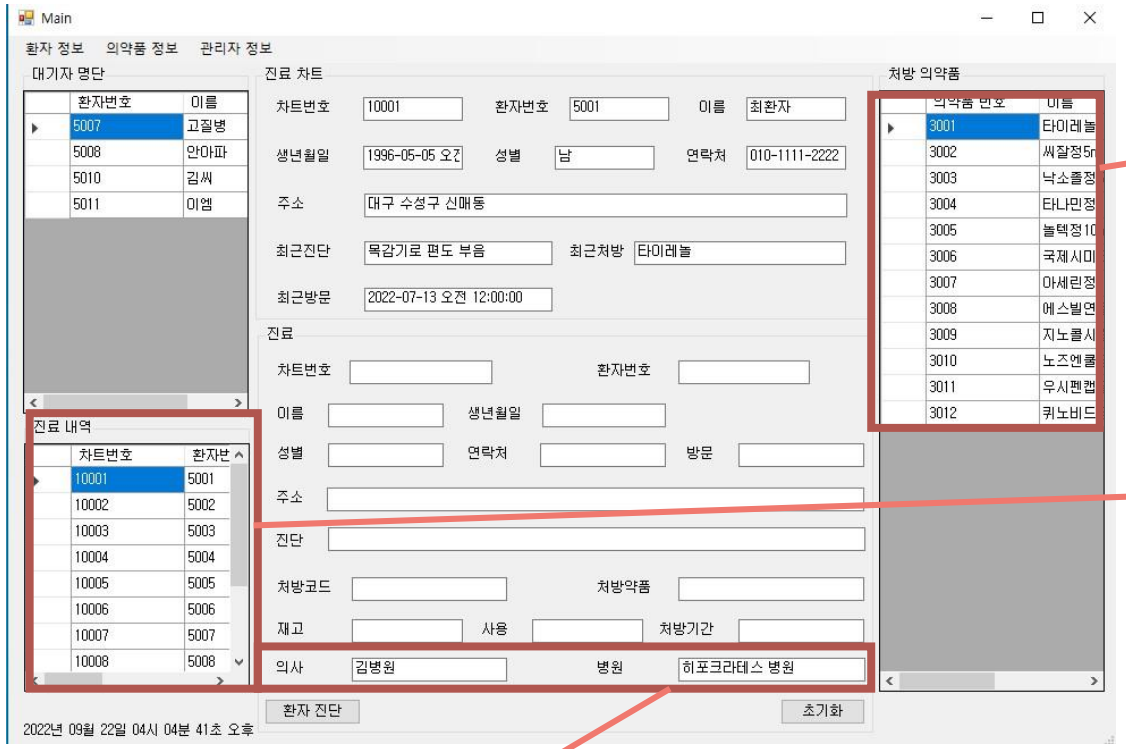
환자 접수하면서 대기자 명단에 추가

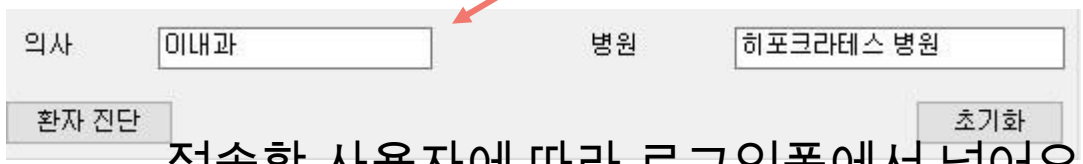
신규 환자 등록



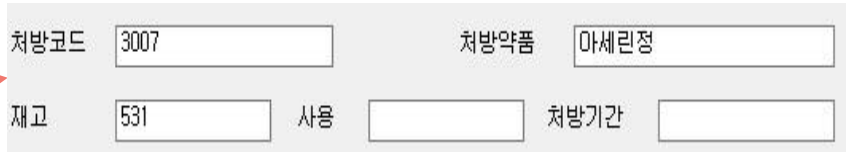
프로그램 실행

03 의사/관리자 창

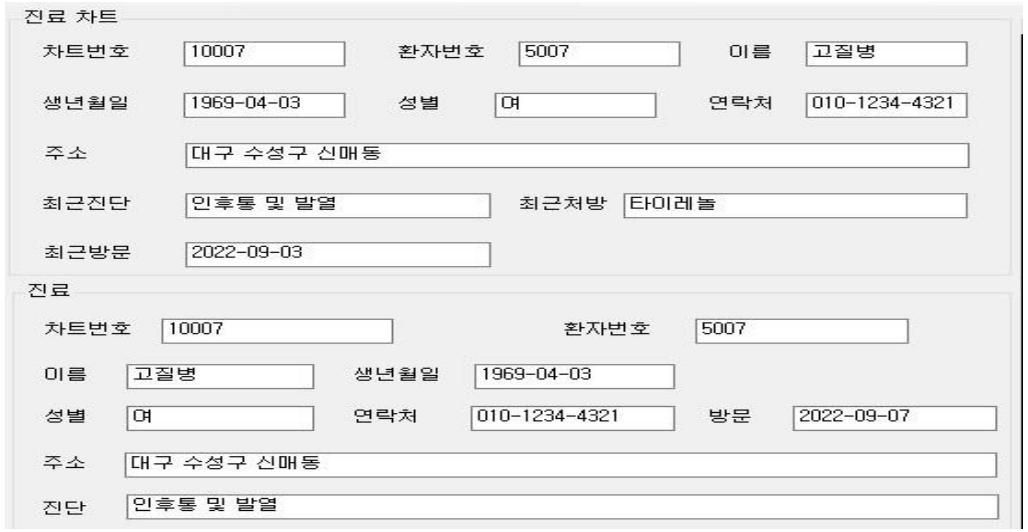




접속한 사용자에 따라 로그인폼에서 넘어오는 이름이 달라짐



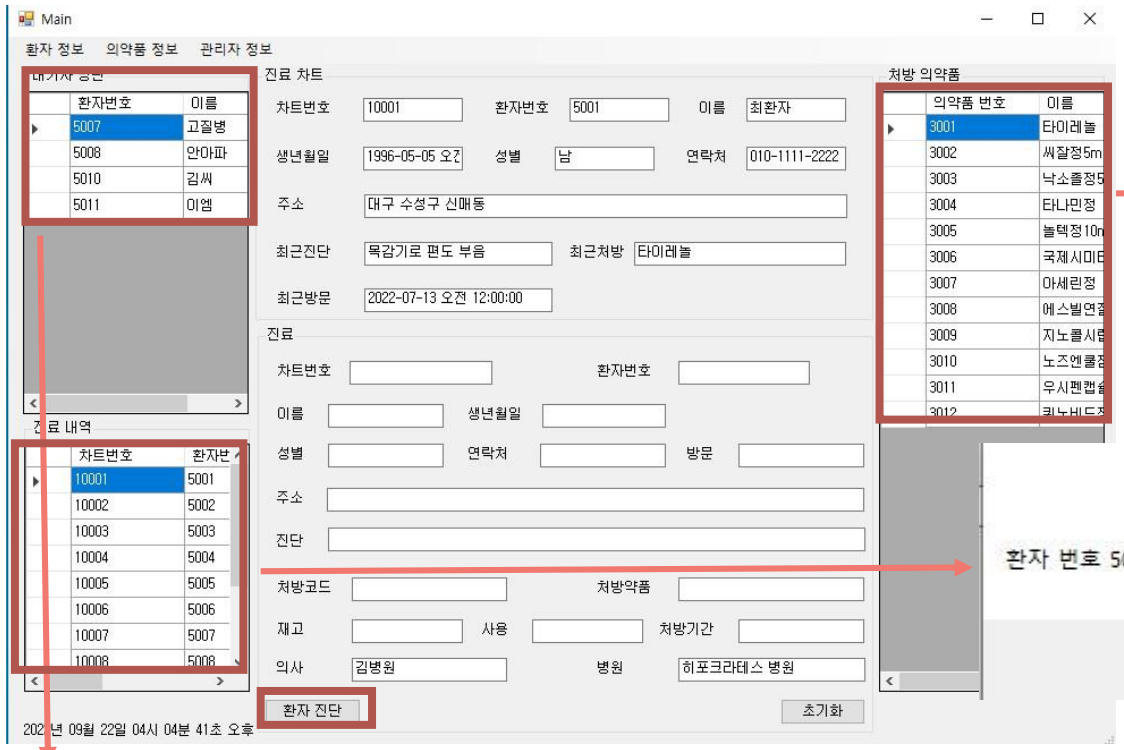
의약품 클릭하면 진료 창에 나타남



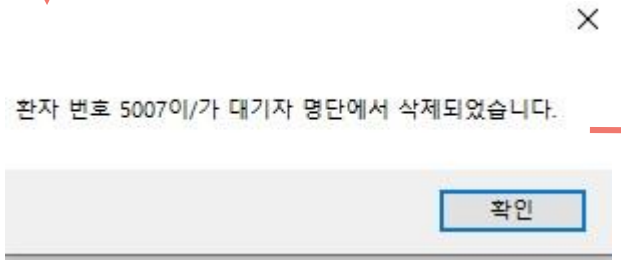
진료 내역의 창을 누르면 진료 차트와 차트 창에 환자 정보 나타남

프로그램 실행

03 의사/관리자 창



	이름	재고수량
	타이레놀	138
	씨잘정5mg	270
	낙소졸정500/2...	103
	타나민정	34
	놀텍정10mg	34
	국제시미티딘정	336
▶	아세린정	507
	에스빌연질캡슐	42
	지노콜시럽	79
	노즈엔쿨점비액	68
	우시펜캡슐	134
	퀴노비드정	367



	환자번호	이름
▶	5008	안아파
	5010	김씨
	5011	이엠
	5013	이개발

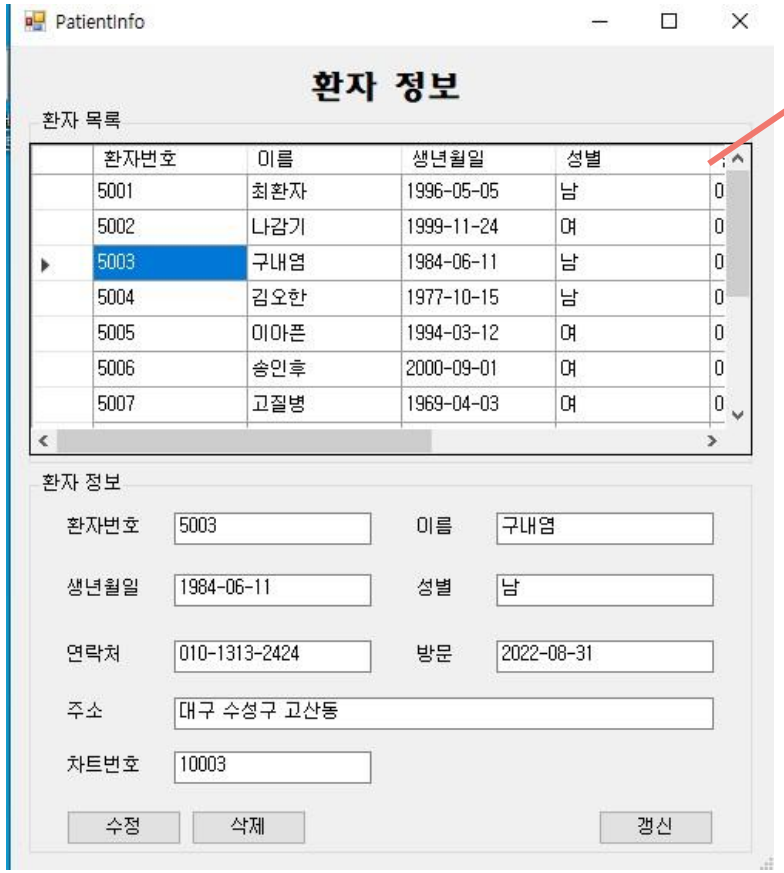
환자 진단 버튼 누르면
3곳의 데이터 수정/삭제

의약품 테이블의 수량/진료 내역 수정

대기환자 테이블에서 해당 환자 삭제

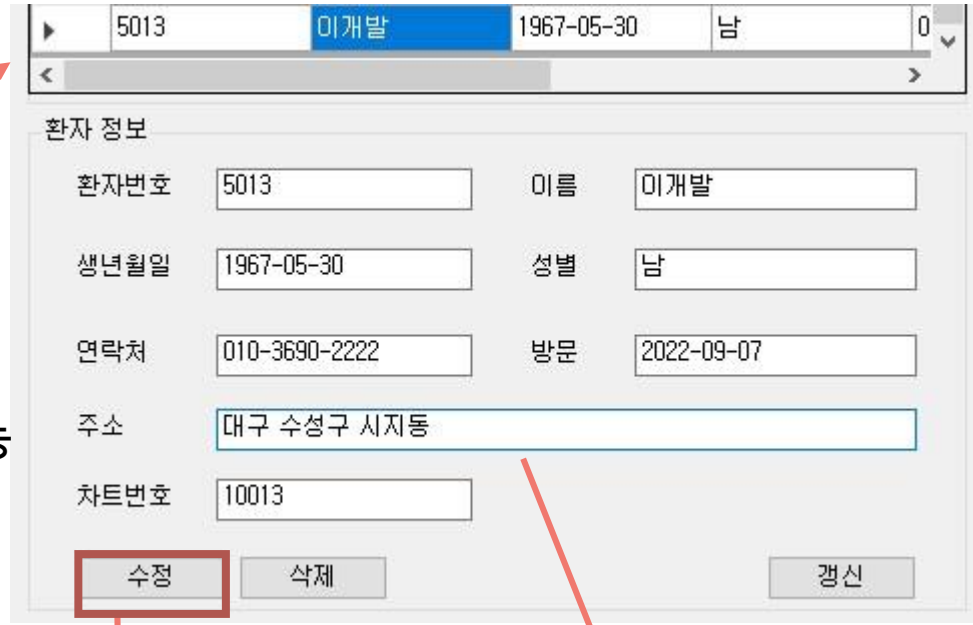
프로그램 실행

03 환자 관리 창

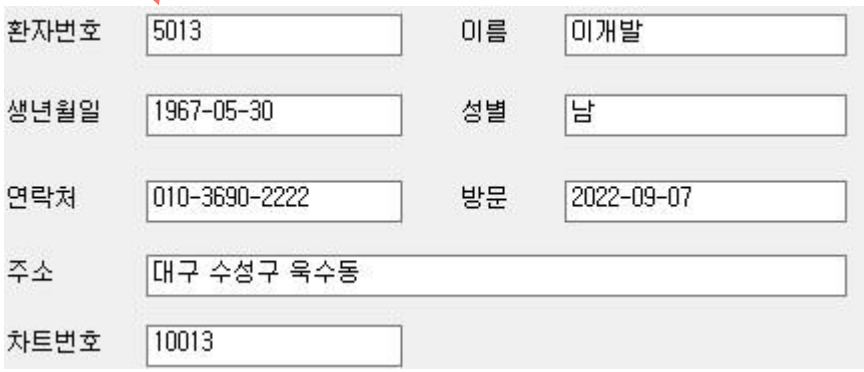


환자 정보 창

셀을 클릭하면 텍스트 박스로 정보 확인 가능

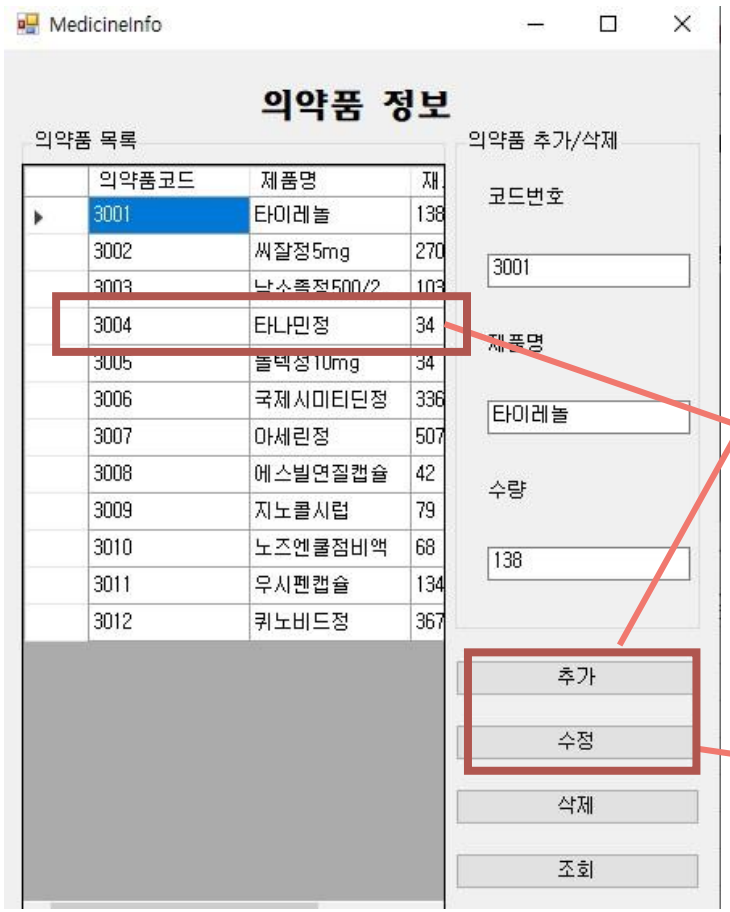


수정 버튼으로 환자 정보를 업데이트

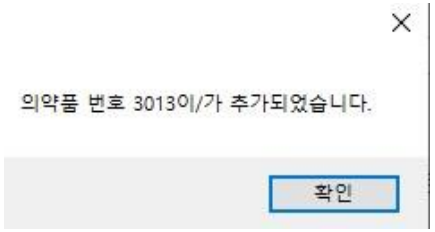


프로그램 실행

03 의약품 관리 창

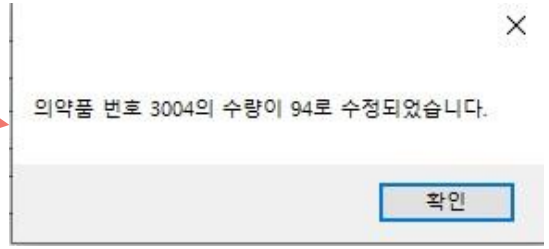


의약품 관리 창



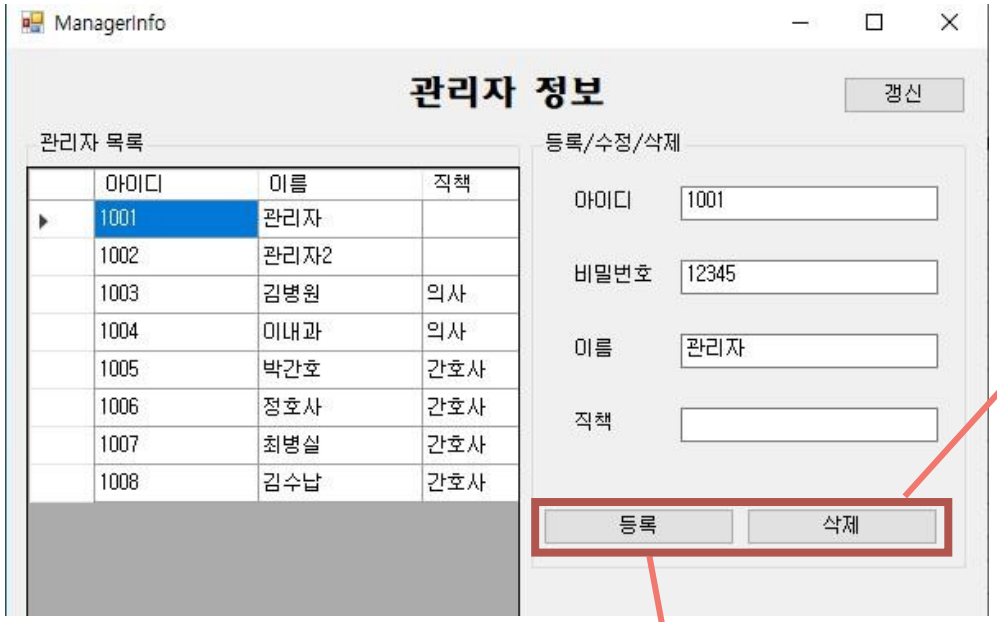
의약품을 추가하여 리스트가 업데이트된 모습

의약품의 수량을 수정한 모습



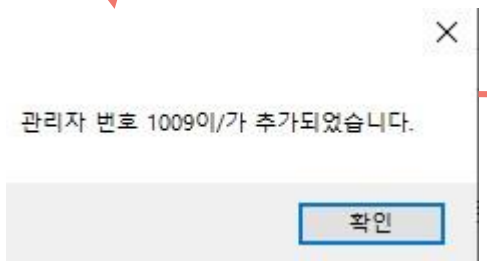
프로그램 실행

03 직원 관리 창



직원 관리 창

관리자가 삭제된 모습



관리자를 등록한 모습

추가 / 보완 사항

04 추가 / 보완

01 약국 Form

주사제와 의약품을 구분하여, 의약품은 약국 DB와 Form을 만들어서 약국에서 관리하고, 병원과 연동되는 것으로 보완 필요

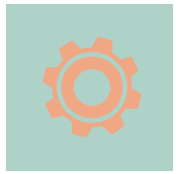
02 관리자 Form

의사와 관리자를 구분하는 것이 필요. 관리자 Form을 만들어서 관리자 Form에서 직원 관리를 하는 것으로 넘기고, 의사 Form에서는 환자 진료만 하도록 구분 필요

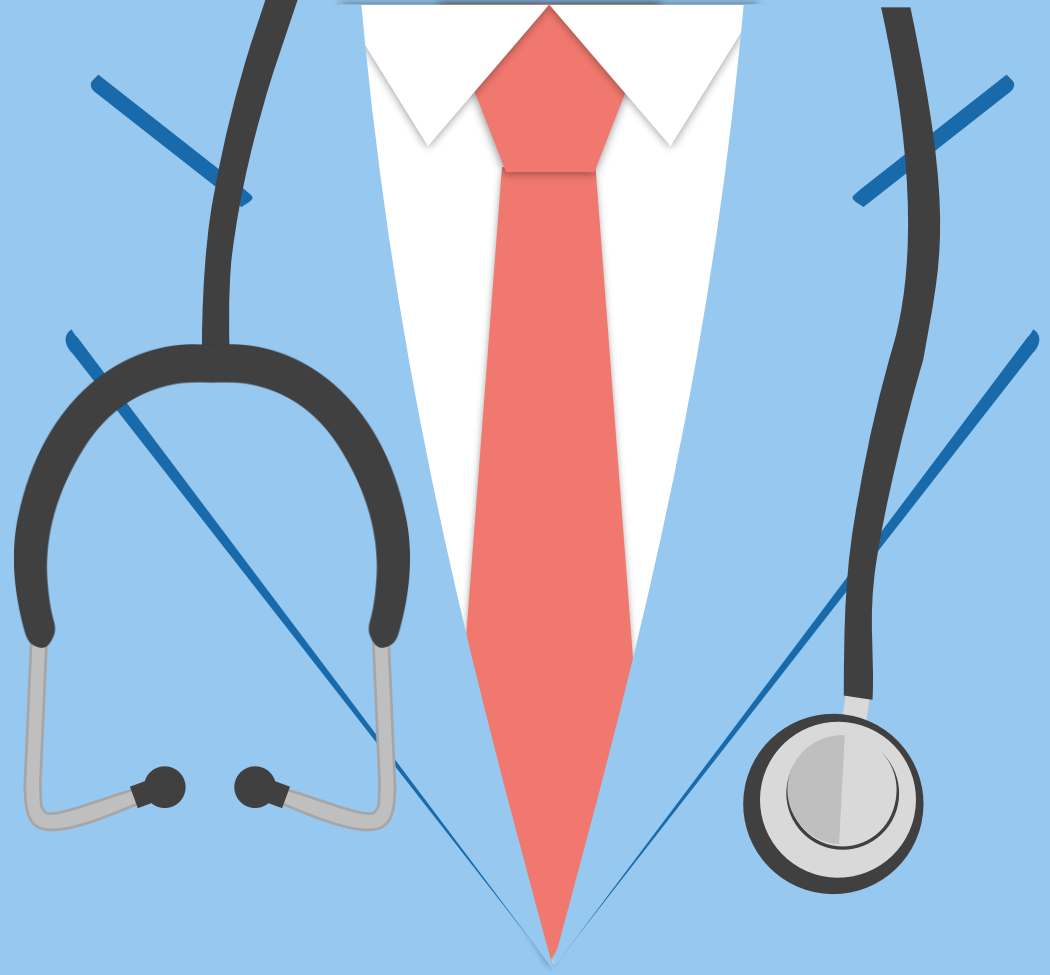
03 매출

관리 기능은 구현되어 있으나, 매출 기능이 없음. 병원비를 계산하는 기능을 구현하고, 이를 토대로 매출액을 산출하는 프로그램으로 업그레이드 필요

04 기기관리



의약품 관리는 구현되어 있으나, 기기관리를 위한 Form은 마련되어 있지 않음. 기기관리 Form을 추가하여, 병원 종합관리 프로그램이 될 수 있도록 업그레이드 필요



THANK YOU

Any Question?